Методичні вказівки
до виконання самостійної роботи
з дисципліни «Транслятори та покоління мов
програмування телекомунікаційних систем»
зі спеціальності «Електроніка, електронні комунікації,
приладобудування та радіотехніка» (освітня програма
«Програмне забезпечення телекомунікаційних систем»)

Міністерство освіти і науки України
Вінницький національний технічний університет

Методичні вказівки до виконання самостійної роботи з дисципліни «Транслятори та покоління мов програмування телекомунікаційних систем» зі спеціальності «Електроніка, електронні комунікації, приладобудування та радіотехніка» (освітня програма «Програмне забезпечення телекомунікаційних систем»)

Вінниця
ВНТУ
2026

Рекомендовано до видання Радою з якості освіти Вінницького національного технічного університету Міністерства освіти і науки України (протокол № 13 від 18.06.2026 р.)

Рецензенти:
Л. Г. Коваль, кандидат технічних наук, доцент
М. О. Притула, кандидат технічних наук, доцент

Методичні вказівки до виконання самостійної роботи з дисципліни «Транслятори та покоління мов програмування телекомунікаційних систем» зі спеціальності «Електроніка, електронні комунікації, приладобудування та радіотехніка» (освітня програма «Програмне забезпечення телекомунікаційних систем») / уклад.: В. І. Макогон, Д. В. Михалевський. Електрон. текст. дані. Вінниця : ВНТУ, 2026. 45 с.

Методичні вказівки містять матеріали для самостійного опрацювання навчальної програми з дисципліни «Транслятори та покоління мов програмування телекомунікаційних систем» для освітньої програми «Програмне забезпечення телекомунікаційних систем». Наведено матеріали для самостійного опрацювання та закріплення теоретичних знань на базі тестів самоконтролю та індивідуальних завдань дослідницького характеру.

ЗМІСТ

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

Методичні вказівки до самостійної роботи з дисципліни «Транслятори та покоління мов програмування телекомунікаційних систем» мають на меті узагальнити набуті теоретичні знання сучасних інформаційних технологій у галузі програмування вбудованих пристроїв, а також передбачає вивчення основних проблем використання мікропроцесорних пристроїв у телекомунікаційних та радіотехнічних системах, а також аналіз та тестування вбудованих пристроїв. Також передбачається надання допомоги в організації самостійної роботи студентів з вивчення навчального матеріалу та підготовки до складання іспиту.

Метою дисципліни «Транслятори та покоління мов програмування телекомунікаційних систем» є засвоєння необхідних знань з основ програмування мікропроцесорних пристроїв з використанням сучасних найбільш поширених засобів програмування, а також формування практичних навичок щодо розробки програмних засобів для мікропроцесорних систем.

Завдання, що вирішуються в процесі вивчення дисципліни, полягають у отриманні теоретичних знань з основ програмування та будови мікропроцесорних систем, отриманні практичних навичок з програмування мікропроцесорних систем, поглиблення основних та формування нових знань і положень про технології розробки і програмування сучасних мікроконтролерних та мікропроцесорних систем з використанням сучасних середовищ розробки.

Вивчення дисципліни передбачає планове, поетапне засвоєння студентами практичних і професійних навичок та застосування в реальних умовах теоретичних знань, отриманих при вивченні дисциплін з циклів загальної та професійної підготовки за вказаною спеціальністю.

Основними завданнями вивчення дисципліни є формування теоретичної бази і практичних навиків для розв’язування складних спеціалізованих задач, пов’язаних із програмуванням сучасних мікропроцесорних систем.

Компетентності, якими повинен оволодіти здобувач в результаті вивчення дисципліни.

Спеціальні компетентності

СК1. Здатність розуміти сутність і значення інформації в розвитку сучасного інформаційного суспільства.

СК2. Здатність вирішувати стандартні завдання професійної діяльності на основі інформаційної та бібліографічної культури із застосуванням інформаційно-комунікаційних технологій і з урахуванням основних вимог інформаційної безпеки.

СК3. Здатність використовувати базові методи, способи та засоби отримання, передавання, обробки та зберігання інформації.

СК4. Здатність здійснювати комп'ютерне моделювання пристроїв, систем і процесів з використанням універсальних пакетів прикладних програм.

СК8. Готовність сприяти впровадженню перспективних технологій і стандартів.

СК12. Здатність проводити роботи з керування потоками навантаження інформаційно-телекомунікаційних мереж.

СК14. Готовність до вивчення науково-технічної інформації, вітчизняного і закордонного досвіду з тематики інвестиційного (або іншого) проєкту засобів телекомунікацій та радіотехніки.

СК15. Здатність проводити розрахунки у процесі проєктування споруд і засобів інформаційно-телекомунікаційних мереж, телекомунікаційних та радіотехнічних систем, відповідно до технічного завдання з використанням як стандартних, так і самостійно створених методів, прийомів і програмних засобів автоматизації проєктування.

СК16. Уявлення про сучасні електронні компоненти та технічні засоби електрозв'язку (побудова і функціонування мікропроцесорів, пристрої збереження та копіювання, документування інформації тощо).

СК17. Уявлення про сучасні підходи, методи та технології програмування в сучасних телекомунікаційних системах та мережах.

СК18 Здатність до застосування існуючих та створення нових програмно-апаратних засобів тестування телекомунікаційного та радіотехнічного обладнання.

СК19. Здатність до створення прикладного програмного забезпечення як для функціонування вузлів телекомунікаційних систем, так і керування в телекомунікаційних мережах.

Програмні результати навчання

РН1. Аналізувати, аргументувати, приймати рішення при розв'язанні спеціалізованих задач та практичних проблем телекомунікацій та радіотехніки, які характеризуються комплексністю та неповною визначеністю умов.

РН2. Застосовувати результати особистого пошуку та аналізу інформації для розв'язання якісних і кількісних задач подібного характеру в інформаційно-комунікаційних мережах, телекомунікаційних і радіотехнічних системах.

РН6. Адаптуватись в умовах зміни технологій інформаційно-комунікаційних мереж, телекомунікаційних та радіотехнічних систем.

РН7. Грамотно застосовувати термінологію галузі телекомунікацій та радіотехніки.

РН8. Описувати принципи та процедури, що використовуються в телекомунікаційних системах, інформаційно-телекомунікаційних мережах та радіотехніці.

РН9. Аналізувати та виконувати оцінку ефективності методів проєктування інформаційно-телекомунікаційних мереж, телекомунікаційних та радіотехнічних систем.

РН10. Спілкуватись з професійних питань, включаючи усну та письмову комунікацію державною мовою та однією з поширених європейських мов (англійською, німецькою, італійською, французькою, іспанською).

РН11. Застосовувати міжособистісні навички для взаємодії з іншими людьми та залучення їх до командної роботи.

РН12. Толерантно сприймати та застосовувати етичні норми поведінки відносно інших людей.

РН15. Застосування розуміння засобів автоматизації проєктування і технічної експлуатації систем телекомунікацій та радіотехніки у професійній діяльності.

РН18. Знаходити, оцінювати і використовувати інформацію з різних джерел, необхідну для розв’язання професійних завдань, включаючи відтворення інформації через електронний пошук.

РН20. Пояснювати принципи побудови й функціонування апаратно-програмних комплексів систем керування та технічного обслуговування для розробки, аналізу і експлуатації інформаційно-телекомунікаційних мереж, телекомунікаційних та радіотехнічних систем.

РН25. Уміння представляти та обговорювати наукові результати іноземною мовою (англійською або іншою, відповідно до специфіки спеціальності) в усній та письмовій формах, приймати участь у наукових дискусіях і конференціях.

РН28. Вміння застосовувати існуючі та розробляти нові програмні продукти, які застосовуються для функціонування та керування телекомунікаційних систем та мереж.

РН29. Здатність застосовувати та розробляти програмно-апаратні засоби для тестування обладнання телекомунікацій.

ТЕМА 1. Огляд актуальних мікропроцесорних систем

1. Яке визначення найбільш повно характеризує сутність мікропроцесора як основного компонента комп’ютерної системи, включаючи його основні функції та роль у виконанні арифметичних і логічних операцій у процесі обробки даних?

А) Мікропроцесор – це периферійний пристрій, що зберігає дані та забезпечує друк інформації.

Б) Мікропроцесор – це основний обчислювальний пристрій комп’ютера, що виконує всі арифметичні, логічні та керуючі операції у процесі обробки даних.

В) Мікропроцесор – це програма для налаштування системи охолодження комп’ютера.

Г) Мікропроцесор – це програмно-апаратний комплекс для графічного моделювання даних.

2. Що із наведеного твердження найбільш точно описує поняття мікроконтролера, з урахуванням його архітектурних особливостей, призначення та відмінностей від стандартних мікропроцесорів у складі комп’ютерних систем?

А) Мікроконтролер – це мікросхема, що містить процесор, пам’ять і периферійні пристрої в одному корпусі та використовується для керування вбудованими системами.

Б) Мікроконтролер – це зовнішній пристрій, призначений для підключення до мережі Інтернет і забезпечення передачі даних.

В) Мікроконтролер – це графічний модуль комп’ютера, що обробляє тривимірні зображення.

Г) Мікроконтролер – це пристрій для друку текстових документів через комп’ютер.

