Л.М. Круподьорова
А.М.Пєтух

Технологія програмування мовою Сі
Частина 2
Лабораторний практикум

Міністерство освіти і науки України
Вінницький національний технічний університет

Л.М. Круподьорова
А.М. Петух

Технологія програмування мовою Сі

Частина 2
Лабораторний практикум

Затверджено Вченою радою Вінницького національного технічного університету як навчальний посібник для студентів напряму підготовки 0804 "Комп'ютерні науки" спеціальностей "Інтелектуальні системи прийняття рішень" та "Програмне забезпечення автоматизованих систем". Протокол № 10 від 27 квітня 2006 р.

Вінниця ВНТУ 2006

УДК 681.31
К 84

Рецензенти:
С.М. Москвіна, кандидат технічних наук, доцент
О.Д. Азаров, доктор технічних наук, професор
О.М. Роік, доктор технічних наук, професор
О.В. Сілагін, к. т. н., директор ТОВ " І Т І "

Рекомендовано до видання Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України

Круподьорова Л.М., Пєтух А.М.
К 84 **Технологія програмування мовою Сі. Частина 2. Лабораторний практикум** – Навчальний посібник. – Вінниця: ВНТУ, 2006. – 184 с.

В навчальному посібнику розглянуто основні способи складання програм при виконанні лабораторних і контрольних робіт згідно з тематичним планом. Детально розглянуто роботу в середовищі програмування, наведені типові варіанти робіт, довідкова інформація та тестові завдання для поточного контролю. Посібник може бути використаний як студентами, так і викладачами при викладанні споріднених дисциплін.

Навчальний посібник призначений для студентів напряму підготовки 0804 "Комп'ютерні науки" спеціальностей "Інтелектуальні системи прийняття рішень" та "Програмне забезпечення автоматизованих систем".

УДК 681.31

Зміст

Введення

Даний посібник містить основні розділи для роботи в середовищі програмування Borland C++ версії 3.1, які дають змогу студентам самостійно оволодіти навичками програмування. В посібнику пояснено правила користування вбудованими системами налагодження та створення багатофайлових програм при написанні курсових та дипломних робіт. Розроблено вісім тематичних лабораторних робіт, в яких наведені типові приклади для складання програм, правила підготовки звітів для захисту лабораторних робіт. Кожна робота має контрольні запитання для самостійної перевірки знань. Наведений розділ з привилами оформлення контрольних робіт для студентів заочної форми навчання. Для полегшення розв'язання задач на ЕОМ в додатку наведений повний перелік помилок, які виникають на етапах компіляції та запуску програм. Написані тести дають можливість перевірити свої знання із вивчення такої цікавої мови програмування як Сі.

1 Робота в інтегрованому середовищі BORLAND C++

1.1 Основні відомості про систему BORLAND C++

Інтегроване середовище програмування – Integrated Development Environment (IDE) подане у вигляді багатовіконного інтерфейсу, підтримує роботу з мишею, має можливість швидкого переходу до роботи з асемблером, здійснює підсвічування різних видів текстів різним кольором (тексту програми, ключових слів, коментарів і ін.). В навчальному посібнику розглядаються основи використання системи Borland C++ версії 3.1 (середовище DOS).

1.2 Ідентифікація файлів

Програми і дані в Borland C++ зберігаються на магнітному диску (МД) у вигляді файлів (фізичних файлів). У файлах можуть зберігатися, наприклад:

- тексти початкових модулів – програм на мові Сі++;
- програми, які вже відкомпільовані;
- програми, готові до виконання;
- початкові дані і результат.

Кожен файл повинен мати унікальний ідентифікатор – ім'я файлу. Воно формується за правилами DOS і складається з двох частин: імені файлу і розширення. Ім'я може містити від 1 до 8 символів. У Borland C++ використовується ряд загальноприйнятих (стандартних для для Сі++) розширень. У таблиці 1 наведені деякі з них.

Таблиця 1 – Розширення імен файлів Borland C++

Розширення	Призначення файлу
.cpp	Текст програми на мові Си++
.bak	Попередня копія файлу (до її заміни)
.obj	Об'єктний модуль функції – результат компіляції
.prj	Файл проекту
.exe	Програма, готова до виконання: exe-файл
.h	Головний (header) файл – інтерфейси функцій бібліотек
.lib	Бібліотеки Borland ++
.tc	Файл для настроювання середовища Borland C++; він відновлює після перерви роботи з Borland C++ середовище в тому вигляді, в якому воно було при завершенні останнього сеансу роботи з Borland C++

1.3 Початок і завершення роботи системи

Систему Borland C++ можна викликати за допомогою відповідного ярлика на робочому столі. Крім того, можна увійти до каталога BC – системи Borland C++, знайти в його підкаталозі BIN файл bc.exe або turbo.exe і запустити його на виконання.

Після успішного виклику системи Borland C++ екран набуде вигляду, наведеного на рис. 1. Верхній рядок екрану – рядок основного меню, під основним меню знаходиться вікно редактора з текстом програми. Самий нижній рядок екрану – рядок стану вікна містить список команд і відповідних їм клавіш, які найчастіше використовуються при роботі в середовищі.

Для виходу з системи треба виконати команду Alt+X.

Маркер закриття вікна Маркер відкриття вікна

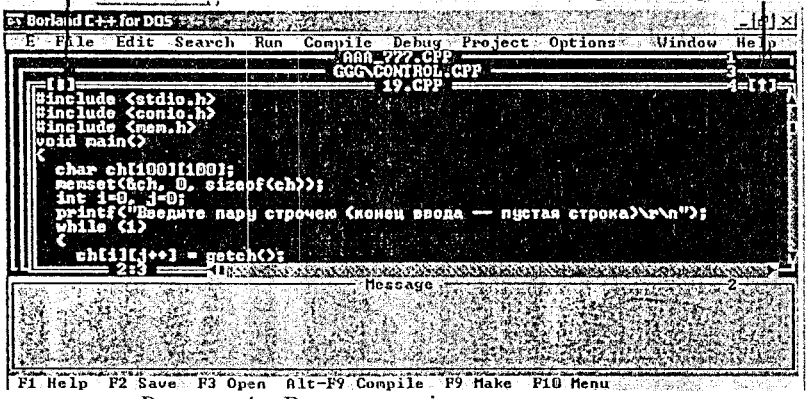


Рисунок 1 – Вид екрану після виклику системи.

Після появи екрану можна виконувати всі види робіт: створювати і розв'язувати програми, створювати і коректувати файли початкових даних і ін. Виконання будь-якої команди нижнього рядка екрану можна викликати клацанням миші на її тексті або натисненням її гарячої клавіші. Призначення основних гарячих клавіш середовища подане в таблиці 3.

1.4 Вікна системи

Вікно – це екран або його частина, де розташована відповідна вікну інформація. У системі є декілька типів вікон. Основні типи вікон: вікно редактора (рис. 1), діалогове (рис. 3), вікно повідомлень. Вікно повідомлень містить інформацію, наприклад, про результати процесу компіляції або компоновки (див. рис. 12 і 14).

В процесі розробки програми в системі Borland C++ використовуються вікна: редактора (Edit), 2 налагоджувальних (повідомлень – Message і перегляду – Watch) і екран користувача. Одночасно на екрані можна бачити декілька вікон.

У кожен момент часу активним може бути тільки одне вікно. Активне вікно має подвійні лінії рамки. Якщо вікна перекриваються, активне вікно завжди знаходиться поверх інших.

Велика частина вікон має такі елементи;

- рядок заголовка вікна;
- маркер закриття вікна (у лівому верхньому кутку);
- смуги прокрутки (справа і знизу);
- куток зміни розміру вікна (правий нижній);
- маркер відкриття вікна на весь екран (на верхній рамці вікна справа); номер вікна (від 1 до 9).

Спеціальний тип вікна – блок діалогу. Він може мати елементи керування: радіокнопки (маленькі квадратні або круглі віконця) для задання режимів роботи, кнопки заданої дії, вікна введення рядка і вікна із списками. Вікна діалогу не можуть бути розкриті на весь екран. Викликати дію будь-якої кнопки можна, якщо підвести до неї курсор миші і клацнути на ній лівою клавішею, або вибрати кнопку за допомогою клавіші Tab і натиснути клавішу Введення.

Вікно редагування має в лівій нижній частині рамки номер рядка і стовпця, в яких розташований курсор. На рис. 1 – це 1:1. Якщо текст у вікні модифікований, зліва від номера рядка і стовпця розташована * .

Рядок заголовка вікна містить його найменування. Рядок заголовка вікна редактора – ім'я файлу, розташованого у вікні. Перші 9 відкритих вікон мають номер вікна. Для активізації вікна можна натиснути клавіші Alt + цифру, яка дорівнює номеру необхідного вікна. Активізувати вікно можна

клацанням лівої клавіші миші на будь-якому місці видимої частини вікна.

Закриття вікна, його відкриття на весь екран і згортання до колишнього розміру виконується натисненням клавіші миші, курсор якої встановлений на відповідний маркер. Закрити активне вікно можна за допомогою команди Alt+ F3.

Для відкриття на весь екран або згортання вікна можна використати клавішу F5 або двічі клацнути лівою клавішею миші, встановивши курсор на верхній рамці заголовка вікна.

Для переміщення вікна треба "відбуксирувати" його в потрібне місце, встановивши курсор миші на верхній рамці вікна: натиснути ліву кнопку і "тягнути" вікно, не відпускаючи її. Або виконати команду Ctrl+F5, потім перемістити вікно за допомогою "стрілок" – клавіш керування положенням курсора. Після переміщення вікна треба натиснути клавішу Введення.

Для зміни розміру вікна треба "відбуксирувати" його правий нижній куток з натиснутою лівою кнопкою миші до потрібного розміру і відпустити її. Без миші змінити розмір вікна можна за допомогою команди Ctrl+F5 потім натиснути клавішу Shift і, не відпускаючи її, змінити розмір вікна за допомогою "стрілок". Потім натиснути Введення.

Смуги прокрутки використовуються для прокрутки вмісту вікна за допомогою миші. Вони мають ліфт – маркер положення курсора щодо початку і кінця тексту. Для прокрутки на один рядок (підняття або опускання тексту на екрані) треба підвести курсор до стрілки в нижній або верхній частині смуги прокрутки і клацнути лівою кнопкою миші. Для неперервної прокрутки треба тримати кнопку натиснутою. Для прокрутки на одну сторінку треба встановити курсор миші на смузі прокрутки вище (для прокрутки на сторінку вгору) або нижче за ліфт і клацнути лівою клавішею миші.

З середовища Сі++ можна активізувати "призначений для користувача екран" – вікно з результатами, виведеними в стандартний файл виведення stdout. Для цього використовується команда Alt + F5. Повернення з екрану користувача в екран середовища виконується натисненням будь-якої клавіші але зазвичай це клавіша Esc.

1.5 Допомога в середовищі Borland C++

При роботі в системі у будь-який момент можна одержати довідкову інформацію, контекстно пов'язану з поточною ситуацією, в спеціальному вікні. Натиснення F1 з будь-якого вікна дає виклик підсистеми Help видачею інформації, найбільш відповідної поточному положенню курсора.

Для виклику підказки можна:

1. Натиснути клавішу F1 при виборі будь-якого елемента в блоці діалогу або будь-якого рядка меню;

2. Виконати команду Ctrl+Fl для отримання підказки щодо оператора мови з вікна редактора; при цьому курсор повинен бути під ключовим словом оператора;
3. Ввести команди Ctrl+Fl, Shift+Fl або Alt+Fl з "порожнього" місця для отримання вікна з Turbo Help Index — списком ключових слів, розташованих в алфавітному порядку, за якими можна одержати підказку;
4. Клацнути мишею на ключовому слові Help, коли воно з'являється в рядку стану, в меню або в блоці діалогу (на клавіші).

На деяких екранах допомоги є ключові слова, виділені кольором (підсвічуванням). Вони позначають можливість виклику відповідної додаткової інформації, що деталізує вибране поняття.

Для отримання інформації про помилку, повідомлення про яку одержане під час компіляції або виконання програми, треба натиснути клавішу F1 під час індикації (підсвічування) повідомлення про помилку у вікні Message. Прибрати підказку або повідомлення про помилку можна за допомогою клавіші Esc.

2 Меню системи BORLAND C++

2.1 Система меню, основні відомості про меню

Borland C++ – це компілятор мови Сі з інтегрованим інтерактивним середовищем розробки програм. Терміном "середовище" визначаються сервісні засоби, які не відносяться до мови програмування, а служать для полегшення процесу розробки програм, зокрема налагодження і тестування програм, та підвищують продуктивність праці програмістів. З цією метою в середовищі Borland C++ використовується система меню і ряд вікон: редактора, результатів, повідомлень, допомоги, меню, ряд клавіш і їх поєднань.

Клавіші Alt, Ctrl і Shift в складених командах використовуються таким чином: спочатку треба натиснути на одну з них і потім, не відпускаючи її, натиснути додаткову клавішу, наприклад, функціональну або алфавітну. Таке натиснення двох клавіш позначається з плюсом або пропуском. Наприклад; ALt+F3, Alt+X, Ctrl+F9, Gtrl+Q X. При натисненні клавіші, написаної в складеній команді через пропуск, додаткову клавішу можна відпустити. Наприклад, для виконання команди Ctrl+Q X треба натиснути Ctrl і, не відпускаючи її, натиснути Q, а потім можна відпустити Ctrl і Q і натиснути X.

Меню системи Borland C++ має деревовидну структуру, що складається з основного меню, підменю і їх команд. Командами називають пункти меню, що викликають певні дії системи, а не розвертання чергового підменю.

Після успішного виклику системи Borland C++ верхній рядок екрана

містить вікно основного меню – можливих підменю роботи системи. Перехід в основне меню з будь-якого вікна здійснюється натисненням клавіші F10. До попереднього рівня меню можна перейти за допомогою клавіші Esc. Вибір необхідної команди основного або додаткового меню здійснюється його підсвічуванням шляхом переміщення курсора за допомогою миші або клавіш керування положенням курсора ←,→,↑,↓. Після вибору пункту натиснення клавіші Введення (Enter) викликає розвертання відповідного додаткового меню або виконання його функцій, якщо цей пункт – команда.

Вибір будь-якого пункту головного меню і розгортання меню другого рівня (підменю) після входу в головне меню можна виконувати також натисненням першої букви імені необхідної команди. Наприклад, після входу в головне меню за допомогою F10 натиснення букви P викликає розвертання меню другого рівня опції Project. Аналогічно після входу в меню другого рівня, наприклад, пункту File, натиснення букви S або s дає вибір команди Save. Далі за текстом позначення команди меню складається з найменувань пунктів меню, підменю і(або) команди. Наприклад: Compile/Link, Debug/Watches/Add watch і ін., де Compile, Debug – імена пунктів головного меню; Watches – ім'я підменю; Link, Add watch – імена команд.

При розгорненому підменю головного меню можна перейти до розгорненого підменю сусіднього пункту головного меню за допомогою клавіш горизонтального переміщення курсора ← і →.

Для розвертання меню другого рівня з будь-якого вікна середовища краще використовувати команди із клавіші Alt і першої букви імені опції головного меню. Призначення пунктів головного меню і команди для переходу у відповідне меню другого рівня подано в таблиці 2.

Таблиця 2 – Призначення підменю головного меню

Пункт	Призначення пункту	Команда
=	Системне меню	
File	Операції з файлами, вихід з системи	Alt+F
Edit	Редагування тексту в активному вікні	Alt+E
Search	Пошук фрагментів тексту, місцерозташування помилок	Alt+S
Run	Трансляція, редагування і запуск програми	Alt+R
Compile	Компіляція (трансляція) програми	Alt+C
Debug	Засоби налагодження програм	Alt+D
Project	Керування проектом	Alt+P
Options	Керування параметрами компіляції і компоновки	Alt+O
Window	Керування вікнами	Alt+W
Help	Звертання до системи оперативної підказки	Alt+H

Деякі пункти випадних підменю можуть мати свої підменю. Такі пункти підменю відмічені темним трикутником справа. Наприклад:

Watches ►, Compiler ►, Environment ►.

Якщо в рядку пункту підменю стоять 3 крапки, то при виборі цього пункту на екрані з'явиться діалогове вікно. Наприклад:

Open...F3, Save as..., Change dir..., Find..., Arguments...

Підсистема Help дозволяє одержати інформацію про меню. Найпростіший спосіб одержати таку інформацію – натиснути F1, коли курсор встановлений на необхідний пункт основного або додаткового меню будь-якого рівня. Повернення з Help виконується натисненням клавіші Esc або закриттям його вікна.

При використанні меню і вікон треба застосовувати такі правила:

1. Перехід з будь-якого вікна в меню другого рівня здійснювати за допомогою команд з табл. 2;
2. Вибір команди в підменю виконувати натисненням на клавіатурі букви цієї команди, виділеної кольором;
3. Перехід від однієї команди до сусідньої виконувати за допомогою клавіш управління положенням курсора;
4. Перехід від одного меню другого рівня до сусіднього виконувати за допомогою клавіш < i > ;
5. Вибрана (підсвічена) команда буде виконана, якщо натиснути клавішу Введення (Enter або ↵) або клацнути на ній мишею;
6. Відмова від вибору команди і повернення в меню вищого рівня або активне вікно екрану виконувати клавішею Esc;
7. Перемикати активні вікна за допомогою клавіш Alt+N, де N- номер вікна;
8. Перехід з будь-якого вікна до деяких найважливіших команд меню виконувати, минувши головне і додаткове меню, за допомогою гарячих клавіш – клавіш оперативного керування Borland C++.

Далі при описі меню в дужках подані гарячі клавіші команд.

У Borland C++ використовуються опції (режими) трьох видів: команди, параметри і діалоги.

Команда визначає додаткове меню або необхідну дію середовища. При її виборі після натиснення клавіші Введення (Enter) негайно реалізується передбачена нею дія. Приклади команд: всі пункти головного меню: Run, Trace info, Make і т.д.

Опція – параметр, пов'язаний з введенням числових або текстових параметрів. Ці опції, як правило, входять до складу вікна діалогу. Опція-параметр має на екрані невелике додаткове вікно, в яке можна ввести значення параметра. Наприклад, меню File має команду Save as. При її

виборі на екрані з'являється вікно діалогу, в якому є додаткове вікно: у нього треба ввести ім'я файлу на МД, в який треба помістити текст, розташований у вікні редактора.

Список основних клавіш оперативного керування Borland C++ поданий в таблиці 3.

Таблиця 3 – Основні клавіші оперативного керування Borland C++

Клавіші	Функції клавіш	Еквівалент меню
F1	Викликати довідкову службу	
F2	Запам'ятати редагований файл на МД з тим самим ім'ям	File/Save
F3	Завантажити текст нового файлу в пам'ять редактора	File/Open
F4	Виконати програму до рядка розташування курсора	Run /Go to cursor
F5	Розкрити поточне вікно на весь екран або згорнути його	Window /Zoom
F6	Перейти в наступне вікно	Window/Next
F7	Виконати (порядкове) трасування в програмі і підпрограмах	Run /Trace into
F8	Виконати порядково програму, а підпрограми – як один оператор	Run/Step over
F9	Створити виконувану програму (exe-файл)	Compile/Make
F10	Перейти з будь-якого вікна в головне меню	
Alt+F1	Викликати попередній екран підказки	Help/Previous topic
Alt+F3	Закрити активне вікно	Window/Close
Alt+F5	Показати вікно виведення результатів роботи програми (файлу stdout)	Window /User screen
Alt+F7	Перейти до попереднього повідомлення про помилку етапу компіляції	Search/Previous error
Alt+F8	Перейти до наступного повідомлення про помилку етапу компіляції	Search/Next error

Продовження таблиці 3

Клавіші	Функції клавіш	Еквівалент меню
Alt+F9	Виконати компіляцію програми, сформувати obj-файл	Compile /Compile
Alt+X	Вийти з Borland C++	File/Quit
Ctrl+F1	Викликати контекстну підказку з мови Сі з середовища редактора	Help /Topic search
Ctrl+F2	Скинути налагоджувальні засоби програми (зняти трасування програми)	Run/Program reset
Ctrl+F3	Викликати стек активних функцій: їх імен і списків їх фактичних параметрів	Debug/Call stack
Ctrl+F4	Вивести на екран значення змінних і виразів для контролю і модифікації	Debug/Evaluate/modify
Ctrl+F7	Додати вираз або ім'я змінної у вікно перегляду (Watch)	Debug/Watches/Add watch
Ctrl+F8	Перемкнути (встановити або скинути точку переривання)	Debug/Toggle breckpoint
Ctrl+F9	Запустити програму на компіляцію, компоновку і (або) виконання	Run/Run

Опції-діалоги викликають вікно діалогу. Приклад вікна діалогу – вікно, в якому можна вибрати файл для введення його у вікно редактора (рис. 3). Вікно може містити додаткові вікна для введення, наприклад, імені файлу, вікна із списками для вибору з них необхідного значення. У вікні діалогу можуть бути кнопки:

- для вибору дії (наприклад, OK, Cancel, Help);
- для настроювання системи на певний режим роботи.

Звичайно кнопки настроювання (радіокнопки) об'єднуються в групи за їх призначенням. Поряд з кнопкою стоїть текст, що визначає параметр, який може бути вибраний за допомогою цієї кнопки. Кнопка установлення параметра може бути в одному з двох станів:

- On – дозволений: використовується символ [X] або (•);
- Off – дозволений/заборонений: використовується [ ] або ( ).

Групи кнопок можуть бути двох типів:

– для вибору значень ряду допустимих параметрів;
– для вибору в групі одного зі всіх параметрів.

У групі 1-го типу можна встановити використання декількох параметрів одночасно. У дозволених параметрів в полі установки розміщується символ X в квадратних дужках, у заборонених – пропуск.

Приклад параметрів з групою кнопок для вибору декількох параметрів:

Options

[X] Case-sensitive

[X] Prompt on replace

У групі 2-го типу можна вибрати тільки один параметр з групи. У дозволеного параметра в круглих дужках з'являється крапка – "дозволений".

Приклад такої групи:

Break Make On

( ) Warning

(•) Errors – вибраний цей параметр.

( ) Fatal errors

2.2 Операції з файлами. Меню File

Вигляд додаткового меню (підменю) File поданий на рис. 2. У правій колонці – гарячі клавіші пунктів меню.

Меню File дозволяє керувати файлами і тимчасово або повністю вийти з середовища Borland C++ в середовище MS-DOS.

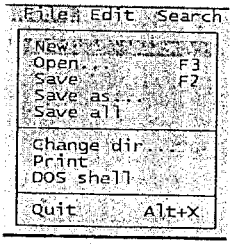


Рисунок 2 – Вигляд додаткового вікна File

Підменю File дозволяє виконувати такі операції з файлами:

1. Створення нового файлу в редакторі (команда New);
2. Завантаження з МД файлу (команда Open) з ім'ям:
   а) заданим в додатковому вікні;
   б) вибраним із списку файлів;
3. Запам'ятовування файлу, зокрема:
   а) з колишнім ім'ям (команда Save);
   б) з новим ім'ям (команда Save as створює нову копію файлу);

4. Зміну поточного каталогу (команда Change dir);
5. Тимчасовий вихід в DOS і повернення в Borland C++ (команда DOS shell); для повернення в систему – команда Exit;
6. Вихід з системи Borland C++ (команда Quit).

Команда File/Open(Файл/Відкрити-F3) викликає вікно діалогу із заголовком Open a file для вибору файлу і введення його у вікно редактора. Приклад вікна для завантаження файлу поданий на рис. 3. Вікно містить:

– вікно-рядок для введення імені файлу або шаблону для виведення списку
– файлів у вікно вибору імені із списку;
– вікно із списком директорій і імен файлів відкритого каталогу;
– кнопки: Open (Відкрити), Replace (Замінити), Cancel (Відмінити) і Help (Підказка);
– інформаційну панель, яка описує вибраний файл.

За допомогою вікна діалогу можна виконати одну з таких дій:

– ввести повне ім'я файлу і вибрати кнопки Replace (Замінити) або Open (Відкрити) файл;
– ввести шаблон (наприклад, для вибору імені файлу із списку);
– вибрати стрілку вниз для вибору шаблону або імені файлу з «передісторії», тобто з раніше введених шаблонів або імен;
– вибрати і проглянути директорії і файли із списку.

Інформаційна панель – це 2 нижні рядки вікна Open a File з інформацією про файл; вони містять шлях до файлу, ім'я файлу, дату і час створення, розмір вибраного файлу.

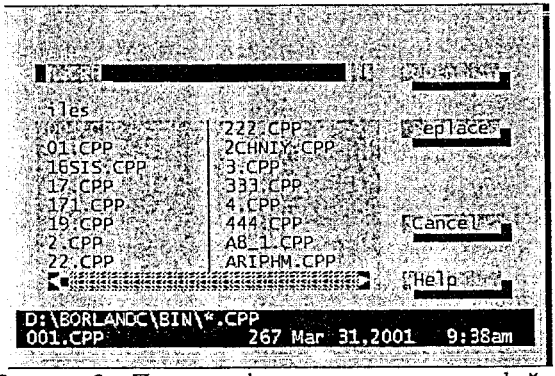


Рисунок 3 – Приклад вікна для завантаження файлу

Команда New очищає пам'ять редактора і переводить його в режим створення файлу. Новостворюваному файлу присвоюється ім'я NONAME.CPP. Його можна змінити при записі файлу на МД.

Команда Save (F2) зберігає файл з активного вікна редактора на МД. Якщо до моменту запису файлу ім'я файлу було NONAME.CPP, редактор запропонує перейменувати файл. Якщо на МД вже знаходиться однойменний файл, він буде заздалегідь перейменований у файл з тим самим ім'ям і розширенням .bak.

Команда Save as записує редагований файл на МД з новим ім'ям, тобто створює копію цього файлу з новим ім'ям. При виборі цієї команди на екрані з'являється додаткове невелике вікно; у цьому вікні потрібно записати ім'я файлу, з яким файл буде поміщений на МД. Потім натиснути кнопку ОК. За замовчуванням дається розширення імені файлу .cpp.

Команда Save All зберігає на МД модифіковані файли зі всіх вікон, а не тільки з активного вікна.

Команда Change dir дозволяє встановити поточний каталог. У цей каталог записуватимуться формовані об'єктні і виконувані файли програми (якщо немає установки директорії для цих файлів в меню Options). Поточний каталог використовується системою також для показу списку файлів за командою F3.

При виборі команди Change dir відкривається діалогове вікно. З нього можна змінити каталог одним з двох способів:

– ввести ім'я необхідної директорії і натиснути клавішу Введення;

– вибрати МД (Drives), в ньому необхідну директорію і натиснути ОК.

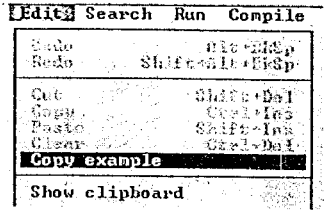
Команда Print дозволяє вивести на друк вміст активного вікна редактора, результатів роботи користувача (Output) або вікна повідомлень про помилки (Message). Команда DOS shell дозволяє тимчасово вийти з системи Borland C++ без вивантаження його з ОП. Для повернення з DOS в систему треба ввести команду EXIT.

Команда Quit (Alt + X) здійснює вихід з системи Borland C++. Якщо перед виходом редагований файл не був збережений на МД, система видасть запит: Чи зберегти файл?

2.3 Редагування файлів. Меню Edit

Вигляд додаткового меню Edit поданий на рис.4.

За допомогою меню Edit можна виконувати різні види робіт над блоками тексту: вирізати або копіювати блок в текстовий буфер (clipboard), вставляти його з буфера, стирати, викликати текстовий буфер у вікно редактора.



Блок – це прямокутний фрагмент тексту, виділений кольором. Для виділення блоку можна використовувати клавішу Shift і клавіші керування положенням курсора.

Рисунок 4 – Вигляд додаткового меню Edit

Це можна зробити і за допомогою миші: натиснути ліву клавішу і, не відпускаючи її, виділити необхідну ділянку тексту.

Команда Undo (Відновити) відміняє дію останньої команди редагування екрана. Команда Redo повторює дію останньої команди редагування.

Команда Cut (Вирізати) видаляє блок і заносить його в текстовий буфер (clipboard). Потім цей текст можна вставити в інший файл або в інше місце цього ж файлу. Для цього використовується команда Paste (Вставити). У текстовому буфері текст залишається виділеним і його можна вставляти багато разів.

Команда Copy (Копіювати) копіює блок в текстовий буфер. Після копіювання цей текст можна вставити в необхідне місце цього ж або іншого файлу. Скопіювати можна також текст з вікна Help.

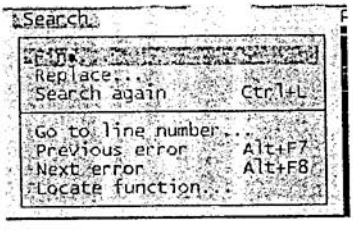
Команда Clear (Стерти) видаляє блок, не заносячи його в буфер.

Команда Copy Example (Копіювати приклад) копіює виділений текст з вікна підказки в текстовий буфер (можна і Ctrl+Ins).

Команда Show clipboard (Показати вміст текстового буфера) відкриває вікно Clipboard, в якому зберігаються фрагменти тексту, скопійовані в буфер.

Для вставки система використовує текст, виділений в поточний момент. Вікно з текстовим буфером аналогічне будь-якому іншому вікну текстового редактора. У ньому можна виділити текст звичайним способом, після чого використовувати його в команді Shift + Ins для вставки в необхідне місце активного вікна редактора.

2.4 Пошук і заміна тексту. Меню Search



Меню Search дозволяє виконувати пошук і заміну тексту новим, пошук місцерозташування функцій і помилок і тексті програми, знайдених під час компіляції. Вид меню Search поданий на рис. 5.

Рисунок 5 – Вигляд команди Search

Команда Find (Знайти) викликає вікно діалогу Find Text. У ньому можна задати текст для пошуку і параметри пошуку. Ця команда може бути викликана клавішами Ctrl+Q F. Вікно діалогу містить 3 стандартні кнопки і 4 групи параметрів: Options, Direction, Scope і Origin. Приклад вікна для пошуку тексту Text поданий на рис. 6.

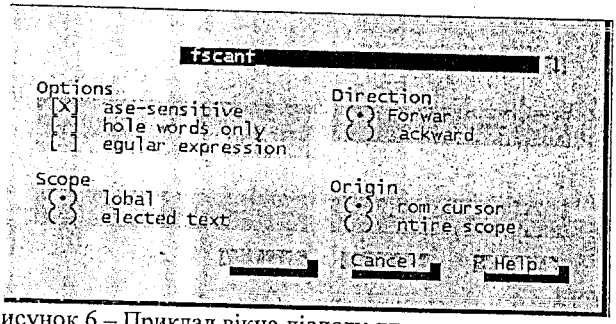


Рисунок 6 – Приклад вікна діалогу для пошуку тексту

Група параметрів Options дозволяє вибрати набір з параметрів:

- Case sensitive – Чутливість до регістра; його треба встановити, щоб система розрізняла при пошуку символи верхнього і нижнього регістрів;
- Whole words only – Тільки цілі слова; встановлюється для пошуку тільки цілих слів, обмежених символами пунктуації або пропусками;
- Regular expression – Регулярні вирази для утиліти GREP.

Direction (Напрям) визначає напрям пошуку Forward (Вперед) або Backward (Назад) щодо точки, в якій розташований курсор перед початком пошуку.

Scope – (Область пошуку) визначає один з параметрів;

- Global – глобальний пошуку по всьому тексту файлу;
- Selected text – пошук у виділеному тексті (у блоці).

Origin (Початок пошуку) визначає один з параметрів:

- From cursor – пошук від курсора;
- Entire scope – вся область пошуку.

Для того, щоб почати пошук, треба ввести рядок для пошуку у вікно введення тексту. При виклику команди пошуку у вікно введення заноситься слово, під яким встановлений курсор. За допомогою клавіші "стрілка вниз" можна викликати список передісторії, з якої можна ввести текст для пошуку.

Команда Replace (Заміна) викликає вікно діалогу. З його допомогою можна ввести шуканий текст і текст, яким треба його замінити.

Приклад вікна діалогу поданий на рис.7.

Порівнянно з командою Find в групі Options додатковий параметр Prompt to replace визначає необхідність видачі запиту на кожну заміну тексту. При виборі кнопок ОК або Change All (Замінити все) відбувається пошук або заміна тексту відповідно до заданих параметрів.

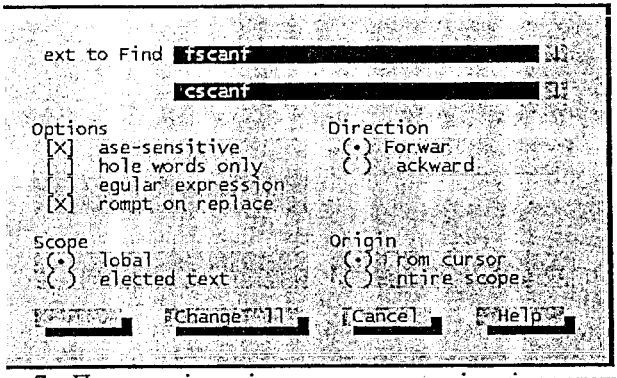


Рисунок 7 – Приклад вікна діалогу для пошуку і заміни тексту

Команда Search again повторює дію останньої команди Find або Replace. Команда Goto line number запитує номер рядка, до якого треба перейти.

Команди Previous error (Попередня помилка) і Next error (Наступна помилка) переміщають точку-мітку рядка у вікні повідомлень (Message) і курсор у вікні редактора до оператора, що викликав попереднє або наступне повідомлення про помилку на етапі компіляції. Ці команди доступні, якщо у вікні Message є повідомлення про помилки.

Команда Locate function (Місцерозташування функції) викликає діалогове вікно, в яке треба ввести ім'я необхідної функції. Приклад вікна для пошуку місцерозташування функції поданий на рис. 8. За допомогою клавіші "стрілка вниз" можна вибрати це ім'я з "передісторії". Команда доступна тільки під час налагодження, наприклад, під час припинення виконання програми після досягнення контрольної точки. Вона шукає заголовок функції, а не її оголошення або виклик. Команда корисна при розробці складних багатофайлових програм. Після набору імені необхідної функції і натиснення кнопки OK або клавіші Введення в редактор завантажується файл (для багатофайлової програми), що містить текст визначення заданої функції, відбувається перехід у вікно редагування цього файлу, причому курсор встановлюється на перший рядок заголовка функції.

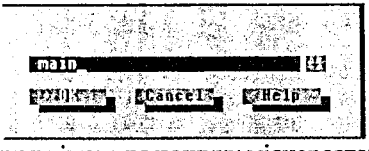


Рисунок 8 – Приклад вікна для пошуку місцерозташування функції.

У програмах, написаних мовою Сі, є значна кількість блоків функцій і складених операторів, взятих у фігурні дужки. Крім того, програмний файл може містити багато парних символів-роздільників, що є обмежувачами:

- {} – програмних блоків;
- [ ] – списків індексних виразів;
- ( ) – параметрів функцій;
- < > – імен бібліотек;
- /* */ – коментарів;
- " – рядкових констант;
- ' – символів.

Ручний пошук парного символу до заданого може бути непростим:

- для пошуку парного символу засобами системи треба помістити курсор на необхідний роздільник, наприклад {;
- для пошуку "вперед" пари для цього роздільника треба ввести команду Ctrl+Q [; за цією командою редактор переміщає курсор вперед по тексту до роздільника, парного щодо вказаного курсором; наприклад, до роздільника } ;
- для пошуку "назад" треба встановити курсор на закриваючу дужку, наприклад, на }; потім ввести команду Ctrl+Q [; редактор переміщає курсор назад по тексту до парного роздільника, наприклад, до {.

Якщо для парного роздільника не знайдений парний символ, редактор не переміщає курсор.

3 Компіляція, компоновка і виконання програми. Меню Run

Вигляд меню Run (Виконання) поданий на рис.9. Меню містить команди, для керування ходом виконання програми.

Меню Run дозволяє виконувати такі види обробки програми:

1. Компіляцію, компоновку і виконання програми (Run);
2. Виконання (трасування) програми під керуванням вбудованого налагоджувача в одному з налагоджувальних режимів;
3. Задання аргументів командного рядка.

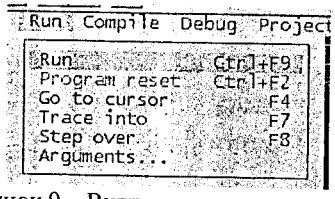


Рисунок 9 – Вигляд екрана з меню Run

Якщо програма до запуску не готова, тобто не відкомпільована і не скомпонована, команда Run (Виконання, запуск) виконує компіляцію одного або декількох файлів початкової програми, компоновку (редагування зв'язків) і виконання програми, якщо при компіляції і компоновці не було помилок.

Якщо програма вже відкомпільована, то Run запустить програму на виконання. При запуску програми на виконання використовуються аргументи, задані за допомогою команди Arguments меню Run. Перервати компіляцію, редагування або виконання програми (наприклад, що зациклилася) можна за допомогою команди Ctrl+Break.

Після нормального завершення програми вікно редактора відновлюється в тому вигляді, в якому воно було до моменту звертання до команди Run. Якщо на будь-якому з етапів обробки програми знайдена помилка, то після закінчення процесу трансляції або компоновки активізується вікно повідомлень і підсвічується перший рядок з повідомленням про помилку. Підказку щодо неї можна одержати натисненням F1.

Після виникнення першої помилки на етапі виконання програми виконання програми припиняється, керування передається текстовому редактору і курсор встановлюється під оператор програми, який викликав помилку, а у вікні stdout (екран користувача) з'являється повідомлення про помилку. Наприклад: Divide error.

Трасування програми – це поетапне виконання програми із зупинками для перегляду і аналізування проміжних результатів її виконання. Для трасування використовують виконання програми: порядкове (F7), виконання функції за одне натиснення клавіші (F8), від попереднього останову або від початку програми до курсора (F4), до чергової контрольної точки (точки переривання).

Якщо система працює в режимі налагодження і в програмі розставлені точки переривання, то виконання програми припиниться перед першим рядком, визначеним як точка переривання. Після припинення можна візуально проконтролювати поточні значення змінних, додати або виключити точки переривання і т.д. Продовжити виконання програми можна за допомогою повторного виклику пункту Run або перейти на покрокове виконання програми. Якщо треба проглянути вікно з результатами роботи програми у файлі stdout (на екрані), треба виконати команди Alt+F5 або User screen меню Window. Повернення з вікна результатів у вікно редактора виконується натисненням будь-якої клавіші, наприклад, клавіші Esc.

Команда Program reset (Скидання програми) – повернення програми до початкового стану завершує сеанс налагодження. При цьому закриваються всі файли, відкриті в ній до цього моменту. Після цього можна коректувати

текст програми і знову виконати її.

Команда Goto cursor (Виконати до курсора) виконує програму від початкового або поточного рядка зупинки виконання до того рядка, в якому розташований курсор. Якщо курсор знаходиться в рядку, який не містить виконуваного оператора, видається попередження:

No code generated for this line

Команду F4 використовують для виконання програми по частинах, до певної точки. Але якщо треба, щоб програма зупинялася на заданому операторі кожного разу, коли вона досягає цього оператора, на ньому треба встановити контрольну точку (точку переривання). Точка переривання встановлюється і знімається командою Ctrl+F8.

Команда Trace into (Трасування до) виконує покрокове виконання програми: головної і всіх викликаних функцій. Від одного виклику команди виконуються оператори поточного рядка або один оператор, розташований в декількох рядках, після чого виконання програми припиняється. Потім можна знову виконати команду Trace into і т.д.

Команда Step over (Зробити крок поверх). Працює, як і попередня, але викликані функції виконуються за одноразове виконання команди.

Команда Arguments (Аргументи) дозволяє сформувати текстовий рядок, що містить параметри виконуваної програми, що запускається з Borland C++ так, ніби вони були задані в командному рядку при виклику програми з MS-DOS. При виклику команди на екрані з'являється додаткове вікно для введення рядка, який формується за тими ж правилами запуску з середовища MS-DOS, тобто аргументи розділяються пропусками. Функція main програми отримує доступ до переданих аргументів. Приклад вікна Program Arguments для введення аргументів командного рядка поданий на рис. 10.

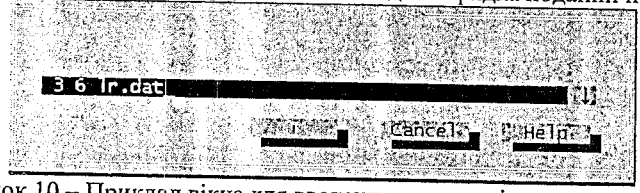


Рисунок 10 – Приклад вікна для введення аргументів командного рядка

3.1 Компіляція програми. Меню Compile

Вигляд меню Compile (Компіляція) поданий на рис. 11.

Після створення програмного файлу процес обробки програми включає компіляцію, компоновку і виконання програми.

Меню Compile дозволяє реалізувати такі види обробки програми:

- компіляцію програми і створення об'єктного файлу (з розширенням .obj);

- компоновку програми і створення виконуваної програми (файлу з розширенням .exe);
- створення програм з декількох початкових програмних файлів;
- отримання докладної інформації про програму;
- видалення всіх повідомлень з Message – вікна повідомлень.

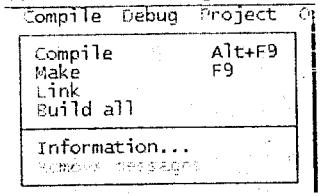


Рисунок 11 – Вигляд меню Compile

Щоб використовувати команди Compile, Make, Build і Link, повинна бути виконана одна з умов:

- у активному вікні редактора повинен бути файл програми або
- у вікні проекту – список файлів програми проекту.

Але якщо активізувати вікна Watch або Message, ці команди будуть заборонені: у відкритому меню команди будуть подані світло-сірим кольором.

Команда Compile(Компіляція) компілює програму: з файлу у вікні редактора типу .C або .CPP формується об'єктний модуль, файл типу .OBJ. Основна частина імені об'єктного модуля, який буде одержаний в результаті трансляції, формується з імені цього файлу.

Під час виконання компіляції на екран видається вікно, в якому відображається інформація про процес компіляції. Після закінчення компіляції у вікні подані результати компіляції. Приклад такого вікна на рис.12. Для видалення цього вікна можна натиснути будь-яку клавішу.

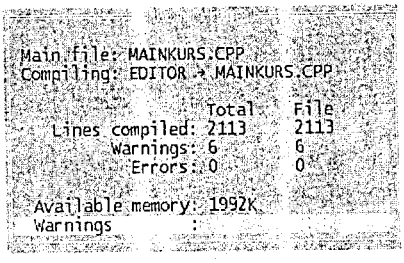


Рисунок 12 – Вигляд вікна з повідомленням про результати компіляції

Якщо на етапі компіляції і(або) компоновки знайдені сумнівні конструкції

мови або помилки, в нижньому рядку вікна замість слова Success (Успіх) розташуються відповідно слова Warnings (Попередження) або Errors (Помилки).

При цьому формується вікно повідомлень (Message) про попередження і(або) про помилки. Переглядання повідомлень з вікна редактора можна виконувати за допомогою команд Alt+F7 (Previous error) і Alt+F8 (Next error).

Приклад вікна з повідомленнями поданий на рис. 13.

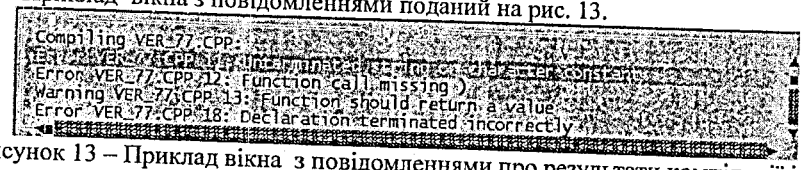


Рисунок 13 – Приклад вікна з повідомленнями про результати компіляції і компоновки

Відразу після закінчення компіляції і(або) компоновки у вікні повідомлень виділяється підсвічуванням перше повідомлення. У вікні з файлом програми підсвічується рядок, в якому знайдено помилку. Перехід від одного повідомлення про результати компіляції до іншого переводить курсор у вікні з файлом програми на рядок, до якого відноситься повідомлення.

Команда Make (Створити exe-файл) створює файл типу .exe. Для цього вона викликає компіляцію, потім компоновку програми за допомогою редактора зв'язків (linker). При виклику команди Make компілюється програма, яка може містити файли, що включаються за допомогою директиви #include. Якщо в процесі компіляції зустрілося звертання до функцій підключуваних файлів, середовище перевіряє, чи були зроблені у файлі з початковим текстом цієї функції які-небудь зміни з моменту його останньої компіляції; якщо зміни були, виконується компіляція файлу цієї функції і формування його obj-файлу. Проте, якщо файл функції не буде знайдений, система використовує існуючий obj-файл без контролю його змін. Підсумком виконання цієї команди є виконувана програма.

