Міністерство освіти і науки України
Луцький національний технічний університет

ОСНОВИ ПРОГРАМУВАННЯ МІКРОПРОЦЕСОРІВ ТА ТЕЛЕКОМУНІКАЦІЙНИХ ЗАСОБІВ

Методичні вказівки до виконання практичних занять
для здобувачів першого (бакалаврського) рівня вищої освіти
освітніх програм « Електроніка», «Автомобільна електроніка» та
«Комп'ютеризовані телекомунікаційні мережі»
галузі знань 17 Електроніка, автоматизація та електронні комунікації
спеціальностей 171 Електроніка , та 172 Електронні комунікації та радіотехніка
денної та заочної форм навчання

Луцьк 2025

УДК 004.43(07)

О - 73

Електронна копія друкованого видання передана для внесення в репозитарій ЛНТУ

Директор бібліотеки _____ Наталія ПОЛІЩУК

Рекомендовано до видання вченою радою факультету комп'ютерних та інформаційних технологій ЛНТУ, протокол № __ від « ___» _____2025 року.

Голова вченої ради ФКІТ _____ Інна КОНДІУС

Розглянуто і схвалено на засіданні кафедри електроніки та телекомунікацій ЛНТУ, протокол № ___ від «___» _____2025 року.

Завідувач кафедри ЕіТК	_____	Валентин ЗАБЛОЦЬКИЙ к.т.н., доц. кафедри електроніки та телекомунікацій ЛНТУ
Укладачі:	_____	Микола ХВИЩУН к.ф.-м.н., доц. кафедри електроніки та телекомунікацій ЛНТУ
	_____	Микола ЄВСЮК к.т.н., доц. кафедри електроніки та телекомунікацій ЛНТУ
Рецензент:	_____	Наталія ЛІЩИНА к.т.н., доц., завідувач кафедри інженерії програмного забезпечення ЛНТУ
Відповідальний за випуск:	_____	Валентин ЗАБЛОЦЬКИЙ к.т.н., доц., завідувач кафедри електроніки та телекомунікацій ЛНТУ

О-73 Основи програмування мікропроцесорів та телекомунікаційних засобів. Методичні вказівки до виконання практичних занять для здобувачів першого (бакалаврського) рівня вищої освіти освітніх програм «Електроніка», «Автомобільна електроніка» та «Комп'ютеризовані телекомунікаційні мережі» галузі знань 17 Електроніка, автоматизація та електронні комунікації, спеціальностей 171 Електроніка, та 172 Електронні комунікації та радіотехніка, всіх форм навчання, уклад. М. В. Хвищун, М.М.Євсюк. Луцьк: ЛНТУ, 2025. 62 с.

Видання містить виклад практичних завдань з дисципліни «Основи програмування мікропроцесорів та телекомунікаційних засобів», яке призначене для здобувачів першого (бакалаврського) рівня вищої освіти освітніх програм «Електроніка», «Автомобільна електроніка» та «Комп'ютеризовані телекомунікаційні мережі».

ЗМІСТ

ВСТУП……... 4

Практичне заняття 1. Основи програмування на С: змінні, оператори та введення/виведення даних ……..… 5

Практичне заняття 2. Управління потоком виконання в С: умовні оператори та цикли ……. 9

Практичне заняття 3. Функції та способи передачі параметрів у мові програмування С ……..……..... 13

Практичне заняття 4. Масиви та вказівники у С……. 18

Практичне заняття 5. Динамічне виділення пам'яті та робота зі структурами у С ……...…… 23

Практичне заняття 6. Робота з файлами в С та Python……. ……... 28

Практичне заняття 7. Робота зі структурами даних та об'єктно-орієнтоване програмування (ООП) у Python …… 30

Практичне заняття 8. Умовні оператори та оператори циклів у Python……. 34

Практичне заняття 9. Таймери та переривання в мікроконтролерах …… 37

Практичне заняття 10. Робота з бітовими операціями в С …….……..... 41

Практичне заняття 11. Організація обміну даними через UART, I2C, SPI …… 43

Практичне заняття 12. Безпровідні протоколи передачі даних (Bluetooth, Wi-Fi)……... 46

Практичне заняття 13. Налаштування середовища розробки та перша програма для мікроконтролера AVR…… ……. 49

Практичне заняття 14. Таймери та переривання в мікроконтролерах…… 52

Практичне заняття 15. Фінальний проєкт. Розробка комплексної системи..... 57

ВСТУП

Сучасний розвиток інформаційних технологій та засобів зв'язку неможливий без використання вбудованих систем, до складу яких входять мікропроцесори, мікроконтролери та телекомунікаційні пристрої. В умовах цифрової трансформації суспільства, стрімкого розвитку Інтернету речей (IoT), автоматизованих систем керування та бездротових технологій, програмування мікропроцесорних пристроїв набуває особливої актуальності.