3. Яким чином можна найбільш точно визначити призначення сигнального процесора у складі обчислювальної техніки, з урахуванням його спеціалізації та відмінностей від інших типів процесорів, таких як центральні чи графічні?

А) Сигнальний процесор використовується виключно для обробки великих обсягів текстової інформації в базах даних.

Б) Сигнальний процесор призначений для обробки цифрових сигналів, таких як звук і відео, та часто використовується у пристроях обробки мультимедійних даних.

В) Сигнальний процесор є аналогом мікропроцесора, що виконує ті самі функції без жодних відмінностей.

Г) Сигнальний процесор відповідає за управління мережею комп’ютера та обробку IP-пакетів.

4. Яке із наступних тверджень найкраще описує функціональне призначення арифметико-логічного пристрою (АЛП) у складі центрального процесора, з деталізацією його завдань у процесі обробки машинних команд?

А) АЛП використовується виключно для підключення комп'ютера до локальної мережі.

Б) АЛП відповідає за завантаження операційної системи після увімкнення комп'ютера.

В) АЛП керує роботою принтерів і інших периферійних пристроїв.

Г) АЛП виконує арифметичні операції додавання, віднімання, множення, ділення та логічні операції, що є основними етапами обробки інформації процесором.

5. Що таке регістри процесора і яку роль вони відіграють у процесі виконання інструкцій, зокрема в аспекті збереження проміжних результатів обчислень і адрес пам'яті, необхідних для швидкодії системи?

А) Регістри процесора – це зовнішні пристрої для запису і збереження великих обсягів даних у довготривалій перспективі.

Б) Регістри процесора – це частина оперативної пам'яті, що використовується виключно для завантаження графічних файлів.

В) Регістри процесора – це внутрішня швидкодіюча пам'ять, що тимчасово зберігає дані та адреси під час виконання інструкцій, забезпечуючи ефективну роботу процесора.

Г) Регістри процесора – це комп'ютерні програми для налаштування BIOS.

6. Яка була головна причина переходу від електронних лампових обчислювальних машин до інтегральної мікросхемної технології, і як це вплинуло на розміри, надійність та ефективність обчислювальної техніки?

А) Інтегральні схеми були створені лише з метою зниження вартості, але не вплинули на розміри або надійність систем.

Б) Заміна ламп мікросхемами дозволила збільшити енергоспоживання, але не мала великого впливу на надійність.

В) Основною причиною була потреба у зменшенні частоти перегріву, підвищенні надійності та зменшенні фізичних розмірів ЕОМ.

Г) Всі старі комп'ютери були знищені, тому довелося створювати нові мікросхеми.

7. Яким чином Intel отримала повні права на мікропроцесор 4004 після завершення розробки, і яке це мало стратегічне значення для подальшого розвитку компанії та мікропроцесорної індустрії в цілому?

А) Intel втратила права на процесор 4004 через фінансові проблеми.

Б) Intel об'єдналась з компанією Busicom у спільне підприємство з розподілом прибутків.

В) Intel викупила всі права на процесор у Busicom за 60 тисяч доларів і розпочала масштабну рекламну кампанію.

Г) Процесор був проданий державі США як стратегічна розробка.

8. Яким чином архітектура першого 8-розрядного мікропроцесора Intel 8008 стала логічним продовженням Intel 4004, і які ключові нововведення в ній дозволили віднести цей процесор до першого покоління мікропроцесорних засобів?

А) Intel 8008 мав ті самі можливості, що й 4004, лише більшу кількість транзисторів.

Б) Intel 8008 був першим мікропроцесором, створеним для графічних обчислень.

В) Intel 8008 був першим мікропроцесором із підтримкою обчислень із плаваючою комою.

Г) Intel 8008 мав 8-розрядну архітектуру, акумулятор, регістри, покажчик стеку і спеціальні команди введення/виведення, на відміну від 4004.

9. Яка основна причина зробила мікропроцесор Intel 8080 більш вдалим, ніж його попередник 8008, та дозволила цьому процесору стати «класичним зразком» для вивчення принципів побудови мікропроцесорних систем?

А) У 8080 була реалізована багатоядерна архітектура з вбудованим графічним адаптером.

Б) 8080 мав складну структуру команд, що дозволяла писати компілятори.

В) Архітектура 8080 була добре структурована, містила розвинену систему команд і зручну регістрову модель.

Г) 8080 використовувався виключно у військових цілях, тому був таємним і унікальним.

10. Яку роль відіграв мікропроцесор Intel 8086 у формуванні архітектури персональних комп’ютерів, і чому саме його використала компанія IBM у своєму першому ПК, відкривши нову епоху розвитку обчислювальної техніки?

А) 8086 був єдиним мікропроцесором, який підтримував операційні системи Apple.

Б) IBM обрала 8086 через його дешевизну та низьку енергоспоживаність, що дозволило створити перший ноутбук.

В) 8086 мав 16-розрядну архітектуру, високу продуктивність і став основою стандарту IBM PC, підтримуючи сучасні ОС.

Г) 8086 був створений спеціально для ігрових приставок і випадково потрапив до IBM.

11. Яке визначення найбільш повно описує архітектуру типового мікроконтролера, що включає в себе елементи обробки, керування та введення/виведення, та як ця архітектура дозволяє реалізувати автономну систему автоматичного керування?

А) Архітектура мікроконтролера передбачає лише просту логіку керування, без вбудованої пам’яті, таймерів та портів.

Б) Архітектура мікроконтролера базується лише на арифметичному блоці без можливостей для взаємодії з зовнішніми пристроями.

В) Типова архітектура мікроконтролера містить систему керування, АЛП, пам’ять даних та програм, порти вводу/виводу, таймери, лічильники та інтерфейси, що робить його повноцінним автономним керуючим пристроєм.

Г) Архітектура мікроконтролера складається лише з пам’яті типу ROM та RAM, без процесорного ядра.

12. Яким чином мікроконтролери AVR, зокрема виробництва компанії Atmel (нині Microchip), здобули популярність у навчанні, прототипуванні та вбудованих системах, і які їх архітектурні особливості сприяли цьому?

А) Мікроконтролери AVR мають складну інструкційну систему та вимагають спеціалізованого обладнання для програмування.

Б) Основною перевагою AVR є їх побудова на RISC-архітектурі, наявність енергоефективного ядра, просте програмування (включно з Arduino) та широке розповсюдження у навчальних закладах.

В) AVR використовуються переважно в серверному обладнанні, що пояснює їхню потужність.

Г) AVR є графічними контролерами і не мають вбудованої пам’яті програм.

13. Які функціональні особливості бітового процесора, вбудованого в мікроконтролери AVR, дають змогу ефективно керувати дискретними пристроями, зокрема в умовах реального часу?

А) Бітовий процесор дає змогу лише виконувати математичні обчислення над цілими числами.

Б) AVR не підтримують операції на рівні окремих бітів і тому не придатні для керування реальними пристроями.

В) Наявність команд маніпуляції окремими бітами (наприклад, встановлення, очищення, інверсії) дозволяє керувати сигналами керування (шлагбауми, лампи, двигуни) максимально швидко й точно.

Г) Бітовий процесор використовується тільки у сигнальних процесорах, а не у мікроконтролерах.

14. Яке призначення має система таймерів і лічильників у мікроконтролерах сімейства AVR, і як вона сприяє реалізації точного хронометрування подій у керуючих системах?

А) Система таймерів і лічильників дозволяє відстежувати час, генерувати затримки, формувати імпульсні сигнали і здійснювати точне керування виконавчими механізмами.

Б) Лічильники служать виключно для обробки числових даних із зовнішніх пам’ятей.

В) Таймери використовуються для збереження текстової інформації.

Г) AVR не містить вбудованих таймерів, а використовує зовнішні модулі.

15. Чому мікроконтролери часто порівнюють з одноплатними мінікомп’ютерами, і які особливості їх конструкції дозволяють реалізувати повноцінні автоматизовані системи управління на базі лише однієї мікросхеми?

А) Мікроконтролери потребують складної багатокомпонентної системи, подібної до персонального комп’ютера, для виконання навіть простих задач.

Б) Мікроконтролери не придатні для автономної роботи без підключення до зовнішнього сервера або комп’ютера.

В) Завдяки інтеграції пам’яті, процесора, інтерфейсів, таймерів і портів, мікроконтролер може функціонувати як самостійний комп’ютер, керуючи пристроями без необхідності зовнішньої підтримки.

Г) Мікроконтролери є лише допоміжними пристроями для ПК і не здатні виконувати обчислення.

Таблиця правильних відповідей

1	2	3	4	5	6	7	8
б	а	б	г	в	в	в	г

9	10	11	12	13	14	15
в	в	в	б	в	а	в

ТЕМА 2. Основні принципи програмування мікропроцесорних систем

1. Яке з тверджень найточніше описує процес виконання команди в мікропроцесорній системі, враховуючи послідовність машинних циклів, які необхідні для вибірки та виконання кожної інструкції?

А) Цикл команди включає вибірку інструкції з пам’яті та її виконання, складаючись із декількох машинних циклів, кількість яких залежить від типу команди.

Б) Всі команди потребують однакової кількості машинних циклів, незалежно від складності.

В) Команди виконуються миттєво, незалежно від структури мікропроцесора.

Г) Команди виконуються без участі пам’яті, лише на основі логіки керування.