Ім'я файлу, в який поміщається виконувана програма, завжди має розширення .exe. Основна частина імені exe-файлу формується таким чином:

- якщо компілюється програма, завантажена в редактор, основна частина імені exe-файла збігається з основною частиною імені файлу, в якому знаходиться функція main;
- якщо за допомогою команди Open Project меню Project задане ім'я проекту, то основна частина імені exe-файлу збігається з основною частиною імені файлу проекту.

Режим роботи команди Make встановлюється за допомогою параметрів команди Options/Make. За цією командою в діалоговому вікні виводяться групи параметрів. Наприклад:

- Break Make On – для визначення умови переривання компіляції;
- After Compiling – для визначення дій після успішної компіляції.

Команда Make контролює відповідність об'єктних модулів їх початковим текстам, а саме, перекомпілює тільки ті підключувані файли, в які внесені зміни після їх останньої компіляції. У такий спосіб команда Make спрощує розробку багатофайлових програм.

Команда Link (компоновка і створення exe-файлу) створює з об'єктних модулів (файлів типу .obj) виконувану програму (файл типу .exe) і записує її у файл, ім'я якого визначається за тими ж правилами, що і для команди Make. Відмінність полягає у тому, що при виконанні команди Link завжди використовуються поточні версії obj-файлів – об'єктних модулів. Команду Link треба виконувати після команди Compile. Приклад вікна з повідомленнями про результати компоновки поданий на рис.14.

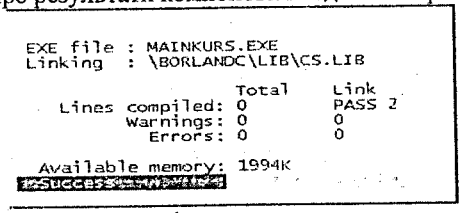


Рисунок 14 – Приклад вікна з повідомленнями про результати компоновки

Команда Build all (Створити все) подібна команді Make. За виключенням: з її допомогою виконується пошук і перекомпіляція всіх початкових файлів незалежно від того, чи були в них зміни.

Команда Information видає вікно діалогу з інформацією про поточний стан системи. Приклад вікна поданий на рис. 15.

Команда Remove messages видаляє всі повідомлення з вікна Message.

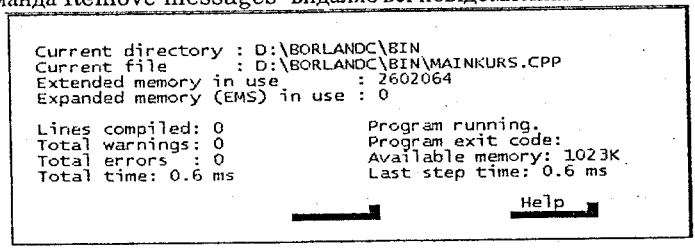


Рисунок 15 – Приклад вікна з інформацією про поточний стан системи

3.2 Засоби налагодження програм. Меню Debug

На рис. 16 показане меню Debug (Налагодження).

Меню Debug дозволяє використовувати такі засоби налагодження:

- перегляд і зміна контрольованих значень;
- перегляд значень фактичних параметрів викликаних функцій;
- робота з точками переривання.

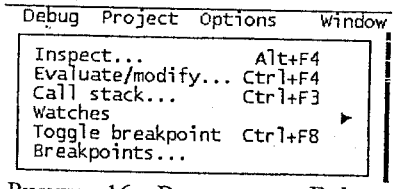


Рисунок 16 – Вигляд меню Debug

Щоб всі можливості системи налагодження були доступні, треба встановити параметр Options/Debugger/Sdurce Debugging (Налагодження на рівні початкового коду) в стан On. Приклад вікна для установки параметрів налагоджувача поданий на рис. 17.

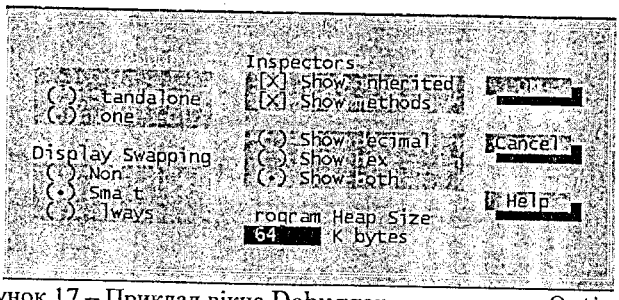


Рисунок 17 – Приклад вікна Debugger пункту меню Options

Команда Inspect відкриває вікно для переглядання вмісту об'єктів.

3.3 Обчислення виразів в процесі налагодження програм. Команда Evaluate/modify

Команда Evaluate/modify (Обчислити/модифікувати) обчислює значення виразу в процесі налагодження, відображає результат обчислення і дозволяє при необхідності модифікувати значення змінних. Команду Evaluate/modify можна викликати під час припинення програми в процесі її виконання, наприклад, за командами F4 або F7. За цією командою на екрані розвертається додаткове вікно (рис. 18), що містить 3 поля і 4 кнопки. Поля: Expression (Вираз), Result (Результат) і New Value (Нове значення).

Кнопки: Evaluate (Обчислити), Modify (Модифікувати), Cancel (Відмінити) і Help (Допомога). Поля Expression і New Value мають "передісторію". Відразу після виклику команди курсор знаходиться в першому полі.

При цьому середовище аналізує найближче оточення курсора і, якщо це можливо, виділяє ідентифікатор або константу, на яку перед звертанням до команди указував курсор у вікні редактора. Виділений ідентифікатор або константа переносяться в поле Evaluate і пропонуються у вигляді обчислюваного виразу. Треба натиснути клавішу Введення, щоб негайно отримати його поточне значення в полі Result. Якщо значення змінної, заданої ідентифікатором, необхідно змінити, треба перейти у вікно New Value і зробити це так само, як і в режимі редагування. Після чого треба натиснути кнопку Modify або Введення. У полі Result з'явиться нове значення змінної. Цього значення набуде змінна програми. Модифікувати можна тільки значення змінних.

Для отримання у вікні значення будь-якої локальної змінної функції у будь-якому місці програми треба вказати ім'я функції, якій належить змінна у форматі: ім'я-функції.ідентифікатор.

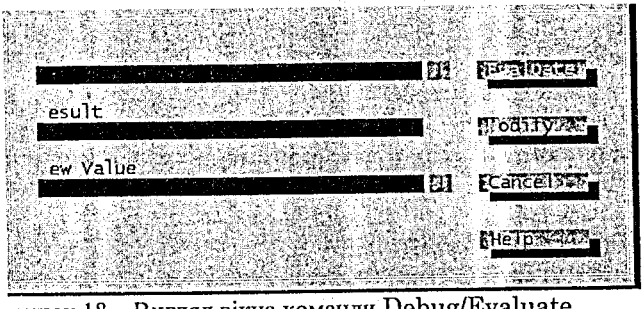


Рисунок 18 – Вигляд вікна команди Debug/Evaluate

Копіювати вираз у вікно Expression можна таким чином: встановити курсор під перший символ виразів, виконати команду Ctrl+F4, натиснути клавішу → – розширення тексту; ідентифікатор (константа), що з'явився у вікні, доповниться символом, розташованим праворуч від нього в тексті програми. Кожне нове натиснення клавіші переведення курсора праворуч приводитиме до копіювання чергового символу з тексту програми в полі Expression. Це полегшує введення довгих виразів і складених ідентифікаторів.

У вікно Expression можна ввести з клавіатури ім'я будь-якої змінної програми або будь-який допустимий вираз, який треба обчислити. Воно не

може містити виклики функцій і ідентифікатори, визначені за допомогою директиви #define. Після натиснення клавіші Введення в полі Result з'явиться відповідне значення результату або діагностичне повідомлення одного з двох типів.

Перший тип повідомлень (Expression syntax) видається у разі синтаксичної помилки виразу. Другий тип відноситься до випадку, коли вираз формально побудований правильно, але його неможливо обчислити через невизначеність (відсутність конкретних значень) деяких змінних. В цьому випадку у вікні Result перераховуються такі змінні.

У разі успішного обчислення значення виразу у вікні Result з'являється його результат. Залежно від типу результату він подається в одній з форм: ціле – у вигляді десяткового числа; дійсне – у форматі %f функції printf; покажчик на char (char *) – у вигляді рядка, тобто послідовності символів ASCII, структурне значення – поелементне.

Після отримання результату курсор залишається у верхньому вікні і можна набрати та ввести для обчислень новий вираз, а у разі імен змінних виду x, *x, x[i] можна змінити значення змінної. Для цього треба перевести курсор у вікно New Value, набрати в ньому необхідне значення і натиснути Введення. Курсор автоматично перейде у верхнє вікно, а у вікні Result з'явиться результат (значення змінної). У разі помилки у вікні Result видається відповідна діагностика.

При роботі з командою Evaluate/modify в нижньому рядку екрана висвічується підказка з двох елементів:

- Fl Help;
- підказка на активний елемент вікна Evaluate and Modify.

Значення підказки наведені в табл. 4.

Для виходу з пункту Evaluate треба натиснути клавішу Esc, кнопку Cancel або закрити це вікно.

3.4 Перегляд стека активних функцій. Команда Call stack

При роботі програми, коли при виконанні однієї функції викликається інша, з якої, у свою чергу, викликається третя і т.д., весь динамічний ланцюжок викликів (функції і їх фактичні параметри), аж до функції, яка виконується в даний момент, зберігається в стеку викликів.

У будь-якій точці, в якій виконання програми припинене, можна проглянути стек викликів функцій за допомогою команди Call stack (Стек викликів): їх імена і списки фактичних параметрів. Приклад вікна поданий на рис. 19.

Таблиця 4 – Підказки в нижньому рядку екрана

Активні елементи	Підказка	Її значення
ВІКНА:		
Expression	Enter expression to evaluate	Введення виразу для обчислення
Result	Exampile result of expresson	Отримання результатів виразу
New Value	Enter new value for expreession	Введення нового значення для виразу
КНОПКИ:		
Evaluate	Evaluate expression	Обчислення виразу
Modify	Modify expression	Модифікація виразу
Cancel	Close the dialog box	Закриття діалогового блоку
Help	View a Help screen abort this dialog box	Показ підказки про цей діалоговий блок

У самому низу вікна (у основи стека) розміщена функція main, яка була викликана першою, а у верхній частині вікна – функція qwerty(), яка викликана останньою і виконується зараз. Кожен елемент стека містить ім'я викликаної функції і значення її фактичних параметрів, для яких вона була викликана; завершити переглядання стека можна за допомогою клавіші Esc, кнопки Cancel або закриттям вікна.

При звертанні до стека викликів виділений кольором останній (верхній) елемент стека. Можна встановити курсор у вікні стека на необхідне ім'я функції і натиснути Введення. При цьому в редактор буде завантажений файл, що містить текст цієї функції, і відбудеться автоматичний перехід у вікно редактора, причому курсор буде встановлений на рядок тексту цієї функції, оператори якої виконувались в ній останніми, тобто в рядок, який містить виклик функції, розташованої над даним рядком стека.

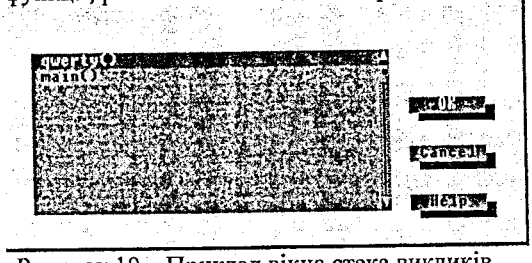


Рисунок 19 – Приклад вікна стека викликів

3.5 Вікно перегляду. Команда Watches, додання, видалення і редагування спостережуваних виразів

Вікно перегляду (Watch) використовується для відображення поточних значень заданих виразів, зокрема змінних, при трасуванні програми і процесі її налагодження. У вікно перегляду виводяться вирази і їх значення, якщо у відповідній точці програми визначені значення змінних, що входять в цей вираз. У одному рядку вікна перегляду виводиться повідомлення про значення одного виразу або змінної: скаляра, масиву, структури, масиву структур або файлу. Приклад вікна Watch поданий на рис. 20.

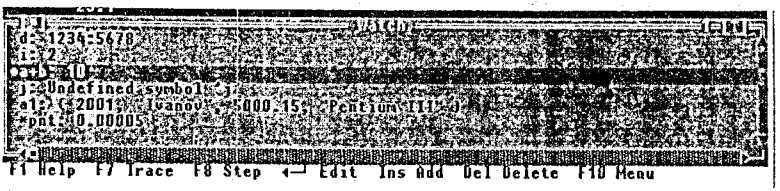


Рисунок 20 – Приклад вікна Watch

Поточний вираз у вікні перегляду помічений точкою зліва. У прикладі на рис. 20 точкою помічений нижній, порожній рядок.

Вирази задаються і їх значення відображаються за тими ж правилами, що і для команди Evaluate/modify підменю Debug.

Команда Watches (Спостереження виразів) меню Debug відкриває підменю з чотирьох команд. Вигляд підменю поданий на рис. 21.

Підменю Debug/Watches має команди для таких видів роботи:

- Add watch – додати вираз у вікно перегляду;
- Delete watch – видалити помічений вираз з вікна перегляду;
- Edit watch – редагувати помічений вираз у вікні перегляду;
- Remove all watches – видалити всі змінні і вирази з вікна перегляду.

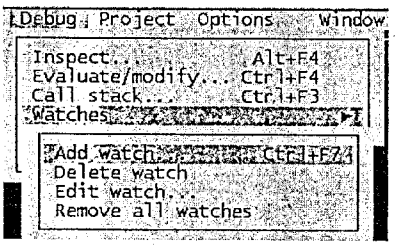


Рисунок 21 – Підменю команди Watches

Для переходу у вікно перегляду (Watch) з іншого вікна використовується команда Alt + номер вікна або клацання мишею в будь-якій видимій частині

вікна Watch.

Коли активне вікно перегляду, рядок оперативної підказки має вигляд:

F1 Help F7 Trace F8 Step ↵ Edit Ins Add Del Delete F10 Menu

Призначення функціональних клавіш оперативної підказки:

– Fl, F7, F8 і F10 мають те ж призначення, що і для вікна редактора та підменю Run (див. табл. 3);

– Ins – викликає діалоговий блок для введення (Add) виразу або імені змінної, значення якої треба контролювати;

– Del – видаляє (Delete) з вікна перегляду рядок, що підсвічується, з виразом або з ім'ям контрольованої змінної;

– ↵ виклик виразу, розташованого в поточному рядку вікна перегляду.

Якщо Введення (↵) або Ins натиснуті при підсвічуванні рядка з виразом, то в діалоговий блок викликається вираз для його редагування (від Введення) або формування нового виразу, що підсвічується (від Ins). Якщо Введення або Ins натиснуті при підсвічуванні порожнього рядка у вікні перегляду, то на екрані з'являється порожній діалоговий блок для введення нового виразу у вікно перегляду.

Натиснення F1 у вікні перегляду викликає інформацію про вікно перегляду, а повторне – викликає вікно Help про підказки системи.

3.6 Додання простежуваних виразів

Команда Add Watch дозволяє додати у вікно перегляду ще один вираз. За цією командою формується діалоговий блок з рядком для редагування і трьома кнопками: OK, Cancel і Help. Введення виразів у вікно перегляду аналогічне введенню виразів у вікно за командою Evaluate/modify.

При одноразовому виконанні команди Add Watch у вікно Watch можна ввести тільки одне ім'я або один вираз. Послідовно викликаючи цю команду, можна задати з її допомогою ряд імен змінних або ряд виразів, які необхідно контролювати в процесі налагодження.

У активному вікні Watch допускається прокручування даних, тобто почергове підсвічування рядків за допомогою клавіш керування положенням курсора ↑ або ↓ – за допомогою смуг прокручування. Вікно перегляду можна розкрити на весь екран і повернути в початковий стан командою F5.

Якщо вираз у вікні Watch має помилку, то в рядку цього виразу видається текст Expression syntax. При правильно введеному виразі негайно виконується спроба обчислити вираз. Якщо до моменту введення були визначені значення всіх змінних, що входять у вираз, він буде обчислений і результат обчислення буде відображений в рядку виразу. Якщо ввести вираз до початку виконання програми, то жодна із змінних, що входять у вираз, не буде визначена і для першої змінної кожного рядка з

виразом буде видана відповідна діагностика. Наприклад:

b: 123

a: Undefined symbol 'a'

a + b: Undefined symbol 'a'

Для керування виведенням значень у вікно перегляду можна використовувати специфікатори формату. Наприклад, для виведення значень елементів масивів і структур. Для виведення значень всіх елементів масиву у вигляді списку значень треба у вікні вказати тільки ім'я масиву. Але якщо масив великий, можна вказати для простежування частину масиву у вигляді:

ім'я-масиву[n], m

де n – індекс першого елемента масиву, який треба бачити у вікні;

m – лічильник повторень, кількість елементів масиву, починаючи з елемента з індексом n, значення яких треба бачити у вікні.

Наприклад, для переглядання п'яти елементів масиву a, починаючи з 10-го елемента, треба за допомогою Ctrl + F7 задати a[10], 5.

Приклад показу значень елементів масиву:

a[10], 5: 12, 23, 34, 45, 56

Для переглядання елементів i-го рядка двовимірного масиву b треба вказати адресу рядка масиву. Наприклад: b[i].

При перегляданні масиву і рядка масиву (формального параметра) у вікно перегляду виводиться адреса масиву або рядка масиву.

Переглядання елементів структури можна встановити:

- без показу імен елементів структури: у вигляді списку значень її елементів; для цього треба ввести у вікно тільки ім'я масиву структур або структури;
- з показом імен елементів структури і їх значень; для цього треба ввести ім'я структури, R (R – від Record); наприклад:

  st, R або st, r

Приклад виведення у вікно переглядання масиву структур і структури без імен і з іменами елементів структур поданий на рис. 22.

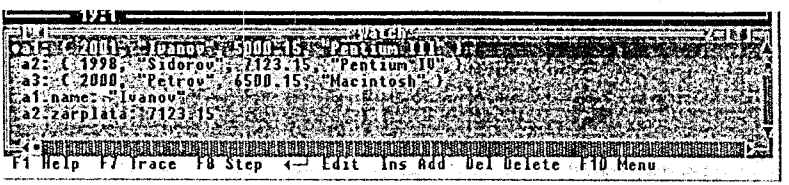


Рисунок 22 – Приклад виведення структур у вікно перегляду

3.7 Редагування виразу у вікні перегляду

За допомогою команди Debug/Watches/Edit watch (Редагувати вираз перегляду) можна викликати для редагування поточний (підсвічений або помічений точкою) спостережуваний вираз.

Приклад вікна редагування поданий на рис. 23.

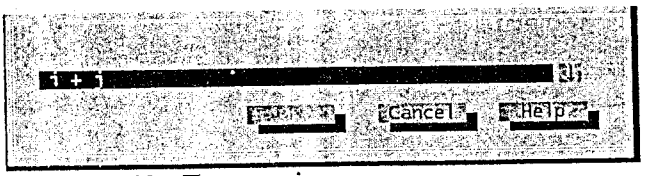


Рисунок 23 – Приклад вікна редагування

За допомогою команди Debug/Watches/Delete watch можна видалити поточну змінну або вираз з вікна перегляду. Якщо активне вікно перегляду, то поточним є вираз, виділений світлим прямокутником і крапкою зліва – покажчиком на початок рядка. Команда стирає рядок з тією змінною (виразом), на яку показує в даний момент покажчик.

Найпростіший спосіб стерти вираз (змінну) з активного вікна перегляду – встановити на рядок з цим виразом покажчик (підсвічування) і натиснути клавішу Del.

Команда Remove all watches видаляє всі змінні і вирази з вікна перегляду.

3.8 Робота з точками переривання

Точки переривання (контрольні точки) – це точки програми, при досягненні яких в процесі виконання програми відбувається зупинка її виконання. Після зупинки можна переглянути значення контрольованих виразів і змінних у вікні перегляду, скоректувати вирази або склад змінних вікна перегляду. Кількість точок переривання в програмі не обмежено. Контрольні точки можна встановлювати і видаляти як до виконання, так і під час зупинки програми.

За командою Debug/Breakpoints (Точки зупинки) відкривається діалоговий блок з сімома кнопками. У ньому показані всі точки зупинки у вигляді таблиці з такими параметрами:

- список точок зупинки (Breakpoint List);
- номер рядка файлу (Line#), в якому встановлена точка зупинки;
- умова зупинки (Condition), необов'язково;
- кількість проходжень програми через цю точку (Pass), після якої треба виконувати зупинку програми.

Приклад діалогового вікна з точками переривання поданий на рис. 24.

При виборі (підсвічуванні) кнопок керування точками переривання в нижньому рядку видається для них підказка. У табл. 5 подані тексти підказок і призначення кнопок. Призначення кнопок Cancel і Help ідентичне призначенню таких самих кнопок в табл. 4.

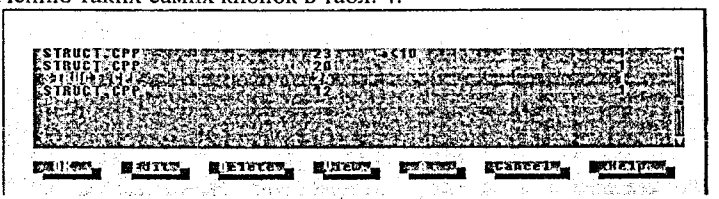


Рисунок 24 – Вид вікна з точками переривання

Таблиця 5 – Підказка в нижньому рядку екрана

Активна кнопка	Підказка	Її значення
OK	Accert the settings inthis dialog box	Прийняти установки цього діалогового блоку
Edit	Edit the currently selected breackpoint	Редагування параметрів поточної вибраної точки
Delete	Delete the currently selected breackpoint	Видалення поточної вибраної точки переривання
View	Position cursur in sourse code at point of the currently selected breackpoint	Установка курсора в початковій програмі в точку поточної вибраної точки переривання

Команда Debug/Toggle breakpoint (Перемикання точки переривання, Ctrl+F8) встановлює або знімає контрольну точку в поточному рядку. Поточний рядок – це рядок з курсором у вікні редактора. Якщо для поточного рядка встановлена контрольна точка, рядок виділяється кольором (яскравістю).

Після чергового запуску програми (наприклад, командою Ctrl+F9) зі встановленими контрольними точками налагоджувач припинить виконання програми перед виконанням того оператора, який міститься в першій (за ходом виконання програми) контрольній точці.

4 Керування проектом. Меню PROJECT

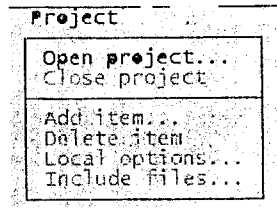
Меню Project дозволяє використовувати в Сі багатофайлові програми, тобто програми, що складаються з модулів, розташованих в декількох програмних файлах. Воно призначене для об'єднання декількох початкових і

об'єктних файлів в процесі створення багатофайлових програм. З його допомогою можна:

- відкрити проект, задавши ім'я файлу проекту;
- закрити проект;
- включити в проект файл програми або змінних;
- виключити з проекту файл програми або змінних;
- задати локальні параметри проекту;
- проглянути файли проекту, що включаються.

Вигляд екрана з меню Project поданий на рис. 25.

Ім'я файлу проекту можна задати викликом команди Open Project. При цьому на екрані з'являється діалоговий блок для вибору або введення імені файлу проекту. Воно повинне мати розширення .prj.



Основна частина імені (до розширення) використовується для формування імені файлу виконуваної програми: при цьому розширення .prj замінюється на .exe. Наприклад, якщо ввести ім'я pr1.prj, то після формування виконуваної програми (exe-модуля) вона матиме ім'я pr1.exe.

Рисунок 25 – Вигляд меню Project

Для вибору імені існуючого файлу проекту можна задати маску імені і потім вибрати файл проекту із запропонованого списку імен. Наприклад:

F:\LEK\LEKC\MNFPR_97\ NFMENU\MENU__PRJ\ *. RRJ.

На рис. 26 поданий приклад діалогового блоку для введення шаблону або імені проекту.

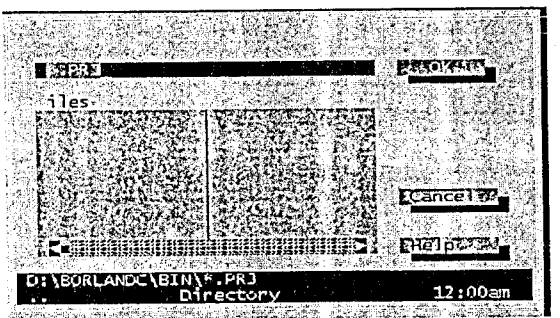


Рисунок 26 – Вигляд діалогового блоку для формування імені проекту

В процесі роботи з проектами в середовищі Borland C++ може бути відкритим тільки один проект. Для роботи з іншим проектом роботу з поточним проектом потрібно закрити за допомогою команди Close project.

Команда Add item використовується для включення файлів в проект. За цією командою відкривається діалоговий блок для вибору файлів проекту. Приклад блоку поданий на рис. 27.

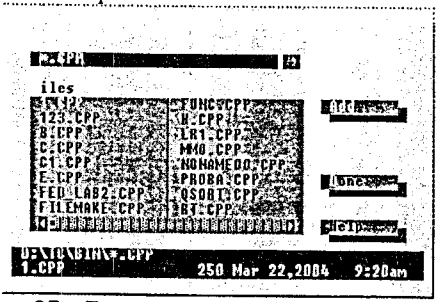


Рисунок 27 – Блок для включення файлів в проект

В результаті формування складу проекту в нижній частині екрана формується вікно проекту із списком файлів проекту. Приклад такого вікна поданий на рис. 28.

Рисунок 28 – Приклад вікна проекту NEW

У кожному рядку вікна подана інформація про один файл. Для компіляції файлу проекту треба активізувати (вибрати, підсвітити) рядок в списку файлів проекту і виконати команду Alt+F9. Після компіляції файлів проекту у вікні виводиться інформація про її результати в стовпцях:

- Lines – кількість рядків початкового файлу;
- Code – об'єм ОП для тексту програми файлу;
- Data – об'єм ОП для даних файлу.

Якщо зі складу проекту треба виключити файл, можна використовувати команду Delete Item. Якщо вікно проекту активне, для керування проектом можна використовувати клавіші Ins (Включити файл) і Del (Виключити файл). Якщо підключений до проекту файл не викликаний у вікно редактора, його можна викликати подвійним натисканням миші в рядку цього файлу у вікні проекту.

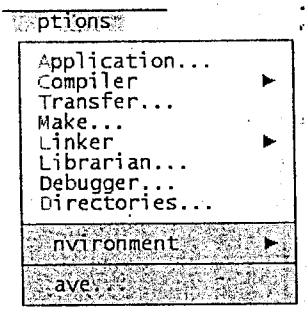
За допомогою команди Local options меню Project можна встановити

локальні параметри проекту: параметри командного рядка для файлу проекту, ім'я і шлях об'єктного файлу, вибрати транслятор.

Команда Include files викликає однойменний блок діалогу із списком файлів, включених директивою #include.

5 Керування системою Borland C++. Меню Options

Меню Options містить команди налаштування середовища, які дозволяють переглядати і модифікувати параметри, що визначають функціонування системи Borland C++. Меню Options подане на рис. 29.



Для встановлення параметрів:
⇐ додатка
⇐ перенесення програм, заданих в =
⇐ роботи менеджера проекта
⇐ компонувальника
⇐ бібліотек
⇐ відладчика
⇐ директорій

⇐ середовища Borland C++

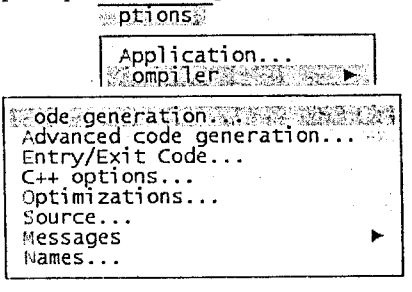
⇐ запам'ятовування налаштувань

Рисунок 29 – Вигляд меню Options

Установлення параметрів додатку, компілятора і умов формування об'єктного і виконуваного модулів.

Команда Options/Application дозволяє встановити параметри додатку: проста або оверлейна програма, використовується в середовищі DOS або Windows і ін..

Команда Options/Compiler відкриває підменю для установлення параметрів компілятора. Вигляд підменю поданий на рис. 30.



Для встановлення параметрів:
⇐ генеруючого коду
⇐ специфічних параметрів
⇐ " "
⇐ " "
⇐ оптимізації
⇐ початкового модуля
⇐ повідомлень про помилки
⇐ специфічних імен

Рисунок 30 – Вигляд підменю команди для установлення параметрів компілятора

За допомогою команди Options/Compiler/Code встановлюються параметри генерації об'єктного коду. Наприклад, один з таких типів моделі пам'яті: Tiny, Small, Medium, Compact, Large, Huge.

За командою Options/Compiler/Messages розгортається додаткове меню для визначення умов видачі повідомлень про помилки і попередження та припинення компіляції. Вигляд меню поданий на рис.31.

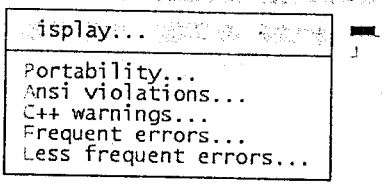


Рисунок 31 – Меню для визначення умов роботи з повідомленнями

За командою Options/Compiler/Messages/Display формується вікно для визначення умов видачі попереджень і завершення компіляції програми. Вигляд вікна поданий на рис. 32.

За командою Options/Make можна встановити параметри системи, які використані при формуванні об'єктного і виконуваного модулів. Наприклад, можна встановити рівень помилок, при яких відбувається переривання роботи менеджера проектів – Break Make On :

( ) Warnings – за попередженнями;
(•) Errors – за помилками;
( ) Fatal errors – за фатальними помилками;
( ) All sources processed – після оброблення всіх початкових файлів.

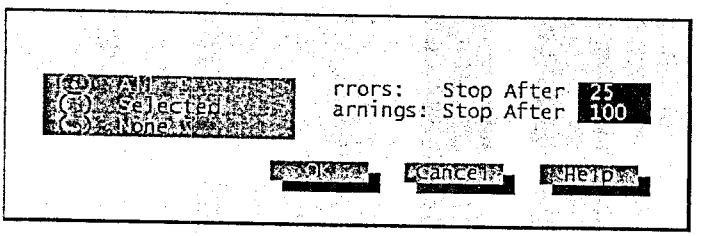


Рисунок 32 – Вигляд вікна для установлення умов компіляції

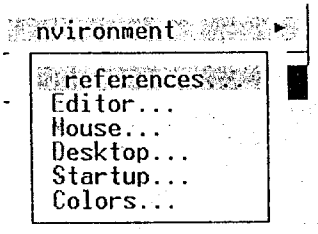
5.1 Установлення параметрів налагоджувача, директорій і середовища

За командою Options/Debugger встановлюються параметри налагоджувача. Наприклад, включення в скомпільований модуль засобів, що дозволяють пов'язати повідомлення про помилки з операторами початкового модуля, режим перемикання вікон редактора і виведення результатів.

За командою Options/Directories відкривається діалоговий блок, в якому можна встановити імена каталогів для:

- Include Directories – файлів, що підключаються з допомогою #include;
- Library Direktories – бібліотечних файлів;
- Output Directory – вихідних файлів;
- Sourse Directories – файлів з початковими модулями.

За командою Options/Environment відкривається підменю такого рівня, за допомогою якого встановлюються параметри середовища. Вигляд підменю команди поданий на рис.33.



Параметри:

⇐ переваги
⇐ текстового редактора
⇐ миші
⇐ робочого столу
⇐ початкових дій
⇐ кольорів всіх елементів екрана

Рисунок 33 – Підменю команди Options/Environment

На рис.34 подано фрагмент вікна діалогу Preferences.

У вікні Preferences за допомогою параметра Screen Size можна встановити кількість рядків на екрані дисплея під час роботи в середовищі Borland C++: 25 або 43/50. Параметром Auto-Save можна задати режим автоматичного збереження налагодження середовища (Environment), робочого столу (Desktop) і проекту (Project). Параметром Source Tracking – порядок відкриття необхідного файлу з початковим текстом: у новому вікні або в поточному вікні редактора. Параметр Save old messages дозволяє запам'ятовувати повідомлення про помилки всіх компіляторів сеансу роботи в середовищі.

Команда Options/Save дозволяє встановити необхідність запам'ятовування і зберегти параметри середовища, робочого столу і розробки проекту. Вигляд вікна для установлення параметрів збереження

налаштувань поданий на рис. 35.

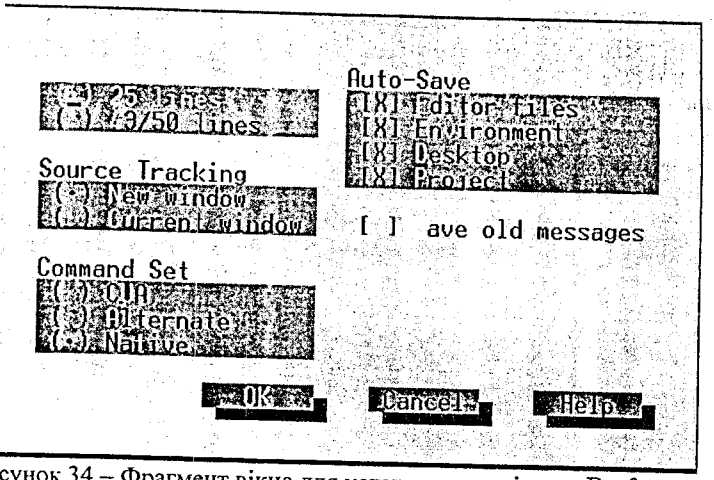


Рисунок 34 – Фрагмент вікна для установлення діалогу Preferences

Призначення параметрів Save Options подане в табл.6.

Таблиця 6 – Призначення параметрів Save Options

Параметр	Що запам'ятовується	Файл, в якому запам'ятовуються параметри
Environment	Параметри середовища	TCCCNFIG.TC
Desktor	Інформація про робочий стіл	prjname.dsk
Project	Файл проекту	Prjname.prj

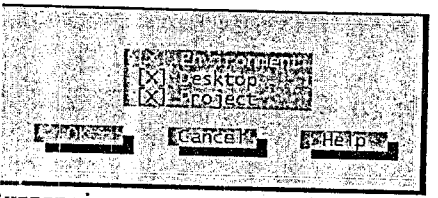
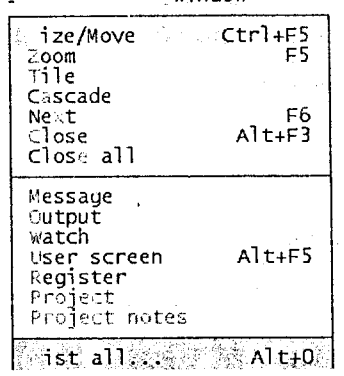


Рисунок 35 – Вигляд вікна для установлення параметрів збереження налаштувань.

6 Керування вікнами. Меню Window

Меню Window містить команди керування вікнами середовища. Вигляд меню Window поданий на рис. 36.



Призначення команд:

⇐ змінити або перемістити вікно клавіатурою
⇐ розкрити або згорнути вікно
⇐ розмістити вікна каскадом
⇐ розмістити вікна «мозаїкою»
⇐ активізувати наступне вікно
⇐ закрити активне вікно
⇐ закрити всі вікна

⇐ відкрити або активізувати вікно повідомлень
⇐ відкрити або активізувати вікно висновку
⇐ відкрити або активізувати вікно перегляду
⇐ показати вікно користувача
⇐ показати вікно з регістрами
⇐ показати вікно з файлами проекту
⇐ відкрити або активізувати вікно з примітками до проекту
⇐ показати список всіх вікон сеансу

Рисунок 36 – Вигляд меню Window

6.1 Керування вікнами на екрані з меню Window

Для переміщення і зміни розмірів вікна можна використовувати мишу (і правий нижній куток вікна) або команду Window/Size/Move (Ctrl + F5 – Змінити/Перемістити) з тією ж метою. Після команди Ctrl + F5 для переміщення вікна використовувати клавіші із стрілками ↑ і ↓, а для зміни розмірів вікна треба натиснути клавішу Shift і змінювати розмір вікна за допомогою тих самих клавіш.

Якщо вікно не займає весь екран, команда Window/Zoom (Розкрити) змінює розмір вікна, відкриваючи його на весь екран. Повторне натиснення F5 повертає початковий розмір вікна. З цією ж метою можна використовувати подвійне клацання мишею на верхній рамці активного вікна, окрім місць розміщення піктограм.

Команда Window/Cascade має в своєму розпорядженні всі відкриті вікна каскадом (уступами): так, щоб верхній край кожного наступного вікна виступав з-під попереднього.

Команда Window/Tile має в своєму розпорядженні всі вікна «мозаїку»: так, щоб були частково видні прямокутники всіх відкритих вікон.

Команда Window/Next (Наступне) робить активним наступне вікно.

Закрити вікно можна командою Window/Close або клацанням миші на маркері закриття вікна в лівому верхньому кутку вікна. Всі вікна можна закрити командою Window / Close all.

6.2 Керування викликом вікон з меню Window

Команди Window/Message (Повідомлення), Window/Output (Виведення), Window/Watch (Переглядання), Window/User screen (Вікно

користувача), Window/Project відкривають і (або) роблять активним відповідне вікно.

Команда Window/Output (Виведення) відкриває і (або) робить активним вікно Output. У ньому можуть бути результати, виведені на екран в процесі виконання програми. Його можна тільки проглядати, писати в ньому не можна.

Команда Window/List all... (Список всіх) викликає вікно Window List. Воно містить список всіх файлів, використовуваних в сеансі роботи з середовищем: всіх відкритих вікон з їх номерами і імен файлів, вікна яких закриті. Приклад вікна поданий на рис. 37.

Команда Window/Close (Закрити) закриває активне вікно. Після цього в списку вікна Window List ім'я файлу закритого вікна з'являється із словом Closed. Приклад вікна Window List із списком вікон поданий на рис. 37.

Команда Window/User screen (Екран користувача) показує екран користувача, наприклад, з виведеними у ньому результатами виконання програми. Повернутися в середовище можна натисненням будь-якої клавіші, але найчастіше це клавіша Esc.

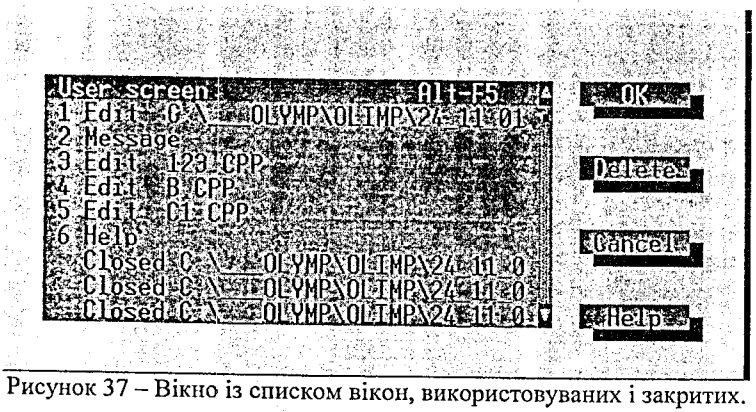


Рисунок 37 – Вікно із списком вікон, використовуваних і закритих.

Команда Window/Project робить активним вікно проекту.

Команда Window/Project notes (Примітки проекту) формує вікно приміток до проекту. В нього можна записати подробиці або рекомендації щодо використання проекту. Цей текст за командою F2 буде збережений у файлі проекту (з розширенням *.prj). Після виходу з середовища і повторного входу в середовище Borland C++ для використання проекту достатньо завантажити його і запустити файл на виконання.

6.3 Вікно повідомлень про помилки – Message

Вікно повідомлень (Message) використовується для виведення повідомлень компілятора і компонувальника про помилки, знайдені в програмі, з метою їх перегляду. Повідомлення можуть бути трьох типів: інформаційні, попередження і про помилки.

Приклад вікна повідомлень компілятора і компонувальника поданий на рис.38.

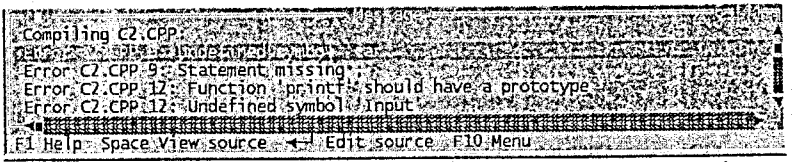


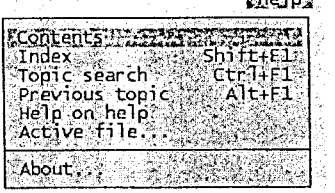
Рисунок 38 – Приклад вікна з повідомленнями компілятора і компонувальника

Список повідомлень вікна Message використовується для пошуку помилки в програмі, що викликала це повідомлення, розташоване у вікні редактора.

Натиснення клавіші Введення при підсвічуванні повідомлення компілятора у вікні повідомлень приводить до переходу у вікно редактора. Причому курсор встановлюється під оператор програми, який викликав повідомлення про помилку. У нижньому рядку вікна редактора при цьому виводиться текст повідомлення про помилку. При підсвічуванні повідомлення про помилку у вікні повідомлень натиснення F1 викликає підказку (пояснення) щодо помилки. Відключити її можна клавішею Esc або командою закриття вікна.

Натиснення F1 з вікна повідомлень за відсутності повідомлень про помилки або при підсвічуванні повідомлення не про помилку, а, наприклад, про ім'я скомпільованого файлу викликає на екран інформацію про призначення вікна повідомлень. Повторне натиснення F1 викликає на екран інформацію про систему допомоги.

7 Система допомоги. Меню Help



Меню Help (Підказка, Допомога) дозволяє використовувати систему підказки, яка містить інформацію зі всіх аспектів інтегрованого середовища і системи Borland C++. Вигляд меню Help поданий на рис. 39.

Рисунок 39 – Вигляд меню Help

Після вибору розділу допомоги натиснення клавіші Введення викликає підказку наступного рівня. У такий спосіб можна одержати до 20 екранів (рівнів) підказки.

Для повернення до попереднього екрану допомоги треба виконати команду Previonus topic (Alt+F1). Щоб прибрати довідкову інформацію з екрану, треба натиснути клавішу Esc або закрити вікно.

Команда Help/Contents (Зміст) відкриває вікно Help з переліком тем системи підказок. Вид вікна Help поданий на рис. 40. З цього вікна можна перейти до будь-якої теми змісту, вибравши підсвічуванням необхідну тему і натиснувши Введення.

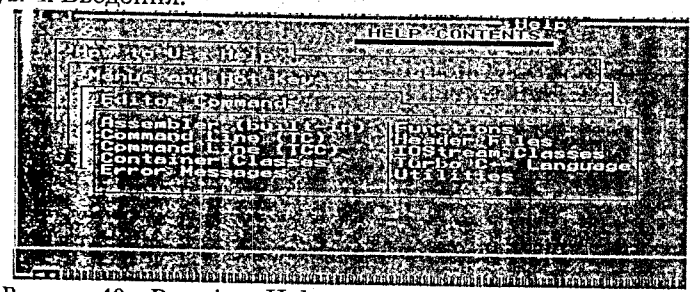


Рисунок 40 – Вид вікна Help з переліком тем системи підказок

Команда Help/Index (Тематичний покажчик) відкриває вікно Help. Під заголовком Turbo Help Index розміщений перелік всіх ключових слів, констант, операцій, імен вбудованих функцій мови і системи, розташованих в алфавітному порядку, за якими можна одержати підказку. Приклад документа вікна поданий на рис. 41.

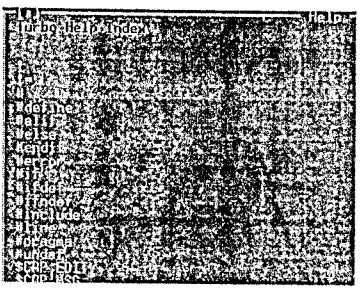


Рисунок 41 – Фрагмент вікна з переліком ключових слів

Команда Help/Topic search (Темати-чний пошук) дозволяє з вікна редактора одержати підказку по операторах мови. Для цього треба підвести курсор під ключове слово оператора, по якому потрібна підказка, і ввести команду Ctrl + Fl. На рис. 42 поданий приклад підказки.

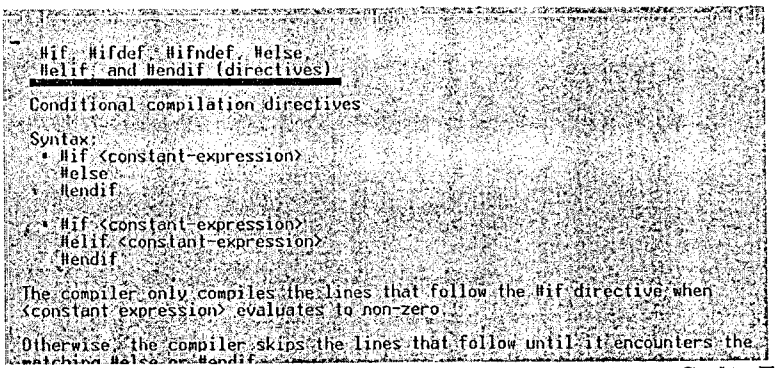


Рисунок 42 – Приклад підказки, викликаної за командою Ctrl + F1

Команда Help/Previous topic (Попередня тема — Alt+F1) відкриває вікно Ctrl + F1 з попередньою темою. Після виходу з системи підказки за цією командою можна повернутися до останнього екрану підказки.