Метою курсу «Основи програмування мікропроцесорів та телекомунікаційних засобів» є формування у студентів практичних навичок програмування апаратних засобів цифрової техніки, розуміння принципів побудови та функціонування мікропроцесорних систем, а також оволодіння методами передавання, обробки та обміну даними між пристроями.

Методичні вказівки до практичних занять містять чітко структуровані теми, які охоплюють основні аспекти програмування мікроконтролерів (на прикладі AVR, STM32, ESP тощо), принципи роботи периферійних пристроїв, організацію вводу/виводу, роботу з портами, перериваннями, таймерами, а також основи реалізації телекомунікаційних інтерфейсів (UART, SPI, I²C, Bluetooth, Wi-Fi).

Практичні роботи покликані закріпити теоретичні знання студентів, розвинути вміння створювати програмне забезпечення для прикладних задач з використанням мов програмування C та Python, асемблер, а також ознайомити з інструментами розробки, такими як Atmel Studio, Keil, Arduino IDE, Thonny IDE тощо. Особливу увагу приділено практичному застосуванню мікропроцесорів у телекомунікаційних системах, інтеграції сенсорних модулів, аналізу протоколів передачі даних, а також створенню проєктів з реального життя, що дозволяє формувати у студентів інженерне мислення та вміння працювати у команді.

Використання даних методичних рекомендацій сприятиме поглибленому засвоєнню матеріалу, розвитку професійних компетентностей, необхідних для роботи у галузі електроніки, автоматизації та телекомунікацій.

Практичне заняття № 1

Тема: основи програмування на С - змінні, оператори та введення/виведення даних

Мета практичної роботи: навчитися оголошувати змінні, працювати з основними операторами та організовувати введення/виведення даних у мові програмування С.

Теоретичні відомості

Основи змінних у С. Змінна – це іменована область пам’яті, яка використовується для зберігання даних. Перед використанням змінної її необхідно оголосити, вказавши тип даних.

Типи змінних у С. Тип змінної визначає обсяг пам'яті, яку вона займає, а також допустимі операції над нею. Основні типи змінних у С:

Цілочисельні типи:

– int (ціле число, зазвичай 4 байти) – використовується для зберігання цілих чисел;

– short int (коротке ціле, зазвичай 2 байти) – займає менше пам’яті, але має обмежений діапазон значень;

– long int (довге ціле, зазвичай 4 або 8 байтів) – має збільшений діапазон значень;

– unsigned int – беззнаковий цілий тип, може приймати лише додатні значення.

Числа з плаваючою комою:

– float (4 байти) – використовується для представлення чисел з плаваючою комою одинарної точності;

– double (8 байтів) – числа з подвійною точністю, точніші, ніж float;

– long double (10-16 байтів) – використовується у випадках, коли потрібна ще більша точність.

Символьний тип:

– char (1 байт) – використовується для зберігання символів ASCII.

Логічний тип:

– bool (1 байт) – приймає значення true або false (доступний у стандарті C99 і вище).

Класифікація змінних за областю видимості:

– локальні змінні – оголошуються всередині функції або блоку {} та доступні тільки в межах цієї функції;

– глобальні змінні – оголошуються поза будь-якою функцією і доступні у всій програмі;

– статичні змінні (static) – зберігають своє значення між викликами функції;

– автоматичні змінні (auto) – тип за замовчуванням для локальних змінних;

– реєстрові змінні (register) – можуть зберігатися в регістрах процесора для швидкого доступу.

Де використовуються змінні:

– цілочисельні (int, short, long) використовуються для підрахунку елементів, індексації масивів;

– типи з плаваючою комою (float, double) застосовуються в обчисленнях, де важлива точність, наприклад у наукових і фінансових розрахунках;

– символьні (char) широко використовуються для роботи з рядками і текстовими даними.

– логічні (bool) використовуються для умовних конструкцій (if, while).

Завдання до практичної роботи

Написати програму, яка:

1) зчитує два цілих числа від користувача;
2) обчислює їхню суму, різницю, добуток і частку;
3) виводить результати на екран.