2. Що таке машинний цикл у контексті програмування мікропроцесорних систем, і яке його значення в синхронізації роботи процесора з зовнішніми пристроями та пам’яттю?

А) Це випадковий інтервал часу, що виникає при зміні команд.

Б) Це тривалість виконання повного програмного циклу зчитування з клавіатури.

В) Це одиниця часу, протягом якої мікропроцесор здійснює звернення до пам’яті або вводу/виводу, синхронізуючи свою роботу з периферією.

Г) Це цикл, протягом якого компілятор перетворює код у машинний.

3. Як формуються мікрокоманди на нижчому рівні програмування мікропроцесорних систем, і яке їхнє значення для реалізації машинних інструкцій?

А) Мікрокоманди створюються у вигляді графічних блоків, які автоматично компілюються.

Б) Мікрокоманди є абстрактними уявленнями про команди користувача і не мають технічного значення.

В) Мікрокоманда – це сукупність керуючих сигналів і адрес, які формуються на основі однієї інструкції та забезпечують її реалізацію в мікропроцесорі.

Г) Мікрокоманди – це лише коди кольорів, які визначають рівні логіки в ПЛК.

4. Яке з наведених тверджень найточніше описує особливості форматів команд у мікропроцесорах і їхній зв’язок із організацією пам’яті та адресацією?

А) Команди зберігаються у вигляді текстових файлів і не залежать від структури пам’яті.

Б) Формат команди визначає її довжину в бітах і розміщення операндів, що безпосередньо впливає на організацію адресного простору та пам’яті.

В) Усі команди мають однакову довжину, незалежно від процесора.

Г) Команди зберігаються лише у кеш-пам’яті і не мають структури.

5. Яке з визначень найкраще пояснює поняття циклу команди, включаючи його фазову структуру, тривалість і залежність від типу інструкції, що виконується мікропроцесором?

А) Цикл команди – це час, за який комп’ютер виконує увімкнення живлення.

Б) Цикл команди – це проміжок між двома натисканнями клавіші.

В) Це цикл, що використовується для оновлення драйверів.

Г) Це інтервал часу, протягом якого відбувається вибірка та виконання команди; складається з декількох машинних циклів, тривалість яких залежить від складності команди.

6. Як система синхронізації тактових сигналів у мікропроцесорі впливає на функціонування програмного автомата та взаємодію з іншими компонентами системи?

А) Синхронізація потрібна лише для живлення екрана.

Б) Сигнали синхронізації використовуються лише у фазі програмування EEPROM.

В) Тактова синхронізація визначає час виконання команд, забезпечуючи узгоджену роботу пристроїв з мікропроцесором через машинні цикли.

Г) Синхронізація не потрібна для виконання програм.

7. Які особливості гарвардської архітектури, використаної в мікроконтролерах AVR, дають змогу ефективно виконувати програми, одночасно зчитуючи коди інструкцій та здійснюючи доступ до оперативної пам’яті?

А) Гарвардська архітектура у AVR передбачає єдину шину даних і команд, як у класичних комп’ютерах.

Б) AVR використовує гарвардську архітектуру, де пам’ять програм (FLASH) і даних (SRAM, EEPROM) мають окремі шини доступу, що дозволяє зчитувати команду й обробляти дані паралельно.

В) Гарвардська архітектура означає, що AVR має тільки один тип пам’яті.

Г) Усі види пам’яті AVR адресуються через один і той самий регістр.

8. Яка структура регістрів загального призначення реалізована в AVR, як вона впливає на програмування, та які спеціальні функції виконують індексні регістри X, Y, Z?

А) Усі 32 регістри мають однакове призначення й не поділяються на групи.

Б) Регістрів у AVR немає, дані передаються лише через оперативну пам’ять.

В) Регістри R0–R31 діляться на молодші (R0–R15), старші (R16–R31), і спеціальні індексні (R26–R31), причому X, Y, Z виконують функції вказівників для непрямої адресації.

Г) Усі регістри можуть використовуватися лише для адрес пам’яті.

9. Як в AVR реалізовано виконання команд із використанням конвеєризації, і яке значення ця технологія має для підвищення швидкодії програмного коду в мікроконтролерах?

А) Команди виконуються лише послідовно, незалежно від апаратної архітектури.

Б) Конвеєризація в AVR означає, що кожна команда виконується в окремому модулі незалежно.

В) Конвеєризація використовується лише в комп’ютерній графіці.

Г) Завдяки конвеєризації AVR виконує одну команду, паралельно витягуючи наступну з пам’яті програм, що зменшує затримки між командами.

10. Яке призначення має регістр стану SREG у AVR і як програміст може використовувати його прапорці під час реалізації арифметичних обчислень та умовних переходів?

А) SREG не впливає на логіку виконання команд.

Б) SREG використовується лише для виведення тексту на дисплей.

В) Прапорці регістра SREG (C, Z, N, V, S тощо) відображають результат арифметичних і логічних операцій і можуть бути перевірені умовними командами переходу.

Г) Прапорці не можуть бути змінені мікропроцесором.

11. Як відбувається адресація пам’яті даних у AVR, які ділянки включає адресний простір SRAM, і чому регістри загального призначення та регістри вводу/виводу не займають місця в основній оперативній пам’яті?

А) Пам’ять даних в AVR розміщена хаотично й не має фіксованих меж.

Б) Адресний простір пам’яті AVR поділений на ділянки для регістрів, I/O, внутрішньої SRAM та зовнішньої пам’яті; регістри не займають місце в SRAM, а лише відсувають її початок.

В) AVR має лише одну область пам’яті для всіх типів даних.

Г) Усі регістри й пам’ять займають однаковий простір.

12. Яке значення має стек у мікроконтролерах AVR, як він організований у пам’яті, та якими командами програміст здійснює запис та зчитування значень у стеку?

А) Стек – це регістр, який зберігає адреси змінних.

Б) Стек – частина Flash-пам’яті, що використовується для зберігання команд.

В) Стек реалізовано в SRAM, керується регістром SPH:SPL, використовується для збереження адрес повернення, та працює за принципом LIFO за допомогою команд PUSH/POP.

Г) AVR не використовує стек.

13. Яка структура організації пам’яті програм у мікроконтролерах AVR, яку роль відіграє FLASH, та які обмеження і можливості пов’язані з її використанням у процесі програмування?

А) FLASH використовується для зберігання змінних і очищається при кожному перезавантаженні.

Б) FLASH зберігає постійні програми, її обсяг залежить від моделі, дозволяє зчитування команд, підтримує до 10 тис. циклів перезапису.

В) FLASH не піддається програмуванню користувачем.

Г) Пам’ять FLASH має нелінійну адресацію.

14. Яким чином команда в AVR визначає операнди для арифметичних або логічних операцій, та які обмеження накладаються на вибір регістрів у залежності від типу операції (зокрема, з константами)?

А) При операціях з константами використовуються лише регістри R16–R31; R0–R15 не підтримують пряме завантаження чисел.

Б) Лише регістри R0–R15 працюють з константами.

В) Будь-які команди можуть використовувати будь-які регістри однаково.

Г) Операнди завжди беруться з EEPROM.

15. Яке призначення мають вектори переривань у системі управління мікроконтролера AVR, і як порядок їхнього розміщення в пам’яті програм впливає на пріоритети обробки подій?

А) Вектори переривань зберігаються в SRAM і не впливають на програму.

Б) Порядок розміщення векторів не має значення, оскільки всі переривання обробляються одночасно.

В) Вектори переривань розміщені на початку FLASH; нижча адреса означає вищий пріоритет – це забезпечує швидкий та передбачуваний порядок обробки.

Г) Переривання не використовуються в сучасних AVR.

Таблиця правильних відповідей

1	2	3	4	5	6	7
а	в	3	б	г	в	б

8	9	10	11	12	13	14	15
в	г	в	б	в	б	а	в

ТЕМА 3. Основи мови програмування Асемблер

1. Яке призначення має директива `.include` в асемблерній програмі для AVR, та як її використання спрощує роботу програміста при взаємодії з конкретною моделлю мікроконтролера, такою як ATmega32A?

А) Вона визначає назву проєкту та шлях до збереження файлу.

Б) Дає змогу вказати компілятору закінчити програму.

В) Забезпечує вставку бібліотеки, яка містить усі імена регістрів і налаштування для конкретної моделі МК, що дозволяє уникати роботи з числовими адресами.

Г) Вона активує всі порти вводу/виводу одночасно.

2. Яким чином сегмент `.CSEG` визначає область пам'яті в AVR, яка зберігає програмний код, і як це впливає на структуру виконання асемблерної програми?

А) `.CSEG` встановлює область пам'яті програм FLASH, у якій розміщується основний виконуваний код, що починається, як правило, з векторів переривань та мітки `reset.`

Б) `.CSEG` позначає область даних у SRAM, яку використовують лише зовнішні пристрої.

В) `.CSEG` застосовується лише в EEPROM для збереження змінних.

Г) `.CSEG` є директивою, що запускає компіляцію.

3. Для чого використовується директива `.org` в асемблерному коді, та яке значення має її точне розміщення в контексті організації векторів переривань чи SRAM?

А) `.org` використовується лише для естетичного форматування коду.

Б) Вона дозволяє встановити адресу початку нового файлу.

В) `.org` встановлює абсолютну адресу комірки в пам'яті (FLASH, SRAM або EEPROM), і неправильне її застосування може призвести до некоректного розміщення команд чи змінних.