Команда Help/Help on help (Інформація за системою підказки) відкриває вікно Help, в якому подані правила користування самою системою підказок. Приклад вікна поданий на рис. 43.

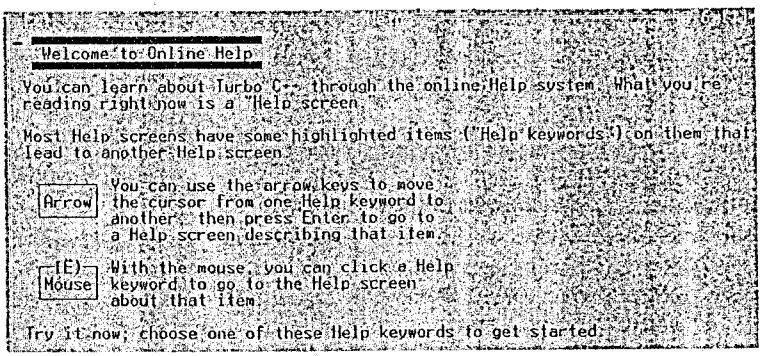


Рисунок 43 – Вікно з правилами користування системою підказок

Команда Help/Active file (Активний файл) викликає вікно Active Help File, яке дозволяє вибрати один з файлів підказок для використання з середовища Borland C++.

- IDE and C++ Language – середовище і мова C++;
- Windows API – інтерфейс Windows-додатків;

- ObjectWindows API – об'єкти інтерфейсу Windows-додатків;
- Turbo Vision API – додатки, що використовують Turbo Vision.

Вигляд вікна для вибору файлу підказок поданий на рис. 44.

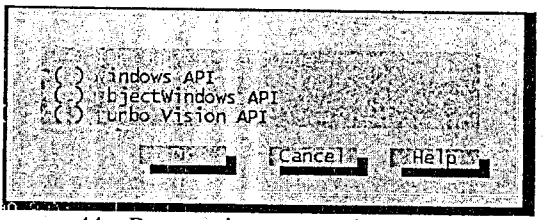


Рисунок 44 – Вигляд вікна для вибору файлу підказок

Команда Help/About відкриває вікно About з інформацією про версію системи. Вигляд вікна поданий на рис.45.

Коли вікно Help активне, з нього можна копіювати текст, наприклад, правило мови або програму прикладу, і вставляти його так само, як з вікна редактора.

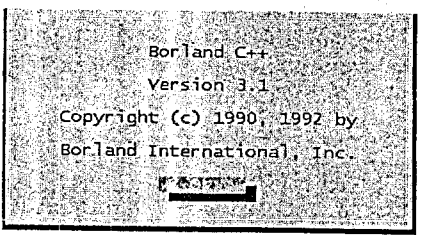


Рисунок 45 – Вигляд вікна з версією системи

8 Текстовий редактор BORLAND C++

Вбудований редактор Borland C++ надає користувачу ряд універсальних засобів для створення і редагування текстів програм і даних. Відразу після успішного виклику системи Borland C++ екран набуває вигляду, поданого на рис. 1. Ознака того, що середовище в стані редагування, тобто вікно редактора активне, – вікно має подвійну рамку.

За допомогою текстового редактора можна виконувати ряд дій, пов'язаних із створенням і редагуванням текстів:

1. Заносити на екран необхідний текст;
2. Видаляти і вставляти символи;
3. Видаляти, відновлювати і вставляти рядки;
4. Розрізати і склеювати рядки;
5. Шукати і замінювати текст рядка;

6. Маніпулювати блоками (частинами тексту).

У редакторі використовується близько 50 команд. Їх можна розбити на 4 основні групи: переміщення курсора і екрана, вставки і видалення, роботи з блоками, інші команди. Підказку щодо команд редактора можна викликати з меню редактора за допомогою клавіші F1 з порожнього місця екрана.

Таблиця 7 – Основні команди редактора

Команди	Клавіші
Основні команди переміщення курсора і екрана	
Курсор вліво на один символ	Ctrl+s або ←
Курсор вправо на один символ	Ctrl+D або →
Курсор вліво на слово	Ctrl+A або Ctrl←
Курсор вправо на слово	Ctrl+F або Ctrl→
Курсор вгору на один рядок	Ctrl+E або ↑
Курсор вниз на один рядок	Ctrl+X або ↓
Екран на один рядок вгору	Ctrl+W
Екран на один рядок вниз	Ctrl+Z
Екран вгору на одну сторінку	Ctrl+R або PgUp
Екран вниз на одну сторінку	Ctrl+C або PgDn
Команди швидкого переміщення курсора	
У початок рядка	Ctrl+Q або Home
У кінець рядка	Ctrl+Q або End
У верхній рядок вікна	Ctrl+Q E або Ctrl+Home
У нижній рядок вікна	Ctrl+X або Ctrl+End
До початку файлу	Ctrl+Q R або Ctrl+PgUp
До кінця файлу	Ctrl+Q C або Ctrl+PgDn
Команди вставки і видалення	
Вкл./викл. Режим вставки	Ctrl+Y або Ins
Вставити рядок перед поточним	Ctrl+N або Введення
Видалити рядок над курсором	Ctrl+H
Видалити текст до кінця рядка	Ctrl+Q Y
Видалити символ зліва від курсора	Ctrl+H або ←Backspece
Видалити символ над курсором	Ctrl+G або Del
Видалити слово праворуч від курсора	Ctrl+T

Продовження таблиці 7

Команди	Клавіші
Команди роботи з блоками	
Помітити блок	Shift + «стрілки»
Перейти до початку блоку	Ctrl+Q B
Перейти до кінця блоку	Ctrl+Q K
Помітити початок блоку	Ctrl+K B
Помітити кінець блоку	Ctrl+K K
Помітити одиночне слово	Ctrl+K T
Зробити невидимим/ видимим блок	Ctrl+K H
Видалити блок	Ctrl+K Y
Надрукувати блок	Ctrl+K P
Скопіювати блок в місце, позначене курсором	Ctrl+K C
Перемістити блок в місце, позначене курсором	Ctrl+K V
Зчитати блок з диска у вікно з курсором	Ctrl+K R
Записати блок на диск	Ctrl+K W
Зсунути блок вправо	Ctrl+K I
Зсунути блок вліво	Ctrl+K U
Скопіювати блок в Clipboard	Ctrl+Ins
Видалити блок	Ctrl+Del
Вставити блок з Clipboard	Shift+ins
Перемістити блок в Clipboard	Shift+Del
Інші команди	
Вкл./викл. автоматичний відступ	Ctrl+ O I
Знайти текст	Ctrl+Q F
Знайти і замінити текст	Ctrl+Q A
Повторити останній пошук	Ctrl+L
Зберегти відредагований текст	Ctrl+K або F2
Знайти парний символ вперед	Ctrl+Q [
Знайти парний символ назад	Ctrl+Q]
Табуляція	Ctrl+I або Tab
Вкл./викл. Режим табуляції	Ctrl+ O T

Вікно редактора

Вікно редактора використовується для створення і корегування файлів Якщо активне вікно редактора (Edit), то рядок оперативної підказки ма вигляд:

F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

Призначення її функціональних клавіш подане в таблиці 3.

Після компіляції у разі виявлення помилок формується вікно Message. Якщо активне вікно повідомлень (Message) і підсвічується повідомлення компілятора: (попередження або помилка), то нижній рядок оперативної підказки має вигляд:

F1 Help Space View source ↵ Edit source F10 Menu

Якщо на екрані розташовані вікно редактора і вікно Message з повідомленнями про помилки й вікно редактора активне, то за допомогою команд Alt+F7 і Alt+F8 з вікна редактора можна проглядати послідовно всі повідомлення, наведені у вікні повідомлень (про помилки, виявлені під час компіляції). При цьому у вікні редактора підсвічується помилковий рядок, а у вікні повідомлень в рядку з відповідним повідомленням зліва встановлюється точка – "вибрано".

Якщо активне вікно Message, то можна послідовно вибирати різні повідомлення про помилки; при цьому у вікні редактора підсвічуватиметься рядок, в якому знайдена відповідна помилка. Якщо при вибраному рядку у вікні Message натиснути Введення, активним стане вікно редактора і в нижній його частині виведеться червоним кольором повідомлення про помилку, а курсор встановиться під оператором, що викликав помилку.

Вікно редактора імітує довгий і широкий лист паперу, фрагмент якого видно у вікні редактора. Максимальна довжина рядка "листа" – 1023 символи. Але для програм доцільно використовувати видиму на екрані довжину рядка, тобто 77 символів. При заходженні курсора за праву межу екрана, екран зміщується вправо. Вертикальна довжина "листа" обмежується кількістю рядків тексту: вона повинна бути не більше 64534. Тексти програм і дані можна записувати на екрані, починаючи з першої позиції рядка. Екран має для текстів 23 рядки. Допускається робота з файлами загальним об'ємом до 6 Мбайтів.

У верхній частині рамки вікна відображається коротке ім'я файлу, завантаженого в редактор, якщо файл розташований в поточному каталозі, встановленому командою Change dir; повне ім'я файлу з ім'ям каталога і підкаталогів, з яких завантажений файл, відображається так само, якщо файл розташований не в поточному каталозі.

У лівій частині нижньої рамки вікна редактора встановлений номер рядка і стовпця, в якому розташований курсор, знак * перед номером рядка, у якому стоїть курсор, означає, що у файл внесені зміни і він повинен бути збережений наново.

Вікно редактора можна швидко зміщувати по тексту за допомогою команд швидкого переміщення курсора (див. табл. 7).

Параметри вікна редактора можна встановити за допомогою меню

установлення режимів середовища: Options/Environment/Editor. Після виклику цієї команди з'являється вікно Editor Options.

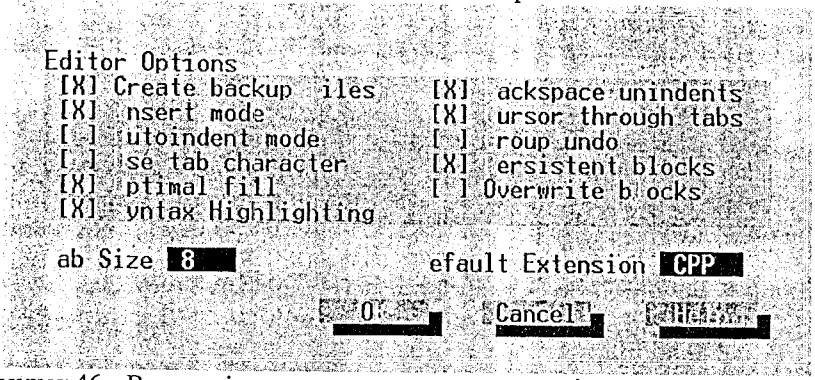


Рисунок 46 – Вигляд вікна для установлення параметрів текстового редактора

Система Borland C++ може функціонувати на будь-якому персональному комп'ютері (ПК). Клавіатура комп'ютера як правило, стандартна. Тривале натиснення будь-якої клавіші викликає багатократне виконання її функцій. Наприклад, натиснувши і утримуючи в натиснутому стані клавішу Пропуск, можна стерти ряд символів рядка. Призначення керуючих клавіш наведене в табл. 8.

Система має можливість перемикання на кирилицю або латинський шрифт, великі або маленькі букви (клавіші Caps Lock або Shift). Перемикання з латинського шрифту на кирилицю може виконуватись, наприклад, за допомогою клавіші Ctrl або лівої і правої клавіші Shift; це встановлюється при завантаженні русифікатора системи.

Курсор – мерехтлива риска – з'являється на екрані відразу після входу в систему Borland C++. Він указує позицію екрану, в яку можна занести або з якої повинен бути видалений символ.

У лівій частині нижньої рамки вікна редактора фіксується поточне положення курсора у вигляді номера рядка екрана і номера позиції в поточному рядку, в якій розташований курсор.

Після введення на екран чергового символу курсор автоматично зміщується до наступної позиції рядка екрана. При формуванні і корегуванні текстів на екрані потрібно встановити курсор в необхідну позицію екрана. Для керування положенням курсора без занесення на екран будь-яких символів служать клавіші Home, End, PgDn, PgUp, ←, ↑, →, ↓, Tab і команди переміщення курсора і екрана. Перелік і призначення команд переміщення курсора подані в таблиці 7.

Таблиця 8 – Призначення керуючих клавіш

Клавіші	Призначення клавіш
Клавіші верхнього ряду клавіатури	
Esc	Escape – вихід, перехід; відміна команди, повернення з меню
F1+F12	Функціональні клавіші
Print Screen	Разом з клавішею Shift – друк вмісту екрану
Scroll Lock	Блокування прокручування
Pause	Припинення виконання програми
Break	Разом з Ctrl перериває виконання програми
Клавіші, розташовані зліва і справа від алфавітно-цифрової клавіатури	
Tab ← →	Клавіша табуляції, переведення курсора вправо на кількість позицій, визначених параметром Tab Size команди Options/Evironment/Edition
Caps Lock	Блокує верхній регістр (великі букви) для букв АЦК
Shift	Перемикання на інший регістр
Ctrl	ConTRoL – керуюча
⇑	Shift – зміна регістра на час натиснення клавіші
Alt	ALTernat – альтернатива (вибір)
←	BackSpace – видалення символу зліва від курсора
Enter	Введення команди : вставка («розрізання») рядка
Керуючі клавіші правої частини клавіатури	
Num Lock	«Цифрове блокування», перемикає режими роботи: цифровий-нецифровий; при включенні режиму цифр включається сигналізація (лампочка) і відключаються клавіші керування положенням курсора ←, →, ↑, ↓
Ins	Переключення режиму вставки / заміни
Del	Видалення символу над курсором
PgUp	PaGe Up – екран на сторінку вгору
PgDn	PaGe DowN – екран на сторінку вниз
Home	Курсор в початок рядка
End	Курсор в кінець рядка
←,→,↑,↓	По символьне переміщення курсора у вказаному напрямі

Редагування текстів в системі Borland C++

В процесі створення і корегування текстів з клавіатури можна виконувати такі види робіт: писати тексти на екран; маніпулювати символами, словами і частинами рядків; маніпулювати рядками; працювати з блоками.

При редагуванні текстів можливі 2 режими роботи:

– вставки – курсор зображений у вигляді символу підкреслення;
– заміни – курсор у вигляді прямокутника, що світиться.

При маніпулюванні символами і частинами рядків можна виконати такі види робіт: замінити помилковий символ; вставити помилково пропущений символ; видалити зайвий символ або ряд символів.

Для заміни помилкового символу треба: встановити режим заміни клавішею Ins; підвести курсор під помилковий символ і натиснути клавішу з необхідним символом – на місці колишнього символу з'явиться той, що потрібен; решта тексту залишається незмінною.

Для вставки символа серед наявних символів рядка потрібно: встановити режим вставки за допомогою клавіші Ins; встановити курсор під символ, перед яким треба вставити символ; натиснути клавішу з необхідним символом; текст разом з курсором, починаючи з позиції, в якій був встановлений курсор, зміститься вправо, а на тому місці, де раніше був курсор, з'явиться вставлений символ.

Видалити один і більше помилкових символів рядка можна за допомогою клавіш:

– ← Backspace – видалить символ зліва від курсора;
– Del – видалить символ над курсором.

У режимі вставки видалений символ зникне, а текст, що знаходиться праворуч від курсора, зсунеться на одну позицію вліво. У режимі заміни на місці видаленого символу з'являється пропуск.

Для видалення необхідного прямокутного блоку треба виділити його кольором (наприклад, за допомогою клавіші Shift і клавіш управління положенням курсора), а потім виконати команду Ctrl+K Y.

Щоб видалити праву частину рядка в режимі вставки або заміни, треба:

1. Встановити курсор під перший символ, починаючи з якого треба видалити текст;
2. Ввести команду Ctrl+Q Y; текст рядка праворуч від курсора до кінця рядка буде видалений.

При роботі з рядками редактор Borland C++ дозволяє:

- видалити рядок;
- вставити порожній рядок;
- "розрізати" рядок на два;
- "склеїти" два сусідні рядки;
- відновити помилково скорегований рядок.

Для видалення рядка треба встановити курсор під будь-який символ рядка, який треба видалити, і ввести команду Ctrl+Y. Для вставки порожнього рядка після даного треба: встановити режим вставки за допомогою клавіші Ins; встановити курсор в кінець рядка, після якого треба вставити новий

рядок, наприклад, за допомогою клавіші End і натиснути Введення.

Для вставки порожнього рядка можна використовувати команду Ctrl+N: якщо курсор на початку рядка тексту, новий рядок вставляється перед даним, якщо курсор в кінці рядка – після даного.

Для "розрізання" даного рядка на два треба: встановити режим вставки; встановити курсор під символ, починаючи з якого треба розрізати рядок, і натиснути клавішу Введення – текст, починаючи від курсора до кінця рядка, переміститься в початок наступного, нового рядка.

"Розрізати" рядок можна також за допомогою команди Ctrl+N, встановивши курсор під символ, починаючи з якого треба "розрізати" даний рядок. Для "склеювання" двох сусідніх рядків треба: встановити курсор в кінець даного рядка; натиснути клавішу Del: до кінця даного рядка приєднається текст наступного.

Другий спосіб "склеювання" рядків: встановити курсор в початок другого "склеюваного" рядка і натиснути клавішу ← Backspace.

Під текстовим блоком розуміється будь-яка кількість тексту від одного до декількох сотень рядків. Частіше це група рядків. З блоком (фрагментом тексту) в системі Borland C++ можна виконувати такі види робіт: копіювати або перемістити в задане курсором місце, копіювати або перемістити в буфер, видалити, записати у файл на МД, ввести текст з файлу у вікно редактора, віддрукувати.

Щоб маніпулювати блоком, треба визначити його розміри, тобто виділити блок. Одночасно в тексті можна виділити тільки один блок. Для виділення блоку можна використовувати один із способів:

- встановити курсор під перший символ тексту блоку і ввести команду Ctrl+K B; це помітить початок блоку; встановити курсор під останній символ блоку і ввести команду Ctrl+K K; це помітить кінець блоку;
- встановити курсор під перший символ тексту блоку; натиснути клавішу Shift і, не відпускаючи її, помітити блок за допомогою клавіш керування положенням курсора.

При виділенні блоку всі рядки, окрім першого і останнього, виділяються повністю.

Після цих команд блок буде виділений підсвічуванням (кольором). Можна вводити команд маніпулювання блоком. Список команд роботи з блоками поданий в таблиці 7. Погасити підсвічування блоку можна командою Ctrl+K K, встановивши курсор в початок блоку, або командою Ctrl+K H.

Для того, щоб записати блок на МД, треба помітити блок і ввести команду Ctrl+K W. Після введення команди на екрані з'явиться додаткове вікно Write Block to File, в яке треба записати і після цього ввести ім'я файлу на

МД, в який треба помістити блок. Ім'я файлу, в який треба помістити виділений блок, можна вибрати із списку у вікні Write Block to File.

Для введення блоку з файлу у вказане курсором місце тексту на екрані треба ввести команду Ctrl+K R – читати текст з файлу в редактор. Після введення цієї команди на екрані з'явиться додаткове вікно Read Block From File, в яке треба записати або вибрати із списку і потім ввести ім'я файлу, з якого треба прочитати текст. Текст буде скопійований з файлу у вікно редактора, починаючи з точки, в якій встановлений курсор. Приклад вікна Read Block From File поданий на рис. 47 .

Для того, щоб скопіювати блок в необхідне місце, треба помітити блок, підвести курсор під символ, після якого треба вставити блок, і ввести команду Ctrl+K C. Для переміщення тексту поміченого блоку треба ввести команду Ctrl+K V.

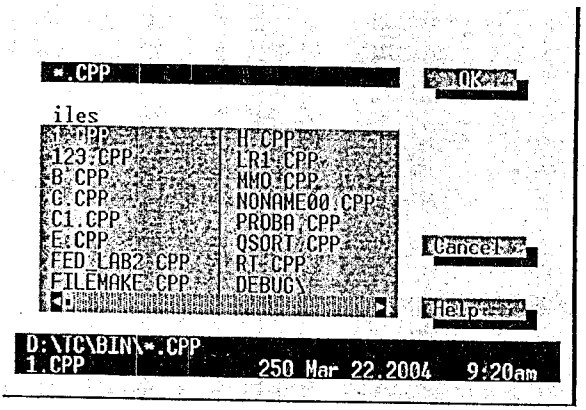


Рисунок 47 – Приклад вікна для введення блоку з файлу на МД

9 Лабораторний практикум

9.1 Лабораторна робота №1.

9.1.1 Складання найпростіших програм мовою Сі

Мета лабораторної роботи – навчитися вирішувати, тестувати і розв'язувати на ЕОМ найпростіші типові задачі, як обчислювального, так і необчислювального характеру. Як правило, в самій умові задачі вже визначений алгоритм її розв'язання. Необхідно записати цей алгоритм, використовуючи основні конструкції мови Сі. Для перевірки правильності роботи програми повинні бути подані необхідні тестові дані.

Хід роботи :

1. Скласти алгоритм програми за індивідуальним варіантом, номер якого визначає викладач.
2. Скласти програму мовою Сі відповідно до розробленого алгоритму.
3. Протестувати програму з використанням нескладних даних і перевірити результат.

Вміст звіту

1. Навести умову і номер індивідуального варіанта.
2. Навести блок-схему розробленої програми та її описання.
3. Навести математичну модель розробленої програми.
3. Навести текст розробленої програми з коментарями.
4. Навести результати тестування програми.
5. Відповісти на контрольні запитання.

Теоретичні відомості

Для виконання лабораторної роботи необхідно знати роботу основних операторів мови Сі та роботу операторів введення-виведення. Для більш детальної інформації слід звернутись до частини першої навчального посібника. В теоретичних відомостях нагадаємо тільки головне. Логічний оператор if називається оператором розгалуження і дія його подібна до аналогічного оператора в мові Паскаль.

Якщо значення виразу ненульове або true (істина), тоді виконується наступний оператор, якщо ж вираз дорівнює нулю або false(хибний), то наступний оператор не виконується.

Наприклад if (i) ++ i;

Дужки, в які взято вираз i, є обов'язковими. Вираз в дужках може бути зовсім довільним, але з однією умовою: він повинен повертати скалярну величину, яку можна порівняти з нулем. Попередній приклад можна переписати в вигляді if (i !=0) ++ i;

Коли величина може дорівнювати нулю? Для типів bool(false), short(0), int(0), float(0.0) і double(0.0) відповідь очевидна. А якщо це

символ або покажчик? Нульовим значенням типу char є \0. Покажчики дорівнюють нулю в тому випадку, якщо вони мають значення Null або 0.

Крім того, оператор if має ще дві форми використання:

if (вираз) оператор1;

else оператор2;

Якщо вираз істинний, то виконується оператор1, якщо ні, то виконується оператор2. Оператор1, оператор2 – це простий або скалярний оператор. Перед ключовим словом else крапка з комою ставиться, якщо оператор1 – простий оператор, і не ставиться, якщо оператор1 – складений оператор.

Приклади умовних операторів повної форми

```
if (x < 0)  y = 1;                    // повна форма
   else     y = 2;                    // і прості оператори
if (a > b) { x = 0; y = 1 }           // повна форма
   else    { x = 1; y = 0 }           // і складені оператори.
```

Є ще така форма:

if (вираз 1) оператор 1

else if (вираз 2) оператор 2 else оператор 3

В мові Сі існує особливий умовний вираз, який іноді називають тернарною операцією ? :. Він призначений для вибору одного з двох виразів для обчислення значень змінної в лівій частині оператора присвоювання. Його форма така: a = (B) ? B1 : B2; де a – ім'я змінної лівої частини оператора присвоювання; B – умова прийняття рішень, вираз будь-якого типу; B1, B2 – вирази, за допомогою яких вираховується значення a (дужки необов'язкові).

Виконання оператора з умовним виразом проводиться таким чином: спочатку вираховується значення B. Потім аналізується результат його обчислення:

– якщо B! = 0, тобто умова істинна. Тоді обчислення змінної a відбувається за допомогою виразу B1;

– якщо B = 0, тобто хибне, – за допомогою виразів B2.

Якщо типи змінної лівої частини оператора присвоювання і результатів виразів B1 і B2 різні, то тип результату обчислення виразу перетворюється в тип змінної a. Приклад вибору за допомогою умовного виразу більшого з двох значень a і b; x = (a > b) ? a : b; або x = a > b ? a : b;

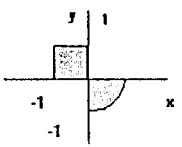
Для того, щоб вибрати одну із декількох альтернатив процесу оброблення даних, тобто одну із N можливих комбінацій залежно від значення ключа, використовується оператор switch. Він має скорочену (без default) або повну форму (з default). Правила виконання оператора switch подібні правилам виконання оператора case в Паскалі.

Приклад програми з використанням оператора switch. При введенні одного із символів, 'y' або 'Y' програма виведе на екран слово "Так", а при введенні символів 'n' або 'N' – слово "Ні".

```
#include <stdio.h>
#include <conio.h>
 void main()
 {
  char c;
  clrscr ();              // очистити екран
  puts("введіть символ 'Y', 'y' або  'n', 'N' \ ");  //введення символу
  c = getchar();
  switch (c)
  {   case 'Y':
      case 'y':  puts (" Так"); break;
      case 'N':
      case 'n': puts ("Ні"); break;
      default : printf("Ви помилились! \n" );
    }
 printf("\nДля завершення програми натисніть будь-яку
        клавішу\n");
 getch ();
 }
```

Приклади розв'язання типових задач

Задача 1



Дано два дійсних числа x, y – координати точки на площині. Потрібно обчислити значення

$$P = \begin{cases} 1 + xy, & \text{якщо } (x, y) \in F; \\ 5, & \text{якщо } (x, y) \notin F. \end{cases}$$

Область F задана графічно.

Програма має такий вигляд:

```
# include <stdlo.h>
# include <conio.h>
  void main ( )
{
     float x, y, p;
     int n;
m1: clrscr() ;   // очищаємо екран
```

```
    puts("Введіть координати точки на площині");
    printf("\n\t x=");
    scanf(" %f" , &x) ;
     printf("\n\t y= ");
    scanf("%f", &y) ;
    if(y<=0&&x>=0&&x*x+y*y<=1 | | x<=0&&y>= 0&&y<=1&&x>=-1)
       p=1 + x*y;
                else  p = 5;
    printf ( " \nЗначення P=%-5.2f\n" , p) ;
    printf(" \nПродовжимо? Так - введіть 7 : ") ;
    scanf("%d", &n) ;
     if (n = 7) goto ml;       else puts ("Кінець роботи") ;
}
```

Задача 2

Увести послідовність літер, що закінчується ознакою кінця файлу (EOF). Підрахувати кількість пропусків, кількість букв 'A' (з врахуванням верхнього/нижнього регістрів) і кількість інших символів. Наведемо розв'язання цієї задачі в двох варіантах: з використанням умовного оператора if і з використанням перемикача — оператора switch:

Варіант 1

```
#include <stdio.h>
#include <conio.h>
void main ( )
{
  int ka=0, kpr=0, kost=0;
  int c;
  clrscr( );
  puts("Введіть послідовність символів, яка закінчується EOF");
  while ((c = getchar()) != EOF)
  {
      if ( c=='a' | |  c== 'A')  ka++ ;
         else if(c==' ' | |  c== '\t'  | | c=='\n' )
          kpr++;
               else kost++;
   }
  printf("\nБуло введено\n,букв A : %3d\n", ka) ;
  printf ("Пропусків : %3d\nІнших:   %3d",  kpr, kost);
  getch();                    // утримуємо екран
}
```

Варіант 2

```
#include <stdio.h>
#include <conio.h>
void main()
{
   int  ka=0, kpr=0, kost=0;
   int c;
   clrscr();
   puts("Введіть послідовність символів, що  закінчується EOF");
   while ( (c=getchar()) !=EOF)
  {
     switch(c)
     {
       case 'a': case 'A': ka++; break;
       case ' ': case '\t': case '\n':  kpr++;break;
       default: kost++;
      }
  }
  printf("\nБуло введено:\ nбукв A: %3d\n", ka) ;
  printf("Пропусків : %3d\nІнших:   %3d",  kpr, kost);
  getch(); // утримуємо екран
}
```

Для демонстрації роботи програми необхідно ввести рядок, що закінчується ознакою кінця файлу EOF. Ця ознака введена спеціально для того, щоб було зручно позначати кінець введення даних. EOF — це ціла константа, визначена у файлі <stdio.h>; її значення вибирається таким, щоб воно відрізнялося від кожного з можливих значень типу char (звичайно EOF — це мінус 1). З цієї причини змінна, що зберігає черговий символ, що вводиться, у програмі описана як int. Як ввести ознаку EOF — залежить від операційної системи. В UNIXі для цього потрібно одночасно натиснути Ctrl+D; у MS-DOS — Ctrl+Z.

Задача 3

Дано дійсне число x і натуральне N. Обчислити

$$\sum_{K=1}^{N}(-1)^k \frac{x^k}{(2k+1)!}.$$

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
```

```
{
  int n, i, k;
  unsigned long fact;
  float  s, x;
  printf("Введіть кількість доданків  : ") ;   scanf ("%d", &n) ;
  printf ("Введіть дійсне число x : ");     scanf ("%f", &x) ;
  for ( k=1, s=0;  k<=n; k++)
    {
     for(i=1, fact=1; i<=2*k+1; i++) fact*=i;
     s+=pow(-1.0,(float)k)*pow(x,(float)k)/fact;
     }
    printf("%4.2f",s);     getch();
}
```

9.1.2 Завдання на програмування

1. Знайти і роздрукувати всі натуральні тризначні числа, які дорівнюють сумі кубів своїх цифр.

2. За введеним числом встановити, у яких позиціях його двійкового коду записані нулі.

3. За трьома введеними дійсними числах з'ясувати, чи можна побудувати трикутник з такими довжинами сторін, і якщо можна, то який це трикутник: рівносторонній, рівнобедрений, прямокутний чи загального виду.

4. Визначити k-у цифру послідовності 182764125216343..., у якій записані підряд куби натуральних чисел.

5. Дано натуральне число *n*. Обчислити суму *k* старших (що знаходяться ліворуч) цифр числа.

6. Дано додатне число a. Знайти k-е число Фібоначчі, таке, що $r_{k-1} \le a \le r_k$. Числа Фібоначчі: $r_1=1$, $r_2=1$, $r_k=r_{k-2}+r_{k-1}$, k=3,4,... .

7. Введіть місяць і день свого народження. З'ясуйте, який найближчий рік буде для вас щасливим. Рік називається щасливим, якщо залишок від ділення суми його цифр на 10 збігається з аналогічним залишком від ділення на 10 суми цифр дня та місяця народження.

8. Дано натуральне число *N*. Підрахуйте суму цифр цього числа, що знаходяться на непарних позиціях, (нумерація позицій йде зліва направо).

9. Обчислити, не використовуючи функцію pow( ), значення функції

$z(x, m) = x^m \sin^m (xm)$ для значень аргументів:

x від -1.1 до 0.3 із кроком 0.2; m від 1 до 5 із кроком 1.

10. У касі є тільки три- і п'ятикарбованцеві купюри (це було в далекому 1980 р.). Скласти програму, що "виплачувала" би такими купюрами будь-яку суму більшу 7 карбованців.

11. За введеним цілим числом M роздрукувати всі тризначні десяткові числа, сума цифр яких дорівнює M. Підрахувати також кількість таких чисел або повідомити про те, що їх немає.

12. Обчислити значення функції

$$F(x)=\begin{cases}\sin(\frac{\pi}{8}+|x| & при\ x<0.3;\\ \sin(x^2\,\pi/2) & при\ x\ge 0.3.\end{cases}$$

для значень аргументу x від -0,5 до 1,2 із кроком 0,1.

13. За введеним натуральним числом *N* визначити, чи є воно досконалим. Досконале число дорівнює сумі усіх своїх дільників, включаючи одиницю і не включаючи себе.

Наприклад: 6=1+2+3 - досконале число; 8=1+2+4 – недосконале.

14. Дано натуральне число *N*. Обчислити суму його цифр.

15. Обчислити суму членів нескінченного ряду

$$\frac{x-1}{x+1}+\frac{(x-1)^3}{3(x+1)^3}+\ldots+\frac{(x-1)^{2n+1}}{(2n+1)(x+1)^{2n+1}}+\ldots$$

з точністю до члена ряду, меншого ε (10^{-5} <= ε <= 10^{-3}).

16. Роздрукувати всі чотиризначні натуральні десяткові числа з діапазону [2000..3000], у записі яких немає двох однакових цифр. Підрахувати кількість таких чисел.

17. За заданим дійсним x обчислити значення $\sqrt{x}$ за такою ітераційною формулою: $y_{i+1}=0.5(y_i+3x/(2y_i^2+x/y_i))$.

Початкове наближення $y_0 = x$. Ітерації припинити при $|y_{i+1} - y_i| < 10^{-5}$.

18. Обчислити значення функції $f(x) = \sin x + \sin^2(x^2) + \sin^3(x^3)$ для значень аргументу x від 0,0 до 1,2 із кроком 0,1.

19. З'ясувати, які цифри (по одній праворуч і ліворуч) треба приписати до числа 1022, щоб отримане число ділилося на 7, 8, 9.

20. Дано натуральне число N. Обчислити суму k молодших (правих) цифр числа.

21. Троє друзів були свідками ДТП. Перший помітив, що номер порушника ділиться на 2, 7 і 11. Другий запам'ятав, що в записі номера беруть участь всього дві різні цифри, а третій – що сума цифр дорівнює 30. Визначити чотиризначний номер порушника.

22. За заданим дійсним x (x<3) обчислити значення $\sqrt{x}$ за формулою подвійної ітерації: $Y_{i+1} = Y_i - 0.5Y_iZ_i$, $Z_{i+1} = 0.25z_i^2(z_i - 3)$. Ітерації припинити при $|y_{i+1} - y_i| < 10^{-6}$.

23. Обчислити значення функції $f(x)=\begin{cases}\ln(1+|x|) & при\ x<-0,2;\\ e^{-(1+x)} & при\ x\ge -0,2.\end{cases}$

для значень аргументу x від -0,8 до 0,6 із кроком 0,1.

24. Дано натуральне число N. Знайти суму цифр числа, що знаходяться на парних позиціях (старша цифра знаходиться на першій позиції).

25. Натуральне число m подати у виді суми квадратів двох натуральних чисел. Видати повідомлення, якщо таке подання неможливе.

26. За заданим дійсним x обчислити значення $\sqrt[3]{x}$ за такою ітераційною формулою: $y_{i+1} = 1/3(2y_i+x/y_i^2)$. Початкове наближення $y_0 = x$. Ітерації припинити при $[y_{i+1} - y_i] < 10^{-5}$.

27. Визначити k-у цифру послідовності **1 1 2 3 5 8 13 21 34...** ,
у який виписані підряд усі числа Фібоначчі.

28. Знайти найбільший загальний дільник (НЗД) двох уведених натуральних чисел, використовуючи алгоритм Евкліда.

Алгоритм Евкліда: віднімаємо від більшого числа менше доти, доки вони не стануть однаковими; отримане в результаті число і є НЗД.

29. Визначити k-у цифру послідовності 1234567891011121314..., у який записані підряд усі натуральні числа.

30. По введеному символу встановити, у яких позиціях його двійкового коду записані одиниці.

31. Визначити k-у цифру послідовності 14916253649... , у який записані підряд квадрати всіх натуральних чисел.

32. Уведіть свій рік, місяць і день народження. Ваш день народження дуже щасливий, просто щасливий чи звичайний? Дуже щасливий – якщо всі залишки від ділення на 7 сум цифр року, місяця і дня збігаються. Просто щасливий – якщо збігаються два будь-яких залишки. Звичайний – якщо збігів немає.

33. Дано дійсні числа A_1, B_1, C_1, A, Y, C. З'ясувати взаємне розташування прямих $A_1 \times x + B_1 \times y = C_1$ і $A \times x + B \times y = C$. Якщо прямі перетинаються, надрукувати координати точки перетину.

34. Знайти всі тризначні числа, які можна подати різницею між квадратом числа, утвореного першими двома цифрами, і квадратом третьої цифри.

35. Протягом доби щогодини проведені 24 вимірювання напруги в мережі. Визначити максимальне значення напруги в мережі в інтервалі (20,6) годину і час, коли воно було зафіксовано.

36. Дано натуральне число N. Обчислити $S = \sum_{k=1}^{N} (-1)^k (2k+1)!$.

37. У виразі ((((1 ? 2) ? 3) ? 4) ? 5) ? 6 замість кожного знака ? поставити знак однієї з операцій +, -, *, / так, щоб результат обчислень дорівнював 35.

38. Назвемо автобусний квиток вдалим, якщо сума цифр його шестизначного номера ділиться на 7(без остатку). Чи можуть два квитки підряд бути вдалими?

39. Трикутник заданий координатами своїх вершин. Знайти його периметр і площу. (Для знаходження довжини сторони трикутника використовувати директиву #define.)

40. Маса однієї молекули води приблизно $3.0 \cdot 10^{-23}$гр. Кварта води дорівнює приблизно 950 грамам. Напишіть програму, що запитує кількість води в квартах і відображає на екрані число молекул у цій кількості води.

41. Обчислити і вивести на екран в вигляді таблиці значення функції F на інтервалі від Хпоч. до Хкін. з кроком dX.

$$F = \begin{cases} ax^2 + b & \text{при } x < 0 \text{ и } b \neq 0 \\ \dfrac{x-a}{x-c} & \text{при } x > 0 \text{ и } b = 0 \\ \dfrac{x}{c} & \text{в інших випадках} \end{cases}$$

де a, b, c — дійсні числа.

Функція F повинна набувати дійсного значення, якщо вираз (Ац АБО Вц) І (Ац АБО Сц) не дорівнює нулю, і цілого значення – в протилежному випадку. Через Ац, Вц і Сц позначені цілі частини значень a, b, c, операції І і АБО — порозрядні. Значення a, b, c, Хпоч., Хкін., dX ввести з клавіатури.

42. Швидкість човна в стоячій воді V км/год, швидкість течії річки U км/год (U < V). Час руху човна по озеру T1 год., а по річці (проти течії) — T2 год. Напишіть програму, що визначає шлях S, пройдений човном.

43. Швидкість першого автомобіля V1 км/год, другого — V2 км/год, відстань між ними S км. Напишіть програму, що визначає відстань між ними через T годин, якщо автомобілі віддаляються один від одного.

44. Швидкість першого автомобіля V1 км/год, другого — V2 км/год, відстань між ними S км. Напишіть програму, що визначає відстань між ними через T годин, якщо автомобілі рухаються назустріч один одному.

45. Дана сторона рівностороннього трикутника. Напишіть програму, яка знаходить площу цього трикутника і радіуси вписаного та описаного кіл.

46. Дані координати трьох вершин трикутника (xl, yl), (x2, y2), (x3, y3). Напишіть програму, яка знаходить його периметр і площу.

47. Дане ціле чотиризначне число. Використовуючи операції / і % напишіть програму, яка знаходить суму його цифр.

48. Дане ціле чотиризначне число. Використовуючи операції / і % напишіть програму, яка знаходить добуток його цифр.

49. Перевірити істинність висловлювання: "Квадратне рівняння $Ax^2+Bx+C=0$ з даними коефіцієнтами A, B, C має дійсні корені". Виведіть "1" якщо висловлювання істинне, і " 0", якщо ні.

50. Перевірити істинність висловлювання: "Дані числа x, y є координатами точки, що лежить в другій координатній чверті". Виведіть " " якщо висловлювання істинне, "0", якщо інакше.

9.1.3 Питання для самоконтроля

1. Чому мова Сі отримала таке поширене використання при програмуванні?
2. З чого складається будь-яка програма мовою Сі?
3. З якої функції починається програма мовою Сі?
4. Що входить до алфавіту мови Сі?
5. Яка відмінність оператора присвоювання від арифметичних операцій Паскаля?
6. Поясніть дію операцій інкремента і декремента.
7. Правила використання коментарів в мові Сі.
8. Які знаки логічних і порозрядних операцій в мові Сі ви знаєте?
9. Які типи даних оголошують змінні цілого типу? Приклади.
10. Які типи даних оголошують змінні дійсного типу? Приклади.
11. Як збільшити діапазон значень для цілого і дійсного типів?
12. Для чого призначений переліковний тип enum ?
13. Які ви знаєте символи перетворення в функції виведення printf?
14. Які ви знаєте керуючі символьні константи ?
15. Пояснити, як працює функція форматного введення.
16. Які символи перетворення функції printf допускаються в функції scanf?
17. Чому при форматному введенні масиву в функції scanf не використовується знак &?
18. Яка функція дозволяє ввести тільки один символ?
19. Як працює тернарна операція в мові Сі?
20. Запишіть вираз мовою Сі: "Збільшити на одиницю вміст комірки пам'яті з адресою Z".
21. Запишіть вираз мовою Сі: "Збільшити значення за адресою x в 5 разів".
22. Які ви знаєте правила перетворення типів?
23. Поясніть пріоритет виконання операцій у мові Сі.

9 2 Лабораторна робота №2

9.2.1 Використання спеціальних символів і операторів мови Сі

Мета лабораторної роботи – навчитися вирішувати, тестувати і налагоджувати на ЕОМ типові задачі з використанням спеціальних символів

і операторів. Необхідно записати алгоритм, використовуючи основні конструкції мови Сі. Для перевірки правильності роботи програми повинні бути подані необхідні тестові вихідні дані.

Хід роботи :

1. Скласти алгоритм програми за індивідуальним варіантом, номер якого визначає викладач.
2. Скласти програму мовою Сі відповідно до розробленого алгоритму.
3. Протестувати програму з використанням нескладних даних і перевірити результат.

Вміст звіту

1. Навести умову і номер індивідуального варіанта.
2. Навести блок-схему розробленої програми та її описання.
3. Навести математичну модель розробленої програми.
4. Навести текст розробленої програми з коментарями.
5. Навести результати тестування програми.
6. Відповісти на контрольні запитання.

Теоретичні відомості

В теоретичних відомостях слід згадати, як вводити і виводити інформацію в мові Сі. Розглянемо чотири функції: printf, scanf, putchar і getchar. Перші дві призначаються для реалізації форматного введення і виведення даних.

Функція printf формально описується таким чином:

printf(" керуючий рядок", аргумент1, аргумент2...)

Керуючий рядок містить об'єкти трьох типів: звичайні символи, які просто виводяться на екран дисплею (копіюються в стандартий вихідний потік); специфікація перетворення, кожна з яких викликає виведення на екран значення чергового елемента зі списку, і керуючі символьні константи.

Кожна специфікація перетворення починається зі знака % і закінчується деяким символом, який задає перетворення. Між знаком % і символом перетворення може бути: знак мінус, який вказує, що перетворений параметр потрібно вирівняти вліво в своєму полі; рядок цифр, який задає мінімальний розмір поля; крапка, яка відокремлює розмір поля від рядка цифр; рядок цифр, який задає максимальне число символів, котрі треба ввести, або кількість цифр, які потрібно вивести справа від десяткової крапки в значеннях float або double.

Далі записується один із символів перетворення :

d – значенням аргумента є дійсне ціле число;

o – значенням аргумента є вісімкове ціле число;

x – значенням аргумента є шістнадцятіркове число;

c – значенням аргумента є символ;

s – значенням аргумента є рядок символів (виводяться до ознаки кінця

рядка);

e –значенням аргумента є дійсне десяткове число в експоненційній формі;

f – значенням аргумента є дійсне десяткове число з плаваючою комою;

q – використовується як %e або %f і вилучає виведення незначущих нулів;

p – значенням аргумента є покажчик (адреса).

Якщо після знака % записаний не символ перетворення, то він виводиться на екран.

Фунція printf використовує керуючий рядок для того, щоб вказати, скільки всього аргументів і їх типи. Аргументами можуть бути змінні, константи, вирази, виклики функцій, головне, щоб їх значення відповідали заданій специфікації. При наявності помилок (наприклад, в кількості аргументів або типів перетворення) результати будуть неправильними.

Серед керуючих символьних констант найчастіше використовуються такі:

\a – для короткочасної подачі звукового сигналу;

\b – для переведення курсора вліво на одну позицію;

\n – для перехода на новий рядок;

\r – для повернення каретки або переведення курсора в початок поточного рядка;

\t – для горизонтальної табуляції;

\v – для вертикальної табуляції.

Наприклад, в результаті запису інструкції виклику функції:

printf(" \t ЕОМ \n % d\n", ic);

спочатку виконується горизонтальна табуляція (\t), тобто курсор зміститься на початок краю екрана, потім на екран виведеться слово ЕОМ, після чого курсор перейде на початок нового рядка (\n), далі буде виведено ціле значення ic за форматом d і на закінчення курсор перейде на початок нового рядка (\n).

Функція scanf формально записується таким чином:

scanf (" керуючий рядок ", аргумент 1, аргумент 2, ...)

Аргументом scanf повинен бути покажчик на відповідні значення (перед іменем змінної записується символ &). Призначення покажчиків розглянемо далі. Scanf розташовує за адресою &і введене значення. В функції scanf допускаються такі символи перетворення функції printf.

Наприклад:

d – на вході очікується десяткове ціле число;

o – на вході очікується вісімкове ціле число;

x – на вході очікується шістнадцятіркове число;

u – на вході очікується поява беззнакового числа;
c – на вході очікується поява одиночного символу;
s – на вході очікується поява рядка символів;
f – на вході очікується поява дійсного числа;
p – на вході очікується поява покажчика (адреси).

Перед символами d, o , x , f може стояти буква l. В перших трьох випадках відповідні змінні повинні мати тип long, а в останньому – double.

Досить потужними операторами мови є оператори циклу. **Циклом** називається участок програми, який повторюється декілька разів. Цикли використовуються тоді, коли деякі дії треба виконати багато разів, кожний раз з новими даними. Кількість повторень визначається залежно від типу оператора заголовку циклу. Початок виконання циклу може виконуватись тільки через оператори заголовку циклу. Завершення циклу любого типу може бути:

–відповідно до умов, визначених заголовком циклу;

–за оператором goto – перехід на оператор поза тілом циклу;

–за оператором break – вихід із циклу або;

–за оператором return – вихід із функції.

В мові Сі є оператори циклу for, while і do-while.

Якщо кількість повторів попередньо відома, то доцільно використовувати оператор for, а якщо кількість повторів визначається умовами, то операторами while або do-while.

Форма оператора циклу for:

```
For (Сп. 1;  Сп 2;  Сп. 3)    //заголовок циклу
        S;                    //тіло циклу
```

Сп. 1 – список операторів, ініціюючих початкове значення, виконуються 1 раз до початку виконання тіла циклу, як правило, для встановлення початкових значень параметрів циклу;

Сп. 2 – список операторів і виразів для перевірки кінця циклу; кінець циклу звичайно визначається на основі аналізу значення параметрів циклу, виконується перед кожним використанням циклу; якщо значення останнього виразу сп. 2 істинно (!= 0), цикл виконується, а якщо хибне (= 0) – завершується.

Сп. 3 – список операторів і (або) для корегування параметрів циклу; виконується після кожного виконання тіла циклу.

S – простий або складений оператор тіла циклу. Всі 3 списки в тілі циклу необов'язкові.

Правила використання оператора циклу for такі:

1. Сп.1 виконується 1 раз до початку виконання тіла циклу;
2. Після виконання тіла циклу виконується Сп.3 ;

3. Поява в любому місці тіла циклу оператора continue дає перехід;
4. Поява оператора break викликає перехід до оператора, який йде після оператора циклу.

Після виходу із цикла за оператором break або goto параметр циклу зберігає значення, при якому завершується цикл. Після нормального завершення циклу (не за goto) значення параметра циклу дорівнює значенню, яке привело до завершення виконання циклу. Наприклад, якщо межа змінних значень параметрів циклу визначена в вигляді:

for (i =1; i <5; i++) ,

то для значень і від 1 до 4 виконуються оператори тіла циклу, а при значенні

i = 5 виконання циклу завершується.

Якщо немає Сп. 1 або Сп. 2, їх крапка з комою(;) повинна залишитись в операторі заголовка циклу; наприклад for(; ;) for(i =1; ; i++), – це безкінечні цикли, з яких треба вийти за допомогою операторів break, return, goto на мітку зовні циклу.

Типовий приклад оператора циклу:

for (i =1; i<20; i++)

і – параметр змінюється від 0 до 19 з кроком 1, тіло циклу виконується 20 разів.

За допомогою скороченої форми оператора for можливо реалізувати в програмі тимчасову затримку процеса виконання програми. Наприклад, для обмеження часу чекання відповіді користувача.

Приклад циклу з пустим оператором S:

for (n=1; n<=10000; n++);

Цей цикл рахує значення від 1 до 10000, нічого більше не роблячи. Символ ; після заголовка циклу – це пустий оператор.

Будь-який список заголовка може містити операцію "кома" , тобто декілька операторів і виразів, розділених комами, які виконуються зліва направо. Таким чином, оператор циклу for може мати якби декілька параметрів циклу, які змінюються синхронно (одночасно). Найчастіше це списки Сп.1 і Сп.3 – оператори присвоювання або звертання до функцій, а Сп.2 – містить вирази відношення або логічні вирази. Якщо немає Сп.1 і Сп.3, то параметр циклу немов не розглядається.

Якщо немає Сп.2 (для нього залишиться ;), то вважається, що умова перевірки кінця циклу істинна, при цьому цикл не може бути завершений за умовою циклу в заголовку циклу (безкінечний цикл), а може бути завершений тільки за оператором goto (перехід на оператор зовні тіла циклу), за допомогою операторів break або return.

Приклад: Визначити суму парних чисел від 1000 000 до 0

for (s=0.0, i=1000 000; i>=2; i- =2) s+=i;

Ініціюються початкові значення, після кожного разу крок параметра циклу зменшується на 2. Після завершення i=0.

Приклади розв'язання типових задач

Компанії необхідно передавати дані по телефону, але є небезпека, що телефони можуть прослуховуватись. Всі дані передаються в вигляді чотиризначних цілих чисел. Представники компанії запропонували написати програму для шифрування даних, щоб зробити їх передачу більш надійною. Програма повинна зчитувати чотиризначне ціле число і шифрувати його таким чином: "Замінити кожну цифру значенням добутку цієї цифри і 7 по модулю 10. Потім поміняти першу цифру і третю, другу і четверту..."