Приклад коду:

```
#include <stdio.h>
int main() {
   int num1, num2;
   printf("Введіть перше число: ");
   scanf("%d", &num1);
   printf("Введіть друге число: ");
```

```
    scanf("%d", &num2);
    printf("Сума: %d\n", num1 + num2);
    printf("Різниця: %d\n", num1 - num2);
    printf("Добуток: %d\n", num1 * num2);
    if (num2 != 0) {
        printf("Частка: %.2f\n", (float)num1 / num2);
    } else {
        printf("Ділення на нуль неможливе!\n");
    }
    return 0;
}
```

Результати тестування наведені в таблиці 1.

Таблиця 1 – Результати тестування програми

Вхідні дані	Сума	Різниця	Добуток	Частка
10, 5	15	5	50	2.00
7, 2	9	5	14	3.50
8, 0	8	8	0	Невизначено

Висновки оформляти у вигляді:

1) що було розглянуто основи роботи з змінними, операторами та функціями введення/виведення;
2) що реалізовано програму для виконання основних арифметичних операцій;
3) вдосконалено навички роботи з printf() і scanf().

Список використаних джерел до практичного заняття №1

1. Brian W. Kernighan, Dennis M. Ritchie. The C Programming Language. 2nd Edition, 2022. 300 p.
2. Stephen G. Kochan. Programming in C. 4th Edition, 2022. 560 p.
3. Deitel P., Deitel H. C How to Program. 9th Edition, 2023. 1000 p.
4. Костенко Д. Програмування на С для початківців. 2-ге видання, 2023. 450с.

5. Офіційна документація C. URL: https://en.cppreference.com/w/c (дата звернення 15.05.2025р.).

Практичне заняття 2

Тема: керування потоком виконання в С: умовні оператори та цикли

Мета практичної роботи:

1) ознайомитися з умовними операторами (if, else, switch) та циклами (for, while, do-while) у мові С;
2) вивчити принципи логічного розгалуження програми;
3) навчитися застосовувати цикли для повторюваних обчислень;
4) виконати практичні завдання для закріплення матеріалу.

Теоретичні відомості

Умовні оператори в С. Умовні оператори дозволяють виконувати певний блок коду залежно від виконання умови.

Приклад запису операторів if-else.

```
int number = 10;
if (number % 2 == 0) {
   printf("Число парне\n");
} else {
   printf("Число непарне\n");
}
```

Якщо умова number % 2 == 0 істинна, виконується код у фігурних дужках {} після if, інакше виконується код після else.

Приклад запису операторів switch-case.

```
int option = 2;
switch (option) {
   case 1:
      printf("Обрано варіант 1\n");
      break;
   case 2:
      printf("Обрано варіант 2\n");
      break;
   default:
      printf("Невідомий варіант\n");
```

```
}
```

Оператори switch-case корисні, коли потрібно перевірити багато варіантів значення змінної.

Розглянемо приклади операторів циклів у С. Цикли дозволяють виконувати певний блок коду кілька разів.

1. Цикл for використовується, коли кількість повторень відома заздалегідь.

```
for (int i = 1; i <= 5; i++) {
   printf("Ітерація %d\n", i);
}
```

2. Цикл while виконується, поки умова залишається істинною.

```
int i = 1;
while (i <= 5) {
   printf("Ітерація %d\n", i);
   i++;
}
```

3. Цикл do-while виконується хоча б один раз, навіть якщо умова хибна.

```
int i = 1;
while (i <= 5) {
   printf("Ітерація %d\n", i);
   i++;
}
```

Порядок виконання роботи

Завдання 1: Перевірка парності числа:

1) створити програму, яка зчитує число з клавіатури;
2) використати оператор if-else для перевірки парності;
3) вивести відповідне повідомлення.

Приклад коду:

```
#include <stdio.h>
int main() {
   int number;
   printf("Введіть число: ");
```

```
    scanf("%d", &number);
    if (number % 2 == 0) {
        printf("Число парне\n");
    } else {
        printf("Число непарне\n");
    }
    return 0;
}
```

Завдання 2: Знаходження всіх дільників числа:

1) написати програму, яка зчитує число з клавіатури;
2) використати цикл for для перевірки дільників числа;
3) вивести всі дільники числа.