Г) Ця директива використовується тільки у макросах.

4. Яку функцію виконує основний програмний цикл у мікроконтролерах AVR, чому його зазвичай зациклюють за допомогою команди `rjmp`, і що може статись, якщо цього не зробити?

А) Основний цикл завершує програму та зберігає її у EEPROM.

Б) Якщо його не зациклити, програма перейде у режим сну.

В) Без команди `rjmp`, після завершення коду, МК продовжить виконання пустих байтів у FLASH, що може призвести до непередбачуваних дій або повторної ініціалізації.

Г) Це лише рекомендація для відладки коду.

5. У чому полягає призначення директиви `.def`, та як її використання допомагає уникати помилок і покращує читабельність асемблерної програми на AVR?

А) Вона визначає початок програми.

Б) Дозволяє прив'язати символьне ім'я до конкретного регістра загального призначення, що спрощує логіку програми і запобігає конфліктам при переназначенні.

В) Використовується для зміни адрес портів вводу/виводу.

Г) Встановлює частоту роботи кварцового резонатора.

6. Яке значення має директива `.byte` у сегменті `.DSEG`, і як вона впливає на організацію змінних в оперативній пам'яті мікроконтролера?

А) Вона формує команду для запису в EEPROM.

Б) Зарезервовує байти лише в регістрах.

В) Дозволяє компілятору виділити певну кількість байтів у SRAM під іменовані змінні, які в подальшому використовуються програмою.

Г) Створює адреси для програмного коду в FLASH.

7. Що таке макрос в асемблері AVR, як він створюється за допомогою директив `.macro` та `.endmacro`, і чим він відрізняється від підпрограми?

А) Макрос – це функція, яку викликає користувач з командного рядка.

Б) Макрос викликається за допомогою адреси в EEPROM.

В) Макрос виконується автоматично при ввімкненні живлення.

Г) Макрос вставляє свій код безпосередньо у місце виклику, не передбачаючи повернення, що збільшує розмір програми, але пришвидшує виконання.

8. Яке призначення регістрів `DDRx`, `PORTx` і `PINx` у керуванні портами вводу/виводу AVR, і як вони взаємодіють для забезпечення правильного функціонування зовнішніх пристроїв?

А) Вони застосовуються лише при використанні UART.

Б) `DDRx` визначає напрямок порту, `PORTx` – логічний рівень або підтягуючий резистор, `PINx` – читання поточного стану виводів.

В) Всі три регістри відповідають лише за внутрішню периферію.

Г) Вони є тільки у Flash-пам'яті і не змінюються програмно.

9. Чому в асемблерних програмах для AVR рекомендується вставляти команду `nop` між записом у порт та читанням із того самого порту, і що відбувається без цієї паузи?

А) Без `nop` МК зависне.

Б) Щоб вивід змінив колір світлодіода.

В) Через наявність синхронізуючої ланки читання з `PINx` одразу після `out` може дати некоректне значення, бо порт ще не оновив стан.

Г) Це вимога стандарту ISO-9001.

11. Яким чином реалізується ініціалізація стеку в мікроконтролері AVR під час старту програми, і чому критично важливо явно вказувати його початкову адресу навіть у випадках, коли вона задається автоматично?

А) Стек в AVR автоматично обнуляється, і його налаштування не потрібне.

Б) Ініціалізація стеку не впливає на роботу програми.

В) Хоча за замовчуванням стек розміщується на кінці SRAM, його потрібно явно ініціалізувати через регістри SPL та SPH для уникнення збоїв у викликах підпрограм чи переривань.

Г) Стек в AVR не використовується, тому ініціалізація не виконується.

12. Як працює підтягуючий резистор у режимі входу мікроконтролера AVR, і чому його вмикання є обов’язковим при підключенні кнопок або вхідних ліній без гарантованого рівня?

А) Підтягуючі резистори лише знижують рівень шуму.

Б) Вони вимикають порт.

В) Підтягуючі резистори підключають вхід до Vcc через PORTx при DDRx = 0, запобігаючи флуктуаціям рівня та збоїв у логіці при відкритому вході.

Г) Їх використовують лише для внутрішніх таймерів.

15. Чому непідключені виводи вводу/виводу AVR повинні бути або підключені через резистори до фіксованого рівня, або налаштовані як виходи, і які небажані ефекти можуть виникнути при порушенні цього правила?

А) Такі виводи просто не працюватимуть.

Б) Вони будуть автоматично вимкнені AVR.

В) Плаваючий вхід може споживати струм через паразитні ємності, створюючи фальшиві спрацьовування або нестабільність програми.

Г) Це потрібно лише для портів, підключених до дисплея.

17. Яким чином реалізується керування світлодіодами через порти AVR, та чому при їх підключенні необхідно дотримуватись полярності та застосовувати обмежувальні резистори?

А) Без резисторів світлодіоди працюють яскравіше.

Б) Резистори необхідні лише при керуванні з EEPROM.

В) Підключення виконується тільки до SPI.

Г) Без резистора струм перевищить допустимий рівень, що може призвести до виходу з ладу світлодіода або порту; полярність визначає напрям струму.

18. Чому важливо виконувати ініціалізацію регістрів загального призначення та оперативної пам’яті при старті програми в AVR, навіть якщо в більшості випадків вони вже обнулені?

А) Це рекомендовано лише для симуляторів.

Б) Регістри й пам’ять завжди обнуляються автоматично.

В) Без явної ініціалізації залишкові значення з попереднього запуску можуть призвести до непередбачуваних результатів під час виконання.

Г) Ініціалізація потрібна лише в мовах високого рівня.

20. Що таке високоімпедансний стан (Hi-Z) у контексті цифрових входів AVR, і чому цей режим важливий для спільного використання ліній даних з іншими пристроями?

А) Hi-Z – це стан, у якому вхід практично не споживає струм, дозволяє не заважати сигналам на шині, з якою він тимчасово не взаємодіє.

Б) Hi-Z дає змогу порту передавати дані з високою потужністю.

В) Hi-Z означає, що порт працює в аналоговому режимі.

Г) Hi-Z активується лише через переривання.

Таблиця правильних відповідей

1	2	3	4	5	6	7
в	а	в	в	б	в	г

8	9	10	11	12	13	14	15
б	в	в	а	в	г	в	в

ТЕМА 4. Мови програмування високого рівня. Огляд, особливості, історія виникнення

1. Яким чином мови програмування високого рівня спростили процес розробки програм порівняно з машинним кодом і асемблером, і яку ключову перевагу вони надали програмістам?

А) Мови високого рівня зберігають програми без компіляції.

Б) Вони дають змогу створювати програми без пам’яті.

В) Вони наближені до природної мови, що дозволяє писати алгоритми без знання архітектури комп’ютера.

Г) Вони дозволяють змінювати прошивку BIOS.

2. Яке з тверджень найточніше описує головну відмінність між мовами програмування низького рівня (машинний код, асемблер) і мовами високого рівня (Pascal, C, Python)?

А) Мови низького рівня мають більший синтаксис.

Б) Мови високого рівня є платформонезалежними, тобто не прив’язані до конкретної апаратури.

В) Мови високого рівня виконуються без процесора.

Г) Мови низького рівня підтримують графіку, а високого – ні.

3. Яким чином виникла перша мова програмування високого рівня, яка її назва, і яка була її головна мета в історичному контексті розвитку обчислювальної техніки?

А) Java була створена першою для розробки мобільних додатків.

Б) Python – перша мова для вебпрограмування.

В) Fortran (1957) був першою мовою високого рівня, створеною для наукових обчислень та автоматизації чисельних методів.

Г) C++ був розроблений для створення баз даних.

4. Яке твердження найкраще описує етап компіляції в мовах програмування високого рівня, таких як C або Pascal, і в чому полягає його основна функція?

А) Компіляція перетворює вихідний код у машинний код, який може бути виконаний процесором.

Б) Компіляція – це процес перевірки орфографії в коді.

В) Компіляція створює копію операційної системи.

Г) Компіляція видаляє помилки у комп’ютері.

5. Чим відрізняється мова C від мов четвертого покоління, таких як SQL або MATLAB, з точки зору наближеності до апаратного рівня та універсальності?

А) C – вузькоспеціалізована для баз даних.

Б) С використовує візуальне середовище.

В) С – мова третього покоління, наближена до апаратури, а SQL – четвертого, орієнтована на опис задач.

Г) Мова С не потребує компіляції.

6. Яка з мов програмування вважається об’єктно-орієнтованою за своєю природою, і яка її головна особливість у порівнянні з процедурними мовами?

А) Pascal – об’єктно-орієнтована, бо використовує функції.

Б) Java – адже програмування базується на класах і об’єктах.

В) С – має об’єкти в пам’яті.

Г) HTML – основа об’єктного програмування.

7. Що з наведеного найточніше визначає мову програмування Pascal, і чому вона здобула популярність у 80-х роках як навчальна мова?

А) Pascal – це система комп’ютерної графіки.

Б) Мова Pascal була створена для розробки вебдодатків.

В) Pascal – структурована мова, спеціально розроблена для навчання алгоритмам і принципам програмування.

Г) Вона призначена виключно для мікроконтролерів.

8. Яку роль відіграє компілятор у мовах програмування високого рівня, і які ключові етапи проходить програма під час трансляції в машинний код?