```
#include <stdio.h>
#include <conio.h>
void main(void)
{
 clrscr();
 int i;
 int n,tmp;
 int tmp1[5];
 textmode(C40);
 printf("Введіть чотиризначне ціле число\n -> ");
 scanf("%d",&n);
 if(n>=10000){ printf("Число містить більше знаків, ніж 4"); goto a;}
 if(n<1000)  { printf("Число містить меньше знаків, ніж 4"); goto a;}
 tmp=n;
 for (i=4; i>=1; i--)
 {
    tmp1[i]=tmp%10;
    tmp/=10;
    tmp1[i]=(tmp1[i]+7)%10;
 }
 printf("Невідсортоване число ->");
 for ( i=1; i<=4; i++ )   printf("%d", tmp1[i]);
 for ( i=1; i<4; i++)
 {
      tmp=tmp1[i];
      tmp1[i]=tmp1[i+2];
      tmp1[i+2]=tmp;
      if(i==2) i=4;
 }
```

```
printf("\nВідсортоване число ->");
for(i=1; i<=4; i++)  printf("%d",tmp1[i]);
a:  getch();
}
```

9.2.2 Завдання для програмування

1. Перевірити істинність висловлювання: "Точка з координатами (x, y) лежить усередині прямокутника, ліва верхня вершина якого має координати (xl, yl), права нижня — (x2, y2), а сторони паралельні координатним осям". Виведіть "1" якщо висловлювання істинне, інакше виведіть "0".

2. Напишіть програму, яка буде в кожному рядку, що вводиться, заміняти символи пропусків і табуляцій, що стоять підряд, на один пропуск і видаляти порожні рядки.

3. Напишіть програму, що перетворить послідовність шістнадцятіркових цифр, що починається з 0x чи 0X, у відповідне ціле. Шістнадцятірковими цифрами є символи 0...9, a...f, A...F.

4. Перевірити істинність висловлювання : "Дане ціле число є непарним тризначним числом". Виведіть "1" якщо висловлювання істинне, інакше виведіть "0".

5. Перевірити істинність висловлювання : "Серед трьох даних цілих чисел є хоча б одна пара збіжних". Виведіть "1" якщо висловлювання істинне, інакше виведіть "0".

6. Перевірити істинність висловлювання : "Серед трьох даних цілих чисел є хоча б одна пара взаємно протилежних". Виведіть "1" якщо висловлювання істинне, інакше виведіть "0".

7. Перевірити істинність висловлювання : "Всі цифри даного тризначного числа різні". Виведіть "1" якщо висловлювання істинне, інакше виведіть "0".

8. Перевірити істинність висловлювання : "Цифри даного тризначного числа утворюють геометричну прогресію". Виведіть "1" якщо висловлювання істинне, інакше виведіть "0".

9. Дано натуральні числа n, m (n<=m). Визначити, скільки з чисел n, n+1, ..., m є номерами високосних років.

10. Дано натуральні числа a, b, c, котрі позначають число, місяць і рік, наприклад, 1.4.1901 – 1 квітня 1901 року. Одержати трійку чисел, відповідних наступному дню.

11. Дано натуральні числа a, y, c, що позначають число, місяць і рік. Перевірити коректность цієї дати (наприклад, 30 лютого 1900 року – некоректная дата). Знайти номер цього дня з початку року. Визначити скільки повних днів залишилося до кінця року.

12. Дано натуральні числа a, b, що позначають число і місяць. На який день тижня припадає ця дата, якщо рік – невисокосний, 1 січня цього року – середа?

13. Обчислити кількість п'ятниць, що припадають на 13-і числа XX сторіччя і сторіччя з номером n, де n – задане натуральне число.

14. Дано натуральні числа a, b, c, що позначають дату (число, місяць і рік) за Юліанським календарем. Одержати цю дату за сучасним календарем. Розбіжність між датами визначається тим, що в Юліанському календарі кожен рік, номер якого ділиться на 4, є високосним, і з цього правила немає ніяких виключень.

15. Дано натуральні числа a1, b1, c1, a2, b2, c2, що вказують дві дати (число, місяць і рік). Обчислити кількість днів, що пройшли між двома цими датами і кількість повних років, що пройшли між двома цими датами.

16. День учителя щорічно відзначається в першу неділю жовтня. Дано натуральне число n, яке позначає номер року. Визначити число, на яке в жовтні вказного року припадає День учителя.

17. Скласти програму, яка за номером дня від початку невисокосного (чи високосного) року обчислює номер місяця і номер дня в даному місяці.

18. У деякій бібліотеці останній четвер кожного місяця – санітарний день. Дано натуральне число n, що означає номер року. Одержати один по одному всі числа, на які в січні, лютому,..., грудні зазначеного року приходиться санітарний день.

19. Напишіть програму, що видаляє символ, який визначається користувачем, із вхідного потоку.

20. Напишіть програму, що друкує гістограму появи різноманітних введених символів у вхідному рядку.

21. Підрахуйте кількість рядків слів, символів у вхідному потоці, вважаючи за слово будь-яку послідовність символів, що не містить пропусків, символів табуляції або переходів на інший рядок.

22. Напишіть програму, що видаляє:

а) "хвостові пропуски" та символи табуляції в введеному рядку;

б) рядки, що складаються лише з одних пропусків;

в) рядки, що містять у собі лише однакові символи.

23. Напишіть програму, що визначає діапазон значень змінних типу int, char, long, short як signed, так і unsigned шляхом друку відповідних значень із стандартних header-файлів.

24. Дано дві змінні цілого типу: A і B. Якщо їх значення не рівні, то присвоїти кожній змінній суму цих значень, а якщо рівні, то присвоїти змінним нульові значення.

25. Дано дві змінні цілого типу: A і B. Якщо їх значення не рівні, то присвоїти кожній змінній максимальне з цих значень, а якщо рівні, то привласнити змінним нульові значення.

26. Дано три змінні: X, Y, Z. Якщо їх значення впорядковані за спаданням, то подвоїти їх; інакше замінити значення кожної змінної на протилежне.

27. Дано три змінні: X, Y , Z. Якщо їх значення впорядковані за збільшенням або убуванню, то подвоїти їх; інакше замінити значення кожної змінної на протилежне.

28. Дано цілочислові координати точки на площині. Якщо точка не лежить на координатних осях, то вивести 0. Якщо координати точки збігаються з початком координат, то вивести "1". Якщо точка не в початку координат, але лежить на осі 0X або 0Y, то вивести відповідно "2" або "3".

29. Дано дійсні координати точки, що не лежить на координатних осях 0X і 0Y. Вивести номер координатної чверті, в якій знаходиться дана точка.

30. На числовій осі розташовані три точки: A, B, C. Визначити, яка з двох останніх точок (B або C) розташована ближче до A, і вивести цю точку та її відстань від точки A.

31. Дано номер деякого року (додатне ціле число). Вивести відповідний йому номер сторіччя, враховуючи, що, наприклад, початком 20 сторіччя був 1901 рік.

32. Дано номер деякого року (додатне ціле число). Вивести число днів цього року, враховуючи, що звичний рік налічує 365 днів, а високосний — 366 днів. Високосним вважається рік, що ділиться на 4, за винятком тих років, які діляться на 100 і не діляться на 400 (наприклад, роки 300, 1300 і 1900 не є високосними, а 1200 і 2000 — високосні).

33. Дано ціле число, що лежить в діапазоні від 1 до 9999. Вивести рядок – словесний опис даного числа вигляду "парне двозначне число", "непарне чотиризначне число" і т.д.

34. Дано номер місяця (1 – січень, 2 – лютий,...). Вивести назву відповідної пори року ("зима", "весна" і т.д.).

35. Дано номер місяця (1 – січень, 2 – лютий,...). Вивести число днів цього місяця для невисокосного року.

36. Напишіть програму, яка залежно від введеної оцінки за п'ятибальною шкалою виводить на екран: «погано», «незадовільно», ... Врахуйте, що оцінок менше 1 і більше 5 не буває.

37. Арифметичні дії з числами пронумеровані таким чином: 1 – додавання, 2 – віднімання, 3 – множення, 4 – ділення. Дано номер дії і два числа A і B (B не дорівнює нулю). Виконати над числами вказану дію і вивести результат.

38. Одиниці довжини пронумеровані таким чином: 1 – дециметр, 2 – кілометр, 3 – метр, 4 – міліметр, 5 – сантиметр. Дано номер одиниці довжини і

довжина відрізка L в цих одиницях (дійсне число). Вивести довжину даного відрізка в метрах.

39. Одиниці маси пронумеровані таким чином: 1 – кілограм, 2 – міліграм, 3 – грам, 4 – тонна, 5 – центнер. Дано номер одиниці маси і маса тіла M в цих одиницях (дійсне число). Вивести масу даного тіла в кілограмах.

40. Робот може переміщатися в чотирьох напрямах ("Пв" – північ, "З" – захід, "Пд" – південь, "С" – схід) і приймати три цифрові команди: 0 – продовжувати рух, 1 – поворот наліво, 2 – поворот направо. Дано символ B – початковий напрям руху робота і число N – послана йому команда. Вивести напрям робота після виконання одержаної команди.

41. Локатор орієнтований на одну із сторін світу ("Пв" – північ, "З" – захід, "Пд" – південь, "С" – схід) і може приймати три цифрові команди: 0 – поворот наліво, 1 – поворот направо, 2 – поворот на 180 градусів. Дано символ B – початкова орієнтація локатора і числа N1 і N2 – дві послані йому команди. Вивести орієнтацію локатора після виконання даних команд.

42. Елементи кола пронумеровані таким чином: 1 – радіус (R), 2 – діаметр (D), 3 – довжина (L), 4 – площа круга (S). Дано номер одного з цих елементів і його значення. Вивести значення решти елементів даного кола (у тому ж порядку). Як значення Pi використовувати 3.14.

43. Елементи рівностороннього трикутника пронумеровані таким чином: 1 – сторона (a), 2 – радіус вписаного кола (R1), 3 – радіус описаного кола (R2), 4 – площа (S). Дано номер одного з цих елементів і його значення. Вивести значення решти елементів даного трикутника (у тому ж порядку).

44. Дано два цілі числа: D (день) і M (місяць), що визначають правильну дату невисокосного року. Вивести значення D і M для дати, наступної за вказаною.

45. Дано ціле число в діапазоні 100 - 999. Вивести рядок - словесний опис даного числа, наприклад: 256 – "двісті п'ятдесят шість", 814 – "вісімсот чотирнадцять".

46. Дано два цілі числа A і B (A < B). Вивести всі цілі числа, розташовані між даними числами (включаючи самі ці числа), у порядку їх зростання, а також кількість N цих чисел.

47. Дано два цілі числа A і B (A < B). Вивести всі цілі числа, розташовані між даними числами (не включаючи самі ці числа), у порядку їх зменшення, а також кількість N цих чисел.

48. Дано дійсне число A і ціле число N (> 0). Вивести A в степені N:

$A^N = A \times A \times ... \times A$ (числа A перемножуються N раз).

49. Вивести на екран числа, квадрат яких не перевищує заданого числа N. Використовуйте цикл for.

50. Дано дійсне число A (> 1). Вивести найбільше з цілих чисел N, для яких сума $1 + 1/2 + ... + 1/N$ буде меншою A, і саму цю суму.

9.2.3 Питання для самоконтролю

1. Які основні характеристики даних типу float, int, char?
2. Поясніть призначення і форму оператора присвоювання.
3. Яке призначення виразів і які типи виразів ви знаєте?
4. Яке призначення арифметичного виразу?
5. Що може бути операндом арифметичного виразу?
6. Назвіть арифметичні операції у порядку зменшення їх пріоритету.
7. У якій послідовності виконуються операції арифметичного виразу?
8. У яких конструкціях мови Сі можна використовувати арифметичний вираз?
9. Яке призначення оператора printf і його параметрів?
10. Що може бути елементом списку аргументів оператора printf?
11. Яке призначення операторів gets, scanf і їх параметрів?
12. Як вивести (видалити) значення змінної програми у вікно перегляду Watch?
13. Як перейти з вікна редактора у вікно перегляду (виведення результатів) назад?
14. Як змінити розмір вікна: редактора, перегляду, виведення результатів?
15. Як виконати програму порядково, до заданого рядка, до кінця?
16. Яке призначення виразів: відношення, логічних; що є їх результатом?
17. Назвіть операції різних типів у порядку зменшення їх пріоритету.
18. У якій послідовності виконуються операції змішаних виразів?

9.3 Лабораторна робота №3

9.3.1 Складання програм на опрацювання масивів

Мета лабораторної роботи – навчитися вирішувати,тестувати і розв'язувати на ЕОМ типові задачі, які працюють з масивами. Необхідно записати алгоритм роботи програми, використовуючи основні конструкції мови Сі, покажчики. Для перевірки правильності роботи програми повинні бути подані необхідні тестові дані.

Хід роботи :

1. Скласти алгоритм програми за індивідуальним варіантом, номер якого визначає викладач.
2. Скласти програму мовою Сі відповідно до розробленого алгоритму.
3. Протестувати програму з використанням нескладних даних і перевірити результат.

Вміст звіту

1. Навести блок-схему розробленої програми та її описання.
2. Навести математичну модель розробленої програми.

3. Навести текст розробленої програми.
4. Навести результати роботи програми.
5. Відповісти на контрольні запитання.

Теоретичні відомості

Масив є структурованим типом даних. Масив — це сукупність елементів одного типу, які використовуються в програмах під одним ім'ям. Кожен масив має ім'я, яке повинно відповідати тим самим правилам, що імена зміних.

Доступ до окремих елементів масиву здійснюється за іменем масиву та індексом (порядковим номером) елемента, який вказує відносну позицію елемента. Індекс — це число, за допомогою якого розрізняються елементи масиву. Елементи — це окремі змінні в масиві. Класичними прикладами масивів є вектор і матриця. Основні властивості масиву:

- всі елементи масиву мають один і той самий тип;
- усі елементи масиву розташовані в пам'яті один за одним. Індекс першого елементу дорівнює нулю;
- ім'я масиву є вказівником-константою, що дорівнює адресі початку масиву (першого байта першого елемента масиву).

Для роботи з масивом у програмі необхідно за аналогією з простими змінними зробити його оголошення на початку головної функції чи блоку. Формат оголошення масиву є таким:

<тип даних> <ім'я масиву> [розмірність масиву];

Тип даних при описанні масивів задає тип елементів масиву і вибирається за тими ж правилами, що і для простих змінних.

Розмірність масиву — це число індексів, що використовуються для посилання на конкретний елемент масиву. Розмірність масиву описується для кожного індексу окремо. Вона визначає кількість значень кожного індексу і повинна бути задана константою, вказаною явно в квадратних дужках, або повинна бути визначена за допомогою директиви #define для автоматичних змінних. Нумерація елементів масиву починається з нуля тому, якщо в одновимірному масиві 50 елементів, то верхня межа зміни індексу дорівнює 49. Тобто перший елемент масиву array означається як array[0], другий array[1], останній array[49]. Якщо двовимірний масив має 10 рядків і 10 стовпців, то індекс рядка набуватиме верхнього значення індексу, тобто 9, і стовпця — 9.

Приклад оголошення масивів:

```
#define k 10
int main()
{ float a[k];      // оголошення масиву a з 10 елементів типу float
  int b[10][10]; //оголошення двовимірного масиву зі 100 елементів цілого
```

```
                    // типу
int c[9];           // оголошення одновимірного масиву з 9 елементів цілого
                    // типу ....
```

Сі підтримує багатовимірні масиви. Найпростішим видом багатовимірного масиву є двовимірний масив, який можна подати як масив одновимірних масивів. Двовимірний масив є матрицею, де перший індекс відповідає за рядок, а другий – за стовпець. Кожна розмірність масиву береться в окремі квадратні дужки. Багатовимірний масив оголошується таким чином:

<тип даних><ім'я масиву>[розмірністьN]...[розмірність2]
[розмірність 1];

Елементи багатовимірного масиву зберігаються в пам'яті в порядку зростання найправішого індексу, тобто по рядках. Це означає, що правий індекс змінюється швидше лівого, якщо переміщатися масивом у порядку розташування елементів в пам'яті. Перед тим як використати масив, необхідно присвоїти значення його елементам. У Сі масиви не ініціалізуються і не обнулюються автоматично. Елементам масиву можна задати початкові значення трьома способами: ініціалізацією, присвоєнням або введенням. При описанні масиву може бути виконана ініціалізація елементів масиву.

Є два методи ініціалізації:

- ініціалізіція за замовчуванням. Якщо не проводиться ініціалізація елементів масиву, то всі елементи статичних та зовнішніх масивів ініціалізуються компілятором нулями, а елементи автоматичних і регістрових масивів розташовуються на неочищених участках пам'яті;
- явна ініціалізіція елементів. Після описання масиву можна записати список початкових значень масиву, які беруться у фігурні дужки.

Наприклад, ініціалізіція зовнішнього масиву з 10 елементів:

```
int array[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
```

Якщо оголошений масив b з 10 елементів

```
int b [10] = {1, 2, 3, 4};
```

то перші чотири елементи масиву ініціалізувати числами 1, 2, 3 і 4. Значення інших шести елементів або дорівнює 0, якщо масив зовнішній чи статистичний, або не визначено, якщо масив автоматичний чи регістровий. При використанні автоматичних масивів програміст повинен присвоїти нульове початкове значення принаймні першому елементові для того, щоб автоматично були обнулені елементи, що залишилися. При ініціалізації багатовимірних масивів можна додати фігурні дужки навколо кожного вимірювання. Наприклад,

```
int  d [2][3] = { {1,2,3},{4,5,6} };
```

Якщо даних рівно стільки, скільки має бути при заповненні всіх елементів, то фігурні дужки в списку можуть бути опущені, тоді такі два записи еквівалентні:

```
int f [2][2] = { {1, 2}, {3, 4} };        int f [2][2] = {1, 2, 3, 4};
```

Масиву символів можна задавати початкові значення, використовуючи рядкову константу. Наприклад, оголошення

```
char c[ ] = {"ABCD"};
```

присвоює елементам масиву с початкові значення як окремі символи рядка "ABCD". Розмір масиву с визначається на основі довжини рядка плюс нульовий символ закінчення рядка, таким чином масив містить п'ять елементів. Символьний масив можна задати окремими символьними константами. Попереднє оголошення еквівалентне такому:

```
char c[ ] = {'A', B', 'C, 'D', '\0'};
```

Для багатовимірних масивів необхідно визначити всю розмірність, крім найлівішої, компілятор визначає число елементів за числом значень у списку ініціалізації. Наприклад, int array [ ] [3] = {0,1,2,3,4,5};
тобто кожен рядок міститиме три елементи, нульовий — 0, 1, 2, перший — 3, 4, 5. Якщо потрібно проініціалізувати не всі елементи, в списку використовуються фігурні дужки, наприклад:

```
int array [ ][3] = { {0}, {3,4} };
```

m [0][0] дорівнюватиме 0, m [1][0] — 3, m [1][1] — 4.
Використовуючи операцію присвоювання, можна в програмі присвоїти значення кожному елементу масиву або, використовуючи оператор циклу і функції введення, можна вводити значення елементів з клавіатури.

Доступ до елементів масиву може виконуватися за допомогою операції індексації або за допомогою механізму покажчиків.

У мові Сі дозволяється лише поелементне звертання до масиву, при якому поточне значення може бути задане константою, змінною чи виразом. Наприклад,

```
int a[9]; a[0] = 0; a[1] = 0; a[2] = 0; a[3] = 0; a[4] = 0; a[5] = 0;
          a[6] = 0; a[7] = 0; a[8] = 0;
```

У даному прикладі обнуляються елементи масиву а. Операції з багатовимірними масивами зручно виконувати, використовуючи оператор циклу for. Наступний фрагмент програми показує як обнулити елементи двовимірної матриці.

```
int    b [9][9], i, j;
 for ( i = 0; i < 9; i++ )
    for(j = 0; j<9; j++ )
        a[i] [j] = 0;
```

Інший спосіб доступу до елементів масиву — використання механізму

покажчиків. Ім'я масиву, що використовується без наступних [ ], є адресою початку масиву. Оскільки ім'я масиву — це покажчик-константа на перший байт першого елемента масиву, то, використовуючи операцію отримання значення за адресою (*), можна виконати доступ до будь-якого елемента масиву. Тобто будуть еквівалентними запис до посилання на 1-й елемент масиву array[i] і *(array+i).

Нехай є опис масиву int a[5], який містить 5 елементів: a[0], a[1], a[2], a[3], a[4]. Адреса і-го елемента масиву дорівнює сумі адреси початкового елемента масиву і зсування цього елемента на і одиниць від початку масиву. Якщо ра — це покажчик на ціле int *ра; , то після виконання оператора ра=&a[0]; ра містить адресу елемента a[0]. Вираз ра+1 вказує на наступний елемент, ра+і вказує на і-й елемент масиву, тобто є адресою a[i], тоді *(ра+і) є вмістом і-го елемента. Оскільки ім'я масиву ототожнюється з адресою його першого елемента, то оператор ра = &a[0]; еквівалентний оператору ра = а; , тому будуть еквівалентними запис до посилання на і-й елемент масиву a[i] і *(a+i). Будь-який масив і індексний вираз можна подати за допомогою покажчика. В той самий час між ім'ям масиву й відповідним покажчиком є істотна відмінність. Треба пам'ятати, що покажчик — це змінна, а ім'я масиву — константа. Тому ра = а; і ра++; допустимі операції. Оператори вигляду а = ра; а++; використовувати не можна, оскільки значення константи постійне і не може бути змінене. Якщо до покажчика додається ціле, компілятор автоматично маштабує ціле, множачи його на число байтів, відповідне типу, зазначеному в оголошенні покажчика.

При роботі з двовимірними масивами також можна використати покажчики. У цьому разі точкою відліку може бути як найперший елемент масиву, так і перший елемент кожного з рядків. Нехай є оголошення

```
int array[4] [2];    // масив array типу int з 4 рядків і 2 стовпців
int *ра;             // покажчик ра на цілий тип
ра=&array[0][0]; // покажчику ра присвоїти адресу першого елемента
                     масиву a[0][0]
```

ра=&array[0][0]; можна записати ра=array; тобто array = =&a[0][0];.

Двовимірний масив розташовується в пам'яті подібно до одновимірного масиву, за рядками, спочатку перший рядок, потім другий, третій і т. д., послідовно займаючи елементи пам'яті. Таким чином,

```
ра= = &array[0][0];
ра+1= = &array[0][1];
ра + 2 = = &array[1][0];
ра + 3 = = &array[1][1];
```

Тоді ра+6 вказуватиме на елемент array[3][0]. Якщо використовується покажчик на перший елемент рядка, то масив подається як масив масивів.

Масив array з попереднього прикладу має чотири рядки, кожен з яких є масивом з двох елементів. Ім'я першого рядка array[0], ім'я другого array[1] і т. д. Ім'я масиву є також покажчиком на цей масив, тому,

```
array [0] = = &array [0][0];
array [l] = = &array [l][0];
array [2] = = &array [2] [0];
array[3] = = &array[3][0];
```

Тоді доступ до елемента масиву, якщо використовується покажчик на перший елемент рядка, можна здійснити за допомогою виразу (array[i]+j), а щоб набути значення, розташованого за цією адресою, необхідно використати вираз *(array[i]+j). Якщо ім'я масиву використовується як адреса початку масиву покажчиків, то вирази *(*(array+i)+j) і array[i][j] є еквівалентними.

Елементи масиву можуть бути будь-якого типу, у тому числі і покажчиками. Масиви покажчиків можна використовувати для роботи з усіма типами даних, але найчастіше їх використовують для зберігання символьних рядків різної довжини. Оголошення масиву покажчиків виконується таким чином:

```
char *name[10]; // оголошення масиву покажчиків name з 10 елементів
```

Елемент оголошення char* вказує, що тип кожного елемента масиву name — покажчик на char. Масиви покажчиків можна ініціалізувати при оголошенні. Наприклад, char *name[ ]={"Понеділок", "Вівторок", "Середа", "Четвер", "П'ятниця", "Субота", "Неділя"}; Компілятор резервує місце для семи покажчиків. Вони набувають початкового значення, що дорівнює адресі початку в пам'яті відповідних рядків символів.

Приклад роботи з покажчиками. Нехай є одновимірний масив, який складається із N дійсних елементів. Необхідно обчислити

- додаток від'ємних елементів масиву;
- добуток елементів масиву, розташованих між максимальним і мінімальним елементами.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
int sum(int *m,int n);          // прототип функції пошуку суми
int *findmax(int *m,int n); // прототип функції пошуку максимума
int *findmin(int *m,int n); // прототип функції пошуку мінімума
long pr(int *n1,int *n2);     // прототип функції пошуку добутка

void main()
{
 int mas[10],i;
```

```
int *min, *max;
clrscr();
   // заповнення масива випадковими  числами
srand(2);
for(i=0; i<10; i++)
      {
      mas[i]=random(40)-20;
      printf("%d", mas[i]);
      }
min=findmin(mas,10);
max=findmax(mas,10);
printf("\nДодаток елементів масиву=%d\n", sum(mas,10));
printf("Мінімальний елемент масиву=%d,
        його номер=%d\n", *min, min-mas);
printf("Максимальний елемент масиву=%d, його номер=%d\n",
        *max,max-mas);
printf("Добуток елементів масиву між ними=%ld\n",
         min<max ? pr(min,max) : pr(max,min));
 printf("\nКінець  виведення. Натисніть будь-яку клавішу...");
 getch();
}
     // функція пошуку суми елементів масиву
int sum(int *m, int n)
{
   int i,s=0;
   for (i=0; i<n; i++, m++)
      if(*m>0) s+=*m;
   return s;
}
  // функція пошуку покажчика на максимальний елемент в масиві
int *findmax(int *m, int n)
{
  int i,*pmax=m++;
  for (i=1; i<n; i++,m++)
      if(*pmax<*m) pmax=m;
  return pmax;
}
  //функція пошуку покажчика на мінімальний елемент в масиві
int *findmin(int *m,int n)
```

```
{
  int i, *pmin=m++;
  for (i=1; i <n; i++, m++)
      if(*pmin>*m) pmin=m;
  return pmin;
}
  // добуток елементів масиву між min і mas
long pr(int *n1,int *n2)
{
  long p=1;
  int *i=n1;
  while(i<=n2)
      p*=*i++;
  return p; }
```

9.3.2 Завдання для програмування

1. Дано дійсну матрицю розміру n×(n+1), дійсні числа $a_1,\ldots,a_{n+1}$, $b_1, \ldots, b_{n+1}$, натуральні числа p, q ($p \le n$; $q \le n+1$). Утворити нову матрицю розміру (n+1)×(n+2) вставкою після рядка з номером p даної матриці нового рядка з елементами $a_1,\ldots,a_{n+1}$ і наступною вставкою після стовпця з номером q нового стовпця з елементами $b_1, \ldots, b_{n+1}$.

2. Дано дійсні числа $a_1,\ldots,a_n$, дійсну квадратну матрицю порядку n ($n \ge 6$). Одержати дійсну матрицю розміру n×(n+1), вставивши у вихідну матрицю між п'ятим і шостим стовпцями новий стовпець з елементами $a_1,\ldots,a_n$.

3. Дано дві дійсні квадратні матриці порядку n. . Одержати нову матрицю:

а) добутком елементів кожного рядка першої матриці на найбільше зі значень елементів відповідного рядка другої матриці;

б) додаванням до елементів кожного стовпця першої матриці добутку елементів відповідного рядка другої матриці.

4. Назвемо припустимим перетворенням матриці перестановку двох рядків або двох стовпців. Дано дійсну квадратну матрицю порядку n. За допомогою допустимих перетворень домогтися того, щоб:

а) один із елементів матриці, що є найбільшим за модулем значенням, розташувався в лівому верхньому кутку матриці;

б) один із елементів матриці, що є найменшим за модулем значенням, розташувався в лівому нижньому кутку матриці.

5. В даній дійсній матриці порядку n знайти найбільший за модулем елемент. Одержати квадратну матрицю порядку n-1 шляхом викидання з

вихідної матриці рядка і стовпця, на перетині яких розташований елемент зі знайденим значенням.

6. Дано квадратні матриці A і B порядку n. Одержати нову матрицю C=A×B – B×A.

7. Дано квадратну матрицю A порядку n і вектор b з n елементами. Одержати вектор: $A^2 \times b$ і вектор (A-E) × b, де E – одинична матриця порядку n.

8. Дано квадратну матрицю A порядку n, вектори X і Y з n елементами. Одержати вектор A×(X + Y).

9. Дано квадратні матриці A і B порядку n. Одержати матрицю A×(B - E)+C, де E – одинична матриця порядку n, а елементи матриці C обчислюються за формулою

$$C_{ij} = \frac{1}{i+j}, \quad i, j = 1,2,\ldots,n...$$

10. Права трикутна матриця A порядку n задана у вигляді послідовності (n +1) n /2 чисел: спочатку йде n елементів першого рядка, потім n— 1 елемент другого рядка, починаючи з другого елемента, і т.д. (з останнього, n-го рядка береться тільки n-й елемент). Крім цієї послідовності дано вектор b з n елементами. Знайти вектор A×b.

11. Відстань між k-м і j-м рядками матриці $A = \|a_{ij}\|$, визначається як $\sum_{J=1}^{N} |a_{kj}| \times |a_{ij}|$. Указати номер рядка, максимально видаленого від першого рядка заданої матриці.

12. Для заданої перестановки A чисел 1, …, 100 знайти максимальне $k \geq 1$, при якому $A^k(i) = i$ для всіх $1 \leq i \leq 100$.

13. Визначити норму заданої матриці $A = \|a_{ij}\|$, тобто максимальне число із сум абсолютних значень кожного рядка матриці.

14. За заданою квадратною матрицею розміром 10×10 побудувати вектор довжиною 19, елементи якого — максимуми елементів, діагоналей, рівнобіжних головний діагоналі.

15. Два рядки матриці назвемо схожими, якщо збігаються множини чисел, що зустрічаються в цих рядках. Знайти кількість схожих рядків у максимальній множині попарно несхожих рядків заданої матриці.

16. Характеристикою рядка цілочислової матриці назвемо суму її додатних парних елементів. Переставляючи рядки заданої матриці, розташувати їх відповідно до росту характеристик.

17. Для заданої цілочислової матриці знайти максимум серед сум елементів діагоналей, рівнобіжних головний діагоналі матриці.

18. Для заданої цілочислової матриці знайти мінімум серед сум модулів елементів діагоналей, рівнобіжних побічний діагоналі матриці.

19. Говорять, що матриця має сідлову точку a_{ij}, якщо a_{ij} є мінімальним у i-му рядку і максимальним у j-му стовпці. Знайти номер рядка і стовпця будь-якої сідлової точки заданої матриці.

20. Знайти максимальний серед всіх елементів тих рядків заданої матриці, що впорядковані (або за зростанням, або за спаданням).

21. Підрахувати кількість стовпців заданої цілочислової матриці розміром 20×20, що складені з попарно різних чисел.

22. Підрахувати кількість рядків заданої цілочислової матриці розміром 20×20, що є перестановкою чисел 1, 2, ..., 20.

23. Серед стовпців заданої цілочислової матриці, що містять тільки такі елементи, що за модулем не більше 10, знайти стовпець з мінімальним добутком елементів.

24. Визначити, чи є лінійно незалежними три заданих вектори цілих чисел довжиною 30.

25. Елемент матриці називається локальним мінімумом, якщо він строго менший всіх наявних сусідніх елементів. Підрахувати кількість локальних мінімумів заданої матриці розміром 10×13.

26. Дано дійсні числа a і b (a<b). Сформувати матрицю P(15,20), елементами якої є дійсні випадкові числа, рівномірно розподілені на відрізку [a,b]. Знайти в матриці два найменших за модулем елементи.

27. У цілочисловому масиві MP(100) непарні елементи збільшити в 2 рази, а у елементів з парними номерами змінити знаки на протилежні.

28. В магазині стоїть черга з N людей. Час обслуговування i-го покупця t_i – випадкова величина, розподілена за законом рівномірної густини в інтервалі [2.5, 10.4]. Одержати $i_1, i_2, \ldots i_N$ час перебування в черзі кожного покупця. Вказати номер тієї людини, для обслуговування якої потрібно мінімальний час.

29. У заданому цілочисловому масиві роздрукувати ті елементи, порядкові номери яких – числа Фібоначчі.

30. Серед N введених дійсних чисел x_i роздрукувати ті числа, які задовольняють умові $|x_i| < i2$. Якщо таких чисел немає – видати повідомлення.

31. Сформувати цілочисловий масив A(N), елементами якого є випадкові числа з діапазону [-8, 10]. Знайти серед його елементів два, модуль різниці яких має найбільше значення.

32. Дано дійсні числа a і b (a>b). Сформувати матрицю XY[17,20], елементами якої є дійсні випадкові числа, рівномірно розподілені на відрізку [a,b]. Визначити суму елементів, номери рядків яких кратні 3, а стовпців – 4.

33. Сформувати цілочисловий масив A(120), елементами якого є випадкові числа з діапазону [-2..3]. Визначити, скільки разів в ньому зустрілися два нульові елементи, що йдуть підряд.

34. Дано цілочисловий масив S[26]. Сформувати матрицю A, перший рядок якої міститиме елементи масиву з парними номерами, а другий – з непарними.

35. Сформувати дійсний масив A1[75], елементами якого є випадкові числа з діапазону [16..53]. Переслати з нього в масив A2 всі елементи, значення яких більше 25,8 і менше 34,7.

36. Дано дійсні числа a і b (a<b). Сформувати матрицю P(15,20), елементами якої є дійсні випадкові числа, рівномірно розподілені на відрізку [a,b]. Знайти в матриці два якнайменших за модулем елементи.

37. Серед стовпців заданої цілочислової матриці, що включають тільки такі елементи, які за модулем не більше 10, знайти стовпець з мінімальною сумою елементів.

38. Задано одновимірний масив. Описати програму, що визначає індекс елемента, значення котрого найближче до середнього арифметичного значення елементів цього масиву.

39. Задано масив X[i], i=1..n, елементами якого є нулі, одиниці та двійки. Переставити елементи масиву так, щоб масив починався нулями, далі йшли одиниці, а потім двійки. Додатковий масив не використовувати.

40. Вказати довжину такого початкового відрізку заданої послідовності чисел, для яких відношення степенів двійки і чисел Фібоначчі, що зустрічаються в ньому, максимальне.

41. У заданій матриці замінити k-й рядок та i-й стовпець нулями, крім елемента, що знаходиться на їх перетині.

42. Матриця розміщена в одновимірному масиві по рядках. Видалити k-й рядок матриці(k задано) з одновимірного масиву. Результат надрукувати по рядках.

43. Задані матриця та вектор. Отримати їх добуток. Надрукувати в рядок.

44. Додати елементи стовпців заданої матриці розміром 4×3. Результат отримати у одномірному масиві розміром 3.

45. Задано квадратну матрицю. Видалити з неї рядок і стовпець, на перетині яких міститься максимальний елемент головної діагоналі.

46. Перевірте, чи в заданій цілочисловій матриці 17×17 суми елементів у всіх рядках та стовпцях рівні між собою.

47. Дано дійсні числа a і b (a<b). Сформувати матрицю X(10,10), елементами якої є дійсні випадкові числа, рівномірно розподілені на відрізку [a,b]. Знайти в матриці рядок з мінімальним елементом і поміняти її місцями з першим рядком.

48. Дано квадратну матрицю розмірності k<2. Знайти суму її елементів, що знаходяться на побічній діагоналі.

49. Сформувати цілочисловий масив A(7), елементами якого є випадкові числа з діапазону [-5, 20]. Знайти серед його елементів два, різниця між якими має найбільше значення.

50. Проведено вимірювання зросту 70 студентів. Дані записані в масиві ROST. Розмістити в масиві NR номери тих студентів, чий зріст менше 180 см, і підрахувати число таких студентів.

9.3.3 Питання для самоконтролю

1. Поясніть оператори оголошення масивів рядкових і дійсних даних.
2. Що таке масив і як він оголошується в мові Сі?
3. У якій послідовності розташовуються в ОП елементи одновимірних масивів і який об'єм ОП вони займають?
4. Що таке покажчик? Що зберігається в покажчику?
5. Що таке операція непрямого посилання? Операція узяття адреси? Як вони застосовуються?
6. Чому важливий тип даних покажчика?
7. Що таке константний покажчик? Наведіть приклад.
8. Які арифметичні дії можна робити над покажчиками? Для чого вони застосовуються?

9.4 Лабораторна робота № 4

9.4.1 Функції, визначені користувачем

Мета роботи. Навчитися працювати з функціями. У запропонованих задачах потрібно логічно незалежні послідовності дій оформити у вигляді окремих функцій, до яких звертатися з функції main(). У задачах, що працюють з масивами, необхідно передбачити можливість ручного введення масиву, а також заповнення його за допомогою датчика випадкових чисел. Хід роботи і оформлення звіту таке ж, як і в попередніх роботах.

Теоретичні відомості

Функції – це фундаментальні конструкції Сі, пов'язані з розв'язанням поставленої задачі або її частини. Функція – це самостійна одиниця програми. Функції для вирішення окремого класу задач можна виділити в окремий файл. Цей файл може бути бібліотекою користувача.

Бібліотеки функцій дозволяють:

- використовувати одні й ті самі функції різними програмістами;
- збільшити структурованість і наочність програми;
- полегшити читання, освоєння і корегування використовуваних програм.

Будь-яка програма на мові Сі містить одну головну функцію з іменем main і будь-яку кількість інших функцій.

Визначення функції – це її текст на мові Сі. В ньому визначається ім'я, формальні параметри, оператори тіла функції і тип повертаного результату. Кожна функція повинна мати текст функції (заголовок і тіло), може мати оголошення функції і повинна мати виклик функції для її виконання. Структура кожної функції ідентична структурі головної функції main. Визначення функції містить заголовок і тіло функції.

Тіло функції – це блок. В ньому можуть бути оголошення змінних і повинні бути оператори тіла функції. В файлі функції або в її тілі можуть бути оголошення зі специфікатором extern, які забезпечують посилання на зовнішні змінні, розташовані в інших файлах. За замовчуванням всі функції зовнішні. При визначенні функції допускається клас пам'яті static, якщо треба щоб функція використовувалась тільки в даному програмному файлі. Для використання зовнішніх нестатичних функцій з інших файлів вони повинні бути оголошені в цих файлах як зовнішні, із специфікатором extern. Наприклад: extern int fun( );

Структура заголовка функції:
[специфікатор класу пам'яті] [тип] ім'я функції ([список формальних параметрів]), де

- специфікатор класу пам'яті – специфікатор, який визначає клас пам'яті функції;
- тип – визначає тип результату, який повертає функція за допомогою оператора return; тип- це ім'я одного з допустимих простих типів даних;
- ім'я функції – ідентифікатор, за допомогою якого функція викликається для виконання;
- список формальних параметрів – визначає типи й імена формальних параметрів-змінних функції, за допомогою яких відбувається обмін даними між функцією, що викликається, та функцією, що викликає у процесі виконання програми.

Елементи заголовка функції в квадратних дужках – необов'язкові.