А) Компілятор зберігає копії коду на сервері.

Б) Компілятор не обробляє помилки коду.

В) Компілятор використовується лише у HTML.

Г) Компілятор аналізує код, перевіряє синтаксис, оптимізує та генерує машинний код, який зберігається у виконуваному файлі.

9. Чим відрізняється процедурне програмування від об’єктно-орієнтованого, і як це впливає на організацію великих програмних систем?

А) Процедурне програмування використовує лише графіку.

Б) У процедурному підході акцент на змінних, в об’єктному – на об’єктах, які поєднують дані та методи, що полегшує масштабування та повторне використання коду.

В) Об’єктне програмування не може бути реалізоване без SQL.

Г) Відмінностей між ними немає.

10. Яку мову програмування створено спеціально для швидкого прототипування, автоматизації систем керування та обробки даних, з особливою популярністю в науковому середовищі?

А) Java

Б) Fortran

В) MATLAB
Г) Pascal

11. Що таке синтаксис у контексті мов програмування, і чому його дотримання є критично важливим для компіляції та виконання програм?

А) Синтаксис визначає структуру та порядок запису інструкцій; порушення його призводить до помилок компіляції.

Б) Синтаксис – набір правил щодо розміру пам’яті.

В) Синтаксис – це довжина коду.

Г) Синтаксис відповідає лише за назви файлів.

12. Яка з мов програмування високого рівня була спеціально розроблена в Microsoft для створення прикладних програм у середовищі .NET, і поєднує риси C++ та Java?

А) VisualBasic
Б) JavaScript
В) C#
Г) Assembly

13. Яким чином мова програмування Python реалізує динамічну типізацію, і яку перевагу це дає програмісту в процесі розробки?

А) Типи не мають значення у Python.

Б) Python не перевіряє типи під час виконання.

В) Типи у Python жорстко закодовані в кожній змінній.

Г) Змінна отримує тип під час виконання, що дозволяє писати код швидше без попереднього оголошення типів.

14. Який вплив на розвиток мов програмування високого рівня мали стандарти, такі як ISO/IEC, і чому важливо мати стандартизовані версії мов, наприклад, C або Ada?

А) Стандарти потрібні лише для електроніки.

Б) Стандарти спрощують графіку в мовах.

В) Стандартизація забезпечує сумісність, довгострокову підтримку коду та унеможливлює різночитання в реалізаціях.

Г) Вона стосується лише документації до мови.

15. Яку роль у навчанні алгоритмічному мисленню відіграють мови програмування високого рівня, і чому їх вивчення є важливою складовою технічної освіти?

А) Вони замінюють вивчення математики.

Б) Дозволяють засвоїти логіку, структуру даних, цикли, умовні оператори – що лежить в основі інженерного мислення і розв’язання практичних задач.

В) Дають змогу обходити складні предмети.

Г) Вивчаються лише для розваги.

Таблиця правильних відповідей

1	2	3	4	5	6	7
в	б	в	а	в	б	в

8	9	10	11	12	13	14	15
г	б	в	а	в	г	в	б

ТЕМА 5. Мова програмування С. Основні інструменти (частина 1)

1. Яким чином алфавіт мови програмування С визначає набір допустимих символів, та чому важливо розрізняти великі й малі літери при створенні ідентифікаторів?

А) Всі літери автоматично перетворюються на великі.

Б) Мова С нечутлива до регістру літер.

В) Усі ключові слова, змінні та функції в С є чутливими до регістру, тобто `Var` і `var` – це різні ідентифікатори.

Г) Ідентифікатори формуються лише з цифр.

2. Яку роль виконують директиви препроцесора в мові С, наприклад `#include` та `#define`, і на якому етапі обробки програми вони використовуються?

А) Вони виконуються під час виконання програми.

Б) Директиви впливають тільки на форматування коду.

В) Препроцесор обробляє їх до компіляції: `#include` вставляє в програму заголовкові файли, а `#define` виконує макрозаміни.

Г) Директиви працюють лише в Windows.

3. Яка з наведених конструкцій найкраще описує повний синтаксис функції в мові С, включаючи заголовок і тіло функції?

А) функція(параметри) { };

Б) тип_повернення ім’я(параметри) { оператори };

В) ім’я: оператори.

Г) return(значення);

4. Що таке лексема в контексті мови С, які типи лексем існують, і чому важливо розрізняти їх при трансляції програми?

А) Лексема – це будь-яке слово.

Б) Лексеми – лише коментарі.

В) Лексема – це мінімальна одиниця мови (ключові слова, ідентифікатори, оператори, розділювачі), з якої формуються вирази.

Г) Лексема – це змінна типу int.

5. Чим відрізняється тернарний умовний оператор `?:` від традиційного `if`, та які переваги його застосування в компактності коду?

А) `вираз ? результат1 : результат2;` дозволяє скоротити запис простої умови в один рядок, зберігаючи функціональність `if-else`

Б) Це форма скороченого порівняння з двома умовами.

В) `.?:` працює лише з числами.

Г) Його використовують тільки в С++.

6. Яка роль оператора `switch` у програмі мовою С, і в яких випадках його доцільно використовувати замість ланцюга умов `if-else`?

А) `switch` працює тільки з логічними значеннями.

Б) Його призначення – зупинити виконання програми.

В) `switch` дозволяє обирати один із варіантів на основі значення цілого виразу, є зручним при роботі з множиною варіантів.

Г) Його використовують лише для циклів.

7. Чому при використанні оператора `switch` у мові С важливо включати оператор `break` у кожному блоці `case`, та що відбувається при його відсутності?

А) Нічого, `break` необов'язковий.

Б) Програма автоматично закриється.

В) `break` працює тільки в циклах.

Г) Без `break` виконання продовжиться у наступний `case`, що може призвести до логічної помилки.

8. У чому полягає відмінність між циклами `while`, `do...while` та `for` у мові С, і які з них є циклами з передумовою або постумовою?

А) Всі три цикли однакові.

Б) `while` і `for` перевіряють умову на початку, а `do...while` – після першого виконання тіла циклу.

В) `do...while` не працює без бібліотек.

Г) `for` не є циклом.

9. Яке призначення мають вкладені цикли у мові програмування С, і як правильно організувати їх структуру для уникнення конфліктів між зовнішніми та внутрішніми лічильниками?

А) Вкладені цикли потрібні лише у графічному інтерфейсі.

Б) Вкладені цикли використовуються, коли потрібно обчислити лише одне значення.

В) Вкладені цикли дозволяють реалізувати багатовимірні алгоритми (наприклад, матриці), і необхідно уникати дублювання імен змінних у межах кожного рівня.

Г) У мові С не допускаються вкладені цикли.

10. Яку функцію виконує оператор `continue` в циклі мови С, і як він змінює стандартну послідовність виконання операторів у тілі циклу?

А) `continue` пропускає решту інструкцій у тілі циклу для поточної ітерації та переходить до наступної перевірки умови.

Б) `continue` запускає підпрограму.

В) `continue` припиняє програму.

Г) `continue` повторює останню команду.

11. Яким чином у мові С можна оголосити змінну, значення якої буде збережено між викликами функції, і яка ключова особливість цієї змінної?

А) Через ключове слово `const`

Б) За допомогою директиви `#define`

В) Через ключове слово `static`, що зберігає значення змінної між викликами функції.

Г) Через `global`.

12. Що таке область видимості змінної в мові С, та як вона впливає на доступ до змінної в межах різних частин програми (функцій, блоків)?

А) Область видимості не має значення.

Б) Змінна може бути доступною лише в межах тієї області (функції, блоку), в якій вона була оголошена, якщо не має модифікатора `extern`.

В) Усі змінні доступні будь-де.

Г) Змінні автоматично глобальні.

13. Як працює оператор `break` у циклах мови С, і чим він відрізняється від логічного завершення циклу за умовою?

А) `break` повторює цикл.

Б) `break` завершує весь код програми.

В) `break` негайно виходить із циклу, незалежно від умови продовження, і передає керування за його межі.

Г) `break` працює тільки у функціях.

14. Чим відрізняється глобальна змінна від локальної в мові С, і яку роль у цьому відіграє її місце оголошення?

А) Глобальна змінна має іншу структуру.

Б) Глобальні змінні створюються автоматично.

В) Глобальні змінні не компілюються.

Г) Глобальна змінна оголошена поза функціями, доступна в усьому коді, тоді як локальна – лише у межах функції чи блоку, де її оголошено.

15. Чому важливо правильно визначати початкові значення змінних перед їх використанням у виразах, зокрема в умовних операціях або циклах, у мові С?

А) Інакше компілятор автоматично поставить нуль.

Б) Компілятор виправляє такі помилки сам.

В) Використання невизначеної змінної призведе до не контрольованого результату, бо в ній може бути випадкове значення з пам’яті.

Г) Немає жодного значення – все працює.

Таблиця правильних відповідей

1	2	3	4	5	6	7
в	в	б	в	а	в	г

8	9	10	11	12	13	14	15
б	в	а	в	б	в	г	в

ТЕМА 6. Мова програмування С. Основні інструменти (частина 2)

1. Яке твердження найкраще описує спосіб оголошення та використання одномірного масиву в мові С, і яку помилку може спричинити звернення за межі допустимого індексу?

А) Масиви автоматично розширюються при потребі.

Б) Звернення за межі призводить до повторення останнього елемента.

В) Масив оголошується як `тип ім'я[розмір];`, і звернення за межі може призвести до пошкодження пам'яті або збоїв.

Г) Масиви мають динамічний розмір за замовчуванням.

2. Як працює нульовий індекс у масиві С, чому відлік елементів починається саме з нуля, і які наслідки неправильного розуміння цього принципу?

А) Адреса масиву вказує на перший елемент, тому `array[0]` фактично є базовою адресою; помилки в індексації можуть призвести до логічних збоїв.

Б) Початковий індекс визначається компілятором.

В) Відлік з нуля введено випадково.

Г) Нульовий індекс означає кінець масиву.

3. Яким чином передається масив у функцію в мові С, і чому зміни, внесені у функції, відображаються на масиві в основній програмі?

А) Масиви передаються за значенням.

Б) Масив автоматично копіюється у функцію.

В) Масив передається за адресою (як вказівник), тому будь-які зміни в ньому відбуваються у вихідному масиві.

Г) Масив не може бути переданий у функцію.

4. Яке з наведених тверджень найточніше описує багатовимірні масиви в мові С, та як формується звернення до елемента масиву типу `intmatrix[3][4];`?

А) Це набір одновимірних масивів.

Б) Звернення до `matrix[i][j]` означає: рядок `i`, стовпець `j`.

В) Індекси вказують тип змінної.

Г) Такі масиви доступні лише в С++.

5. Яким чином працює масив символів у мові С, та яку ключову роль відіграє символ `'\0'` у кінці рядка?

А) Це символ пропуску.

Б) `'\0'` – перший символ у кожному рядку.

В) Символ `'\0'` (null) позначає кінець рядка, тому функції роботи з рядками зчитують до нього.

Г) `'\0'` видаляє всі символи.

6. Що таке структура в мові С, як вона оголошується, та у чому її перевага порівняно з масивами при зберіганні гетерогенних даних?

А) Структура містить лише числа.

Б) Структура – це масив функцій.

В) Структура дозволяє об'єднати змінні різних типів в одну логічну одиницю, на відміну від масиву, де всі елементи одного типу.

Г) Структура створюється автоматично з масиву.

7. Як відбувається ініціалізація структури в мові С, і яким чином можна звертатися до її членів?

А) Через квадратні дужки.

Б) Звернення не можливе.

В) Тільки через спеціальні функції.

Г) За допомогою крапки: `struct_name.member`, або через вказівник – `->`.

8. Що таке вказівник у мові С, як він оголошується, та чому важливо знати тип змінної, на яку він вказує?

А) Тип не має значення для вказівника.

Б) Вказівники – це лише адреси без типу.

В) Тип вказівника визначає інтерпретацію вмісту пам'яті: `int *ptr;` означає, що ptr містить адресу, де зберігається `int`.

Г) Вказівники доступні лише у мовах високого рівня.

9. Як працює оператор розіменування* у мові С, та чим він відрізняється від оператора оголошення вказівника?

А) У виразі `*ptr` оператор розіменування повертає значення, на яке вказує `ptr`, тоді як у `int *ptr;` – * вказує, що ptr є вказівником.

Б) * завжди означає множення.

В) Вони однакові.

Г) * не використовується у вказівниках.

10. Яким чином працює оператор взяття адреси & у мові С, і яку роль він відіграє при передачі змінних у функції за адресою?

А) Оператор & потрібен лише для виводу на екран.

Б) & використовується тільки в масивах.

В) `&x` повертає адресу змінної `x`, що дозволяє передати її у функцію для модифікації значення без повернення результату.

Г) Оператор & замінює команду `scanf`.

11. Чим відрізняється звернення до елемента структури за вказівником (`->`) від звичайного доступу через змінну (`.`), і в яких випадках кожен підхід використовується?

А) Обидва використовуються взаємозамінно.

Б) `.` використовується в масивах, а `->` – у рядках.

В) `->` дозволяє створювати структуру автоматично.

Г) `.` – для об'єктів структури, `->` – для вказівників на структури.

12. Яке значення має нульовий вказівник (`NULL`) у мові С, у яких ситуаціях він використовується, і чому важливо перевіряти його перед доступом до пам'яті?

А) `NULL` використовується для очищення екрана.

Б) `NULL` – синонім нуля як значення.

В) `NULL` означає, що вказівник не посилається на жодну дійсну область пам'яті, і спроба розіменування спричинить аварійне завершення.

Г) `NULL` – це спецсимвол у рядках.

13. Що таке динамічне виділення пам'яті у мові С, які функції бібліотеки `<stdlib.h>` для цього використовуються, і як звільнити раніше виділену область?

А) Пам'ять виділяється автоматично.

Б) `malloc`, `calloc` – для виділення, `free` – для звільнення.

В) Потрібна команда `save()`.

Г) Виділення працює лише з масивами.

14. Як вказівники дозволяють реалізовувати функції, які змінюють значення декількох змінних одразу, і чому це не можна зробити, передаючи значення напряму?

А) Значення передаються у будь-якому вигляді.

Б) Передача за значенням не змінює оригінальні дані, а через вказівники можна безпосередньо змінювати вміст переданої пам'яті.

В) Змінні змінюються лише через масив.

Г) Потрібна функція `copy()` з бібліотеки.

15. Чому вказівники є необхідними для роботи з масивами та рядками в мові С, і як взаємозв'язок між іменем масиву та його адресою пояснює їхню поведінку?

А) Вказівники не використовуються з масивами.

Б) Ім'я масиву – це змінна, яка зберігає довжину.

В) Ім'я масиву фактично є вказівником на перший елемент, тому доступ до елементів здійснюється як через індексацію, так і через арифметику вказівників.

Г) Масиви працюють без вказівників.

Таблиця правильних відповідей

1	2	3	4	5	6	7
в	а	в	б	в	в	г

8	9	10	11	12	13	14	15
в	а	в	г	в	б	б	в

ТЕМА 7. Мова програмування С. Структурування складних програм

1. Яким чином структурне програмування в мові С дозволяє розділити велику програму на логічні блоки, та яку роль у цьому відіграє організація функцій?

А) Усі блоки повинні міститися в одній функції.

Б) Програма структурована у вигляді безкінечного циклу.

В) Кожен логічний блок реалізується окремою функцією, що спрощує розуміння, повторне використання та тестування окремих частин програми.

Г) Розбиття на блоки в С не передбачене.

2. Яким чином можна реалізувати повторне використання функцій у різних частинах великої програми, і яку користь це дає при модернізації коду?

А) Усі функції мають бути дубльовані вручну.

Б) Функції зберігаються в окремих модулях і підключаються до основного коду, що дозволяє оновлювати логіку лише в одному місці.

В) Повторне використання не рекомендується.

Г) Функції не можна використовувати двічі.

3. Яке призначення має бібліотека функцій у мові С, та як вона використовується при структуруванні програм, що мають багато повторюваних обчислень?

А) Бібліотека – це лише довідковий файл.

Б) Бібліотеки використовуються лише в Linux.

В) Бібліотека – це набір заздалегідь реалізованих функцій, які підключаються до програми, що спрощує розробку і скорочує код.

Г) Бібліотека створює візуальний інтерфейс.

4. Як здійснюється підключення стандартної або користувацької бібліотеки функцій у мові С, і чим відрізняється #include<...> від #include "..."?

А) Кутові дужки вказують на стандартну бібліотеку, а лапки – на локальні (користувацькі) файли.

Б) #include використовується тільки з файлами .exe.

В) Обидва варіанти працюють однаково.

Г) #include лише виводить повідомлення.

5. Яким чином у мові С відбувається поділ програмного коду на окремі файли .c та .h, і чому це є ключовим для підтримки великих проектів?

А) Це потрібно лише при програмуванні графіки.

Б) Усі функції повинні бути в одному файлі.

В) Код функцій зберігається в .c, прототипи – в .h; це дозволяє ізолювати реалізацію від інтерфейсу і спростити редагування.

Г) Поділ потрібен тільки при компіляції з Python.

6. Яким чином можна уникнути багаторазового включення одного й того самого заголовкового файлу при компіляції великого проєкту, та чому це важливо?

А) Повторне включення не впливає на компіляцію.

Б) Компілятор сам визначає, що файл вже підключено.

В) Потрібно перейменовувати всі файли вручну.

Г) За допомогою умовних директив #ifndef, #define, #endif, які обмежують повторне включення одного й того самого файлу.

7. Яким чином можна організувати обробку файлів (читання і запис) у мові C за допомогою стандартної бібліотеки stdio.h, і які функції для цього застосовуються?

А) Файли можна відкривати лише вручну.

Б) Стандартна бібліотека не підтримує роботу з файлами.

В) Функції fopen, fclose, fscanf, fprintf дозволяють працювати з текстовими та двійковими файлами в C.

Г) Робота з файлами відбувається через HTML.

8. Чому при роботі з файлами у мові C важливо перевіряти, чи відкрився файл успішно, і як це робиться правильно?

А) Файл завжди відкривається, якщо існує.

Б) Помилки відкриття ігноруються компілятором.

В) Необхідно перевіряти, чи fopen повертає ненульове значення, бо відкриття може провалитися через відсутність файлу або права доступу.

Г) Помилка відкриття викликає перезавантаження системи.