Тип результату функції, який повертається за допомогою return, може бути будь-яким простим типом: арифметичним, символьним, переліковим, покажчиком на будь-який допустимий тип (в тому числі на скаляр, масив, структуру, файл або функцію). Якщо у заголовку функції не вказаний її тип і вона повертає результат за допомогою return, то за замовчуванням тип дорівнює int.

Елементи списку формальних параметрів відокремлюються комами. Кожний елемент списку формальних параметрів – це оголошення одного формального параметра функції у вигляді:

ім'я типу ім'я формального параметра

Приклади заголовків функцій:

- int f1(int a, float b) – функція повертає за допомогою оператора return значення типу int; a і b – параметри;
- float f2(int c, char *d) – функція повертає за допомогою оператора return значення типу float; c і d – параметри;
- void print( ) – заголовок функції без параметрів, не повертає ніякого значення за допомогою функції return.

Виконання функції продовжується до тих пір, поки не зустрінеться оператор return або не виконаються всі оператори тіла функції. В тілі функції може бути, а може і не бути оператор return. Якщо функція повертає результат в точку виклику, то оператор return обов'язковий, а якщо не повертає, оператора return може і не бути, і виконання функції завершиться після закінчення тіла функції.

Якщо функція повертає значення результату за допомогою return, то таку функцію можна використовувати у виразах правої частини операторів присвоювання, а також у всіх виразах, де допустимі значення результату функції, в тому числі у списках фактичних параметрів інших функцій. Такі функції – аналог функцій в мові Паскаль. Але їх можна також викликати для виконання і не із виразів, аналогічно процедурам мови Паскаль. Наприклад:

y = cube (x); z=sin(x); print f (" % f \n", a+cos(x));

де cube, sin, printf, cos – імена функцій, які повертають результат за допомогою оператора return.

Якщо вказано, що функція повертає значення типу void, то це означає, що функція не формує і не повертає результат за допомогою return і її не можна використовувати у виразах. В тілі такої функції може бути нуль і більше операторів return, але без „виразу" після return. Результатом її виконання можуть бути значення, які передаються через список параметрів, значення глобальних змінних або виведення значень. Таку функцію треба викликати за іменем зі списком фактичних параметрів, як у Паскалі.

Приклад виклику функції, яка не повертає значення результату:

free(ptr); – вивільнення динамічно виділеної ОП.

В мові Сі можливо використовувати функції, які повертають результат за допомогою return і з допомогою списку формальних і фактичних параметрів. Наприклад:

i=fscanf(f,"%d %f",& a,&b); – через return

або

fscanf(f,"%d%f",&a,&b); – через список формальних і фактичних параметрів.

Приклад визначення функції:

```
# include <stdio.h>
# include <conio.h>
```

```
/* Заголовок функції визначає тип результату int, за замовчуванням
формальні і фактичні параметри  x і y – типу int */
  //----Функція обчислення суми двох значень;
  static add (int, int y)      // Заголовок функції
  { return (x+y); }            //функція повертає x+y
   //----головна функція:
  void main ( )
  {
    clrscr( );
        print f (" %d \n", add(2,3));  // виклик функції
        print f (" \n Для завершення програми натисніть будь-яку
                        клавішу \n")
         getch( );
}
```

Приклади розв'язання типових задач

Приклад функції для знаходження добутку двох матриць:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#nclude <conio.h>
#define RND (rand0/32768.0)
#define MEMORY (n,m)  (float*) malloc((n)*(m) *sizeof (float))
// функція umn_m() – добуток матриць:  A(n,m) *B(m,p) = C(n,p)
void unm_m(float *a,  float *b,  float *c,  int n,  int m,  int p)
 {
   int i, j, k;
   float s;
   for(i=0; i<n; i++)
        for (j=0;  j<p;  j++)
         {
            for (s=k=0;  k<m;  k++)
                s+=* (a+i*m+k) ** (b+k*p+j);
                 *(c+i*p+j)=s;
         }
  }
// функція print_m() – виведення матриць
void print_m (char *name, float *a, int n, int m)
{
```

```
  int i, j;
  puts (name);
  for (i=0; i<n; i++, putchar( '\n' ) )
      for ( j=0; j<m; j++ )
          printf("%8.3f", *(a+i*m+j));
   putchar('\n');
}
void main ( )
 {
   int n, m, l, i, j;
   float *A, *B, *C;
   srand(2227);
m1:
    clrscr( ) ;
     printf("Введіть число рядків і стовпців матриці A : ");
     scanf("%d%d", &n,&m);
     printf("Введіть число стовпців матриці B : ");
     scanf("%d",&l );   // виділяємо пам'ять під матриці A, B та C
A=MEMORY(n,m);
B=MEMORY(m,l);
C=MEMORY(n,1) ; // формуємо і виводимо матриці A та B
  for(i=0; i<n; i++)
       for(j=0;  j<m;  j++)
          *(A+i*m+j) = RND*10-5;  // елементи A[i,j] – в  інтервалі (-5,5)
  print_m("\n\t Матриця A", A, n, m);
    for (i=0; i<m;  i++ )
       for(j=0;  j<l;  j++ )
        *(B+i*l+j )=RND*9-2;    // елементи B[i,j] – в  інтервалі (-2,7)
  print_m("\t Матриця B", B, m, l );
        // перемножуємо матриці:  C=A*B,   виводимо результат
  umn_m(A, B, C, n, m, l) ;
  print_m("\t Матриця C=A*B", C, n, l);
  free(A);   free (B) ;   free (C) ;    // звільняємо пам'ять
  printf("\n Продовжимо? Так – введіть 7  :  ");
  scanf("%d", &n);
  if (n=7)  goto m1;
}
```

Приклад функції, яка циклічно зсуває вправо на n бітів ціле число x.

```
#include <stdio.h>
#sinclude <conio.h>
  int rzsuv ( int x, int n)
{
   int y;
   y = (x>>n) + (x<<(16-n) );
   return y;
}
 void main ( )
 {
   int x,n,y;
   clrscr ();
   puts( "Введіть число і кількість бітів  ");
   scanf ("%d%d",&x,&n);
   y=rzsuv(x,n);
   printf("Число  x  зсунуте на 12 бітів вправо =  %d",n,y);
   getch();
 }
```

9.4.2 Завдання для программування

1. Дано натуральне число n. Роздрукувати число, що вийде після виписування цифр числа n у зворотному порядку. (Для одержання нового числа скласти функцію.)

2. Написати і протестувати функцію, що перетворить рядок вісімкових цифр в еквівалентне їй ціле десяткове число.

3. Дано дві квадратні матриці. Надрукувати ту з них, що має мінімальний "слід" (тобто суму елементів головної діагоналі). Використовувати функцію для знаходження сліду матриці і функцію виведення матриці.

4. Написати і протестувати функцію, що за заданим натуральним числом визначає кількість цифр у ньому і їхню суму.

5. Написати і протестувати функцію, що за заданим рядком Str формує новий рядок, що складається тільки з цифр, які входять у Str.

6. Написати і протестувати функцію, що обчислює $y = \sqrt[3]{x}$ $(0<|x|<2)$, використовуючи ітераційну формулу

$$Y_{i+1} = Y_i + \frac{1}{3}\left(Y_i - \frac{Y_i^4}{X}\right).$$

Початкове наближення $y_0 = x$. Ітерації припинити при $|y_{i+1} - y_i| < 2 \cdot 10^{-6}$.

8. З'ясувати, скільки простих чисел знаходиться в інтервалі $[n,m]$, і роздрукувати їх. Для визначення, чи є чергове число простим, скласти функцію.

9. Написати і протестувати функцію для знаходження в прямокутній матриці номера рядка, що має максимальну суму елементів.

10. Написати і протестувати функцію, що перетворить рядок двійкових цифр в еквівалентне їй ціле десяткове число.

11. Написати і протестувати функцію, що у рядку, переданому їй як параметр, замінює кожен другий елемент на заданий символ.

12. Написати і протестувати функцію для додавання і віднімання дійсних матриць. Одним з формальних параметрів повинна бути ознака виду операції.

13. Написати і протестувати функцію, що визначає, чи розташовуються букви в заданому символьному рядку за абеткою.

14. Дано дійсні числа a, b, c, d, e, f. Змінній s присвоїти значення 1, якщо обидва рівняння $ax^2 + bx + c = 0$ і $dx^2 + ex + f = 0$ мають дійсні корені і при цьому всі корені першого рівняння лежать між коренями другого рівняння. У іншому випадку зміній s присвоїти значення 0. (Для знаходження коренів квадратного рівняння використовувати функцію.)

15. Написати і протестувати функцію, що перетворить ціле без знака в його вісімкове символьне подання (бібліотечні функції для перетворення числа в рядок і формат виведення "%o" не використовувати).

16. Розробити функцію, що повертає найменше загальне кратне трьох заданих натуральних чисел.

17. Написати і протестувати функцію, що визначає, чи входить кожна літера в заданий рядок не більше двох разів.

18. Знайти натуральне число з інтервалу [n1,n2] з максимальною сумою дільників. (Для знаходження суми дільників числа використати функцію.)

19. Площа трикутника, заданого координатами своїх вершин, знаходиться за формулою

$$S = 0.5 \cdot \left| x_1 y_2 + x_2 y_3 + x_3 y_1 - x_1 y_3 - x_2 y_1 - x_3 y_2 \right|$$

Використовуючи функцію для обчислення площі трикутника, визначити площу опуклого чотирикутника ABCD, заданого координатами своїх вершин.

20. Поле шахової дошки визначається парою натуральних чисел, перше з який задасть номер вертикалі, а друге – номер горизонталі. Дано натуральні числа k, l, m, n. Потрібно, якщо можливо, з поля (k, l) одним ходом коня потрапити на поле (m, n). Якщо ні, то визначити, чи можна це зробити за два ходи. У випадку успіху вказати проміжне поле. Використати функцію.

21. Написати і протестувати функцію, що перетворить ціле без знака в його двійкове символьне подання (бібліотечні функції для перетворення числа в рядок не використовувати).

22. Написати і протестувати функцію, що переставляє в прямокутній матриці рядки в зворотному порядку.

23. Дано натуральні числа n і m. Написати і протестувати функцію, що повертає результат операції додавання двох чисел, утворених k молодшими цифрами числа n и k старшими цифрами числа m.

24. Написати і протестувати функцію, що перетворить рядок шістнадцяткових цифр в еквівалентне їй ціле десяткове число.

25. Написати функцію для обчислення значення

$$S(n) = \sum_{i=1}^{n} \frac{(2i)!}{(n+1)!}$$

Обчислити з її допомогою значення S(n) для n від 12 до 24 із кроком 4.

26. Дано прямокутну дійсну матрицю. Перевірити, чи упорядковані за неспаданням суми елементів рядків цієї матриці. Використати функцію для знаходження суми елементів рядків матриці.

27. Написати і протестувати функцію для визначення полярних координат точки за її прямокутними декартовими координатами. Залежність полярних і декартових координат:

$$r = \sqrt{x^2 + y^2}; \varphi = arctg(x/y).$$

28. Написати і протестувати функцію перестановки рядків матриці відповідно до вектора транспозиції.

29. Написати програму, яка знаходить максимальний і мінімальний елементи числового масиву. Обов'язково використати функцію.

30. Задано масив розміру N. Вивести спочатку його елементи з парними індексами, а потім – з непарними. Обов'язково використати функцію.

31. Задано цілочисловий масив A розміру 10. Вивести номер першого з тих його елементів A[i], які задовольняють подвійній нерівності: A[0]< A[i]< A[9]. Якщо таких елементів немає, то вивести 0. Обов'язково використати функцію.

32. Задано цілочисловий масив розміру N. Перетворити його, додавши до парних чисел перший елемент. Перший і останній елементи масиву не змінювати. Обов'язково використати функцію.

33. Замінити всі додатні елементи цілочислового масиву розміром 10 на значення мінімального. Обов'язково використати функцію.

34. Задано масив розміру 10. Переставити в зворотному порядку елементи масиву, розташовані між його мінімальним і максимальним елементами.

35. Задано масив розміру N. Здійснити циклічний зсув елементів масиву вліво на одну позицію. Обов'язково використати функцію.

36. Задано масив розміру N і число k (0 < k < 5, k < N). Здійснити циклічний зсув елементів масиву вправо на k позицій. Обов'язково використати функцію.

37. Перевірити, чи утворюють елементи цілочисельного масиву розміру N арифметичну прогресію. Якщо так, то вивести різницю прогресії, якщо ні – вивести 0. Обов'язково використати функцію.

38. Перевірити, чи утворюють елементи цілочисельного масиву розміру N геометричну прогресію. Якщо так, то вивести знаменник прогресії, якщо ні – вивести 0. Обов'язково використати функцію.

39. Задано масив ненульових цілих чисел розміру N. Перевірити, чи чергуються в ньому парні і непарні числа. Якщо чергуються, то вивести 0, якщо ні, то вивести номер першого елементу, що порушує закономірність. Обов'язково використати функцію.

40. Задано масив ненульових цілих чисел розміру N. Перевірити, чи чергуються в ньому додатні і від'ємні числа. Якщо чергуються, то вивести 0, якщо ні, то вивести номер першого елементу, що порушує закономірність. Обов'язково використати функцію.

41. Задано цілочисельний масив розміру N. Обнулити всі елементи масиву, що зустрічаються більше двох разів. Обов'язково використати функцію.

42. Дано число k (0<k< 11) і матриця розміру 4×10. Знайти суму і добуток елементів k-го стовпця даної матриці. Обов'язково використати функцію.

43. Дано квадратну матрицю порядку M. Замінити нулями елементи матриці, що лежать вище головної діагоналі. Обов'язково використати функцію.

44. Дано квадратну матрицю порядку M. Дзеркально відобразити її елементи щодо горизонтальної осі симетрії матриці. Обов'язково використати функцію.

45. Дано квадратну матрицю порядку M. Дзеркально відобразити її елементи щодо головної діагоналі матриці. Обов'язково використати функцію.

46. Дано два впорядкованих за збільшенням одновимірних масиви. Скласти з них третій масив, впорядкований за спаданням. Обов'язково використати функцію.

47. Дано квадратну матрицю розміром n×n. Знайти суми елементів, паралельних щодо головної діагоналі і визначити найбільшу з одержаних сум.

Обов'язково використати функцію.

48. Територія академмістечка задається прямокутною матрицею, яка складається з 0 і 1 , що відображає інформацію про територію. 1-якщо будова, 0-якщо зелене насадження. Написати програму, яка за заданою матрицею: підраховує площу, відведену під будови, якщо одиниця на плані відповідає 20м×20м=400кв.м; визначити загальний периметр зеленого насадження, якщо сторона однієї клітинки рівна 20м. Обов'язково використати функцію.

49. Знайти середні арифметичні значення двох масивів A[1..5], B[1..5] і порівняти їх між собою. Надрукувати більше з них. Обов'язково використати функцію.

50. Дано координати вершин п'ятикутника. Визначити площу кожного трикутника, з якого він складається. Обов'язково використати функцію

9.4.3 Питання для самоконтролю

1. Дайте означення розробки зверху вниз. Яке відношення має це поняття до структурного програмування?
2. Яка функція є в будь-якій програмі на Сі?
3. З яких частин складається функція?
4. Що таке аргумент? А параметр?
5. Що таке бібліотечні функції? Що потрібно зробити, щоб ваша програма дістала доступ до бібліотечної функції?
6. Якщо при оголошенні покажчика на функцію пропустити деякий елемент, в результаті можна одержати оголошення функції, що повертає покажчик. Вкажіть, про що йде мова?
7. Як передати функції масив? Чи застосовується цей самий спосіб передачі до рядків?
8. Як забезпечити повернення з функції декількох величин?
9. Назвіть чотири основні елементи, необхідні при роботі із змінним списком параметрів. (Підказка: всі вони починаються з va_.)
10. Дайте означення рекурсії. Які ресурси активно витрачаються при використанні рекурсії?

9.5 Лабораторна робота № 5

9.5.1 Обробка символьних даних

Мета даної роботи – одержання навичок обробки символьної інформації, текстів з використанням стандартних функцій мови Сі, а також за допомогою своїх власних функцій.

Теоретичні відомості

Відомо, що рядки – це послідовність символів, які визначаються за допомогою коду ASCII, як в мові Паскаль. Рядкова змінна – це така змінна, що може зберігати послідовність символів, подібно до рядка, на якому надруковані різні літери або інші символи. Обробка рядкових змінних в Сі має непростий характер, наприклад, в мові Visual Basic це робити зручніше. Рядкова змінна називається змінною-масивом типа char.

Символьним змінним можна присвоювати початкові значення. Це можливо зробити прямо в операторі оголошення. В дійсності можна не вказувати довжину рядка. Далі написані три оператори є еквівалентними:

```
char Radock [4]= { 'c', 'a', 't', '\0' };
char Radock [4]= "cat";
char Radock [ ]= "cat";
```

Якщо текст береться в подвійні лапки, Сі розглядає його як рядок; це

означає, що компілятор автоматично додає в кінець рядка нульовий символ. В першому прикладі його додають явним чином. Якщо в оголошенні рядка пропускається значення індексу, Сі підраховує, скільки символів в літерному рядку, додає одиницю, щоб врахувати нульовий символ, й створює рядок-змінну, яка точно відповідає ініційованому тексту. Якщо записати

```
char Mas [5]= "Україна";
```

то це буде помилка, тому що змінна має менший розмір, ніж їй присвоюється. Якщо забрати 5, то буде створений рядок потрібної довжини.

Щоб працювати з рядками, потрібно в свої програми включати заголовочні файли, такі, як string.h stdio.h.

Розглянемо три простих дії над рядками.

Перша: Це знайома нам операція конкатенації або склеювання рядків. Виконати конкатенацію рядків – це означає взяти другий з них і приставити його в кінець першого рядка; утвориться новий рядок, добуток двох рядків.

Для цього використовується функція strcat:

```
{ char Name [15]= "Igor";
  char NN [ ]= "Ivanov";
  strcat (Name; NN)';
  puts (Name);
}
```

Рядок, в який записується результат, повиннен бути досить довгим, щоб вмістити обидва рядки, інакше виникне помилка.

Друга: Функція strcpy( ) – копіювання рядка з другого місця на місце першого, вилучаючи значення попереднього: strcpy (st1, st2);

Рядок st1 повинен бути більшим за st2.

Третя: Для того, щоб знайти точний розмір рядка використовують функцію strlen( ). Вона повертає ціле без знака, яке дорівнює довжині рядка, не враховуючи нульового символу. Функцію можна використовувати в операторі присвоювання: Length = strlen (st1);

Для введення рядків використовуйте функцію gets( ). Цю функцію можна використовувати і для того, щоб звільнити буфер клавіатури після формального введення. Іноді в буфері залишаються незатребувані символи; наступна функція може їх використати, із-за чого буде помилковий результат. Використовуйте оператори типу:

```
char Trash [80];
gets (Trash);
```

Для виведення рядка користуйтесь функціями puts(st) і printf("%s", st);

Прототипом функції strlen є: int strlen (char *pointer);

тобто функція чекає, що pointer буде покажчиком на символьну змінну. Звідки функція strlen "знає", скільки символів у рядку? Вона шукає

завершальний нуль-символ \0. Початок рядка передається функції в якості аргумента; це покажчик. (Згадайте, що покажчик містить адресу початку змінної).

В Сі ім'я масиву є константним покажчиком на його перший елемент. Так як рядки в Сі – масиви символів, ім'я рядка є покажчиком початку рядка.

Нульовий символ є важливим моментом. Якщо ви напишите програму, яка створює рядок із окремих елементів, то не додавши в кінець нульового символу – чекайте неприємностей. Функції Сі не цікавляться тим, скільки елементів ви дали масиву, який створює рядок при його оголошенні. Вони шукають /0. Обшуковують пам'ять байт за байтом, поки не зустрінуть нуль. В результаті з пам'яті буде витягнута купа непотрібної інформації.

Є ще багато функцій для роботи з рядками. Самі поширені це:

strncpy – для копіювання загального числа символів із одного рядка в інший.

```
puts ( "\n введіть рядок");
gets ( sour); puts("Скільки символів копіювати?");
scanf ( "%d",&num);
strncpy ( Res, sonr, num);
printf ( " Перші %d символів: %s", num, Res);
```

strcmp() – функція порівняння двох рядків. Її прототип

– int strcmp (const *char string1, const *char string2);

strncmp () – функція порівняння частини рядків. Її прототип

– int strcmp (const *char string1, const *char string2, num);

Повертане значення менше нуля якщо string1 менше ніж string2 (тобто string1 іде за алфавітом раніше ніж string2). Значення дорівнює нулю, якщо рядки збігаються. Значення більше нуля, якщо string2 менше string1. Майте на увазі, шо strcmp() розрізняє регістри і великі букви йдуть за алфавітом раніше малих.

strchr () – пошук одного рядка в іншому. Її прототип

– char *strchr (const char *string, int char_to_find);

Функція повертає покажчик на знайдений в рядку string символ char_to_find Якщо такого немає,то повертає нуль. Для визначення позиції символу в рядку треба використовувати арифметику покажчиків. Треба відняти від покажчика, що повертає значення, покажчик на початок рядка; результат буде цілим зсувом, відповідаючим положенню символа.

Пошук першого проміжка:

```
PLOC=strchr (string, '_');
POS= PLOC - string;
```

– strrchr() – шукає останнє входження в рядок, подібна попередній;

– strcspn() – шукає в рядку перше входження будь-якого символа із другого рядка;

– strstr() – шукає перше входження підрядка, повертає покажчик, тобто
char *strstr()

Приклад: Напишіть блок операторів, які знаходять останню кому в рядку.

```
pPosi=strrchr (Astring, ',');
posi=pPosi-Astring;
```

Напишіть блок операторів, які вивчають рядок на предмет першого входження в нього однієї з десяткових цифр.

```
strcspn (A string, "123456789");
```

Є ще багато функцій для роботи з рядками, деякі з них використовуються трансляторами мови С++, в разі потреби звертайтесь до довідкової інформації.

Приклади розв'язання типових задач

Задача 1

Ввести послідовність слів і роздрукувати ті з них, перед якими стоять тільки лексикографічно менші, а за ними – тільки більші слова.

```
#include <stdio.h>
#include <conio.h>
# include <string.h>
#define DL_STR 80              // довжина рядка, що вводиться
#define KOL_SL 40              // кількість слів у рядку
 void main ()
{
  int i, j, k;
  char str[DL_STR] , *sl[ KOL_SL ], *razd=" . ,;:!?";
vvod:
   clrscr( ) ;
   puts ("Введіть рядок") ;
   gets(str);
   sl[k=0] =strtok(str, razd) ;       // виділяємо перше слово
   while ( (sl[++k] = strtok (NULL, razd)) !=0) ; //виділяємо інші слова
   puts("\n Список упорядкованих слів : ") ;
   for( i = 0; i<k; i++)              // цикл за словами
     { for( j=0;j<1;j++)
           if (strcmp(sl[i] , sl[j]<0)  goto sled_sl;
       for(j=k-1; j>i; j --)
         if (strcmp(sl[i], sl[j])>0)  goto sled_sl;
        puts (sl[i]);
sled sl:
     }
```

```
    printf("\n Продовжимо? Так:-введіть 7 : " );
    scanf("%d", &i); getchar();
    if (i==7)  goto vvod;
  }
```

Задача 2

Серед уведених слів роздрукувати спочатку ті, котрі починаються і закінчуються однією і тією самою буквою, а потім – всі інші (слова виводити по одному у рядку).

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
#define DL_STR 80     // довжина рядка, що вводиться
#define KOL_SL 40     // кількість слів у рядку

void main ( )
{
  int i, l, k;
  char *s1[KOL_SL], *p, *razd=", . ? ! ; : ",  str[DL_STR];
  clrscr( );
  printf("Уведіть рядок слів довжиною не більше %d
          символів\n", DL_STR-1) ;
  gets(str);
  str[DL_STR-1]+'\0';    //на випадок неприпустимо довгого рядка
  k=0;
  puts("\n\t\tОсь слова в потрібному порядку:\n");
  p=strtok(str, razd);
  while(p)
  { l=strlen(p);
    if(*p==*(p+l-1))  puts(p);  else s1[k++]=p;
    p=strtok(NULL, razd);
  }
   for (i=0; i<k; i++)
   puts(s1[i]);
   getch();          //утримуємо екран
}
```

Задача 3

Роздрукувати слова введеного тексту, змінивши кожне слово в такий спосіб: літери слова, що стоять до першої голосної, перенести в кінець слова.

```
#include  <stdio.h>
#include <string.h>
#include <conio.h>
#include <alloc.h>
void main()
{
 char  *str, *razd=", . ; : ? ! ", *glas="aeoiuy", b1, *s1;
 int  j, len, p, k, N;
 clrscr( );
 puts("Яка максимальна довжина рядка?");
 scanf("%d",&N);  getchar( );
 str=(char*) malloc(N) ;
 puts("Уведіть рядок з слів");
 gets(str);
 puts("Ось перетворений рядок:\n");
 s1=strtok(str, razd) ;
 while(s1)                          // цикл обробки чергового слова
  {
   len = strlen(s1);
   for(j=0; j<len; j++)              //шукаємо в слові першу голосну
   if (strchr(glas, s1[j])!= NULL) break;
   j%=len;              // якщо голосних не було, присвоїмо j=0,
      // щоб не змішувати вхолосту слово по колу
   for(k=0; k<j; k++)
    {      // j раз зміщуємо слово на одну літеру вліво циклічно
     b1=s1[0];
    for (p=0;  p<len-1;p++)
      s1[p] = s1 [p + 1 ];
      s1[p] = b1;
     }
     printf ("%s ", s1) ;
    s1=strtok(NULL, razd);   //виділяємо чергове
                             //слово з вихідного рядка
 } getch( );
}
```

Задача 4

Скласти програму, що друкує аргументи командного рядка в прямому порядку, якщо першим з них йде опція -s, і в зворотному порядку, якщо перший аргумент – опція -r (саму опцію не роздруковувати).

Наприклад, при виклику C:\prog -r *мені не дають ананаси*
повинно бути видане: *ананаси дають не мені*

```
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main(int argc, char *argv[])
{
  clrscr ( ) ;
  if(argc<2)
  {
    puts (" Потрібні аргументи в командному рядку ! ");
    exit (-1 );
  }
 if ( argv [1] [0] =='-' && argv[1] [1 ]  =='s'  &&  argv[ 1 ] [2] == '\0')
  { ++ argv;  // будемо друкувати аргументи, починаючи з третього
    while  (--argc>1) printf ( ( argc>2) ? "%s" : "%s\n", *++argv);
  }
 else if (strcmp(argv[1],"-r") == 0)
       while (--argc>1) printf ((argc>2) ? "%s" : "%s\n", *(argv+argc));
    else puts("Першим повинен бути аргумент -s чи -r");
  getch();
}
```

9.5.2 Завдання для лабораторних робіт

1. Написати і протестувати функцію STRP(str1, str2), що повертає покажчик на перше входження символу з рядка str2 у рядок str1. Якщо жоден символ рядка str2 не входить у рядок str1, то повернути NULL.

2. Ввести речення, слова в якому розділені пропусками і комами. Роздрукувати це речення, видаливши з нього ті слова, що зустрілися в ньому більше одного разу.

3. Написати і протестувати функцію DELETE(s1, s2), що видаляє з рядка s1 усі символи, що зустрічаються в рядку s2.

4. Дано два символьні рядки, що складаються тільки з цифр (довжина кожного більше 10 символів). Вважаючи, що в цих рядках знаходяться дуже довгі цілі числа, сформувати третій рядок – суму цих чисел.

5. Дано довільний текст. Відредагувати текст так, щоб:

а) між словами був рівно один пропуск;

б) речення в тексті розділялися рівно двома пропусками.

6. Написати і протестувати функцію ESCAPE(str1, str2), що при копіюванні тексту з str1 у str2 перетворить літери "новий рядок" і "табуляція" у видимі послідовності літер \n і \t. Написати також функцію, що виконує зворотне перетворення.

7. Ввести два речення і роздрукувати найдовші слова, які є спільними для цих речень. Якщо потрібних слів немає – повідомити про це.

8. Якщо перший аргумент командного рядка – опція *-a*, то роздрукувати інші аргументи без їхніх перших символів, а якщо першою йде опція *- r*, то роздрукувати аргументи через один у зворотному порядку. (Якщо аргументів немає – видати повідомлення.)

9. Виконати вирівнювання по правому краю введеного тексту, для чого до кожного рядка застосувати функцію WIDE(str, k), що рівномірно вставляє пропуски між словами так, щоб довжина рядка str стала рівною k. (Величина k повинна бути більше довжини найдовшого рядка тексту.)

10. Написати і протестувати функцію I_TO_B(n,s,b), що переводить ціле число n у рядок s, який є числом в системі числення з основою b.

11. Скласти програму, що реверсує(перевертає) кожне слово рядка str.

12. Ввести рядок, що складається тільки з цифр і букв. Роздрукувати ті групи цифр, у яких цифра 7 зустрічається не більше двох разів. (Група цифр – це послідовність цифр, обмежена буквами.)

13. Написати і протестувати функцію STRP(str1, str2), що повертає покажчик на останнє входження символу з рядка str2 у рядок str1. Якщо жоден символ рядка str2 не входить у рядок str1, то повернути NULL.

14. Ввести речення, слова в якому розділені проміжками і комами. Роздрукувати ті слова, що є оберненими для інших слів у цьому реченнні. Якщо потрібних слів немає – повідомити про це.

15. Дано масив символьних рядків. Якщо в командному рядку не задані аргументи, роздрукувати всі рядки, а якщо задана опція - n – роздрукувати останні n рядків.

16. Роздрукувати ті пари слів, відстань між якими найменша. (Відстань між словами – це кількість позицій, у словах, в яких символи цих слів не збігаються. Наприклад, відстань між словами МИШКА і КІШКА дорівнює двом.)

17. Написати і протестувати функцію NXT_BLNK(str, pos), що аналізує рядок str, починаючи з позиції pos, і повертає номер першого знайденого пропуску. Якщо пропуску немає, повертається 0; якщо pos<0 чи більше довжини рядка, то повертається мінус 1.

18. Роздрукувати введене речення, вилучивши з нього слова, що складаються менше ніж із трьох букв.

19. Зашифрувати текст у такий спосіб: записати його в матрицю по рядках, а потім переписати по спіралі від центра. Прочитати зашифрований текст.

20. Роздрукувати введені слова, які відрізняються від останнього, перетворивши їх у такий спосіб: перенести останню букву в початок слова; залишити в слові тільки перші входження кожної букви.

21. Якщо перший аргумент командного рядка опція -d, то роздрукувати введений далі текст без вхідних у нього цифр, а якщо зазначена опція -r, роздрукувати введений далі текст так, щоб цифри йшли в зворотному порядку. В інших випадках роздрукувати текст таким, яким він був уведений.

22. Написати і протестувати рекурсивну функцію REVERSE(str), що перевертає даний рядок на тому самому місці. Порівняти час її роботи і час роботи нерекурсивної версії.

23. Написати і протестувати функцію STRP(str1, str2), що визначає, чи зустрівся в рядку str1 будь-який символ з рядка str2. Функція повинна повертати номер позиції першого символу рядка str1, що збігається з будь-яким символом рядка str2, чи мінус 1, якщо збігів немає.

24. Роздрукувати рядок, що виходить із уведеного рядка в такий спосіб: кожна цифра заміняється на взяту в круглі дужки послідовність символів '+' (якщо цифра парна) чи '-' (якщо цифра непарна), довжина якої дорівнює числу, яке зображується цифрою.

25. Роздрукувати введений рядок, видаливши з нього слова з непарними номерами і перевернувши слова з парними номерами. Наприклад, з рядка

щоб то не було повинне вийти *от олуб*

26. Написати і протестувати функцію ISSUBSTR(str1, str2), що з'ясовує, чи є рядок str1 підрядком рядка slr2. Функція повинна повертати номер позиції, з якої починається підрядок, або мінус 1, якщо підрядок не знайдений.

27. Роздрукувати введений рядок, вилучивши з нього ті символи, що знаходяться між дужками '(' ')'. Самі дужки не видаляти. Якщо хоча б однієї дужки немає – повідомити про це.

28. Якщо в командному рядку задані опції -x - n <зразок>, то необхідно роздрукувати ті рядки введеного тексту з їхніми номерами, у яких не знайдений зазначений зразок, а при вказанні опції -x <зразок> роздрукувати ті самі рядки, тільки без номерів.

29. Роздрукувати ті слова, у яких або букви упорядковані за алфавітом, або кожна буква входить у слово не менше двох разів (тобто слова типу BEER, ABBA).

30. Написати і протестувати рекурсивну функцію STOI(n, str), що перетворить рядок десяткових цифр у ціле число.

31. Увести рядок, у який можуть входити тільки цифри і букви. Роздрукувати ті групи букв, у яких буква А зустрічається не менше двох разів. (Група букв – це послідовність букв, обмежена цифрами.)

32. Скласти частотний словник тексту, що вводиться. Роздрукувати його за алфавітом, а праворуч від кожного слова – частоту, з якою воно зустрілося.

33. Написати і протестувати функцію ISSUBSTR(str1, str2), що з'ясовує, чи є рядок str1 підрядком рядка str2. Функція повинна повертати покажчик на початок підрядка або NULL, якщо підрядок не знайдений.

34. Дано цілочисельний арифметичний вираз, записаний як рядок, у десятковій системі числення. Перевірити правильність запису й обчислити значення цього виразу. Вираз записується без дужок, операції виконуються в порядку їхнього проходження.

35. В уведеному слові підрахувати кількість різних пар букв. (Наприклад, у слові babacabacd 5 різних пар букв.)

36. Написати і протестувати функцію STREND(str1, str2), що повертає 1, якщо рядок str1 розташований наприкінці рядка str2, і 0 – у протилежному випадку.

37. Увести довільний текст. Обчислити середнє число слів у реченні і середню довжину речення.

38. Визначити, чи є введений рядок правильним записом цілого десяткового числа без знака.

39. Якщо в командному рядку задані опції -x - n <зразок>, то необхідно роздрукувати ті рядки введеного тексту з їхніми номерами, у яких зазначений зразок знайдений, а при вказанні опцій – < зразок> роздрукувати ті ж рядки, тільки без номерів.

40. Перевірити, чи є в заданому тексті баланс відкриваючих і закриваючих круглих дужок.

41. Скласти частотний словник тексту, що вводиться. Слова разом з їхніми частотами роздрукувати в порядку спадання частот.

42. Роздрукувати в порядку, зворотному алфавітному, усі букви, що входять у текст не менше трьох разів.

43. Увести довільний текст. Обчислити середню довжину слів у тексті і середню відстань (у позиціях) між розділовими знаками.

44. Визначити, чи є введений рядок символів правильним записом формули. Формула має такий вид:

<формула>::=<цифра> | (<формула><знак><формула>)

<знак>::=+|-|*

<цифра>::=0|1|2|3|4|5|6|7|8|9

45. Виділити з рядка str1 усі слова, що починаються з голосної букви, а з рядка str2 - слова, що починаються з приголосної. Утворити рядок str3, що

складається з виділених слів обох рядків. У новому рядку слова повинні розділятися двома пропусками.

46. Увести рядок, у який можуть входити тільки цифри і букви. Роздрукувати ті групи букв, у яких буква А зустрічається не менше двох разів. (Група букв - це послідовність букв, обмежена цифрами.)

47. Якщо в командному рядку задано опцію -u, то літери, що вводяться в нижньому регістрі, повинні перетворюватись в літери верхнього регістра. Якщо вказана опція –l, здійснити зворотне перетворення.

48. З уведеного тексту роздрукувати тільки ті слова, що симетричні (RADAR, ANNA), або в яких букви впорядковані в порядку, зворотному алфавітному (ТІК, ZONA).

49. Написати і протестувати функцію NXT_BLNK(str, pos), що аналізує рядок str, починаючи з позиції pos, і повертає покажчик на перший знайдений проміжок. Якщо проміжка немає, або pos<0 чи більше довжини рядка, то повертається NULL.

50. Зашифрувати текст методом Гронсфельда. Ключем є кінцева послідовність цифр, що записують підряд над символами тексту, який шифруємо. Цифра, що стоїть над літерою, є величиною зсуву (тобто говорить про те, на скільки треба просунутися вперед по таблиці кодування від поточного символу, щоб одержати для нього заміну). Протестувати написану програму.

9.5.3 Питання для самоконтролю.

1. Дайте означення рядка. Який символ в Сі позначає кінець рядка?
2. Що означає термін конкатенація?
3. Як записується символьна літеральна константа? Рядкова буквена константа?
4. Що не пишеться в назві рядка, якщо вона передається якій-небудь функції обробки рядків.
5. Чому важливо очищувати вхідний буфер після форматного введення?
6. В чому різниця між максимальною довжиною і поточною довжиною рядка? Як в Сі визначається поточна довжина? Яка функція визначає поточну довжину?
7. Де знаходяться прототипи функцій оброблення рядків?
8. Яка функція використовується для копіювання перших n символів із одного рядка в інший?
9. В чому різниця між функціями strchr() і strrchr()?
10. За допомогою якого ключового слова можна визначити синонім типу?

9.6 Лабораторна робота №6

9.6.1 Робота із структурами

Метою даної лабораторної роботи є надбання навичок у визначенні і використанні структур.

Теоретичні відомості

Структури в Сі відносяться до найбільш важливих і гнучких типів даних. Структура є множиною змінних – однієї або декількох – згрупованих під одним ім'ям. Кожна змінна структури має своє ім'я і визначення. Більше того, кожна із змінних може містити свою форму даних.

Структура є корисною в будь-якому випадку, коли декілька змінних потрібно розглядати як одне ціле. Структури можуть містити в собі інші структури, а також масиви. Можна створити цілий масив структур. Структуру потрібно оголосити, а потім її використовувати.

Загальною формою оператора означення структури є

```
Struct ім'я {
Тип змінної ім'я змінної;
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _
тип змінної ім'я змінної;
};
```

Ім'я – це початкова назва, під якою дана конструкція буде відома Сі. Для кожної змінної відводиться окремий рядок. Наприклад, структура має три змінні

```
Struct circle {
        int radius;
        int xcoord;
        int ycoord;
};
```

Ці три змінні містять радіус круга і координати його центра. В структурі потрібно вказати, яка змінна їй належить. Це робиться за допомогою операції-точки, тобто точки, яка розділяє частини імені структури.

Розглянемо приклад:

```
Struct circle {
        int radius;
        int xcoord;
        int ycoord;
} Mycircle;
My circle.radius = 15;
```

Тут створюється екземпляр, який має назву Mycircle, структури з ім'ям circle. Останній оператор присвоює значення 15 змінній circle, яка належить

до Mycircle. Зверніть увагу, що крапка з комою після } перед рядком, який задає ім'я екземпляра, не ставиться. Якщо потрібно, можна створити декілька екземплярів структур одночасно, написавши:

```
Struct circle Mycircle, Yourcircle , Ourcircle;
```

Цей самий принцип працює і при створенні екземплярів відразу після визначення структури.

```
Struct circle {
        int radius;
        int xcoord;
        int ycoord;
} Mycircle, Yourcircle , Ourcircle;
```

Для цього можна переміщувати весь вміст одного екземпляра в інший, наприклад: Yourcircle = Mycircle;

Альтернативним засобом оголошення і роботи зі структурою є використання ключового слова typedef, яке дозволяє економити час при наборі програми. Ці рядки еквівалентні попередньому:

```
typedef Struct circle {
        int radius;
        int xcoord;
        int ycoord;
} circle;
```

Щоб створити екземпляр структури, використовується оператор:

```
circle Mycircle;
```

Різниця в тому, що тепер ви не пишете ключове слово struct. Далі все таке ж, як і раніше. Це були прості структури; складні структури, це, наприклад, масив структур – можливо, найкорисніша із усіх конструкцій. Крім того, структура може містити масиви і масив може містити структури. Розглянемо приклади програм, які показують, як використовувати структури.

Оголошується структура computer і змінна pibm типу computer. Всі описи і визначення подані перед рядком main(). Далі буде показано, що в цьому випадку змінна pibm є глобальною. Рядки pibm.model, pibm.mem, pibm.sp викликають звертання до відповідних елементів структури pibm типу computer, яким раніше були присвоєні значення.

```
# include<stdio.h>
struct computer { int mem;
                  int sp;
                  char model [20];};
/* оголошення структури типу computer із 3-х елементів mem, sp, model*/
  struct computer pibm={ 512, 10, "ПЭВМ" };
```

```
/*ініціалізація змінної pibm типу computer*/
main()
{
  printf(" персональна ЕОМ % s \n \n", pibm.model);
  printf(" об'єм операційної пам'яті-- %d Кбайтів \n", pibm.mem);
  printf(" потужність- %d млн. опер. в секунду \n",pibm.sp);
}
```

Унарна операція & дозволяє отримати адресу структури. Припустимо, що задано оголошення

Struct date { int d, m, y;} day;

Тут day – структура типу date, яка має три елементи: d, m і y. Рядок виду struct date *db; встановлює той факт, що db – це покажчик на структуру типу date. Тепер для вибору елементів d, m, y структури необхідно використовувати конструкції: (*db).b, (*db).m, (*db).y. Дійсно, db – адреса структури, *db – сама структура. Круглі дужки тут необхідні, тому що точка (.) має вищий за зірочку (*) пріоритет. Для аналогічних випадків в мові Сі є ще одна додаткова операція -> (два символа A15 і D16). Вона теж вибирає елемент структури і дозволяє подати розглянуті вище конструкції в більш простому вигляді: db->d, db->m, db->y.

Поля і об'єднання є особливим різновидом структур. Поле – це послідовність сусідніх двійкових розрядів (бітів) всередині одного цілого значення. Воно може мати тип signed int або unsigned int і займати від 1 до 16 бітів. Поля розміщуються в машинному слові в напрямку від молодших до старших розрядів. Наприклад, структура

struct prim { int a:2; unsigned b:3; int:5;
int c:1; unsigned d:5; } i, j;

забезпечує розміщення згідно з вказаним розміром полів.

В полях типу signed крайній лівий біт є знаковим. Наприклад, таке поле шириною 1 біт може зберігати значення -1 і 0, тому що будь-яка ненульова величина буде інтерпретуватись як 1.

Поля використовуються для упакування значень декількох змінних в одне масивне слово з метою економії пам'яті. В оголошенні полів найчастіше використовується модифікатор unsigned. Поля не можуть бути масивами і не мають адрес, тому для них не можна застосувати унарну операцію &.

Використання полів розглянемо на прикладі такої програми:

```
# include <stdio.h>
struct { unsigned a:1;
         unsigned b:1;
         unsigned y:1;
         unsigned c:2;
```

```
    } /*оголошення змінної f, яка містить три однорозрядних і одне
дворозрядне поле (число), за двокрапкою – розмір поля*/
 main ()
 { int i;
   printf ("розмір f=%d байта \n", sizeof (f));
   f.a=f.b=1;                /* в поля f.a і f.b записуються одиниці*/
   for ( i=0;  i<2;  i++)
   {
    f.y = f.a && f.b; /*вираховується кон'юнкція f.y змінних f.a і f.b */
    printf (" цикл  %d;  f.y = %d\n ", i,  f.y );
    f.b=0;                    /*  в поле  f.b записується нуль       */
   }
   f.c=f.a+ !f.b;  /* арифметичне додавання знач. f.a (воно дорівнює 1)
                      і заперечення  знач. f.b (заперечення нуля теж
                      дорівнює  одиниці); в результаті f.c=2   */
    printf ( "f.c = %d",   f. c );
}
```

Оголошення виду unsigned a:1 визначає однобітове поле a; в нього можна записати тільки одне із двох значень: нуль або одиницю.

Оголошення unsigned c:2; визначає двобітове поле c; в нього вже можна записати одне із 4-х значень: 0, 1, 2 або 3. Функція sizeof(f) дозволяє визначити розмір в байтах одної структури f. Аналогічно з її допомогою можна знаходити і розміри других об'єктів (змінних, масивів і т.д.)

Звертання до любого поля здійснюється так, як до елемента структури. Наприклад: f.a – звертання до однобітового поля a.