9. Яким чином реалізується модульність при створенні великих проєктів мовою C з кількох .c та .h файлів, і як компілятор та компоновщик обробляють ці компоненти?

А) Компоновщик автоматично об’єднує всі файли в один.

Б) Кожен файл компілюється окремо, а потім всі об’єктні файли зв’язуються в один виконуваний файл за допомогою лінкера.

В) Компіляція всіх файлів відбувається одночасно без розділення.

Г) У мові C не можна використовувати більше одного файлу.

10. Яким чином структура великого C-проєкту дозволяє спростити командну розробку програмного забезпечення, і які переваги дає поділ на окремі модулі?

А) Один програміст має писати всю програму.

Б) Поділ ускладнює спілкування між розробниками.

В) Кожен розробник може працювати над окремим модулем незалежно, використовуючи спільні заголовки, що зменшує конфлікти й підвищує ефективність.

Г) Модулі потрібні лише в системному програмуванні.

11. Яким чином у мові С забезпечується зв’язок між різними модулями (файлами), які містять змінні та функції, та як уникнути дублювання імен при цьому?

А) Для доступу до змінної, оголошеної в іншому модулі, використовують ключове слово extern, що дозволяє компілятору знайти її визначення.

Б) Усі файли повинні мати різні імена.

В) Змінні автоматично унікальні.

Г) Не можна звертатись до змінних інших файлів.

12. Яким чином можна обмежити доступ до змінної або функції лише в межах одного модуля, і чому це важливо для надійності великої програми на С?

А) Таке обмеження суперечить принципам С.

Б) Змінні завжди глобальні.

В) Використання ключового слова static перед змінною або функцією дозволяє приховати її від інших модулів, забезпечуючи інкапсуляцію.

Г) Це досягається за допомогою бібліотеки secure.h.

13. Яка функція використовується для перевірки досягнення кінця файлу під час читання даних у мові С, та чому така перевірка є необхідною в циклічному читанні?

А) file.end()

Б) eof(file)

В) feof(file_pointer) перевіряє, чи було досягнуто кінець файлу, що необхідно для запобігання нескінченному читанню.

Г) close(file)

14. Яким чином можна реалізувати функцію логування даних у зовнішній файл у програмі на С, і чому важливо закривати файл після запису?

А) Файл закривається автоматично при main().

Б) Логування можна робити лише через printf.

В) Для логування потрібна база даних.

Г) Логування – це запис через fprintf у файл, відкритий у режимі "a"; закриття fclose звільняє ресурси і запобігає втраті даних.

25. Яку роль у структурованій програмі на C відіграє головна функція main(), і як її оголошення впливає на запуск та завершення програми?

А) main() є змінною середнього рівня.

Б) main() може бути будь-де в коді.

В) main() – це точка входу, її сигнатура визначає параметри запуску; завершення через return може передавати код успішності.

Г) main() запускається з бібліотеки.

Таблиця правильних відповідей

1	2	3	4	5	6	7
в	б	в	а	в	г	в

8	9	10	11	12	13	14	15
в	б	в	а	в	в	г	в

ТЕМА 8. Використання об’єктно-орієнтованого програмування

1. Яким чином використання принципів об’єктно-орієнтованого програмування (ООП) у розробці прошивок для мікроконтролерів дозволяє підвищити масштабованість та повторне використання коду?

А) ООП дозволяє створювати абстракції апаратних ресурсів у вигляді класів, полегшуючи повторне використання та розширення функціональності.

Б) Об’єкти не підтримуються апаратно мікроконтролерами.

В) ООП ускладнює адаптацію програм до нових платформ.

Г) ООП можна застосовувати лише в десктопному середовищі.

2. Яка з переваг використання класів у прошивках для мікроконтролерів є ключовою при створенні драйверів для периферійних пристроїв, таких як датчики, екрани, двигуни тощо?

А) Клас не може взаємодіяти з апаратурою.

Б) Класи вимагають зовнішньої пам’яті.

В) Класи дозволяють інкапсулювати логіку взаємодії з пристроєм, роблячи інтерфейс керування простим, незалежним та переносимим.

Г) Класи використовуються тільки у графіці.

3. Що таке інкапсуляція в об’єктно-орієнтованому програмуванні, і як вона застосовується при розробці мікроконтролерних програм?

А) Це захист коду від компіляції.

Б) Інкапсуляція означає приховування внутрішньої реалізації об’єкта (змінних, методів) від зовнішнього коду, що спрощує інтерфейс та захищає логіку.

В) Це створення окремих змінних для кожної функції.

Г) Це передача значення між класами.

4. У чому полягає перевага використання поліморфізму при розробці коду для мікроконтролерів, що керують декількома типами пристроїв через спільний інтерфейс?

А) Поліморфізм знижує швидкість виконання.

Б) Його можна реалізувати лише в C#.

В) Він дозволяє створювати єдиний інтерфейс, через який можна взаємодіяти з різними типами пристроїв, що мають власну реалізацію.

Г) Поліморфізм дозволяє уникнути оголошення змінних.

5. Яке ключове призначення має наслідування в ООП, і як цей механізм застосовується в мікроконтролерних бібліотеках (наприклад, Arduino)?

А) Наслідування дозволяє копіювати змінні в масив.

Б) Наслідування дає змогу створювати нові класи без об’єктів.

В) Наслідування дає змогу створювати нові класи, що розширюють або змінюють поведінку базового класу, не змінюючи його код.

Г) Це механізм для копіювання файлів.

6. Чому застосування ООП у мікроконтролерах потребує уважного контролю використання пам’яті, і які можливі ризики при надмірному використанні об’єктів?

А) Мікроконтролери мають автоматичний збірник сміття.

Б) Більше об’єктів прискорює виконання.

В) Пам’ять об’єктів стискається при компіляції.

Г) Об’єкти займають пам’ять у стеку або купі, а їх надмірне використання може призвести до переповнення пам’яті або збоїв у реальному часі.

7. Яке обмеження слід враховувати при використанні віртуальних функцій у прошивках для мікроконтролерів?

А) Віртуальні функції потребують таблиці віртуальних викликів (vtable), що збільшує обсяг коду та час виконання.

Б) Вони не можуть бути оголошені в заголовкових файлах.

В) Віртуальні функції доступні тільки в Python.

Г) Вони автоматично компілюються в асемблер.

8. У яких мовах програмування, що підтримують ООП, найчастіше реалізуються прошивки для мікроконтролерів, і чому?

А) Java через JVM.

Б) Python через скриптові можливості.

В) C++ завдяки низькорівневому доступу до апаратури та підтримці об’єктної моделі.

Г) HTML через простоту синтаксису.

9. Яким чином шаблони (templates) у C++ дозволяють реалізовувати гнучкі бібліотеки для мікроконтролерів, наприклад, універсальні драйвери для UART, SPI або I2C?

А) Шаблони – це тільки частина HTML.

Б) Шаблони дозволяють створювати функції або класи, які адаптуються до типів пристроїв під час компіляції без дублювання коду.

В) Шаблони не можна використовувати з периферією.

Г) Шаблони лише оформлюють код.

10. Яке обмеження може виникати при використанні конструктора і деструктора у класах в мікроконтролерних прошивках, особливо при ініціалізації апаратних ресурсів?

А) Якщо ресурс ініціалізується в конструкторі, але не вивільняється в деструкторі, це може призвести до зависання або витоку пам’яті.

Б) Деструктори запускаються автоматично при помилці.

В) Їх використання зменшує розмір програми.

Г) Конструктори заборонені в класах.

11. У чому полягає особливість реалізації таймерів, UART, SPI або інших периферійних модулів через об’єкти, та яку структуру класу зазвичай застосовують для цього?

А) Кожен об’єкт створює копію периферії.

Б) Периферію не можна інкапсулювати в об’єкт.

В) Класи зазвичай мають методи `init()`, `start()`, `stop()`, а також приватні поля, що містять конфігурацію – це забезпечує чітку модульність.

Г) Для периферії використовують лише глобальні змінні.

12. У чому перевага використання шаблонних класів при роботі з апаратними абстракціями мікроконтролера, наприклад, для створення універсальних класів керування портами або таймерами?

А) Вони зменшують кількість методів.

Б) Шаблони непотрібні для ООП.

В) Шаблони дозволяють створювати гнучкі об’єкти, які адаптуються до різного апаратного коду без дублювання функцій.

Г) Вони автоматично виводять текстові повідомлення.

13. Як впливає використання вбудованих функцій (`inline`) в ООП-підході до мікроконтролерів, і коли їх доцільно застосовувати?

А) `inline` потрібна лише для оголошення класу.

Б) Вона робить функції недоступними для налагодження.

В).Вона використовується тільки в Arduino IDE.

Г) Вбудовані функції зменшують накладні витрати на виклик функції, що критично у швидкодійних мікроконтролерних задачах

14. Яку роль може відігравати шаблонний клас-перевірка (`wrapper`) у мікроконтролерній системі, що інкапсулює взаємодію з ресурсами через API?

А) Wrapper-клас потрібен тільки для мережі.

Б) Це частина стандартної бібліотеки.

В) Wrapper дозволяє створити універсальний інтерфейс для роботи з різними реалізаціями однієї функціональності, спрощуючи тестування.

Г) Wrapper автоматично обробляє інтеррукти.

15. Яким чином використання класів у мові C++ для мікроконтролерів узгоджується з принципами ефективного кодування, і як можна зменшити накладні витрати на ООП?

А) Класи завжди збільшують об’єм коду.

Б) Використання `inline`, відмова від віртуальності, обмежене використання динаміки – все це дозволяє поєднати ООП із високою ефективністю.

В) Класи перетворюють C на інтерпретовану мову.

Г) Не можна оптимізувати класи.

Таблиця правильних відповідей

1	2	3	4	5	6	7
а	в	б	в	в	г	а

8	9	10	11	12	13	14	15
в	б	а	в	в	г	в	б

3 ЗАВДАННЯ ДЛЯ ПІДГОТОВКИ ДО ІСПИТУ

1 модуль

1. Історія створення і розвитку засобів програмного забезпечення мікропроцесорів.
2. Мікропроцесор, принципи організації застосування і актуальність.
3. Архітектура, будова і особливості програмування мікропроцесорів.
4. Програмне забезпечення мікропроцесорів. Особливості і етапи створення програм.
5. Особливості програмування мікропроцесорних засобів і їх відмінність від інших типів програмування.
6. Етапи створення програми і особливості її відлагодження для мікропроцесорів.
7. Типи архітектур мікропроцесорів і їх особливості. Програмування і сучасна будова, а також апаратне виконання мікропроцесорів.
8. Особливості програмування мікропроцесорів, що входять до засобів безпеки.
9. Етапи компіляції програми на Ассемблері, види і типи компіляторів. Основні етапи розвитку і програмні модулі. що входять до складу компіляторів і середовищ розробки.
10. Особливості програмування AVR-мікроконтролерів в середовищі AVR Studio. Основні етапи створення програми і її відлагодження.
11. Особливості програмування мікроконтролерів на мові С.
12. Особливості програмування і синтаксис команд на мові Асемблер.
13. Команди групи переміщення даних на мові Ассемблер.
14. Команди групи арифметичних операцій на мові Ассемблер.
15. Команди групи логіних операцій на мові Ассемблер.
16. Система переривань та її організації при програмуванні контроллерів.
17. Які типи пам’яті присутні в мікроконтролерах серії AVR?
18. Для чого використовуються мітки?
19. Як здійснюється звернення мікроконтролера до периферійних пристроїв.
20. Директиви асемблера.
21. Операнди, формати даних
22. Арифметичні і логічні інструкції.
23. Інструкції розгалуження. Інструкції пересилки даних.
24. Що таке керуючий оператор і керуюча конструкція?
25. Що таке складений оператор?
26. Що таке вираз?
27. Які функції стають доступними при підключенні файлу і як їх використовувати?

28. Якими операторами може бути реалізоване розгалуження у програмі?

29. Опишіть оператор вибору switch.

30. Навіщо потрібен оператор break в конструкції switch?

31. Чи можна об’єднувати розділи case?

32. Чи можна вкладати умовні оператори один в другий?

33. Які різновиди визначення умови входу в цикл в С ви знаєте?

34. Що таке тіло циклу? Ітерація циклу?

35. Які принципові відміни між циклами з перед- і постумовою?

36. У якому випадку цикл не буде виконано жодного разу?

37. У якому випадку цикл буде виконано хоч раз за будь-яких умов?

38. Опишіть синтаксис і призначення кожної секції оператора for.

39. Опишіть механізм дії оператора for.

40. Що таке лічильник циклу?

2 модуль

1. Скільки і якого типу може бути лічильників у циклі for в С?

2. Параметри яких типів можна використовувати у циклі for?

3. Чи допустимо використання оператора кома при визначенні умови виконання циклу?

4. Які обмеження на глибину вкладеності циклів ви знаєте і з чим вони пов’язані?

5. Що таке «безкінечний цикл»?

6. У яких випадках можна використовувати безкінечні цикли?

7. Про що потрібно пам’ятати, щоб не організувати безкінечний цикл?

8. Які різновиди циклів С ви знаєте і чим вони відрізняються?

9. Назвіть чотири оператори безумовного переходу.

10. Для чого призначений оператор return? оператор break?

11. Опишіть відмінність оператора continue від оператора break.

12. Опишіть механізм дії оператора goto.

13. Опишіть функції exit()та abort(). Як вони функціонують?

14. Як оголосити вказівник? Як ініціалізувати?

15. Що таке розіменування вказівника?

16. Які операції називають операціями адресної арифметики?

17. Як змінюється значення вказівника при застосуванні до нього операції інкременту?

18. Як можна звернутися до елементу масиву за допомогою вказівника на його нульовий елемент?

19. Як обчислити різницю значень двох вказівників?

20. За яких умов до вказівників можна застосовувати оператор присвоювання?

21.Як оголосити двовимірний масив? У яких випадках можна опустити визначення кількості елементів в оголошенні масиву?

22. Що означає і для чого призначений символ ‘\0’?

23. Як склеїти два рядка? Що потрібно врахувати при цьому?

24. Який заголовковий файл потрібно підключити для перетворення символьних рядків у числа?

25. Опишіть структуру заголовка функції та правила оголошення формальних параметрів.

26. Що таке прототип функції? Де у програмі може бути розміщений прототип функції?

27. Яке значення може повертати функція?

28. Як мають бути узгоджені між собою формальні і фактичні параметри?

29. Скільки операторів return може бути в описі функції?

30. Чи завжди виклик функції приводить до передачі управління за адресою цієї функції?

31. Які існують варіанти оголошення змінної структурного типу?

32. У яких випадках застосовують оператор точка, а у яких – стрілочка?

33. Які є обмеження на типи полів структури?

34. Як ініціалізують поля структури?

35. Як визначити розмір пам’яті, яку займає змінна структурного типу?

36. Як можна звернутися до поля структури, яка є полем іншої?

37. Як звернутися до поля елемента структури у масиві структур?

38. Як звернутися до елемента масиву, який є полем структури?

ПЕРЕЛІК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Грищук Ю. С. Мікроконтролери: архітектура, програмування та застосування в електромеханіці : навч. посіб. Харків : НТУ «ХПІ», 2019. 384 с.
2. Зиков І. С., Межерицький А. О. Подорожняк І. П. Хавіна С. Г. Програмування мікропроцесорів у захищеному режимі : навч-метод. посібник. Харків : ДІСА ПЛЮС, 2018. 264 с.
3. Поджаренко В. О., Кучерук В. Ю., Севастьянов В. М. Основи мікропроцесорної техніки : навч. посіб. Вінниця : ВНТУ, 2006. 226 с.
4. Цирульник С. М., Азаров О. Д., Крупельницький Л. В., Трояновська Т. І. Програмування мікроконтролерів AVR : навч. посіб. Вінниця : ВНТУ, 2018. 111 с.
5. Шаховська Н. Б., Голощук Р. О. Алгоритми і структури даних : навч. посіб. / за заг. ред. В. В. Пасічника. Львів : Магнолія, 2011. 215 с.
6. Шпак З. Я. Програмування мовою С : навч. посіб. Друге вид., допов. Львів : Видавництво Львівської політехніки, 2017. 436 с.
7. Кичак В. М., Макогон В. І., Васильківський М. В. Інформаційна система для оцінювання рівня дрібної моторики та стресостійкості операторів дистанційно-керованих пристроїв. *Вісник Хмельницького національного університету*. 2020. № 2. С. 72-77/
8. Городецька О. С., Гикавий В. А., Онищук О. В. Компютерні мережі. Вінниця : ВНТУ, 2017. 129 с.
9. Internet of Things for Industry and Human Application : In Volumes 1-3. Vol. 1. Fundamentals and Technologies / V. S. Kharchenko (ed.), Ministry of Education and Science of Ukraine, National Aerospace University KhAI, 2019. 605 p.
10. Цирульник С. М., Азаров О. Д., Крупельницький Л. В., Трояновська Т. І. Мікропроцесорна техніка : навч. посіб. Вінниця : ВНТУ, 2017. 123 с.
11. Основи програмування на мовах Сі та Сі++ для початківців. URL: http://cppstudio.com/uk/ (дата звернення: 12.04.2026).
12. Основи Інтернету речей. URL: http://edu.asu.in.ua/course/view.php?id=4#section-1 (дата звернення: 19.04.2026).
13. Интерфейс I2C и Arduino. URL: https://soltau.ru/index.php/arduino/item/371-interfejs-i2c-i-arduino (дата звернення: 19.04.2026).

Електронне навчальне видання

Віталій Іванович Макогон
Дмитро Валерійович Михалевський

Методичні вказівки до виконання самостійної роботи з дисципліни «Транслятори та покоління мов програмування телекомунікаційних систем» зі спеціальності «Електроніка, електронні комунікації, приладобудування та радіотехніка» (освітня програма «Програмне забезпечення телекомунікаційних систем»)

Рукопис оформив В. Макогон

Редактор Н. Кравчук

Оригінал-макет виготовлено РВВ ВНТУ

Підписано до видання 10.08.2026 р.
Гарнітура TimesNewRoman.
Зам. № Р2026-113

Видавець та виготовлювач
Вінницький національний технічний університет,
Редакційно-видавничий відділ.
ВНТУ, ГНК, к. 114.
Хмельницьке шосе, 95,
м. Вінниця, 21021.
press.vntu.edu.ua;
Email: rvv.vntu@gmail.com
Свідоцтво суб'єкта видавничої справи
серія ДК No 3516 від 01.07.2009 р.