В мові Сі можна вводити імена для нових типів даних за допомогою ключового слова **typedef** (визначити тип). Наприклад, опис typedef int Integer; робить слово integer синонімом int. Тепер його можна використовувати в оголошеннях типа, так само як int.

```
          /* Приклад визначення типу */
#include<stdio.h>
typedef float REAL; /* ім'я REAL робиться синонімом float (тепер в
                       описах відповідного типу можна
                       використовувати і float, і  REAL) */
main()
{
  REAL a;
  printf("ввести значення a\n");
  scanf("%f", &a);
```

```
  printf("a=%f",a);
}
```

Результати роботи програми:

```
  Ввести значення a
  9.4567
  a=9.456700
```

Приклади розв'язання типових задач

Задача 1

Визначити структуру, що описує поняття комплексного числа. Написати і протестувати функції для введення, виведення комплексного числа, а також для додавання двох комплексних чисел, заданих в алгебраїчній формі.

```
#include <stdio.h>
  typedef struct
  {
    int real;
    int imag;
  } Complex;
void main 0
{
  Complex c1, c2, c3, read( ) ;
  void add(Complex, Complex, Complex*), print(Complex);
  cl=read 0 ; c2=read0 ;
  add(c1, c2, &c3);
printf("При додаванні "); print (c1) ; printf(" i ") ; print (c2);
printf ("\n отримали "); print(c3) ;
}
  Complex read 0
{
   Complex c;
    puts("Уведіть дійсну і уявну частини числа:");
    scanf("%d %d", & (c.real) , & (c.imag));
return c;
 }
   void print (Complex c)
{
   printf ("%d+i*(%d)" , c.real, c.imag);
}
  void add (Complex c1, Complex c2, Complex* c3)
```

```
 {
    c3->real=c1.real+c2.real;
    c3->imag=c1.imag+c2.imag;
 }
```

Задача 2

Визначити структуру, що описує поняття раціонального числа. Написати і протестувати функції для скорочення, виведення раціонального числа, а також для ділення двох раціональних чисел.

```
#include<stdlib.h>
#include <conio.h>
#include <stdio.h>
#include <math.h>
   typedef struct
 {
    int chis;
    int znam;
 }   Racion;
   void  put (Racion A)
 {
  if (A.chis*A. znam<0)  printf ("-");
   A.chis=abs(A.chis);
   A.znam=abs(A.znam) ;
  printf ("%d/%d",  A.chis,  A.znam);
 }
  Racion SORK(Racion A)
 {
  int i, min;
 if (abs(A.chis)   > abs(A.znam))  min=abs(A.znam) ;
  else min=abs (A.chis);
   for ( i=min;  i>1;  i-- )
 if(A.chis% i ==0  && A.znam%i ==0)  break;
 A.chis/=i;
 A.znam/=i;
  return A;
}
    void DIV (Racion A, Racion B, Racion *C)
{
  A= SORK(A);
```

```
 B= SORK(B);
 C->chis=A.chis*B.znam;
 C->znam=A. znam*B.chis;
 *C=SORK(*C);
 }
void main0
{
 Racion A, B, C;
 puts("Введіть чисельник і знаменник 1-го, а потім - 2-го дробу ");
 scanf ("%d%d%d%d",  &A.chis,  &A.znam,  &B.chis,  &B.znam);
 DIV (A, B, &C);
 printf("При діленні  ") ; put(A) ; printf ("  на "); put(B) ;
 printf("\n Отримали "); put (C);
}
```

Задача 3

За введеним роком, місяцем і числом визначити порядковий номер дня в році.

```
  typedef struct
          {
            int year;
            int month;
            int day;
          } DATE;
       int tab_days[2][13] = { {0,31,28,31,30,31,30,31,31,30,31,30,31},
                               {0,31,29,31,30,31,30,31,31,30,31,30,31} };
     int day_of_year  (DATE d)
     {
      int i,  k;
      k= d.year%4 == 0 &&d.year %100 !=0 | | d.year%400 ==0;  // чи
                          високосний рік
      if (d.month<1 | |  d.month >12)  return -1;
      if (d.day < 1 | |  d.day > tab_days[k][d.month] )  return -1;
      for ( i=1;  i<d.month;  i++ )
            d.day += tab_days [k ][ i ];
       return d.day;
      }
   #include <stdio.h>
    void main( )
   {
```

```
    DATE  d;
    int  N;
  puts ("Введіть число, номер місяця і рік.");
  scanf ("%d%d%d", &d.day, &d.month, &d.year);
  N = day_of_year(d) ;
  if  (N  <0) puts("Неправильно введено дату!");
    else  printf("В %d  році цей день має номер  %d\n", d.year, N);
}
```

9.6.2 Завдання на програмування

1. Ввести переліковні типи **масть, достоїнство.** З їхньою допомогою описати як структуру змінну **карта**. Скласти і протестувати функцію Б'Є(К1, К2, КМ) , яка перевіряє, чи б'є карта К1 карту К2, зважити на те, що масть КМ є козирною.

2. Описати як структуру змінну **час** (з полями години, хвилини, секунди). Скласти і протестувати функцію: СЛІД_СЕК(1, tl.d), яка присвоює параметру tl час на d секунд більше, ніж час t (може відбуватися зміна доби); ІНТЕРВАЛ (tl, t2, d), яка обчислює час d, що пройшов від часу tl до часу t2.

3. Ввести переліковні типи **вертикаль, горизонталь** для позначення клітин шахової дошки. Скласти і протестувати функцію: ХІД_КОНЯ (К1, К2), яка обчислює, за скільки ходів кінь може перейти з поля К1 на поле К2.

4. Увести структуру (з полями **чисельник** і **знаменник**) для опису поняття **раціональне** число. Скласти і протестувати функцію: РІВНО(А,В), яка перевіряє, чи дорівнюють один одному раціональні числа.

5. Увести структуру (з полями **чисельник** і **знаменник**) для опису поняття **раціональне** число. Скласти і протестувати функцію: МАКС (Х, N), яка повертає найбільше з масиву Х[N] раціональних чисел.

6. Увести структуру (з полями **чисельник** і **знаменник**) для опису поняття **раціональне** число. Скласти і протестувати функцію: ДОД(А,В,С). яка записує в С результат додавання раціональних чисел А і В;

7. Увести структуру (з полями **чисельник** і **знаменник**) для опису поняття **раціональне** число. Скласти і протестувати функції: МІН (А, В), яка повертає найменше з двух раціональних чисел А і В.

8. Увести структуру (з полями **чисельник** і **знаменник**) для опису поняття **раціональне** число. Скласти і протестувати функції: УМН (А, В, С), яка записує в С результат добутку раціональних чисел А і В.

9. Ввести структуру (з полями **число, місяць, рік**) для опису поняття **дата**. Скласти і протестувати функцію, що обчислює інтервал (у днях), що пройшов між двома датами.

10. Ввести структуру (з полями **число, місяць, рік**) для опису поняття

дата. Скласти і протестувати функцію, що за порядковим номером дня в році визначає число і місяць року, які відповідають цьому дню.

11. Ввести структуру (з полями **число, місяць, рік**) для опису поняття **дата**. Скласти і протестувати функцію, що за введеною датою роздруковує дату на N днів уперед.

12. Визначити структури, що описують **кулю** і **точку** в тривимірному просторі. Скласти і протестувати функцію, що перевіряє, чи знаходиться точка усередині заданої кулі.

13. Ввести структуру для опису поняття **алгебраїчний поліном**. Скласти і протестувати функції для введення полінома і виведення полінома.

14. Ввести структуру для опису поняття **алгебраїчний поліном**. Скласти і протестувати функції для додавання поліномів і віднімання поліномів.

15. Ввести структуру для опису поняття **алгебраїчний поліном**. Скласти і протестувати функції для множення поліномів і ділення поліномів.

16. Ввести структуру для реєстрації автомашин. Вона повинна мати такі поля: дату реєстрації (структура з полями – день, місяць, рік), марку машини; рік випуску; колір; номер. Написати і протестувати функцію реєстрації нової машини.

17. Ввести структуру для реєстрації автомашин. Вона повинна мати такі поля: дату реєстрації (структура з полями - день, місяць, рік), марку машини; рік випуску; колір; номер. Написати і протестувати функцію видалення машини з реєстраційного списку.

18. Масив структур містить інформацію про студентів групи: у першому полі знаходиться прізвище, у другому – вік, у третьому – ріст, у четвертому – середній бал за сесію і т.д.(і-й елемент масиву описує і-го студента). Студент називається середньостатистичним за k-м параметром; якщо на ньому досягається мінімум модуля різниці середнього арифметичного чисел k-го стовпця і значення k-го параметра цього студента. Аналогічно визначається унікальний за k-м параметром студент (на ньому досягається максимум). Студент називається самим середнім, якщо він є середньостатистичним за найбільшою кількістю параметрів. Аналогічно визначається самий унікальний студент. З'ясувати, хто в групі є: самим середнім; самим унікальним; самим середнім серед самих унікальних; самим унікальним серед самих середніх.

19. У будинку N поверхів і три ліфти. Кожен ліфт може бути вільним чи зайнятим. Людина стоїть на одному з поверхів і збирається викликати або найближчий вільний ліфт, або найближчий зайнятий, прямуючий в ту сторону, де знаходиться людина. Роздрукувати початкову конфігурацію (розміщення, зайнятість і напрямок руху ліфтів, місце розташування людини), а також номер ліфта, що буде викликаний. Використовувати функції ВВЕДЕННЯ, ВИВЕДЕННЯ, ВИБІР_ЛІФТА.

20. Нехай ЕОМ не вміє працювати з дійсними числами, а має тільки операції і функції для роботи із символами, рядками і цілими числами. Реалізувати функції для введення дійсних чисел. (Числа вводяться як рядки, розділяються на цілу і дробову частини, і над ними, як над цілими числами, з урахуванням міжрозрядних переносів, виконуються операції.)

21. Нехай ЕОМ не вміє працювати з дійсними числами, а має тільки операції і функції для роботи із символами, рядками і цілими числами. Реалізувати функції для виведення дійсних чисел. (Числа вводяться як рядки, розділяються на цілу і дробову частини, і над ними, як над цілими числами, з урахуванням міжрозрядних переносів, виконуються операції.)

22. Нехай ЕОМ не вміє працювати з дійсними числами, а має тільки операції і функції для роботи із символами, рядками і цілими числами. Реалізувати функції для додавання дійсних чисел. (Числа вводяться як рядки, розділяються на цілу і дробову частини, і над ними, як над цілими числами, з урахуванням міжрозрядних переносів, виконуються операції.)

23. Нехай ЕОМ не вміє працювати з дійсними числами, а має тільки операції і функції для роботи із символами, рядками і цілими числами. Реалізувати функції для віднімання дійсних чисел. (Числа вводяться як рядки, розділяються на цілу і дробову частини, і над ними, як над цілими числами, з урахуванням міжрозрядних переносів, виконуються операції.)

24. Визначити структуру, що описує рівнобедрений прямокутний трикутник з катетами, паралельними осям координат, і нижнім лівим прямим кутом. Написати і протестувати функцію, що повертає покажчик на новий трикутник – область перетинання двох заданих. Якщо перетинання немає – повертається NULL.

25. Визначити структури, що описують точку в полярній і декартовій системах координат. Скласти і протестувати функцію для одержання декартових координат точки, якщо задані її полярні координати.

26. Визначити структури, що описують точку в полярній і декартовій системах координат. Скласти і протестувати функцію для обчислення відстані між двома точками, заданими в декартовій системі координат.

27. Визначити структуру – найважливіші історичні дати. Її поля – рік, подія. Написати і протестувати функцію, яка сортує структури за кожним із полів.

28. Визначити структуру – найважливіші історичні дати. Її поля – рік, подія. Написати і протестувати функцію, яка підраховує середній інтервал між датами.

29. Визначити структуру, що описує прямокутник зі сторонами, паралельними осям координат (прямокутник задається двома точками – лівою нижньою і правою верхньою). Написати і протестувати функцію, яка повертає покажчик на новий прямокутник – область перетинання двох

прямокутників. Якщо перетинання немає – повертається NULL.

30. Ввести переліковні типи **вертикаль, горизонталь** для позначення клітин шахової дошки. Скласти і протестувати функцію ХІД_ФЕРЗЯ (К1, К2), яка перевіряє, чи зможе ферзь за один хід перейти з поля К1 на поле К2.

31. Описати структуру і помістити в неї такі дані: житель, прізвище, місто, адреса (вулиця, будинок, квартира). Створити масив структур та функцію „Іронія долі", котра друкує прізвища двох мешканців зі списку С, які живуть у різних містах за однаковою адресою.

32. Описати структуру з ім'ям STUDENT, що містить такі поля:
прізвище і ініціали; номер групи; успішність (масив з п'яти елементів).
Написати програму, що виконує такі дії: введення з клавіатури даних у масив, що складається з десяти структур типу STUDENT; записи повинні бути впорядковані за зростанням номера групи; виведення на дисплей прізвищ і номерів груп для всіх студентів, включених у масив, якщо середній бал студента більше 8.0; якщо таких студентів немає, вивести відповідне повідомлення.

33. Описати структуру з ім'ям STUDENT, що містить такі поля: прізвище і ініціали; номер групи; успішність (масив з п'яти елементів).

Написати програму, що виконує такі дії: введення з клавіатури даних у масив, що складається з десяти структур типу STUDENT; записи повинні бути впорядковані за зростанням середнього бала; виведення на дисплей прізвищ і номерів груп для всіх студентів, що мають оцінки 4 і 5; якщо таких студентів немає, вивести відповідне повідомлення.

34. Описати структуру з ім'ям AEROFLOT, що містить такі поля:
назва пункту призначення рейсу; номер рейсу; тип літака.

Написати програму, що виконує такі дії: введення з клавіатури даних у масив, що складається із семи елементів типу AEROFLOT; записи повинні бути впорядковані за зростанням номера рейсу; виведення на екран номерів рейсів і типів літаків, що вилітають у пункт призначення, назва якого збіглася з назвою, введеною із клавіатури; якщо таких рейсів немає, видати на дисплей відповідне повідомлення.

35. Описати структуру з ім'ям AEROFLOT, що містить такі поля: назва пункту призначення рейсу; номер рейсу; тип літака.

Написати програму, що виконує такі дії: введення з клавіатури даних у масив, що складається із семи елементів типу AEROFLOT; записи повинні бути розміщені за абеткою по назвах пунктів призначення; виведення на екран пунктів призначення і номерів рейсів, що обслуговуються літаком, тип якого введений із клавіатури; якщо таких рейсів немає, видати на дисплей відповідне повідомлення.

36. Описати структуру з ім'ям WORKER, що містить такі поля: прізвище і ініціали працівника; назва займаної посади; рік влаштування на роботу.

Написати програму, що виконує такі дії: введення з клавіатури даних у масив, що складається з десяти структур типу WORKER; записи повинні бути розміщені за алфавітом; виведення на дисплей прізвищ працівників, чий стаж роботи в організації перевищує значення, введене з клавіатури; якщо таких працівників немає, вивести на дисплей відповідне повідомлення.

37. Описати структуру з ім'ям TRAIN, що містить такі поля: назва пункту призначення; номер потяга; час відправлення.

Написати програму, що виконує такі дії: введення з клавіатури даних у масив, що складається з восьми елементів типу TRAIN; записи повинні бути розміщені за абеткою по назвах пунктів призначення; виведення на екран інформації про потяги, що відправляються після введеного з клавіатури часу; якщо таких потягів немає, видати на дисплей відповідне повідомлення.

38. Описати структуру з ім'ям TRAIN, що містить такі поля: назва пункту призначення; номер потяга; час відправлення.

Написати програму, що виконує такі дії: введення з клавіатури даних у масив, що складається із шести елементів типу TRAIN; записи повинні бути упорядковані за часом відправлення потяга; виведення на екран інформації про потяги, що направляються в пункт, назву якого введено з клавіатури; якщо таких потягів немає, видати на дисплей відповідне повідомлення.

39. Описати структуру з ім'ям TRAIN, що містить такі поля: назва пункту призначення; номер потяга; час відправлення.

Написати програму, що виконує такі дії: введення з клавіатури даних у масив, що складається з восьми елементів типу TRAIN; записи повинні бути упорядковані за номерами потягів; виведення на екран інформації про потяг, номер якого введений із клавіатури; якщо таких потягів немає, видати на дисплей відповідне повідомлення.

40. Описати структуру з ім'ям MARSH, що містить такі поля: назва початкового пункту маршруту; назва кінцевого пункту маршруту; номер маршруту.

Написати програму, що виконує такі дії: введення з клавіатури даних у масив, що складається з восьми елементів типу MARSH; записи повинні бути упорядковані за номерами маршрутів; виведення на екран інформації про маршрут, номер якого введений із клавіатури; якщо таких маршрутів немає, видати на дисплей відповідне повідомлення.

41. Описати структуру з ім'ям MARSH, що містить такі поля: назва початкового пункту маршруту; назва кінцевого пункту маршруту; номер маршруту.

Написати програму, що виконує такі дії: введення з клавіатури даних у масив, що складається з восьми елементів типу MARSH; записи повинні бути упорядковані за номерами маршрутів; виведення на екран інформації про маршрути, що починаються чи закінчуються в пункті, назву якого

введено з клавіатури; якщо таких маршрутів немає, видати на дисплей відповідне повідомлення.

42. Описати структуру з ім'ям NOTE, що містить такі поля: прізвище, ім'я; номер телефону; дата народження (масив із шести чисел).

Написати програму, що виконує такі дії: введення з клавіатури даних у масив, що складається з восьми елементів типу NOTE; записи повинні бути упорядковані за датами народження; виведення на екран інформації про людину, номер телефону якої введений із клавіатури; якщо такої немає, видати на дисплей відповідне повідомлення.

9.6.3 Питання для самоконтролю

1. Дайте визначення структури. Що називається ярликом структури?
2. Яке призначення операції-точки?
3. В чому різниця використання typedef і struct при визначенні структури?
4. Як звернутися до елемента масиву, який входить в структуру? До структури, яка є елементом масиву?
5. Що потрібно для включення структури в структуру більшого розміру?
6. Що таке поле? Для чого воно використовується?
7. Що таке об'єднання? Які функції воно виконує в мові Сі? Поясніть на прикладі.
8. За допомогою якого ключового слова можна визначити синонім типу?

9.7 Лабораторна робота №7

9.7.1 Організація роботи з файлами

Мета роботи. Навчитися працювати з файлами. У запропонованих задачах потрібно заздалегідь створити необхідні файли засобами системи, якщо в програмі відбувається звертання до них. При написанні програм імена вхідного і вихідного файлів рекомендується задавати як аргументи командного рядка. Необхідно видавати діагностику при помилкових ситуаціях у роботі з файлами.

Теоретичні відомості

Мова Сі крім стандартного введення даних із клавіатури і виведення результатів на екран надає також можливість обміну при операціях введення / виведення з зовнішніми пристроями, у тому числі, з файлами на диску.

У Сі не передбачені ніякі певні структури файлів (такі як файли послідовного чи прямого доступу): усі файли розглядаються як послідовності, потоки байтів. Дисковим файлом називають поіменований набір даних

одного типу.

Для файлу визначений маркер (покажчик читання / записування). Він визначає поточну позицію, до якої здійснюється доступ.

З початком роботи будь-якої програми автоматично відкриваються деякі стандартні потоки, наприклад, стандартне введення (його ім'я – stdin) і стандартне виведення (його ім'я – stdout). За замовчуванням вони пов'язані з клавіатурою й екраном термінала, відповідно. Тому в тих функціях уведення / виведення, що використовувалися дотепер, не вказувалося з якого потоку беруться і куди розміщуються дані: це відомо за замовчуванням.

Однак можливо у своїй програмі відкривати інші потоки, пов'язувати їх або з файлами на диску, або з фізичними пристроями (наприклад, принтером) записувати чи зчитувати з них інформацію. Для цього служать функції введення / виведення верхнього рівня.

Доступ до потоку здійснюється за допомогою покажчика. Покажчик на файл описується в такий спосіб:

FILE *fp;

Тип FILE – це структура, визначена в <stdio.h> за допомогою засобу typedef, яка містить деяку інформацію про файл: наприклад, прапорці стану файлу, розмір буфера, покажчик на буфер і ін. Описаний покажчик можна пов'язати з конкретним файлом у момент відкриття даного файлу. Це здійснюється за допомогою функції

fopen ("шлях до файлу", "тип доступу"),

яка повертає покажчик на файл чи NULL у випадку помилки.

Наприклад, у результаті виконання оператора

fp = fopen (" ex1.txt ", "w") ;

файл ex1.txt відкритий для записування, а в програмі на цей файл можна посилатися за допомогою покажчика fp (тобто функція fopen() бере зовнішнє подання файлу – його фізичне ім'я – і ставить йому у відповідність внутрішнє логічне ім'я, що далі і буде використовуватися в програмі). Як тип доступу можуть бути зазначені такі параметри:

" w " – існуючий файл відкривається для записування, при цьому його старий вміст затирається. Маркер файлу вказує на його початок. Якщо файл ще не існував, то він створюється (якщо це можливо);

" r " – існуючий файл відкривається для читання (з початку). Якщо файл не існує – це помилка;

" a " – файл відкривається для дозаписування в його кінець. Про інші типи доступу можна прочитати в таблиці 9.

Після закінчення роботи з файлом він повинний бути закритий. Для цього використовується функція

fclose (покажчик_файлу).

Для читання/записування даних у файл існують функції, аналогічні уже відомим функціям введення / виведення

fprintf( ), fscanf( ), fputs( ), fgets( ), getc( ), putc( ), fgetc( ), fputc( ).

Функції getc( )/fgetc( ), putc( )/fputc( ) за своїми діями ідентичні, відмінність складається тільки в тому, що getc( ) і putc( ) реалізовано як макровизначення, а fgetc() і fputc( ) – як дійсні функції.

Прототипи усіх файлових функцій, а також необхідні константи знаходяться у файлі <stdio.h> .

Всі файли поділяються на два основні типи: текстові і двійкові (інакше бінарні), пов'язані з різними типами потоків даних. Згадаємо, що текстовий потік складається із послідовності ASCII кодів, поданих у вигляді рядків; кожний рядок завершується символом \n. Кінець текстового потоку помічається символом кінця файлу (^Z). Навпаки, двійковий потік є собою послідовністю бітів, такі дані не можна інтерпретувати ззовні тексту програми.

Таблиця 9 – Режими доступу до файлу

Режими	Призначення
r	Доступ тільки для читання. При відсутності вказаного файлу fopen() повертає нульовий покажчик (NULL).
w	Доступ тільки для записування. При відсутності вказаного файлу fopen() створює файл з таким ім'ям. Якщо файл з таким ім'ям існує, його вміст стирається.
a	Режим додання даних. Якщо файл з вказаним ім'ям не існує, fopen() створює його. В існуючий файл дані додаються до уже створених раніше.
R+	Доступ для читання і записування при відсутності вказаного файлу fopen() створює файл з таким ім'ям. Якщо файл з таким ім'ям існує, дані записуються поверх існуючих даних, починаючи з першої позиції в файлі.
W+	Режим для читання і записування. Якщо файл з вказаним ім'ям не існує, fopen() створює його. Якщо файл з таким ім'ям існує, колишній вміст файла стирається.
A+	Доступ для читання і додання. Якщо файл з вказаним ім'ям не існує, fopen() створює його. В існуючий файл дані додаються до існуючих в файлі даних.

Подібно до текстових і двійкових потоків існують текстові і двійкові файли. Текстові файли містять дані в звичайній формі, наприклад: програми на Сі. Двійкові або бінарні файли називаються форматними файлами, містять дані іншого роду. Засіб інтерпретації даних залежить від того, як програма

читає такий файл. Документ, сформований текстовим процесором, є прикладом двійкового файлу.

Для роботи з дисковим файлом потрібно відкрити потік даних, пов'язаних з цим файлом. Частиною цієї процедури є визначення області оперативної пам'яті для оброблення потоку; ця область має назву **файлового буфера**. В Сі доступ до файлового буфера відбувається шляхом задання адреси в оперативній пам'яті; ця адреса звичайно зберігається в покажчику, який ви повинні передати оператору відкриття файлу.

Приклад виклику функції fopen();

```
FILE *FileOpend;
pFileOpend = fopen(FileName, "r");
```

Залежно від запропонованих операцій з файлами існує декілька режимів відкриття файлу. Якщо ви будете тільки читати дані з файлу – він повинен бути відкритим для читання. Для записування даних в файл, він повинен бути відкритим для запису.

Таблиця 9 містить опис варіантів відкриття файлів тільки в текстових режимах; для відкриття файла в двійковому режимі необхідно до наведених в таблиці описів відповідного режиму додати символ b.

Наприклад, наведений нижче фрагмент коду Сі запитує у користувача ім'я файлу, потім намагається відкрити його в режимі тільки для читання в двійковій формі:

```
FILE *pDiskFile;
puts("введіть ім'я файлу, який читають");
gets(EnterredFilename);
pDiskFile = fopen(EnterredFilename, "rb");
```

Значення покажчика pDiskPointer повино бути передано відповідному оператору, який повинен виконати читання даних з відкритого файлу. При відсутності вказаного файлу fopen() повертає нульовий покажчик (NULL). Таке ж значення покажчика повертається в випадку неможливості знайти файл, наприклад, через відсутність необхідного обладнання.

Залежно від завдання, яке вирішує ваша програма, ви або читаєте дані з файлу або записуєте їх. Існує три способи зробити це залежно від характеру даних у файлі. Ці способи допускають виконання трьох можливих варіантів введення-виведення:

- прямий;
- символьний;
- форматний, що аналогічно функціям форматного, рядкового і символьного введення-виведення.

При використанні **прямого введення-виведення** читання або записування в файл відбувається байтовими блоками певного розміру.

Згаданий блок повинен десь знаходитись в пам'яті; для визначення його положення використовується покажчик типу void.

Зверніть увагу на використання операції sizeof; це кращий спосіб гарантувати, що прочитаєте необхідну кількість байтів. Якщо згадати, що ім'я масиву або структури є константним покажчиком; то потрібно використовувати ім'я структури в якості покажчика місця, куди повинна бути записана структура.

Наведемо приклад функції, яка зчитує дані з відкритого файлу в масив структур. Функція використовує оператор виклику функції fread() і відповідні йому операції. Звернемо увагу, що ця функція звертається до функції FileExists() для перевірки існування файлу; при відсутності файлу виводиться повідомлення про помилку і відбувається вихід із функції без читання даних:

```
int FileLoad(char *FileName,
                        struct NameRecord *pNames)
{
   FILE *pDickFile;
   Int counter = 1;
   If((pDiskFile = FileExists(FileName) == NULL))
        {
             puts("File Does Not Exist");
             return 0;
        }
   else
        {
             pDiskFile = fopen(FileName,"r");
             while(!feof(pDiskName))
             {
                  fread(&pNames[counter++],
                       sizeof(struct NameRecord),
                       1, pDiskFile);
             }
             fclose(pDiskFile);
        }
   return counter – 1;
}
```

Рідше використовують функції введення-виведення файлових потоків, які працюють з рядком або з одним символом. Нижче наведені прототипи цих функцій:

int fgetc(FILE *file_pointer);
int fputc(FILE *file_pointer);
char *fgets(char *buffer, int numch, FILE *fpointer);
char *fputs(const char *str, FILE file_pointer);

Функція fgetc() повертає значення одного символу з файла, на який поставлений покажчик; putc() записує в файл один символ. Функції fgets() і fputs() використовуються для роботи з цілими рядками.

Також існують функції введення-виведення файлового потоку, які є аналогами функцій scanf() і printf(). Це функції fscanf() і fprintf(). Їх прототипи:

int fprintf(FILE *file_pointer, const char *format...);
int fscanf(FILE *file_pointer, const char *format...);

Для роботи з файлами використовуються такі функції:

- fread() – призначена для прямого читання даних з відкритого файлу; її прототип:

int fread(void *buffer, int blocksize, int blocks, FILE *fptr);

Передача функції покажчика (buffer) використовується для визначення в пам'яті місця зберігання даних; це може бути покажчик на змінну типу структура або на рядок. Крім того, функції треба передати розмір в байтах (blocksize) зчитуваного блоку даних і кількість блоків (bloks), яку вона повинна повернути. Функції також потрібен покажчик на відкритий файл (fprt); це значення покажчика повертається функцією fopen(). Сама функція fread() повертає кількість прочитаних блоків.

Наприклад, для читання одного блоку даних із файлу в структуру з іменем Name; ім'я структури – Namedef:

fread(&Name, sizeof(struct Namedef), 1, pFilePointer);

- функція fwrite() призначена для записування даних в відкритий файл. Її прототип зберігається в файлі stdio.h, має вигляд:

int fwrite(void *buffer, int blocksize,
int blocks, FILE *fprt);

Ця функція виконує записування даних із області пам'яті, визначеної покажчиком buffer. Вона записує необхідну кількість (blocks) блоків заданого розміру (blocksize) в байтах в відкритий файл, який визначає покажчик fprt. Функція повертає кількість записаних блоків. Оператор для записування даних із структури з іменем Name в файл, визначений покажчиком pFilePointer буде мати такий вигляд:

fwrite(&Name, sizeof(sttruct NameDef), 1, pFilePointer);

- функція fclose() – закриває файл і пов'язаний з ним потік даних. Її прототип:

```
int fclose(FILE *file_pointer);
```

При успішному закритті потоку функція повертає нуль, в випадку помилки – повертає 1. file_pointer є значення, повертане при першому виклику функції fopen() для підготовки файлу до роботи. Таким чином, оператор закриття файлу має вигляд:

```
fclose(pFilePointer);
```

Якщо ви бажаєте повернути індикатор позиції в початкове положення, скористайтесь функцією rewind(). Вона викликається таким чином:

```
rewind(pFilePointer);
```

і дозволяє встановити індикатор позиції на початок файлу.

- І остання функція в цьому розділі – це функція fseek().

Її прототип:

```
int fseek(FILE *file_pointer, long offset, int origin);
```

Ця функція виконує зміну на величину offset значення індикатора позиції файлу, на який вказує file_pointer. Значення origin визначає початкове положення, відносно якого буде проводитись зміщення. Нульове значення цієї величини відповідає початку файлу; при значенні, рівному 1, відлік буде вестись від поточної позиції в файлі; при значенні, рівному 2, відлік буде вестись від кінця файлу. Зміщення позиції буде виконано на кількість байтів, що визначаються значенням offset. Таким чином, щоб встановити позицію, яка відстоїть на два записи від кінця файлу, потрібно викликати функцію з такими параметрами:

```
fseek(pFilePointer, 2*(sizeof(struct Record) ,2));
```

Існує ще декілька операцій з файлами, про які ви повинні мати уявлення. Кожна з цих операцій – переіменування і вилучення файлів – має еквівалент серед команд операційної системи. Часто буває доречно перекласти таку роботу на саму операційну систему, щоб не завдавати собі зайвий клопіт, якщо помилково будуть вилучені потрібні файли.

Але існує така задача, як відкриття і робота з тимчасовими файлами. В процесі роботи може знадобитися тимчасове видалення даних з ОП і зберігання їх на диску. В цьому випадку після завершення роботи програми необхідно вилучити створені тимчасові файли, щоб не забруднювати диск.

Функція С rename() змінює ім'я існуючого файлу даних.

Її прототип:

```
int rename(const char *old_name,  const char *new_name);
```

Якщо старе ім'я Old_Name, а нове New_Name, то переіменування виконано успішно. Збій при виконанні функції може бути пов'язаний з тим, що в дійсності старого імені не існує. Те ж саме відбудеться, якщо файл з новим іменем уже існує або ви спробуєте вийти за межі диска (приклад,

спробуйте перейменувати файл на диску С в файл на диску А, оскільки така операція уже буде не переіменуванням, а переміщенням файлу).

Іноді файл потрібно видалити.

Функція Сі remove() призначена для видалення файлу. Це найнебезпечніша із всіх функцій. Її прототип має вигляд:

remove(Filename);

Функція повертає нуль в випадку успіху і мінус 1 в іншому випадку. Причиною помилки може бути відсутність файлу з вказаним ім'ям, існування у файлу ознаки тільки для читання (в нього не можна записувати) або те, що він відкритий і знаходиться в роботі.

Приклади розв'язання типових задач

Задача 1

Демонстрація основних функцій роботи з файлами.

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int n;
 char str[50],str1[50],ch;
 FILE *fp;  // заповнюємо файл
 fp=fopen("ex.txt","w");
 clrscr();
 printf("Записуємо дані у файл\n");
 puts("Введіть символ");
 scanf("%c",&ch);
 putc(ch, fp);
 puts("Введіть рядок");
 scanf("%s",str);
 fputs(str, fp);
 fprintf(fp,"\n");
 puts("Введіть ціле число");
 scanf("%d",&n);
 fprintf(fp,"%d",n);
 fclose(fp);
 if((fp=fopen("ex.txt","r"))!=NULL)
 {
  printf("Читаємо дані з файлу\n");
  printf("Символ");  ch=getc(fp); putchar(ch);
```

```
 printf("\n");
 printf("Рядок "); fgets(str1, 50, fp); printf("%s",str1);
 fscanf(fp,"%d",&n); printf("Число %d",n);
 fclose(fp);
}
else printf("\nНе можна відкрити файл для читання!");
getch();
}
```

Другий параметр функції fgefs() – кількість N символів, що зчитуються, включаючи '\0'. Ця функція припиняє роботу після читання N-1 символів або після читання '\0'. У будь-якому випадку в кінець рядка додається '\0'. Функція fgets( ) повертає або адресу прочитаного рядка, або NULL після закінчення файлу чи у випадку помилки.

У випадку закінчення файлу чи помилки функція getc( ) повертає EOF. Функція putc( ) повертає записану літеру або (у випадку помилки) – EOF. Функція fputs( ) повертає код останнього прочитаного символу, якщо все в порядку, і EOF у випадку помилки. Ця функція **не переводить** рядок автоматично.

Задача 2

У командному рядку передається ім'я файлу, що зчитується. Кожен третій його символ переписується у вихідний файл, що має те ж саме ім'я і розширення .red.

```
#include<conio.h>
#include <stdio.h>
#include<string.h>
void main(int argc, char *argv[])
{
 clrscr();
 int ch, count=1;
 FILE *in,*out1;
 char name[20];        // під ім'я вихідного файлу
 if (argc<2)
    puts ("Потрібно ім'я файлу в командному рядку!");
 else
 {
     if((in=fopen(argv[1],"r")) != NULL)
     {
        name[0]='\0';
        int d=strlen(argv[1]);
```

```
        strcpy(name, argv[1]);
        name[d-3]='\0';
        out1=fopen(name,"w" ) ;
        while((ch=getc(in))!=EOF)
        {
         if(count++ % 3 == 0)
         putc(ch, out1);
        }
        fclose(in);
        fclose(out1);
      }
      else  printf("\n Не можна відкрити файл %s", argv[1]);
    }
    getch();
}
```

Усі наведені вище функції обробляли файл послідовно – символ за символом. Мова Сі надає можливість працювати з файлом і як з масивом: безпосередньо досягати будь-якого певного байта. Для позиціювання файлу служить функція

fseek (покажчик на файл, зсув від початкової точки, початкова точка)

Другий аргумент має тип long, його значення може бути більшим і меншим нуля. Він показує, як далеко (у байтах) варто просунутися від початкової точки. Третій аргумент є кодом, що визначає положення початкової точки у файлі. Установлено такі значення для коду

0 – початок файлу;

1 – поточна позиція;

2 – кінець файлу.

У випадку успіху функція fseek( ) повертає 0, а якщо була помилка (наприклад, спроба виходу за ліву границю файлу), тоді повертається мінус 1.

9.7.2 Завдання для лабораторної роботи

1. Експертами компанії із перевезень вантажів на верблюдах встановлено, що верблюд безпечно для життя може перевезти 7640 лозин. Дані про кожен караван верблюдів знаходяться у файлі. Для кожного каравану набір даних є групою рядків: ім'я погонича, число верблюдів, число кошиків, перевезених кожним верблюдом, і число лозин у кожному кошику. Роздрукувати стан верблюдів. Наприклад,

ТОМ ДЖОНС верблюди в порядку

БОБ УЭЙТ неприпустимі вантажі:
верблюд 3 перевозить 7645 лозин
верблюд 8 перевозить 8006 лозин

2. У файлі знаходяться тільки цілі числа. Визначити, чи має послідовність чисел, що знаходяться у файлі, довжину, виражену непарним числом, і якщо так, то змінній middle присвоїти значення середнього елемента файлу. У іншому випадку присвоїти цій змінній значення першого числа файлу.

3. Створити файл, що містить відомості про книги в бібліотеці. Структура запису: шифр книги, автор, назва, рік видання, місце розташування (номер стелажа, полку). Передбачити можливість коректування файлу за кодом корегування, що вводиться, наприклад: 1 – видалити запис (за шифром XXX); 2 – додати новий запис; 3 – змінити запис (за введеним прізвищем автора і назвою книги);

4. У текстовий файл вставити пропуски таким чином, щоб кожен рядок мав довжину 80 символів (проміжки в рядку повинні бути вставлені рівномірно).

5. Кожен рядок файлу містить назву гірської вершини і її висоту. Використовуючи структуру для опису поняття ***вершина,*** одержати назву найвищої вершини за даними файлу.

6. Кожен рядок файлу містить такі дані: стать, ім'я, зріст. Роздрукувати середній жіночий зріст і ім'я найвищого чоловіка за даними файлу. Використовувати структуру для опису поняття ***людина.***

7. У файлі знаходиться текст програми мовою Сі з коментарями. Створити вихідний файл, у який переписати вміст вихідного файлу, забравши коментарі з тексту програми.

8. Написати програму, що працює в одному з двох режимів. Якщо в поточному каталозі є файл tabl_umn.txt, то роздрукувати по рядках його вміст. У іншому випадку створити файл із таким ім'ям і записати туди таблицю множення для чисел від 2 до 9.

9. Використовуючи структуру для визначення поняття студент, (що складається з полів П.І.Б, курс, група, оцінки в сесію) роздрукувати прізвища й імена відмінників першого курсу і частку їх від загального числа відмінників. (Дані знаходяться у файлі.)

10. Використовуючи структуру з полями стать, П.І.Б, вік, роздрукувати кількість дівчат з іменем "Elena" і імена тих, кому 19 років. (Дані знаходяться у файлі.)

11. Дано довільний текст обсягом не менше 1000 символів. Відредагувати його таким чином, щоб усі рядки, крім останнього, мали фіксовану довжину n. Правила редагування:

- слова не переносяться;
- розділовий знак не відокремлюється від слова, за яким він стоїть;

- рядки вирівнюються за рахунок пропусків, що рівномірно вставляються.

12. У текстовому файлі підрахувати кількість рядків, що починаються з букви Т.

13. У файлі знаходяться дійсні числа. Визначити кількість чисел у найдовшій зростаючій послідовності елементів файлу.

14. Написати програму записування у файл і читання з файлу елементів масиву структур для реєстрації автомашин з полями: марка машини; рік випуску; колір; номер.

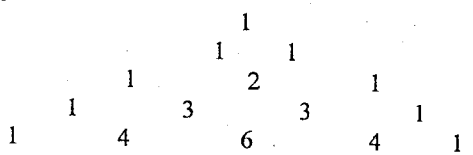
15. В текстовому файлі підрахувати кількість рядків, які закінчуються літерою 'S'.

16. Перевірити наявність балансу усіх видів дужок у текстовому файлі.

17. Дано текстовий файл F1. Переписати його вміст у файл F2, розбивши на рядки таким чином, щоб кожен рядок або закінчувався крапкою, або містив 40 літер, якщо серед них немає крапки.

18. Створити 2 файли, що містять відомості про гравців хокейних команд "Динамо" і "Шахтар". Структура записів файлів: прізвище, ім'я гравця; число закинутих шайб; число зроблених голевих передач.
За даними, що вибираються з цих файлів, створити новий файл, що містить дані про шість самих результативних гравців обох команд.

19. Трикутник Паскаля – таблиця чисел, що є біноміальними коефіцієнтами. У цій таблиці на бічних сторонах рівнобедреного трикутника стоять 1, а кожне з інших чисел дорівнює сумі двох чисел, що стоять над ним ліворуч і праворуч.



У рядку з номером n + 1 виписані коефіцієнти розкладання бінома $(a + b)^n$.
Написати програму, що працює в одному з двох режимів. Якщо в поточному каталозі існує файл "tr_pasc.txt", то роздрукувати його вміст у вигляді, як на рисунку. У іншому випадку створити файл із таким ім'ям і записати туди трикутник Паскаля n-го порядку. Параметр n (n<12) задається в командному рядку.

20. Написати програму порівняння двох файлів: повинен друкуватися перший рядок, яким вони різняться. Якщо файли ідентичні, то видати повідомлення.

21. Створити файл, що містить дані про те, які з 5 запропонованих дисциплін бажає слухати студент. Структура запису: прізвище студента; № групи; середній бал; 5 дисциплін, де '*' вказує на обрану дисципліну.
Створити файл, що містить дані про тих, хто бажає прослухати дисципліну XX. Якщо бажаючих більше 10, то відібрати тих студентів, у яких вищий середній бал.

22. Дано текст, у якому початок кожного абзацу відзначено символом '@'. Відредагувати цей текст за такими правилами:

а) перший рядок має відступ *k* позицій;

б) усі рядки тексту, крім останніх рядків абзаців, повинні мати фіксовану довжину *n;*

в) при редагуванні слова не переносяться. Розділовий знак не відокремлюється від слова, за яким він стоїть. Рядки вирівнюються за рахунок додатково внесених проміжків.

23. Створити файл, у який записати результати змагань за 6 видами спорту літньої Олімпіади 1992 р. Написати програму, що виконує таку функцію: видає на друк список призерів країни NNN.

24. Створити файл, у який записати результати змагань за 6 видами спорту літньої Олімпіади 1992 р. Написати програму, що виконує таку функцію: видати таблицю призерів (золото, срібло, бронза) із запитуваного виду спорту.

25. Створити файл, що містить дані про товари, які зберігаються на складі: шифр, найменування товару, кількість одиниць, вартість одиниці. Усі записи повинні бути відсортовані в порядку зростання шифру товару.
Мати можливість корегування за введеним кодом:

змінити/додати запис про товар із шифром XXX.

26. Створити файл, що містить дані про товари, які зберігаються на складі: шифр, найменування товару, кількість одиниць, вартість одиниці. Усі записи повинні бути відсортовані в порядку зростання шифру товару.
Мати можливість корегування за введеним кодом:

видалити запис про товар із шифром XXX.

27. Створити файл, що містить дані про товари, які зберігаються на складі: шифр, найменування товару, кількість одиниць, вартість одиниці. Усі записи повинні бути відсортовані в порядку зростання шифру товару.
Мати можливість корегування за введеним кодом: одержати інформацію про всі товари, у найменуванні яких присутній заданий ключ.

28. Створити файл, що містить дані про асортимент взуття в магазині. Структура запису: артикул, найменування, розмір, кількість пар, вартість однієї пари. Артикул починається з букви D для жіночого, з M – для чоловічого, із C – для дитячого взуття. Написати програму, що видає таку інформацію: наявність і вартість взуття артикула XX.

29. Написати програму, котра зчитує із текстового файлу три речення і виводить їх в зворотному порядку.

30. Написати програму, яка прочитує текст з файлу і виводить на екран тільки речення, що містять введене з клавіатури слово.

31. Написати програму, яка прочитує текст з файлу і виводить на екран тільки рядки, що містять двозначні числа.

32. Написати програму, яка прочитує англійський текст з файлу і виводить на екран слова, що починаються з голосних букв.

33. Написати програму, яка прочитує текст з файлу і виводить його на екран, міняючи місцями кожні два сусідні слова.

34. Написати програму, яка прочитує текст з файлу і виводить на екран тільки ті речення, що не містять ком.

35. Написати програму, яка прочитує текст з файлу і визначає, скільки в ньому слів, що складаються не більше ніж з чотирьох букв.

36. Написати програму, яка прочитує текст з файлу і виводить на екран тільки цитати, тобто речення, взяті в лапки.

37. Написати програму, яка прочитує текст з файлу і виводить на екран тільки речення, що складаються із заданої кількості слів.

38. Написати програму, яка прочитує англійський текст з файлу і виводить на екран слова тексту, що починаються і закінчуються голосними буквами.

39. Написати програму, яка прочитує текст з файлу і виводить на екран тільки рядки, що не містять двозначних чисел.

40. Написати програму, яка прочитує текст з файлу і виводить на екран тільки речення, що починаються з тире, перед яким можуть знаходитися тільки символи пропуску.

41. Написати програму, яка прочитує англійський текст з файлу і виводить його на екран, замінивши кожну першу букву слів, що починаються з голосної букви, на велику.

42. Написати програму, яка прочитує текст з файлу і виводить його на екран, замінивши цифри від 0 до 9 на слова «нуль», «один», ..., «дев'ять», починаючи кожне речення з нового рядка.

43. Написати програму, яка прочитує текст з файлу, знаходить щонайдовше слово і визначає, скільки разів воно зустрілося в тексті.

44. Написати програму, яка прочитує текст з файлу і виводить на екран спочатку питальні, а потім окличні речення.

45. Написати програму, яка прочитує текст з файлу і виводить його на екран, після кожного речення додаючи скільки разів зустрілося в ньому введене з клавіатури слово.

46. Написати програму, яка прочитує текст з файлу і виводить на екран спочатку речення, що починаються з однобуквених слів, а потім всі інші.

47. Списати процедуру ***Lines(t)***, яка порядково друкує вміст непорожнього текстового файлу ***t***, вставляючи в початок кожного рядка, що друкується, його порядковий номер (він повинен займати чотири позиції) і пропуск.

48. Вважаючи, що непорожній текстовий файл f розбитий на рядки, довжина кожного з який не перевищує 80 символів, описати процедуру ПЕРЕТВОР(f, f80), що доповнює короткі рядки файлу f пропусками справа, формує текстовий файл f80, усі рядки якого мають довжину 80 символів.

49. У текстовому файлі записана непорожня послідовність дійсних чисел, розділених пропусками. Розділити файл на два файли, в першому будуть числа, які кратні 10, а в другому – всі інші.

50. Описати процедуру ПРИСВ(t1, t2), що переписує в текстовий файл t1 вміст текстового файлу t2, попередньо вилучаючи з нього всі цифри.

9.7.3 Питання для самоконтролю

1. Що таке файловий буфер? Як пов'язаний буфер з потоком даних?
2. Назвіть фукцію для підготовки файлу до роботи. Що повертає ця функція?
3. Вкажіть різницю між двійковими і текстовими режимами доступу до файлу.
4. Яку операцію Сі можна використовувати, щоб забезпечити збіг розміру блоку даних, які читаються із файлу, з розмірами відповідної конструкції, в якій ви маєте намір зберігати ці блоки даних в пам'яті?
5. Вкажіть різницю між послідовним і довільним доступом до файлу. Яка функція використовується для встановлення індикатора позиції в задане місце в файлі?
6. Що таке логічний (фізичний) файл?
7. Поясніть правила введення із файлу рядкових і дійсних даних.
8. Поясніть правила форматного виведення даних символьного, рядкового і дійсного типів.
9. Що таке бінарний файл?
10. Як оголосити бінарний файл?
11. Які типи даних можна зберігати в бінарному файлі?
12. Яка функція використовується для зв'язку бінарного логічного файла програми з фізичним файлом?
13. Які функції використовуються для обміну між ОП і бінарним файлом?
14. Поясніть призначення, параметри і повертане значення для функцій роботи з бінарними файлами: fopen, fread, fwrite, feof, fseek, rewind, remove.
15. Поясніть процес обробки даних в вашій лабораторній роботі.

9.8 Лабораторна робота №8

9.8.1 Робота з графічним модулем мови Сі

Мета роботи. Навчитися працювати з різноманітними функціями графічного модуля. У запропонованих задачах потрібно, користуючись засобами графіки, відтворити на екрані необхідний рисунок. Бажано користуватись шрифтами, штриховкою та іншими засобами графічного модуля.

Теоретичні відомості

Для організації графіки фірма BORLAND випустила ряд універсальних графічних драйверів – програм, що забезпечують взаємодію програмної та апаратної частини комп'ютера. Графічні каталоги знаходяться в каталозі BGI і мають таке ж розширення, а саме *.bgi (BGI – Borland Graphics Interface). Ім'я файлу звичайно відображає, для роботи з яким саме графічним режимом він призначається. В нашому випадку це файл egavga.bgi. Крім шаблону в програмі, потрібно в меню Options →Linker→Libraries… поставити хрестик в рядку Graphics library, якщо це ще не зробили до вас.

Дисплей персонального комп'ютера може працювати в одному із двох режимів: текстовому і графічному. Розглянемо роботу в графічному режимі. Потрібно нагадати, що графічний драйвер мови Сі подібний до графічного драйвера мови Паскаль і, за деяким винятком, використовуються такі самі графічні функції.

Ініціювання графічної системи. Обробка помилок

1. Вибір драйвера і графічного режиму виконує функція detectgraph(&gd, &gm);. Відбувається тестування апаратури відеоадаптера, автоматичний вибір потрібного драйвера і графічного режиму. Цілочисельні змінні, покажчики, на які передаються функції в якості аргументів, повертають номер драйвера і номер режиму.

2. Завантаження драйвера, ініціалізацію графічної системи виконує функція initgraph (&gd, &gm, "шлях до BGI-файлу");.

Значеннями змінних gd, gm повинні бути номер потрібного драйвера і номер бажаного графічного режиму. Якщо файл з вказаним драйвером знайдений, то йому динамічно виділяється потрібна пам'ять і відбувається його завантаження. Якщо пам'яті немає в достатній кількості, функція завершується аварійно з кодом "-5". Після ж нормального розміщення драйвера в пам'яті, функція initgraph() організовує внутрішній графічний буфер, динамічно запитує блок ОП розміром 4 Кбайти (за замовчуванням). Потім відбувається ініціалізація графічних змінних і адаптер дисплея переводиться в графічний режим. При цьому екран очищується і покажчик поточної позиції встановлюється в лівий верхній куток. Якщо BGI-файли

знаходяться в поточній директорії, то як третій параметр функції initgraph() можна задати порожній рядок initgraph (&gd, &gm, " ");

3. Обробка помилкових ситуацій

Функція graphresult(); повертає код завершення останньої використаної графічної функції.

Для різних типів помилок передбачені свої коди завершення:

Константи	Код помилки	Повідомлення
GrOK	0	Немає помилки
grNoInitGraph	-1	Графіка неініційована
grNoLoadMem	-5	Дефіцит пам'яті для завантаження драйвера
……	……	……
grFotFound	-8	Файл шрифта не знайдений

Є можливість роздрукувати повідомлення, відповідне значенню коду помилки. Для цього використовується функція

grapherrormsg(код помилки);

Вона повертає покажчик на рядок, який містить опис коду завершення, переданого як аргумент.

Звичайна початкова послідовність дій при роботі з графікою виглядає таким чином:

```
# include <stdio.h>
# include <stdlib.h>
#include <graphics.h>
void main()
{
int gd, gm, error;
detectgraph( &gd, &gm );
initgraph( &gd, &gm, " " );
error = graphresult ( );
if ( error !=grOk )
{ puts( "помилка графіки " );
puts( " grapherrormsg( error ) );
exit (1);
}
   //Тіло програми
closegraph ( );
exit (0);
}
```

4. Переключення режимів

Для тимчасового переходу в текстовий режим дисплейного адаптера в графічній бібліотеці передбачена функція

restorecrtmode ();

Вона відновлює той текстовий режим, який був перед зверненням до функції initgraph( ); текстовий вміст екрана не зберігається, тому що його знищила функція initgraph( ). Повернутись до графічного режиму можна за допомогою функції setgraphmode(gm) ;

Аргументом її є цілочисловий номер режиму, допустимий для даного драйвера. Можна відновити колишній графічний режим, якщо його номер був завчасно визначений з допомогою функції getgraphmode ( ); і збережений в програмі.

Отримати максимальне значення номера графічного режиму можна за допомогою функції getmaxmode ( ).

Після закінчення роботи з графічною системою необхідно звільнити пам'ять, виділену для потреби графіки, почистити буфер відеоадаптера, відновити попередній текстовий режим. Всі ці дії виконує функція

closegraph ( );

5. Установка кольорів, шрифтів, стилів ліній і стилів зафарбовування

Відеорежимом називається набір параметрів, які підтримує відеокарта. Зображення формується із точок, які називаються пікселями (pixel). Самим відомим атрибутом відеорежиму є роздільна здатність екрана. Наприклад, 640×480. Початок системи координат знаходиться в лівому верхньому кутку екрана. Тому всі зображені об'єкти будуть мати додатні координати як по x, так і по y.

Відеорежими різняться глибиною пікселів (pixel depth). Цей параметр визначає кількість різних значень, яких набуває окремий піксель, і, відповідно, кількість відображених кольорів. Наприклад, в відеорежимі з глибиною 8 бітів кожний піксель може мати один із 256 різних кольорів ($2^8 = 256$). В режимах з 16-ти бітною глибиною пікселів підтримується зображення до 65 536 кольорів. Глибина пікселів звичайно дорівнює 4, 8, 16, 24 або 32 біти. Ми з вами будемо працювати в одному із самих простих графічних режимів – 640×480 з глибиною кольору 4 біти. Тобто в нашому розпорядженні буде 16 кольорів. Такий режим називається VGA.

Для вказання кольору в програмі можна користуватись константами переліковного типу або відповідними їм цілими значеннями.

Кольором фону керують за допомогою функції setbkcolor (колір); Поточне значення кольору лінії рисунка встановлюється функцією setcolor(колір);

Стандартна кольорова палітра така:

Black	0	чорний
Blue	1	синій
Green	2	зелений
Gray	3	сірий
Red	4	червоний
Magenta	5	малиновий
Brown	6	коричневий
Lightgray	7	світло-сірий
Darkgray	8	темно-сірий
Lightblue	9	світло-синій
Lightgreen	10	світло-зелений
Lightgyan	11	світло-бірюзовий
Lightred	12	світло-красний
Lightmagenta	13	світло-малиновий
Yellow	14	Жовтий
White	15	Білий

В графічному режимі при виведенні текстового повідомлення є можливість вибору одного із декількох шрифтів, розмірів виводимих символів і направлення тексту. Ці параметри задаються за допомогою функції

settextstyle (шрифт, направлення, розмір);

Допустимі значення для параметра шрифт

0	DEFAULT_FONT	(стандартний)
1	TRIPLEX_FONT	(типу триплекс: TRIP.CHR)
2	SMALL_FONT	(зменшений; в файлі LITT.CHR)
3	SANS_SERIF_FONT	(прямий; в файлі SANS.CHR)
4	GOTHIC_FONT	(готичний; в файлі GOTH.CHR)

Допустимі значення для параметра направлення:

0	HORIZ_DIR	(зліва направо)
1	VERT_DIR	(знизу вверх)

Аргумент, керуючий розміром шрифту, може змінюватись від 1 до 10. Для стандартного шрифту ця величина показує, в скільки разів треба збільшити кожний символ (це шрифт визначений на матриці 8×8, тобто , якщо розмір дорівнює 4, то символи будуть збільшені до матриці 32×32 пікселя). Для інших шрифтів цей параметр задає не лінійну, а експотенційну шкалу масштабування. Базовий варіант символу відповідає розміру, який дорівнює 4. Тому, якщо розмір дорівнює 7, то символи збільшуються в 2 рази, якщо 8 – то в 3 рази; якщо – 9, то в 4 рази.

Використання вертикального розташування рядка робить символи нижчими і ширшими, ніж в горизонтальному рядку. Це пов'язано ефектом неадекватності пікселів на деяких типах дисплеїв.

Символи тексту завжди виводяться суцільними тонкими лініями.

Для встановлення характеру і товщини ліній геометричних об'єктів використовується функція setlinestylle(вигляд, зразок, товщина);

Допустимі значення для параметра товщина

1	NORM_WIDTH	(лінія в один піксель);
3	THICK_WIDTH	(лінія в 3 пікселя).

Коди для параметра **вигляд** (тільки для курсово-лінійних графічних параметрів)

0	SOLID_LINE	(суцільна)
1	DOTTED_LINE	(з крапок);
2	CENTER_LINE	(з крапок і рисок);
3	DASHED_LINE	(пунктирна);
4	USERBIT_LINE	(визначається користувачем).

Параметр **зразок** задається тільки, якщо **вигляд** = 4 (інакше він – 0). З його допомогою можна задати будь-який періодично повторюваний рисунок лінії з періодом до 16 пікселів. Якщо в лінії потрібен світний піксель, в шаблоні задається біт, рівний 1; якщо ні – 0.

Наприклад, шаблон для пунктирної лінії може бути таким: 0×3333, що відповідає послідовності бітів 0011001001100l1.

В графічному режимі є можливість зафарбувати виділену на екрані замкнуту область яким-небудь способом.

Для встановлення стилю зафарбовування використовується функція setfillstylle (тип зафарбовування, колір);

Допустимі значення параметру тип зафарбовування

0	EMPTY_FILL	– штрихування кольором фону;
1	SOLID_FILL	– суцільне штрихування вказаним кольором;
2	LINE_FILL	– штрихування горизонтальними лініями;
3	LTSASH_FILL	– штрихування похилими лініями // //;
4	SLASH_FILL	– штрихування потовщеними лініями // //;
5	BKSLASH_FILL	–штрихування лініями \\\;
6	LTBKSLASH_FILL	– штрихування лініями \\\;
7	HATCH_FILL	– прямокутне горизонтальне штрихування;
8	XHATCH_FILL	– похиле штрихування;
9	INTERLEAVE_FILL	– похиле перекриваюче штрихування;
10	WIDE_DOT_FILL	– заповнення рідкими фарбами;
11	CLOSE_DOT_FILL	– заповнення густими фарбами.

Робота з вікнами і координатами

- Очистка екрана виконується функцією cleardevice( );
- Максимальні координати по вертикалі і горизонталі, відповідно:

getmaxy( ), getmaxx( ).

- Відкриття вікна в графічному режимі setviewport(x1,y1,x2,y2, clip);

де x1, y1 – координати лівого верхнього кутка;
x2, y2 – координати правого верхнього кутка;
clip – відсікання.

Якщо параметр clip дорівнює 1, то ті елементи зображення, які не поміщаються в вікні, будуть відсічені, якщо він дорівнює 0, то межі вікна ігнорується. При успішному виконанні цієї функції покажчик поточної графічної позиції переміститься на початок координат.

- Очистку графічного вікна виконує функція: clearviewport ( )
- Поточні координати отримуємо за допомогою функцій getx( ), gety( ).
- Перевстановлення покажчика позиції виконують функції moveto (x,y);
  moverel (dx, dy);
  де x,y – нові координати в системі координат вікна;
  dx, dy – приріст відносно старих координат в вікні.

Записування пікселя в відеопам'ять виконує функція: putpixel(x, y, колір).
Наприклад, в результаті виконання фрагменту

```
For ( i = 0; i < 160; i++ )
{   putpixel(i, 10, GREEN);   putpixel(160+i, 10, RED);  }
```

буде виведено зелено-червону пряму.
Виведення графічного тексту в вікно outtext(Sp); або outtext (x,y,Sp);
Перша – з поточної позиції, друга – з позиції (x, y).

Приклад: Суцільна діагональ червоного кольору товщиною в 3 пікселя на зеленому екрані:

```
{ setbkcolor(GREEN);
 sercolor(RED);
 setlinestyle(SOLID_LINE, 0, 3);
 line(0, 0, getmaxx( ), getmaxy( ));
// Після натискання на клавішу очищаємо екран
 getch();
 cleardevice();
// Визначимо вікно виведення і очищаємо його.
 setviewport (100, 100,getmaxx ( ) – 50, getmaxy ( ) – 50, 1);
 clearviewport ( );
 rectangle ( 100,100, getmaxx()-50, getmaxy()-50);
 settextstyle (TRIPLEX_FONT, HORIZ_DIR, 3);
 setbkcolor (BLUE);
 setcolor (WHITE);
//виводимо текст в лівому верхньому кутку вікна
```

```
    outtextxy(1,1 "А ось і нове вікно ");
    getch();
    closegraph();
    return( 0 );
}
```

Графічні примітиви

Основне призначення графічних примітивів – забезпечити програміста зручним набором програмних засобів для рисування різних геометричних об'єктів. Розглянемо спочатку функції, призначенні для рисування об'єктів контурного типу. Спосіб взаємодії виведених прямих ліній встановлює функція Setwritemode (режим). Можливі коди параметра режим

0	COPY_PUT
1	XOR_PUT

Якщо встановлено режим **0**, то накреслена лінія "затирає" те, що було на екрані. Якщо ж встановлено режим **1**, то для комбінування лінії з існуючим на екрані зображенням використовується операція виключення (або XOR в асемблері). Гарна властивість цієї операції в тому, що виведення двічі на одне і те саме місце лінії приводить до її стирання і відновлення початкового зображення на екрані (це використовується при програмуванні рухомих об'єктів).

Функція setwritemode( ) працює без "побічних ефектів" тільки з курсиво-лінійними зображеннями (в випадку кривих 2-го порядку бувають непрогнозовані наслідки роботи даної функції).

Лінія рисується будь-яким способом: line(x1,y1,x2,y2); або linerel(dx,dy); - лінія рисується з поточної точки до нових координат; або lineto(x, y);

Приклад: Два способи накреслення ромба.

1 спосіб:

```
setwritemode (XOR_PUT);   setcolor (GREEN);
moveto (100, 10); lineto(50,90); lineto(100,170);
setcolor (RED); lineto(150, 90); lineto(100, 10);
```

2 спосіб:

```
setbkcolor (WHITE);  setcolor (GREEN);
line (100, 10, 50, 90);  line ( 50, 90, 100, 170);
setbkcolor (RED);
line (100, 170, 150, 90); line(150, 90, 100, 10);
```

Контур прямокутника можна нарисувати за допомогою rectangle(x1,y1,x2,y2); де x1, y1 – координати лівого верхнього кутка; x2, y2 –координати правого нижнього кутка.

Якщо спробувати нарисувати квадрат, то на багатьох моніторах це не виходить тому, що піксель має форму прямокутника, витягнутого по

вертикалі. Необхідно виконати корегування кількості пікселів по горизонтальній і вертикальній сторонах квадрата. Істинні пропорції пікселів, необхідних для такого корегування, можна взнати за допомогою функції

getaspectratio(&xasp, &yasp);

Відношення xasp/yasp і є відношенням горизонтального і вертикального розмірів пікселя. Тому, якщо горизонтальний рядок квадрата має розмір G пікселів, то довжина вертикального рядка повинна бути:

(int)(G * (float)xasp/yasp);

Накреслити ламану лінію дозволяє функція drawpoly(кількість вершин, покажчик на масив цілих);

Кожна пара чисел масиву інтерпретується як пара координат чергової вершини ламаної.

Для зображення кривих використовують функції: circle(x, y, радіус) – для рисування кола;

для рисування дуги кола: arc(x, y, поч_ кут, кін_ кут, радіус).

Кути задаються в градусах і відраховуються проти часової стрілки. Нульовий кут відповідає горизонтальному направленню вектора зліва направо. Значення кутів перетворюється до еквівалентних значень з інтервалу [0..360]. Таким чином, arc(x, y, -45, 45, r) і arc(x, y, 675, -315, r) задають одну і ту ж дугу в четвертину кола.

Паралелепіпед з зафарбованою передньою гранню можна нарисувати, використовуючи функцію bar3d(x1, y1, x2, y2, глибина, 'dax');

Паралелепіпед обрамляється зовнішнім контуром. Якщо параметр **глибина** буде дорівнювати 0, то отримаємо обрамлений прямокутник. Якщо параметр **'dax'** дорівнює 1 (TOP_ON), то рисується верхня грань; якщо він дорівнює 0 (TOP_OFF), то верхня грань не прорисовується (це корисно, якщо треба поставити один на інший декілька паралелепіпедів).

Приклад: Паралелепіпеди, які стоять один на одному:

```
setbkcolor (WHITE);
setcolor (GREEN);
bar3d (250, 50, 250, 150, 15, 1);
bar3d (220, 150, 260, 180, 15, 1 );
bar3d (300, 150, 340, 180, 15, 0);
bar3d (300, 50, 340, 150, 15, 1);
```

Побудувати і зафарбувати багатокутник дозволяє функція

fillpoly(n, покажчик на масив цілих); де n –кількість вершин.

Координати кожної вершини задаються двома величинами цілого типу. Ця функція завжди з'єднує першу точку списку вершин з останньою, замикає контур і зафарбовує його. Шаблон і колір зафарбування може бути заданий функціями setfillstyle( ) і setfillpattern( ).

Звичайний еліпс рисує функція ellipse (x, y, поч_кут, кін_кут, rx, ry); rx, ry – довжини півосей еліпса в пікселях. Зафарбовується поточним кольором. Осі еліпса завжди паралельні осям координат.

Зафарбований еліпс з контуром можна отримати, виконавши функцію

fillellipse (x, y, rx, ry); де x, y – координати центра;

rx, ry – довжина півосей еліпса в пікселях. Зафарбовується поточним кольором. Осі еліпса паралельні осям координат.

Зафарбований круговий сектор з контуром рисує функція

pieslice(x, y, поч_ кут, кін_кут, радіус);

Після приведення кутів до діапазону [0..360] сектор рисується від меншого значення кута до більшого, тому неможливо зобразити сектор, який перетинає додатній напрямок осі 0X. Контур (дуга і два радіуси) рисується після зафарбування сектора, причому тип і товщина лінії беруться із установок функції setlinestile( ). Якщо потрібен сектор без контура, це можна створити так:

```
setcolor (BLACK); setbkcolor (BLUE);
setwritemode (XOR_PUT);
setfillstyle (WIDE_DOT_FILL, RED);
pieslice (200, 100, 45, 90, 50);
```

Приклад кругових секторів з написами на них.

```
setbkcolor (BLUE);
setcolor (RED);
setfillstyle (1, 3);
x= getmaxx ( ) / 2;
y= getmaxy( ) / 2;
pieslice (x, y, 270, 360, 100);
setfillstyle (1, 2);
pieslice (x, y, 0, 270, 100);
settextstyle (1, 0, 2);
moveto (x-20, y-40);   outtext("75%");
moveto (x+20, y+20);   outtext("25%");
```

Зафарбований еліптичний сектор з контуром, отримаємо, використавши функцію setctor (x, y, поч_кут, кін_кут, rx, ry);

Зафарбовування довільної замкнутої області виконується функцією

floodfill (x, y, границя); де x, y – координати точки всередині області.

Контур, який обмежує область, повинен бути замкнутим, інакше "фарба прольється" на іншу частину екрана. Колір контура повинен збігатися зі значенням параметра межі.

Зафарбовування (колір, тип) встановлюється функцією setfillstyle( ).

Приклад: сім концентричних червоно-блакитних кіл.

```
setlinestyle(0, 0, 1);
setbcolor(GREEN);
for (i =7;  i>=1;  i-- )
if ( i % 2 )
{       setcolor  (BLUE);
        circle ( x, y, 5*i );
        setfillstyle (  SOLID_FILL, BLUE);
        floodfill  ( x, y, BLUE );
}
else
{
   setcolor  ( RED );
   circle  ( x, y,  5*i );
   setfillstule  (SOLID_FILL, RED);
   floadfill   (x, y, RED);
}
```

Приклад: рух прямокутника зліва направо:

```
setlinestyle  (0, 0, 1);
setwritemode (XOR_PUT);  // для стирання ліній
setbkcolor  (BLUE);          //для   встановлення кольору
setfillstyle  (1, RED);
restangle  (1, 90, 10, 110);  //нарисували зафарбований
bar  (1, 90, 10, 10);          //червоний прямокутник
getch ();                      //після натискання будь-якої клавіші
                               //він "поїде" вправо
for ( i=11;  i<=300;  i++)
{
line  ( i- 10,  90,  i-10,  110 );        // знищуєм сторону зліва
line( i,  90, i, 110 );                   //дорисовуємо лінію справа.
```

Зафарбовування прямокутника виконує функція bar(x1, y1, x2, y2); Сторони його паралельні осям координат; контур не вимальовується. Стиль зафарбовування встановлюється функцією

setfillstyle( );

x1,y1 – координати лівого верхнього кутка,

x2,y2 – координати правого нижнього кутка.

Приклади розв'язання типових задач

Задача 1

Зобразити на екрані прямокутник і забезпечити можливість його пересування за допомогою клавіш керування курсором.

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <graphics.h>
#define BGIPATH "C:\\borlandc\\bgi"
// Директорія, що містить файли *.bgi. Можливо, на Вашій машині
// необхідно змінити значення BGIPATH.
#define ESC 27  // Коди клавіш, що повертаються getch()
#define UP_ARROW     72
#define DOWH_ARROW   80
#define RIGHT_ARROW  77
#define LEFT_ARROW   75
void Initialize ( ) ;
void MoveRec ( ) ;
 int GraphDriver;        // Номер драйвера
 int GraphMode;          // Номер графічного режиму
 int MaxX, MaxY;         // Максимальна роздільність екрана
 int MaxColors;          // Максимальне доступне число кольорів
 int ErrorCode;          // Код завершення графічної функції
void main ( )
{
   Initialize ( ) ;
   MoveRec ( ) ;
   closegraph ( ) ;
}
void Initialize ( )
{
   detectgraph (&GraphDriver, &GraphMode);
   initgraph (&GraphDriver, &GraphMode, BGIPATH);
   ErrorCode = graphresult ();
   if(ErrorCode != grOk )
   {
      printf ("Помилка ініціалізації: %s\n",
                          grapherrormsg ( ErrorCode ) );
      exit (1);
```

```
  }
  MaxColors=getmaxcolor ( ) +1;  //Визначаємо максимальне число
                                 // кольорів
  MaxX=getmaxx ( );              //Визначаємо розмір екрана
  MaxY=getmaxy ( );
}
void MoveRec ( )
{
  int i, x=0, y=0, x1, y1, x2, y2;
  int scalex=1, scaley=1;
  setbkcolor  (BLUE) ;
  setfillstyle (1, WHITE);
 //Зобразити  прямокутник у центрі екрана – початкова позиція
 x1=MaxX/2-70;    y1=MaxX/2-20;
 x2=MaxX/2+70;    y2=MaxY/2+20;
 bar(x1, y1, x2, y2);
 // Очікувати натискання клавіші. Вихід при натисканні
 //  клавіші <ESC>
 for ( i=getch0; i != ESC; i=getch0 )
 {
 if ( i ) continue;  // Алфавітно-цифрова клавіша  ( i=1; )
 switch ( i=getch0  )
 {
      case UP_ARROW      :  x=0;  y=y-3; i=1; break;
      case DOWH_ARROW    :  x=0;  y=y+3; i=1;  break;
      case RIGHT_ARROW   :  x=x+3;  y=0; i=1;  break;
      case LEFT_ARROW    :  x=x-3;  y=0; i=1;  break;
      default            :  i=0;
 }
 // чи не  вийшли за межі екранаі
     if (( x1+x*scalex<0) | | (x2+x*scalex>MaxX) )   x=1;
     if (( y1+y*scaley<0) | | (y2+y*scaley>MaxX) )   y=1;
 //Вилучити старе зображення – зафарбувати кольором фону
     setfillstyle (1, BLUE);
     bar (x1, y1, x2, y2);
 //Нарисувати прямокутник  у нових координатах, використовуючи
 //масштабування
     setfillstyle (1, RED);
     x1+=x*scalex;   x2+=x*scalex;
```

```
    y1+=y*scaley;  y2+=y*scaley;
    bar (x1, y1, x2, y2);
  }
}
```

Задача 2

Одержати на екрані картину "зоряного неба", причому заповнення екрана повинно виконуватись в два прийоми: спочатку верхня частина екрана, потім нижня.

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <graphics.h>
void Initialize();
void Pixel();
int GraphDriver, GraphMode, MaxX, MaxY, MaxColors, ErrorCode;
void main ()
{
 Initialize();
 Pixel();
 closegraph();
}
void Initialize()
{
 GraphDriver = DETECT;//Автовизначення типу графічного адаптера
 initgraph (&GraphDriver, &GraphMode,"");
 MaxColors=getmaxcolor ( ) + 1 ;
 MaxX = getmaxx () ; MaxY = getmaxy () ;
}
void Pixel()
{
   int x,y, h, w, color;
   long int  i;
   cleardevice();
   rand();  // Виведення у верхній екран
            // Вивести 2000 пікселів на екран
   for(i=0; i<2000; ++i)
   {
    x=1+random(MaxX-1);//Одержати координати чергового пікселя
    y=1+random(MaxY/2-1);
```

```
    //Вибір кольору і виведення пікселя на екран
    color = random (MaxColors);
    putpixel (x, y, color);
   }
   getch();
  // Виведення у нижній частині екрана з використанням
  // відсічення
  setviewport (0, MaxY/2, MaxX, MaxY, 1);
  rand();
  for ( i=0; i<20000; ++i)
  {
   x=1+random(MaxX - 1);
   y=1+ random(MaxY - 1);
   color = random (MaxColors);
   putpixel (x, y, color);
  }
  getch();
}
```

Для збереження зображень і їх подальшого відтворення використовують функції **getimage, imagesize, putimage**. Для того, щоб запам'ятати бітове зображення, потрібно виділити необхідний об'єм пам'яті прямокутника екрана за допомогою функції imagesize.

Її прототип:

unsigned far imagesize(int left, int top, int right, int bottom); де left, top – координати лівого верхнього кутка прямокутника;

right, bottom – координати правого нижнього кутка прямокутника.

Функція getimage запам'ятовує в ОП бітове зображення прямокутної частини екрана. Її прототип:

```
void far getimage(int left, int top, int right, int bottom,
                  void far *bitmap);
```

Перші 4 параметри подібні попереднім, а bitmap – безтиповий (void) покажчик на область ОП, в якій запам'ятовується зображення прямокутника. Функція putimage призначена для відтворення в заданому місці екрана зображення прямокутника, який запам'ятала функція getimage. Її прототип:

```
void far putimage(int left, int top, void far *bitmap, int op);
```

де left, top – координати лівого верхнього кутка прямокутника;

bitmap – покажчик на область ОП, де зберігається зображення;

op – умова виведення кольорів пікселів;

Можливі такі значення:

COPY_PUT	0	виведення копії зображення на екран

Для коду кольорів зображення:

XOR_PUT	1	виключне АБО
OR_PUT	2	об'єднуюче АБО
AND_PUT	3	логічне І
NOT_PUT	4	інверсія коду

При використанні значення op=0 на екран виводиться копія зображення. При виведенні на те саме місце екрана того ж зображення з параметром op=1 старе зображення стирається з екрана.

Задача 3

Створити програму для переміщення квадрата по діагоналі екрана, використовуючи функції для роботи з динамічною пам'яттю.

```
#include <graphics.h>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <dos.h>
void main()
{ int  gdriver = DETECT, gmode, errorcode;
   int n = 20, x, dx, y, dy, i;
   unsigned int size;
   void * ptr;
 initgraph(&gdriver, &gmode, " ");    // – ініціалізація графіки
// Орнамент – суцільне заповнення білим кольором:
 setfillstyle ( 1, 15 );
 bar(10, 30, 40, 60);                 // – рисування білого квадрата
// Запит ОП для квадрата: }
 size = imagesize(10, 30, 40, 60);    // – розмір необхідної ОП
 ptr = malloc(size);                  // – запит ОП
// Запам'ятовування в буфері бітового образу частини екрана:
 getimage(10, 30, 40, 60, ptr);
 setbkcolor ( 0 );                    // – колір фону чорний
 cleardevice();     // – очищення екрана і зафарбовування чорним кольором
 dx = getmaxx() / n ;                 // – крок переміщення квадрата по x
 dy = getmaxy() / n ;                 // – крок переміщення квадрата по y
  x = 0; y = 0;                       //– початкові координати квадрата
 for( i = 1; i <= n; i++ )
 { putimage(x, y, ptr, 0);            // – виведення білого квадрата
   delay (300);                       // – затримка програми
```

```
    putimage(x, y, ptr, 1);          // – стирання квадрата
      x = x + dx; y = y + dy;        // – нові координати квадрата
  }
  free(ptr);                         // – звільнення ОП для квадрата
  closegraph();                      // – закрити графічний режим
}
```

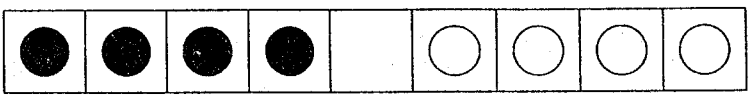
9.8.2 Завдання для лабораторних робіт

1. Змініть наведену вище програму так, щоб було можливо: збільшити або зменшити швидкість переміщення прямокутника по екрану; при пересуванні прямокутника по екрану за ним "тягнувся слід", тобто повинно бути видний весь попередній маршрут. Вмикання / вимикання "сліду" повинне виконуватися натисканням якої-небудь клавіші (наприклад, пропуску).

2. Змініть програму, наведену вище, так, щоб передбачити можливість зміни кольору прямокутника в процесі руху, наприклад, при натисканні функціональних клавіш (кілька різних кольорів). Забезпечіть також можливість прокручування (скролінга) зображення, тобто при досягненні прямокутником краю екрана, він повинен плавно перейти на протилежний край.

3. Одержати на екрані зображення діючого електронного годиника, що показує поточний час. Шаблони використовуваних цифр повинні відповідати звичайному для електронних годиників семисегментному шаблону.

4. Ігрова дошка має 9 лунок, в яких лежать 4 чорних і 4 білих кульки так, як показано на малюнку. Треба передвинути чорні кульки на місце білих, а білі – на місце чорних за найменше число ходів. Кульку можна пересувати або в сусідню з ним пусту лунку, або в пусту лунку, яка знаходиться безпосередньо за найближчим шаром.



5. Побудувати секторну і стовпчасту діаграму, яка показує співвідношення у введеному тексті таких груп символів:

а) голосних,
б) сонорних приголосних (л, м, н, р),
в) глухих приголосних,
г) парних глухих і дзвінких приголосних,
д) шиплячих приголосних,
е) інших букв: й, ч, ь .

6. В центрі екрану сидить "жук", який може переміщатися по прямій на вказану відстань і повертати праворуч і ліворуч на заданий кут. Крім того, в жука є "перо", яке може залишати слід, що відтворює рух жука. Ці дії виконуються за допомогою команд:

forward(x)	(переміщення на відстань x);
left(alfa)	(поворот наліво на alfa градусів)
right(alfa)	(поворот направо на alfa градусів)
PenUp	(підняти перо, тобто не залишати сліду)
PenDown	(опустити перо, тобто залишати слід)

Реалізувати ці команди у вигляді окремих функцій і з їх допомогою вивести на екран зображення.

7. В точці P знаходиться собака, а в точці Q – кішка. Відстань між ними рівно 100 м. Кішка біжить уздовж прямої S з постійною швидкістю V м/с. Собака біжить у напрямку кішки із швидкістю 10 м/с. Знайти траєкторію собаки за перші 15сек. гонитви.

Вказівка. Замініть криву ламаною: за першу секунду кішка пробігає відрізок QQ', а собака – PP'; за другу, відповідно, Q'Q" і P'P" і т.д. На початку кожної секунди собака "уточнює" рішення про напрям гонитви.

8. Здійснити обертання зафарбованого прямокутника в площині екрана навколо однієї з своїх вершин.

9. Зобразити на екрані приладову дошку, на якій знаходяться лампи, тумблери і прямокутні кнопки (з тінню). Забезпечити можливість: вмикати / вимикати лампи (супроводжувати звуком), маніпулювати тумблерами; натискувати на кнопки (тінь зникає) і повертати їх в початкове положення.

10. Чотири жуки, розташовані у вершинах квадрата із стороною a, одночасно починають повзти у напрямку до наступного за годинниковою стрілкою жука з постійною швидкістю v. Зобразіть траєкторії жуків за перші T секунд після початку руху. Вказівка: можна замінити траєкторії жуків ламаними – перші T секунд жуки повзуть прямо (перша ланка ламаної), потім повертають і ще T секунд повзуть прямо (друга ланка ламаної) і т.д.

11. На екрані зображено коло радіусом 20 з центром в точці (100,100). Необхідно надати можливість керування розміром і положенням кола

клавіша ">" збільшує радіус на 5 точок;

клавіша "<" зменшує радіус на 5 точок;

клавіші керування курсором " ⇐ "," ⇑ "," ⇒ "," ⇓ ";

переміщають коло по екрану.

12. Здійснити обертання зафарбованого трикутника в площині екрана навколо центра ваги цього трикутника.

13. На екрані – символ X. Клавішами керування курсора можна переміщати його по екрану. Спроба виходу за межі екрана повинна позначатися звуковим сигналом. За натисненням клавіші "пропуск "символ починає переміщатися із

залишенням за собою сліду. Повторне натиснення клавіші повертає його в колишній стан.

14. На екрані зображений прямокутник. Необхідно надати можливість керування його розмірами і положенням:

клавіша ">" збільшує ширину на 5 точок;
клавіша "<" зменшує ширину на 5 точок;
клавіша "+" збільшує висоту на 5 точок;
клавіша "-" зменшує висоту на 5 точок;

15. Побудувати графік функції

$f(t) = 3\ \mathrm{sint} + i * 2\ \sin 4t$ в комплексній площині.

16. Побудувати графіки функцій (полярна система координат):

$r = 2 \cos 4\varphi$ і $r = \cdot a \cos 2\varphi / \cos j$ при $(a > 0)$ · строфоїда.

17. Побудувати зрізи функції

$$F = x^4 \cdot 4x^2y^2 + y^4.$$

18. Побудувати еліпс, рівняння якого має вигляд:

$$f(x, y) = 5x^2 + 3y^2 - 18.$$

19. Побудувати графік функції (декартова система координат):

$y = 0.5x + \sin x$ на $[-2\pi, 2\pi]$;

20. Побудувати графік функції (декартова система координат):

$y = x^5 + x^4 - 5x^3 - 2x^2 + x - 4$ на $[-2,9; 2,9]$;

21. Побудувати графік функції (декартова система координат):

$y = |\sin x| - |\cos x|$ на $[0, \pi]$.

22. Побудувати графіки функцій (декартова система координат):

$y = (x - 3) / (x^2 + 2)$ на $[-1,0; 4,0]$.

23. На площині координатами своїх впорядкованих вершин заданий довільний багатокутник без самоперетину і точка а, що знаходиться зовні багатокутника. Визначити число вершин, видимих з точки а.

24. На площині сидять N жуків. Кожний з них з постійною швидкістю, рівною 1, повзе до наступного (N-й до першого). Якщо відстань між жуками стає меншою або рівною 1, переслідуваний жук вважається з'їденим. Зобразити еволюцію системи.

25. В будинку N поверхів і 3 ліфти. Кожний з ліфтів може бути вільним або зайнятим. На одному з поверхів стоїть людина. Він повинен викликати найближчий вільний ліфт, а якщо такого немає, найближчий зайнятий, який їде у бік поверху, де стоїть людина. Показати картину руху ліфтів.

26. Скласти програму для керування розмірами кола і його положенням на екрані. Початкове коло має центр в точці (100, 100) і радіус r = 20. Керування виконується клавішами:

« > » збільшує радіус кола на 5 точок,

« < » зменшує радіус кола на 5 точок;
клавіші керування курсором викликають переміщення кола у відповідному напрямку; «введення» завершує роботу програми.

27. Скласти програму для довільного рисування на екрані. Рисунок – це слід курсора, який переміщується за допомогою клавіш керування курсором. Необхідно забезпечувати можливість стирання зображення і режим, в якому курсор не залишає слід.

28. Задане шестирозрядне десяткове натуральне число зобразити цифрами за 9-сегментним шаблоном, який використовується при поштовій індексації.

29. Зобразити довільний будиночок (дерев'яний або цегляний), відобразити матеріал стін (колоди, цегла) і даху (шифер, черепиця і т. п.). Розміри будиночка, цегли, вікон і т.д. задаються введенням.

30. Розподіл швидкості вітру по кожному з восьми напрямів заданий масивом з восьми чисел. Побудувати "розу вітрів" з указанням напрямів.

31. Зобразити на екрані шахівницю (разом з буквено-цифровим позначенням горизонталей і вертикалей) і випадковим чином розставлені на ній шашки.

32. Розробити і одержати на екрані рисунок обкладинки якого-небудь підручника разом з назвою, прізвищами авторів, рисунками, що відображають суть предмету і так далі.

33. Побудувати фігуру Ліссажу: $x = \alpha\sin((\omega\upsilon t + \varphi x)$; $y=\beta\sin(\omega\upsilon t + \varphi y)$. Параметри α, β, $\omega\upsilon$, ωy, φx, φy задаються введенням.

34. За правилами креслення зобразити проекцію якого-небудь тіла і проставити розміри.

35. Одержати в аксонометрії або диметрії кубик Рубіка в будь-якому розібраному вигляді. Кольори елементарних кубиків можна зобразити різним штрихуванням або півтонами.

36. Зобразити вулицю, що йде вдалину і складається з двох рядів однотипних будинків. Врахувати невидимі частини будівель.

37. Одержати на екрані рисунок павутини з центром в довільній (заданій) точці, з довільним числом променів. Павутина утворена промінням і багатокутниками.

38. Зобразити на екрані достатньо складну квітку (жоржина, ромашка з випадковим числом пелюсток, калина, волошка і так далі).

39. Розробити і реалізувати на екрані вітальну листівку до свята (Новий рік, день народження і так далі) з динамічними елементами (миготлива гірлянда, салют, свічки, що горять, миготливі написи і так далі).

40. Розробити і реалізувати на екрані політичний або екологічний плакат з динамічними (рухомими або миготливими) ефектами.

41. Розробити і реалізувати на екрані заставку до однієї з популярних телепрограм («Поле чудес», «КВК» і так далі).

42. Зобразити на екрані працюючий годинник із стрілочним індикатором (3 стрілки). За відсутності таймера — імітувати його.

43. Зобразити діючий конвеєр, що транспортує будь-які однотипні предмети.

44. Екран — судина з киплячою рідиною. На дні у випадковій точці утворюється кулька; при русі вгору вона росте, а дійшовши до поверхні — лопається. Якщо дві кульки стикаються, вони зливаються в одну. Реалізувати цей процес.

45. Зобразити на екрані довільний пейзаж, натюрморт або інтер'єр. Потім випадковими точками або прямими заповнювати екран до повного зникнення картини (зручніше реалізовувати ефект керуванням палітрою). Передбачити зворотний процес: світанок або прояв фотозображення.

46. Побудувати в центрі екрану трикутник заданого розміру і заповнити його довільним (жорстко заданим, випадковим або заданим з клавіатури) зображенням. Виробити багатократне дзеркальне віддзеркалення зображення від кожної сторони трикутника до заповнення всього екрана.

47. Одержати на екрані картину, яку бачить машиніст рухомого потягу: рейки, шпали, стовпи, придорожні будови і так далі. Врахувати повороти, стрілки, зміну швидкості потягу, зустрічні склади і так далі.

48. Зобразити фінальну сцену якої-небудь театральної вистави: на екрані: довільне зображення; справа і зліва на нього насуваються завіси. На завісі — напис: «КІНЕЦЬ».

49. Зобразити модель атома довільного хімічного елемента: ядро і електрони, що обертаються по своїх орбітах. Розподіл електронів по орбітах задається. У підготовленому файлі зберігається розподіл електронів по орбітах для всієї системи Менделєєва; користувач задає тільки номер або позначення хімічного елемента.

50. Реалізувати на екрані картину святкового салюту: зліт, розриви, падіння піротехнічних ракет і їх осколків (з декількох стволів). Світлові ефекти бажано супроводжувати звуковими.

9.8.3 Питання для самоконтролю

1. Як виконується підключення графічного драйвера в мові Сі?
2. Яка є функція затримки зображення на екрані?
3. Як виконується переміщення фігур по екрану?
4. Як виводиться зображення тексту на екрані?
5. Як працювати з шрифтами?

10 Методичні вказівки до виконання контрольної роботи студентами заочної форми навчання

Навчальним планом для студентів заочної форми навчання передбачено виконання контрольної роботи з дисципліни «Основи програмування і алгоритмічні мови». Конкретна тематика завдань узгоджується з викладачем під час установчої сесії.

Студенти заочної форми навчання для захисту повинні подати контрольну роботу оформлену належним чином: робота повинна мати стандартний титульний лист, де вказується назва університету, кафедри, дисципліни, що вивчається, прізвище та ім'я студента і викладача. На другій сторінці повинно бути завдання, яке студент отримує від викладача після закінчення курсу лекцій. Завдання включає в себе чотири задачі за різними темами і два теоретичних питання. Кожна задача повинна мати алгоритм розв'язання мовою блок-схем. Правила написання блок-схем наведені в посібнику автора Круподьорової Л.М. «Алгоритмізація і програмування», виданому в ВНТУ в 2005 році. До кожної задачі повині бути дані пояснення щодо її розв'язання і розумна кількість коментарів. Результат роботи задачі повинен бути роздрукований з екрана комп'ютера засобами Paint та Word. Теоретичні питання повинні містити основні положення розглядуваного питання та приклади типових фрагментів програм. Вважаючи на те, що тематика задач для студентів заочної форми навчання постійно змінюється, автори не наводять перелік типових завдань для студентів заочної форми навчання.

11 Тести

11.1 Оператори мови Сі

1. Основні типи в Сі
   а) void, char, float, int, double;
   б) integer, longint, float, real;
   в) float, int, char, void, long.
2. Чи є правильним такий запис в Сі: for (int i = 0; i < 10; i++)
   а) так;
   б) ні.
3. Сі – це
   а) структурована мова;
   б) не структурована мова;
   в) блочно-структурована мова.
4. Чи правильним є такий запис:
   int x = 19;
   char ch = '1';
   float f = 20;

```
  void main(void)
{
 ch = x;      x  = f;
 f = ch;      f = x;
}
```

а) так;

б) ні.

5. Інкрементація – це:

а) збільшення на одиницю;

б) зменшення на одиницю.

6. Що виведе така програма:

```
#include <stdio.h>
  int main(void)
  {
 int x = 7;
 x  = x<<1;          x = x<<3;
 x = x<<2;           x = x>>1;
 x = x>>2;
 printf("%d",x);
 return 0;
}
```

а) 7;

б) 96;

в) 24.

7. Найдіть альтернативу такому виразу: y = x > 9 ? 100 : 200

а) if(x > 9) y = 100; else y = 200;

б) if(x > 9) y =200; else y = 100;

в) if(x > 9) y = 100; else exit(200);

8. В мові Сі оператор switch допускає до

а) 34 операторів case;

б) 257 операторів case;

в) 16384 операторів case.

9. Цикли do-while і while відрізняються:

а) тим, що в циклі do - while спочатку виконуються оператори в тілі цикла, а потім перевіряється умова виходу, а в циклі while навпаки;

б) нічим;

в) тільки записом.

10. Що виведе така програма:

```
#include <stdio.h>
```

```
void main(void)
{
 int mas[100],*pmas,i;
 pmas = (int *)  mas;
 for (i = 0; i < 100;i++)      *(pmas + i) = i;
 printf("%d",*pmas);
}
```

а) 100;
б) 0;
в) 1.

11. Найдіть помилку в такій програмі:

```
void main(void)
{
 int x,*p;
 *p =x;
}
```

а) нема помилки;
б) є, покажчик *p має невизначений адрес, тому не можна присвоїти *p = x;
в) є, функція main() не повертає значення.

12. Бітові поля – це:
а) нічого нового;
б) звичайна структура;
в) доступ до окремих бітів.

13. Об'єднання – це:
а) область пам'яті яка використовується для збереження змінних різних типів;
б) різновид структури;
в) нічого, бо об'єднань не існує.

14. Що виведе програма

```
#include <stdio.h>
void main(void)
{
 int x = 10;
 printf("%d",&x);
}
```

а) нічого, бо програма містить помилки;
б) 10;
в) адресу змінної x.

11.2 Масиви і рядки

1. Дано фрагмент коду:

```
int count[10], i;
for (i = 0; i < 100; i++)
{
    count[i]= i;
    printf("%d ", count[i]);
}
```

Що відбудеться під час виконання програми?

а) компілятор видасть повідомлення про помилку;

б) буде надруковано 10 цифр;

в) буде надруковано 99 цифр + "сміття";

г) буде надруковано 100 цифр + "сміття";

д) буде надруковано 9 цифр.

2. Оголошений масив:

```
int sample[];
```

Правильно, що вирази sample і &sample[0] еквівалентні?

а) так;

б) ні;

в) свій варіант.

3. Назвіть три способи передачі масиву як аргумент функції.

4. Яким способом допустимо створити двовимірний цілочисловий масив розміром 10 на 10?

а) int a[ ][ ] = new int[10,10];

б) int a[10][10]= new int[ ][ ];

в) int a[ ][ ] = new int[10][10];

г) int [ ]a[ ] = new int[10][10];

д) int [ ][ ]a = new int[ ][ ].

5. Рядок – це :

а) об'єкт типу string;

б) масив символів, що завершується нульовим байтом;

в) свій варіант.

6. Дано фрагмент коду:

```
void func (double x[32])
{
/* ..... */
```

}

Функції передано:

а) масив, що складається з 32 елементів;

б) масив, що складається з 31 елементу;

в) покажчик на перший елемент масиву.

7. Чому дорівнює об'єм пам'яті, зайнятий цілочисловим масивом типу int, який складається з 5 рядків і 10 стовпців?

а) 200 байтів;

б) 100 байтів;

в) 50 байтів.

8. Функція одержує як аргумент двовимірний масив, що складається з 10 рядків і 10 стовпців. Назвіть допустимий запис.

а) void func(int x[][]);

б) void func(int x[10][]);

в) void func(int x[][10]);

г) void func(int x[10][10]).

9. Значенням якого виразу є елемент a[1][2]?

а) *((int *)a + 4);

б) *((int *)a + 12);

в) *((int *)a + 3);

г) *((int *)a + 2).

10. Дано масив, що проініціалізований

char str[10]= "Я люблю Сі";

Оголосіть його іншим способом.

11. Для розв'язання яких з наступних задач потрібні масиви, а в яких можна обійтися і без них?

а) дано 50 чисел. Знайти їх середнє арифметичне;

б) дано 50 чисел. Визначити, скільки серед них відмінних від останнього числа;

в) дано 100 чисел. Надрукувати спочатку всі від'ємні, а потім всі інші;

г) дано число *a*. Визначити перший від'ємний член послідовності

$x_1, x_2, x_3,..$, де $x_1 = a$, $x_n = tg(x_n\text{-}1)$.

12. Використана індексація покажчика:

```
void putst (char *s)
{
```

```
    register int t;
    for (t = 0; s[t]; t++) putchar (s[t]);
}
```

Змініть доступ до елементів рядка за допомогою покажчика.

13. Дано фрагмент коду:

```
double a[15][8], b[15][8];
bool t;
```

Які з вказаних операцій допустимі?

а) a = b;
б) a += b;
в) t = a!=b;
г) scanf("%d", &a);
д) a[0]= a[15];

14. Чому при роботі з масивами надається перевага використанню покажчиків?

а) покажчики зручніші у використанні;
б) операції адресної арифметики виконуються швидше, ніж індексація;
в) покажчики дозволяють контролювати вихід за межі масиву;
г) свій варіант.

11.3 Покажчики. Динамічні масиви

1. Яке оголошення є оголошенням покажчика?
   а) int p*;
   б) int *p;
   в) int &p.
2. Який фрагмент коду робить приведення типу без втрат?
   а) int *p; long a = (long)p;
   б) int *p; long a = (int)p;
   в) int *p; long a = (char)p.
3. Що виконує оператор new?
   а) створює нову змінну типу int і повертає покажчик на цю змінну;
   б) створює нову змінну будь-якого типу і повертає покажчик на цю змінну;
   в) створює новий клас.
4. Для чого використовується NULL?
   а) для того, щоб відзначити, що покажчик ні на що не вказує;
   б) для того, щоб відзначити, що місце в пам'яті на яке вказує покажчик, містить нульове значення;

в) для того, щоб обнулити місце в пам'яті, на яке вказує покажчик.

5. Що виконує оператор delete?
   а) видаляє динамічну змінну, і повертає використовувану нею пам'ять до "купи";
   б) видаляє покажчик на динамічну змінну, а сама змінна продовжує існувати;
   в) знищує пам'ять, на яку вказує покажчик.
6. "Купою" (heap) називається:
   а) спеціальна зарезервована область пам'яті, використовувана для динамічних змінних;
   б) так програмісти називають корзину;
   в) місце куди відкладаються тимчасово не використовувані динамічні змінні.
7. Що робить така інструкція?

```
p = &v;
```

   а) робить p вказуючим на v;
   б) робить v вказуючим на p ;
   в) ініціалізує змінну p .
8. Чи еквівалентні такі присвоєння?

```
p2 = p1;
*p2 = *p1;
```

   а) так;
   б) ні;
   в) не знаю.
9. Що робить така інструкція?

```
p1 = new int;
```

   а) створює нову змінну типу int, присвоюючи її адресу p1;
   б) робить p1 покажчиком на нову безіменну змінну;
   в) створює нову змінну типу int з ім'ям p1.
10. Яким буде виведення на екран такого фрагмента коду?

```
v1 = 0;
p1 = &v1;
*p1 = 42;
cout << v1 << *p1;
```

   а) 00;
   б) 042;
   в) 4242.
11. Чи є a і p покажчиками одного типу?

```
int a[10];
typedef int* IntPrt;
```

IntPrt p, p2
а) не знаю;
б) ні;
в) так.

12. Чи коректна інструкція видалення масиву a? delete a;
а) іноді;
б) так;
в) ні.
13. Чому еквівалентний фрагмент коду? int *p; p++;
а) збільшенню адресованого значення;
б) збільшенню адреси значення;
в) зміна покажчика.
14. Чи є помилка у фрагменті коду? void *p; int *a; p = a;
а) іноді;
б) так;
в) ні.
15. У чому недолік фрагмента коду?
int *p = new int; int *a = new int; p = a;
а) у втраті пам'яті;
б) у втраті покажчика;
в) ні у чому.
16. Змінні створені за допомогою оператора new називаються...
а) статичними;
б) динамічними;
в) автоматичними.
17. Чи коректний вираз? int *p; p = 28;
а) так;
б) ні;
в) так, але позбавлений значення.

Література

1. М. Эллис, Б. Страструп Справочное руководство по языку C++ с комментариями: Пер. с англ. - Москва: Мир, 1992.
2. Стенли Б. Липпман C++ для начинающих: Пер. с англ. В 2 Т. - Москва: Унитех; Рязань: Гэлион, 1992.
3. Бруно Бабэ Просто и ясно о Borland C++: Пер. с англ. - Москва: БИНОМ, 1994.
4. В.В. Подбельский Язык C++: Учебное пособие. - Москва: Финансы и статистика, 1995.
5. Ирэ Пол Объектно-ориентированное программирование с использованием C++: Пер. с англ. - Киев: НИИПФ ДиаСофт Лтд, 1995.
6. Т. Фейсон Объектно-ориентированное программирование на Borland C++ 4.5: Пер. с англ. - Киев: Диалектика, 1996.
7. Т. Сван Освоение Borland C++ 4.5: Пер. с англ. - Киев: Диалектика, 1996.
8. Г. Шилдт Самоучитель C++: Пер. с англ. - Санкт-Петербург: BHV-Санкт-Петербург, 1998.
9. У. Сэвитч C++ в примерах: Пер. с англ. - Москва: ЭКОМ, 1997.
10. К. Джамса Учимся программировать на языке C++: Пер. с англ. - Москва: Мир, 1997.
11. Скляров В.А. Язык C++ и объектно-ориентированное программирование: Справочное издание. - Минск: Вышэйша школа, 1997.
12. Х. Дейтел, П. Дейтел Как программировать на C++: Пер. с англ. - Москва: ЗАО "Издательство БИНОМ", 1998.- 640с.
13. Климова Л.М. СИ++. Практическое программирование. – Москва:"Кудиобраз", 2001.- 586с.

Додаток А

Повідомлення про помилки

У текстах повідомлень про помилки використовується ряд стандартних змінних. Їх перелік поданий в табл. А.1.

Таблиця А.1 – Змінні, використані в повідомленнях про помилки

Змінна	Призначення змінної
'argument'	Аргумент
'class'	Ім'я класу
'errorcode'	Код помилки
'filename'	Ім'я файлу з розширенням або без нього
'function'	Ім'я функції
'group'	Ім'я групи
'identifier'	Ідентифікатор
'language'	Найменування мови
'linenum'	Номер рядка у файлі
'member'	Ім'я елементу даних або функції
'message'	Повідомлення
'module'	Ім'я модуля
"num'	Числове значення
'number'	Реальне число (дійсне)
'option'	Параметр (режим)
'parameter'	Ім'я параметра
'path'	Ім'я шляху
'reason'	Висновок, основа, виведення
'segment'	Ім'я сегменту
'size'	Розмір
'specifier'	Тип описувача (специфікатора)
'symbol'	Ім'я символу
'type'	Ім'я типу
'XXXXh'	Чотирирозрядне шістнадцятіркове число, що стоїть перед символом h

Повідомлення про помилки на етапі компіляції

Таблиця А.2 – Критичні (фатальні) помилки

Bad call of intrinsic function	Помилковий виклик вбудованої функції
Compiler table limit exceeded	Перевищена межа таблиці компілятора
Error directive: message	Помилка директиви: message
Error writing output file	Помилка запису у вихідний файл
Irreducible expression tree	Недопустиме дерево виразів. Вираз довгий і складний. Треба спростити його
Register allocation failure	Збій при розподілі регістрів
Unable to create output file filename	Неможливо створити вивідний файл
Unable to open filename	Неможливо відкрити файл filename
Unable to open input file filename	Неможливо відкрити ввідний файл
Out of memory	Вичерпана ОП

Таблиця А.3 – Помилки (ERROR)

(expected	Очікується (
) expected	Очікується)
, expected	Очікується,
< expected	Очікується <
{ expected	Очікується {
} expected	Очікується }
286/287 instructions not enabled	296/287 команди не дозволені
Ambiguity between functionl and function2	Двозначність між functionl і function2
Ambiguous member name name	Двозначне ім'я пам'яті name
Array allocated using new may not have an initializer	Не можна використовувати нову ініціалізацію масиву
Array bounds missing]	При описанні масиву відсутня закриваюча квадратна дужка
Array must have at least one element	Масив повинен мати хоча б один елемент
Array of references is not allowed	Посилання на масив недопустиме

Продовження таблиці А.3

Array size too large	Розмір масиву більше допустимого.
Assembler statement too long	Довжина оператора асемблера більше допустимої (більше 480 байтів)
Attempting to return a reference to local variable identifier	Спроба повернути посилання на локальну змінну identifier
Bad define directive syntax	Неправильний синтаксис директиви define
Bad file name format in include directive	Неправильний формат імені файлу в директиві include
Bad ifdef directive syntax	Неправильний синтаксис директиви ifdef
Bad syntax for pure function definition	Неправильний синтаксис чистої (реєнтабельної) функції
Bad undef directive syntax	Неправильний синтаксис директиви undef
Bit field cannot be static	Бітове поле на може бути static
Bit field tooTarge	Бітове поле дуже довге
Bit fields must be signed or unsigned	Бітове поле повинно бути цілим, із знаком або без нього
Bit fields must have integral type	Бітове поле повинно мати хоча б один біт
Bit fields must have integral type	Бітове поле повинно бути цілочислового типу
Body already defined for this function	Текст цієї функції вже визначений
Call of nonfunction	Звертання не до функції
Cannot access an inactive scope	Неможливий доступ до неактивної області
Cannot allocate a reference	Не можна розмістити посилання
Cannot call main from within the program	Не можна викликати main зсередини програми
Cannot cast from typel to type2	Не можна привести typel до type2
Cannot convert typel to type2	Не можна перетворювати typel в type2
Cannot define a pointer or reference to a reference	Не можна визначати покажчик або посилання на покажчик за допомогою define
Cannot overload main	Не можна перенавантажувати main

Продовження таблиці А.3

Cannot use tiny or huge memory model with Windows	Не можна використовувати під Windows модель пам'яті tiny або huge
Case .bypasses initialization of local variable	Не можна обходити ініціалізацію локальної змінної
Case outside of switch	Case поза оператором switch
Case statement missing:	У оператора case відсутня :
Character constant must be one or two characters long	Символьна константа може бути завдовжки в 1 або 2 символи
Compound statement missing }	У складеному операторі відсутня }
Conflicting type modifiers	Конфліктуючі типи модіфікаторів покажчика
Constant expression required	Обов'язковий константний вираз
Constant variable variable must be Initialized	Константна змінна variable повинна бути ініціалізована
Conversion of near pointer not	Перетворення покажчика типу near неприпустимо
Conversion operator cannot have a return type specification	Оператор перетворення не може повернути тип описувача
Could not find a match for argument	Не знайдений зв'язок для аргументу(ів)
Could not find file filename	Файл filename не знайдений
Declaration does not specify a tag or An identifier	Оголошення не має специфікації або ідентифікатора
Declaration is not allowed hare	Оголошення тут неприпустимо
Declaration missing	У описі полів структури або об'єднання відсутня ;
Declaration syntax error	Помилка в синтаксисі оголошення
Declaration terminated Incorrectly	Некоректне переривання оголошення
Declaration was expected	Очікується оголошення
Default argument value redeclared	Переоголошено значення аргументу за замовчуванням
Default argument value redeclared for Parameter parameter	Переоголошено значення аргументу за замовчуванням для параметра parameter

Продовження таблиці А.3

Default expression may not use local Variables	За замовчуванням вираз не може використовувати локальних змінних
Default outside of switch	За замовчуванням (default) поза switch
Default value missing	За замовчуванням значення відсутнє
Default value missing following	За замовчуванням відсутнє значення параметра parameter
Define directive needs an identifier	У директиві define відсутній ідентифікатор
Define array size missing]	При оголошенні розміру масиву відсутня]
Division by zero	Ділення на нуль. Початковий файл містить ділення або залишок від ділення константного виразу на нуль
Do statement must have while	Оператор do повинен мати while
Do-while statement missing (	Відсутня (у операторі do-while
Do-while statement missing)	Відсутня) у операторі do-while після умовного виразу
Do-while statement missing ;	Відсутня ; у операторі do-while
Duplicate case	Дублювання оператора case.
Enum syntax error	Синтаксична помилка при визначенні enum
Expression expected	Очікується вираз
Expression of scalar type expected	Очікується вираз або скалярний тип
Expression syntax	У виразі синтаксична помилка
Extern variable cannot be initialized	Зовнішня змінна не може ініціалізувати
Extra parameter in call	Зайвий параметр у виклику функції
Extra parameter in call to function	Зайвий параметр при виклику функції
File must contain at least one external declaration	Файл повинен мати хоча б одне зовнішнє оголошення
File name too long	Ім'я файлу дуже довге (більше 64 символів)
For statement missing (	Відсутня (у операторі циклу for
For statement missing)	Відсутня) у операторі циклу for
For statement missing ;	Відсутня ; у операторі циклу for

Продовження таблиці А.3

Function function cannot be static	Функція function не може бути static
Function function should have a Prototype	Функція function повинна мати прототип
Function call missing)	При виклику функції відсутня)
Function calls not supported	Виклик функції не підтримується
Function definition.cannot be a typedefed declaration	Визначення функції не може бути оголошено за допомогою typedef
Function should return a value	Функція повинна повернути значення оператором return
Functions may not be part of a struct Or union	Функція не може бути частиною структури або об'єднання
Global anonymous union not static	Глобальний анонім типу union не має специфікатора static
Goto bypasses initialization of a local variable	Goto обходить ініціалізацію локальної змінної
Goto statement missing label	У оператора goto відсутня мітка
Identifier expected	Очікується ідентифікатор
Identifier identifier cannot have a type qualifier	Ідентифікатор identifier не може мати тип уточнювача (специфікатора)
If statement missing (	Відсутня (у операторі if
If statement missing)	Відсутня) у операторі if
Illegal character character (0xvalue)	Неприпустимий символ character (0xvalue) у вхідному файлі
Illegal initialization	Неприпустима ініціалізація (константний вираз)
Illegal octal digit	Неприпустима вісімкова цифра (наприклад, 8 або 9)
Illegal pointer subtraction	Неприпустиме віднімання покажчиків
Illegal structure operation	Неприпустима операція над структурою
Illegal to take address of bit field	Неприпустиме отримання адреси бітового поля
Illegal use of floating point	Неприпустиме використання плаваючої точки
Illegal use of member pointer	Неприпустиме використання елементу покажчика

Продовження таблиці А.3

Illegal use of pointer	Неприпустиме використання покажчика
Implicit conversion of typel to type2 not allowed	Неявне перетворення з type1 в тип type2 не дозволяється
Improper use of typedef identifier	Неправильне використання ідентифікатора, визначеного в typedef
Incompatible type conversion	Несумісне перетворення типів. Наприклад, функції в нефункцію, структури або масиву в скаляр або назад, а також дійсного числа в покажчик або назад
Incorrect command-line option	Некоректний параметр в командному рядку
Incorrect configuration file option	Некоректний параметр файлу конфігурації
Incorrect number format	Некоректний формат числа
Incorrect use of defaut	Некоректне використання оператора default. Наприклад, немає двокрапки (:) після ключового слова "default"
Invalid combination of opcode and	Неприпустима комбінація коду операції
Invalid indirection	Неприпустиме посилання (*)
Invalid macro argument separator	Неприпустимий роздільник макроаргументів
Ivalid pointer addition	Додавання покажчиків неприпустимо
Invalid use of dot	Неприпустиме використання крапки. Ідентифікатор повинен знаходитися безпосередньо за крапкою (.)
Last parameter of operator must have type int	Останній параметр в operator повинен мати тип int
Linkage specification not allowed	Розширена специфікація не дозволена
Lvalue required	Необхідна адреса змінної. У лівій частині оператора присвоювання повинен знаходитися адресний вираз
Macro argument syntax error	Аргумент в макроозначенні повинен бути ідентифікатором
Macro expansion too long	Макророзширення дуже довге (більше 4096 символів)

Продовження таблиці А.3

main must have a return type of int	main повинна повернути тип int оператором return
Member identifier expected	Очікується ідентифікатор елементу
Member pointer required on right side Of .* or ->*	Елемент покажчик обов'язково повинен бути з правого боку від .* або ->*
Memory reference expected	Очікується посилання на елемент
Misplaced break	Неприпустиме місцерозташування оператора break: поза циклом або оператором switch
Misplaced continue	Неприпустиме місцерозташування оператора continue: поза циклом або оператором switch
Misplaced decimal point	Неприпустиме місцерозташування десяткової крапки. Наприклад, усередині значення порядку
Misplaced elif directive	Неприпустиме місцерозташування директиви elif
Misplaced else	Неприпустиме місцерозташування оператора else, поза відповідного йому оператором if
Misplaced else directive	Неприпустиме місцерозташування директиви else
Misplaced endif directive	Неприпустиме місцерозташування директиви endif
Multiple declaration for identifier	Багатократне оголошення identifier
Must take address of a memory Location	Повинна бути задана адреса розміщення в ОП
Need an identifer to declare	У оголошенні немає ідентифікатора
new and delete not supported	new і delete не підтримуються
No: following the ?	Немає : після ? у умовному виразі
No file name ending	Немає обмежувача імені файлу. Ім'я файлу в сператорі #include не містить закриваючої лапки (") або кутової закриваючої дужки (>)
No file names given	Не дано ім'я файлу
No type information	Немає інформації про тип

Продовження таблиці А.3

Nonportable pointer conversion	Неприпустиме перетворення покажчика
Not a valid expression format type	Неприпустимий вираз для типу формату
Not an allowed type	Недозволений тип
Numeric constant too large	Числова константа дуже довга
Only member functions may be const or volatile	Тільки параметр функції може бути const або volatile
Operand of delete must be non-const pointer	Операнд функції delete не повинен бути покажчиком-константою
Operator [] missing]	У операторі [] відсутня]
Operator delete must return void	Оператор delete повинен повернути тип void
Operator must be declared as function	Оператор повинен бути оголошений як function
Operator must be declared with one or Two parameters	Operator повинен бути оголошений з одним параметром або без параметрів
Overlays only supported in medium large, and huge memory models	Оверлеї використовуються тільки з моделями пам'яті medium, large і huge
Parameter names are used only with function body	Імена параметрів використовують тільки в тілі функції
Parameter number missing name	Параметр number не має імені
Pointer to structure required on left side of -> or ->*	Зліва від -> або ->* повинен бути покажчик на структуру
Reference initialized with typel needs lvalue of type type2	Посилання, ініціалізоване типом type1, вимагає значення типу type2
Reference member member is not Initialized	Посилання на елемент member не ініціалізоване
Reference member needs a temporary for initialization	Посилання елементу member вимагає робочої ініціалізації
Reference variable variable must be Initialized	Посилання на змінну variable повинно бути ініціалі зоване
Repeat count needs an lvalue	Повторний рахунок вимагає іменного виразу
Side effects are not allowed	Побічний ефект недопустимий
Sizeoff may not be applied to a	Sizeof непридатний до функції

Продовження таблиці А.3

Size of identifier is unknown or zero	Розмір identifier невідомий або рівний нулю
Sizeof may not be applied to a function	Sizeof непридатний до функції
Size of identifier is unknown or zero	Розмір identifier невідомий або рівний нулю
Size of the type is unknown or zero	Розмір типу невідомий або рівний нулю
Specifier has already been included	Specifier був вже підключений
Statement missing ;	У операторі відсутня;
Structure required on left side of . Or .*	Зліва від . або .*обов'язково повинна бути структура
Structure size too large	Розмір структури дуже великий У даній конфігурації системи немає такого об'єму пам'яті
Subscripting missing]	У індексної змінної відсутня]
Switch selection expression must be Of integral type	Вираз в операторі switch повинен бути цілочислового типу
Switch statement missing (	У операторі switch відсутня (
Switch statement missing)	У операторі switch відсутня)
The value for identifier is not within the range of an int	Значення identifier не входить в діапазон
Too few parameters in call	Дуже мало параметрів в звертанні до фунції: менше, ніж в прототипі
Too few parameters in call to Function	Дуже мало параметрів в звертанні до функції
Too many decimal points	Дуже багато десяткових крапок
Too many default cases	Дуже багато альтернатив у case, тобто більше одного оператора default в одному операторі switch
Too many error or warning messages	Дуже багато повідомленнь про помилки або попередження
Too many exponents	Дуже багато експоненти: більше одної в константі з плаваючою комою
Too many initializers	Дуже багато ініціалізації: більше допустимого описом

Продовження таблиці А.3

Too many types in declaration	Дуже багато типів в описі: більше одного з допустимих (char, int, float, double, struct, union, enum або typedef-ім'я)
Too much global data defined in file	Дуже багато глобальних даних в одному файлі: їх об'єм більше 64 Кбайтів
Two consecutive dots	Дві послідовні крапки. Наприклад, десяткові або в складеному імені
Two operands must evaluate to the same type	Два операнди повинні мати значення одного типу
Type mismatch in default argument value	Невідповідність типу, визначеного за замовчуванням, зі значенням аргументу
Type mismatch in default value for parameter parameter	Невідповідність типу та значення параметра parameter, заданого за замовчуванням
Type mismatch in parameter number	Невідповідність типу в параметрі number. Фактичний параметр не може бути перетворений в тип, оголошений в прототипі цієї функції
Type mismatch in parameter number in call to function	Невідповідність типу в параметрі number при виклику функції function
Type mismatch in parameter parameter	Невідповідний тип в параметрі parameter
Type mismatch in parameter parameter in call to functibn	Невідповідний тип в параметрі parameter при виклику function
Type mismatch in redeclaration of Identifier	Невідповідність типу при оголошенні identifier
Type name expected	Очікується ім'я типу
Type qualifier identifier must be a struct or class name	Тип identifier повинен бути структурою або класом
Type typename may not be defined here	Тип typename не може бути розміщений тут
Unable to execute command command	Неможливо виконати команду command
Unable to open include file filename	Неможливо відкрити включаємий файл

Продовження таблиці А.3

Undefined label identifier	Невизначена мітка identifier. Мітка використовується в операторі goto функції, але не визначена в тілі функції
Undefined structure structure	Невизначена структура structure
Undefined symbol identifier	Невизначений ідентифікатор
Unexpected .}	Несподівана }
Unexpected end of file in comment started on line number	Несподіваний кінець файлу в коментарі, що починається в рядку line number. Початковий файл завершується усередині коментаря. Причиною помилки може бути відсутність символів закриття коментаря (*/)
Unexpected end of file in conditional started on line number	Несподіваний кінець файлу з умовним запуском в рядку line number
Unknown language, must be C or C++	Невідома мова, потрібно Сі або C++
Unknown preprocessor directive: Identifier	Невідома директива препроцесора identifier
Unterminated string or character Constant	Незавершений рядок або рядкова константа: знайдена непарна лапка (") при визначенні рядкової константи
Use . or -> to call function	Використовуйте . або -> для виклику функції
User break	Переривання від користувача: введена команда Ctrl-Break під час компіляції або компоновки
Value of type void is not allowed	Значення типу void неприпустимо
Variable variable has been optimized And is no longer available	Змінна variable була оптимізована і більше недоступна
Variable identifier is initialized More than once	Змінна identifier ініціалізована більше одного разу
While statement missing (	Відсутня (у операторі while
While statement missing)	Відсутня) у операторі while
Wrong number of arguments in call of Macro	Неправильна кількість аргументів при звертанні до макросу

Таблиця А.4 – Попередження (Warnings)

Ambiguous operators need parentheses	Неоднозначний оператор, потрібні дужки. Наприклад, спільно використовуються без дужок дві операції зсуву, відношення або порозрядної логічної операції. Це повідомлення також видаватиметься, якщо оператор додавання або віднімання без дужок використовується з оператором зсуву
Array size for delete ignored	Розмір масиву для delete проігнорований
Array variable identifier Is near	Змінна-масив identifier має модель пам'яті near
Assigning type to enumeration	Присвоєння type переліковному типу
Bit fields must be signed or unsigned int	Бітове поле повинно бути цілим, знаковим або беззнаковим
Both return and return With a value used	Використовується подвійний return: просто return і return з поверненням значення
Call to function function with no prototype	Звертання до функції function, що не має прототипу
Call to function with no prototype	Звертання до функції, що не має прототипу
Code has no effect	Програма неефективна. Це повідомлення з'являється, коли компілятор знаходить оператори, що не приводять ні до яких дій
Condition is always true	Умова завжди істинна
Condition is always false	Умова завжди помилкова
Constant is long	Довга константа: ціла десяткова константа > 32 767 або вісімкова (шістнадцятіркова) константа > 65535 без наступної за нею букви l або L
Constant out of range in Comparison	Значення константи в операторі порівняння перевищує comparison
Conversion may lose significant digits	При перетворенні може бути втрачено багато значущих цифр. Це може бути, якщо в процесі присвоєння або з іншої причини виникає перетворення з long або unsigned long в тип int або unsigned int

Продовження таблиці А.4

Declare type type prior To use in prototype	Оголошення типу type до використання в прототипі
Division by zero	Ділення на нуль
Function should return a Value	Функція повинна повертати значення, але не повертає його за допомогою оператора return
Hexadecimal value contains more than 3 digits	Шістнадцятіркова константа дуже велика
identifier declared but never used	Ідентифікатор оголошений, але ніколи не використовується
identifier is assigned a value that is never used	Ідентифікатору присвоєне значення, але воно ніколи не використовується
identifier is declared as both external and static	Ідентифікатор оголошений як external і static
Initialization is only partially bracketed	Ініціалізація тільки часткова, перервана
Initializing enumeration type with	Ініціалізація enumerations типом type
Maximum precision used for member pointer type type	Максимальна точність використовується для елементу покажчика типу type
Mixing pointers to signed and unsigned char	Змішення покажчиків на знаковий і без-знаковий символ
No declaration for function function	Немає оголошення функції function
Nonportable pointer Comparison	Неприпустиме порівняння покажчиків. Наприклад, порівняння покажчика із змінною, що не є покажчиком (виключаючи ситуацію порівняння з нульовою константою)
Nonportable pointer conversion	Неприпустиме перетворення покажчика
Parameter parameter is never used	Параметр parameter ніколи не використовується
Possible use of identifier before definition	Можливо використання змінної identifier до її визначення
Possibly incorrect Assignment	Треба взяти присвоювання в круглі дужки, if (a = b)...повинен бути переписаний як: if ((a=b) !=0) ...

Продовження таблиці А.4

Redefinition of macro is not identical	Перевизначення константи macro неідентичне її попередньому визначенню
Structure passed by value	Структура передається значенням. Доцільна її передача за допомогою адреси (&) структури
Superfluonus & with function	Непотрібне використання & з функцією
Suspicious pointer Conversion	Сумнівне перетворення покажчика
Temporary used for Parameter number	Тимчасово використовується для параметра number
Temporary used for parameter number in call to function	Тимчасово використовується для параметра number при виклику function
Temporary used for Parameter parameter	Тимчасово використовується для параметра
Temporary used for parameter parameter in call to function	Тимчасово використовується для параметра parameter при виклику function
Temporary used for initialize identifier	Тимчасово використовується для ініціалізації identifier
Undefined structure structure	Невизначена структура structure
Unknown assembler Instruction	Невідома команда на асемблері
Unreachable code	Недосяжний оператор. За операторами break, continue, goto або return йде не мітка або кінець циклу (програми)
Void functions may not return a value	Функції з void-типом не можуть повертати значення. У тілі функцій, описаної як void (неповертаюча значення), знаходиться оператор return з поверненням значення. Це значення буде проігнороване

Повідомлення про помилки на етапі компонування

Фатальна помилка на етапі компоновки призводить до негайної зупинки компонувальника; exe-файл видаляється.

Таблиця А.5 – Критичні (фатальні) помилки компонування

32-bit record encountered	Знайдений 32-бітовий запис
Bad character in parameters	Помилковий символ в параметрах
Bad object file record in library file filename near module file offset 0xxxxxxxxx	Помилковий запис об'єктного файлу в бібліотечному файлі filenamэ файлового модуля з near-зсувом 0xxxxxxxx
Bat version number in parameter block	Помилковий номер версії в блоці параметра
filename (linenum): File read error	Filename (linenum): Помилка читання з файлу
filename (linenum): Incompatitle attribute	Filename (linenum): Несумісні атрибути
filename (linenum):Missing internal name	Filename (linenum): Відсутнє внутрішнє ім'я
filename (linenum): Syntax error	Filename (linenum): Синтаксична помилка
General error	Загальна помилка
General error in library file in module module near module file offset 0xyyyyyyyy	Загальна помилка в бібліотечному файлі в модулі module – файлового модуля з near-зсувом 0xyyyyyyyy
General error in module module near module file offset 0xyyyyyyyy	Загальна помилка в модулі module – файлового модуля з near-зсувом 0xyyyyyyyy
Internal linker error errorcode	Внутрішня помилка компонувальника errorcode
Invalid initial stack offset	Помилковий початковий зсув стека
Invalid segment definition in module module	Помилкове визначення сегменту у модулі module
Linker stack overflow	Стек компонувальника переповнений
Limit of 254 segments for new executable file exceeded	Вичерпаний ліміт в 254 сегменти для нового виконуваного файлу
New executable header overflowed 64 K	Новий виконуваний заголовок перевищив 64 Кбайти

Продовження таблиці А.5

Not enough memory	Немає достатньої пам'яті, придатної до використання. Повідомлення може бути видане, коли недостатньо наявної пам'яті для виконання підпроцесу або запит на отримання пам'яті не може бути задоволений
Out of memory	Вихід за межі пам'яті. Об'єм пам'яті вичерпаний. Ситуація виникає, якщо використовується пам'ять об'ємом 640 Кбайтів і вільної пам'яті в комп'ютері вже немає. Треба розбити початковий файл на ряд простіших і коротших файлів і відкомпілювати їх окремо
Relocation item exceeds 1MB DOS limit	Розміщення даних перевищило ліміт DOS в 1 Мбайт. DOS не підтримує розміщення виконуваних файлів об'ємом більше 1 Мбайта
Relocation offset overflow	Переповнення розміщення зсувів. Розміщення сегмента обмежене 64 Кбайтами. Тільки для 32-бітових об'єктних модулів
Relocation table overflow	Переповнення таблиці розміщень. Файл зв'язків містить базу даних об'ємом, що перевищує допустимий стандарт DOS. Ця база створюється головним чином при виклику far-функцій. Тільки для 32-бітових об'єктних модулів
Segment segment exceeds 64 K	Сегмент segment перевищив 64 Кбайти
Segment too large fog segment table	Сегмент дуже великий для таблиці
Stud program exceeds 64 K	Коренева програма перевищує 64 Кбайти
Table limit exceeded	Перевищений ліміт таблиці
Unable to open file filename	Неможливо відкрити файл Filename
Unable to open dpralmem.dll	Неможливо відкрити файл dpralmem.dll

Продовження таблиці А.5

Unknown option	Невідомий параметр
Write fallad, disk full	Запис помилковий, диск повний
Terminated by user	Зупинено користувачем

Таблиця А.6 – Помилки (ERROR)

Помилка, знайдена компонувальником, не приводить до зупинки його роботи або видалення файлів .exe або .map. Але файл .exe не може бути виконаний через помилку. Треба усунути помилку і перекомпонувати exe-файл.

Automatic data segment exceeds 64 K	Сегмент автоматичних даних перевищує 64 Кбайти
Common seqment exceeds 64 K	Зовнішній сегмент перевищує 64 Кбайти
Fixup overflow at seg:xxxxh, target =seg:xxxh in module module	Переповнення з фіксованою крапкою в seg:xxxxh, адресат=seg:xxxh в модулі module
Fixup overflow at seg:xxxxh, target =symbol in module module	Переповнення з фіксованою крапкою в seg:xxxxh, адресат = symbol в модулі module
Imported references from VIRDEFs not supported	Посилання, що імпортується, з VIRDEF не підтримується
Invalid entry point offset	Помилковий зсув точки входу
Invalid limit specified for code segment packing	Помилково визначене обмеження для упакування сегмента коду
Invalid size specified for segment aligment	Помилково визначений розмір для розміщення сегмента
Proqram entry point may not reside in an overlay	Точка входу програми не може бути в оверлеї
Undefined symbol symbol in module module	Невизначений символ symbol в модулі module
User break	Переривання користувачем
'symbol' defined in module 'module' is duplicated in module 'module'	Ім'я symbol, визначене в модулі module, дубльоване в модулі module, тобто повторно визначене зовнішнє ім'я
'symbol' is duplicated in module 'module'	Ім'я symbol дубльоване в модулі module
Too many error or warning messages	Дуже багато повідомлень про помилки або попередження

Таблиця А.7 – Попередження (Warnings)

Warnings – це попередження, причину яких треба встановити. Якщо з'явилися попередження, .exe- і .map – файли створюються.

Attempt to export Non-public symbol Symbol	Спроба експортувати незагальний символ Symbol
Debug information in module inodiiln will be ignored	Інформація налагоджування в модулі module буде проігнорована
Duplicate ordinal number in exports	Дублювання ординального number в експорті
filename (linenum): Duplicate external name in exports	filename (linenum): Дублювання зовнішнього імені в екепорті
filename (linenum): Duplicate internal name in exports	filename (linenum): Дублювання внутрішнього імені в експорті
filename (linenum): Duplicate internal name in imports	filename (linenum): Дублювання внутрішнього імені в імпорті
Invalid entry at segment: xxxxh	Помилковий вхід segment :xxxxh
No automatic data segment	Сегмент неавтоматичних даних
No module definition file specified:	Файл не визначений в модулі: використовується умовчання
No -program starting address defined	Не визначена адреса початку програми
No stack	Немає стека
No stub for fixup at segment: xxxxh in module module	Немає кореня сегменту segment: xxxxh в модулі module
Overlays ignored in new executable image	Оверлей проігнорований в новому здійснимому файлі
Possible reference to undefined extern	Можливе посилання до невизначеного xxxx: :i in module module
Stack size is less than 1400h.It has been reset to 1400h	Розмір стека менше 1400h. Він буде встановлений в 1400h
symbol conflicts with module module in module module	symbol не узгоджений з модулем module в модулі module
symbol defined in module modulel is duplicated in module module2	symbol, визначений в модулі modulel, дубльований в модулі module2
symbol is duplicated in module module	symbol дубльований в модулі module

Повідомлення про помилки на етапі виконання програми

Таблиця А.8 – Повідомлення, викликані при роботі з бібліотеками, файлами і функціями

Arg list too big	Список аргументів дуже довгий: перевищує 128 байтів
Attempted to remove current directory	Спроба пересилання поточної директорії
Cross-device links	Перехресний зв'язок пристроїв. Помилка виникає при спробі передати файл з одного пристрою на інший, використовуючи функцію rename
Exec format error	Помилка формату exec. Спроба виконати невиконувану програму
Invalid access code	Помилковий код доступу
Invalid argument	Неправильний один з аргументів функції
Invalid data	Неправильні дані
Invalid environment	Помилкове середовище
Invalid format	Помилковий формат
Invalid function number.	Помилковий номер (ім'я) функції
Invalid memory block address	Помилкова адреса блоку пам'яті
Memory arena trashed.	Область пам'яті засмічена
No more files.	Більше немає файлів
Nor same device.	Немає такого пристрою
No such device.	Не знайдено пристрій
No such file or directory	Немає такого файлу або директорії. Вказаний файл не існує або компонент існуючої директорії не вказаний в шляху
Path not found.	Шлях не знайдений
Permission denied	Доступ неможливий: режим доступу до файлу не дозволяє це
Too many open files	Дуже багато відкрито файлів. Немає доступніших дескрипторів (handle), оскільки дуже багато відкритих файлів

Таблиця А.9 – Повідомлення з системи Help про помилки при роботі з бібліотеками, файлами і модулями:

Bad header in input LIB	Помилковий заголовок вхідної бібліотеки
Could not allocate memory for per module data	Неможливо розміщення в ОП даних модуля
Could not create list file 'filename'	Неможливо створення списку файлів filename
Could not write output	Неможливо виведення в output
DOS reported error: Permission denied opening 'filename '	Помилка повідомлена DOS: Дозволу немає на відкриття filename
DOS reported error: Permission denied opening 'filename' for output	Помилка повідомлена DOS: Дозволу немає на відкриття filename для виведення
DOS reported error: Permission denied opening 'filename' to 'filename'	Помилка повідомлена DOS: Дозволу немає на перейменування filename в filename
Error changing file buffer size	Помилка в зміні розміру буфера файлу
Error opening 'filename'	Помилка відкриття filename
Error opening 'filename' for output	Помилка відкриття filename для виведення
Error renaming 'filename' to 'filename'	Помилка перейменування filename в filename
Library too large, please restart with /P 'size'	Бібліотека дуже велика, будь ласка, стартуйте знову з /P size
M ibrary too large, restart with library paqc size 'size'	Бібліотека дуже велика, потрібен рестарт з вказанням size — розміру сторінки
Not enough memory for command-line buffer	Недостатньо пам'яті для буфера командного рядка
Object module 'filename' is invalid	Помилковий об'єктний модуль filename
Out. Of memory	Вихід за межі пам'яті. Розбити початковий файл на ряд коротших файлів і відкомпілювати їх окремо
Out. Of memory creating extended	Вихід за межі пам'яті при створенні розширеного словника

Продовження таблиці А.9

Out. Of space allocating per module debug struct	Вихід за межі простору при розміщенні в модулі структури налагодження
Output device is full	Вивідний пристрій заповнений
'path' - path is too long	path — дуже довго
Record length 'len' exceeds available buffer in module 'module'	Довжина запису len перевищує доступний буфер в модулі 'module'
The combinations '+*' or '*+' are not Allowed	Поєднання +* або *+ неприпустимо
Unexpected char 'X' in command line	Неприпустимий символ X в командному рядку
User break, library aborted	Переривання користувачем. Бібліотека закрита

Таблиця А.10 – Помилки -попередження:

Added file 'filename' does not begin t'otmctly, ignored	У файлу filename, що додається, некоректний початок, проігноровано
'filename1 couldn't be created, original won't be changled	filename не був створений, оригінал не буде змінений
'filename' file not found	Файл filename не знайдений
Ignored 'module', path is too long	Проігноровано module, шлях дуже довгий
Invalid page size value ignored	Невірне значення розміру сторінки, проігноровано
Memory full listing truncated!	Пам'ять заповнена, виведення усічене!
'module' already in LIB, not changed!	Module вже в бібліотеці, не змінений!
Module not found in library	Модуль не знайдений в бібліотеці
'reason' – extended dictionary not created	Висновок – розширений словник не створений
Results are safe in file 'filename'	Результати надійні у файлі filename
Unknown command line switch 'X' ignored	Невідомий перемикач X в командному рядку проігнорований

Продовження таблиці А.10

Use /e with TLINK to obtain debug information from library	Використовуйте /e з компонувальником для отримання налагоджувальної інформації з бібліотеки

Таблиця А.11 - Повідомлення про помилки, пов'язані з математичними обчисленнями, і помилки етапу виконання програми

Ці повідомлення розташовані у файлі math.h:

Argument domain error	Аргумент функції поза допустимим діапазоном
Argument singularity	Вироджений, неприпустимий аргумент
Overflow range error	Помилка переповнення діапазону
Underflow range error	Помилка втрати значущості з виходом з допустимого діапазону
Total loss of significance	Повна втрата значення
Floating point unit stack overflow	Переповнення стека модуля при роботі з плаваючою комою

Таблиця А.12 – Повідомлення етапу виконання програми – з розділу Run-time Errors, викликаного за командою Help\Contents\Error Messages:

Abnormal program termination	Ненормальне завершення програми
Divide error	Спроба ділення на нуль при роботі з даними цілого типу
Floating point error: Divide by zero	Помилка при роботі з плаваючою комою: спроба ділення на нуль
Floating point error: Domain	Помилка при роботі з плаваючою комою
Floating point error: Overflow	Помилка при роботі з плаваючою комою; переповнення
Floating point error: Partial loss of Precision	Помилка при роботі з плаваючою комою. Часткова втрата значущості
Floating point error: Underflow	Помилка при роботі з плаваючою комою; втрата значущості
Floating point error: Stack fault	Помилка при роботі з плаваючою комою; стек помилковий
Null pointer assignment	Спроба записати значення за неініціалізованим покажчиком
Stack overflow	Стек переповнений

Навчальне видання

Л. М. Круподьорова
А.М. Пєтух

Технологія програмування мовою Сі

Частина 2
Лабораторний практикум

Навчальний посібник

Оригінал-макет підготовлено Круподьоровою Л.М., Пєтухом А.М.

Редактор Т.О. Старічек

Науково-методичний відділ ВНТУ
Свідоцтво Держкомінформу України
серія ДК № 746 від 25.12.2001
21021, м. Вінниця, Хмельницьке шосе, 95, ВНТУ

Підписано до друку 5.02.2007 р.
Формат 29,7х42 ¼
Друк різографічний
Тираж 75 прим.
Зам. № 2007-017

Гарнітура Times New Roman
Папір офсетний
Ум.друк.арк. 10.5

Віддруковано в комп'ютерному інформаційно-видавничому центрі
Вінницького національного технічного університету
Свідоцтво Держкомінформу України
серія ДК № 746 від 25.12.2001
21021, м.Вінниця, Хмельницьке шосе, 95, ВНТУ