Приклад коду:

```
#include <stdio.h>
int main() {
    int num;
    printf("Введіть число: ");
    scanf("%d", &num);
    printf("Дільники числа %d: ", num);
    for (int i = 1; i <= num; i++) {
        if (num % i == 0) {
            printf("%d ", i);
        }
    }
   printf("\n");
    return 0;
}
```

Звіт про виконану роботу та порядок оформлення

Звіт повинен містити:

1) тему і мету роботи;
2) теоретичні відомості: опис умовних операторів та циклів;

3) код програм та їх пояснення;
4) результати виконання програм (скріншоти або текстові приклади роботи);
5) висновки про виконану роботу.

Список використаних джерел до практичного заняття №2

1. Бабич А. В., Костін Д. С. Програмування мовою С: керування потоком виконання. Київ: Видавничий дім «Освіта», 2022. 224 с.

2. Сисоєв М. О. Основи структурного програмування на С. Харків: Наука і техніка, 2023, 312 с.

3. Kernighan B. W., Ritchie D. M. The C Programming Language. 2nd Edition. Prentice Hall, 2021, 272 p.

4. Perry G. C Programming Absolute Beginner's Guide. 4th Edition. Que Publishing, 2022,384 p.

5. Kochan S. Programming in C. 4th Edition. Addison-Wesley, 2023, 552 p.

Практичне заняття 3

Тема: функції та способи передачі параметрів у мові програмування С

Мета практичної роботи: ознайомитися з поняттям функцій у С, їхньою структурою, видами та способами передачі параметрів; навчитися створювати та використовувати функції для розв'язання практичних задач.

Теоретичні відомості

Функції є фундаментальною складовою мови програмування С, що дозволяє структурувати код, підвищувати його читабельність та повторно використовувати окремі блоки. Функція – це самостійний блок коду, який виконує певну задачу та може бути викликаний з іншої частини програми.

Структура функції. Функція в С складається з заголовка та тіла. Заголовок визначає тип повертаємого значення, ім'я функції та список параметрів. Тіло містить набір операторів, що виконуються при виклику функції.

```
return_type function_name(parameter_list) {
   // Тіло функції
}
```

, де :

- return_type – тип даних, який функція повертає (наприклад, int, void);
- function_name – ім'я функції;
- parameter_list – список параметрів (може бути порожнім).

Розглянемо основні види функцій:

1) бібліотечні функції: вбудовані в стандартну бібліотеку С (наприклад, printf, scanf);
2) користувацькі функції: створені програмістом для виконання специфічних задач.

Передача параметрів. У С існують основні способи передачі параметрів у функцію. За значенням функція отримує копію значення аргументу. Зміни цього значення всередині функції не впливають на оригінальну змінну.

```
void increment(int num) {
   num = num + 1;
}
```

При виклику increment(&a); значення а буде збільшено на 1.

Порядок виконання роботи

1. Створення функції для обчислення факторіала числа:

1) Оголосіть функцію unsigned long long factorial(int n), яка приймає ціле число n та повертає його факторіал.

2) Реалізуйте функцію, використовуючи ітеративний підхід:

```
unsigned long long factorial(int n) {
    unsigned long long result = 1;
    for (int i = 1; i <= n; ++i) {
        result *= i;
    }
    return result;
}
```

У головній функції main() запросіть у користувача введення цілого числа та виведіть результат обчислення факторіала, викликавши функцію factorial.

2. Реалізація функції, що повертає найбільше з трьох чисел:

1) оголосіть функцію int max_of_three(int a, int b, int c), яка приймає три цілі числа та повертає найбільше з них;

2) реалізуйте функцію, використовуючи умовні оператори:

```
int max_of_three(int a, int b, int c) {
    int max = a;
    if (b > max) {
        max = b;
    }
    if (c > max) {
        max = c;
    }
    return max;
}
```

У функції main() запросіть у користувача введення трьох цілих чисел та виведіть найбільше з них, викликавши функцію max_of_three.

Варіанти завданнь для практичної роботи

Для кожного студента визначається індивідуальне завдання згідно з варіантом, поданим у таблиці 2. Номер варіанта може відповідати номеру студента у списку групи.

Таблиця 2 – Варіанти завдань до практичного заняття

Варіант	Завдання
1	Написати функцію для обчислення факторіала числа (ітеративний метод).
2	Написати функцію для обчислення факторіала числа (рекурсивний метод).
3	Реалізувати функцію, яка повертає найбільше з трьох чисел.
4	Реалізувати функцію, яка повертає найменше з трьох чисел.
5	Написати функцію, яка перевіряє, чи є число простим.
6	Написати функцію, яка обчислює суму цифр цілого числа.
7	Реалізувати функцію, яка знаходить найбільший спільний дільник (НСД) двох чисел.
8	Реалізувати функцію, яка знаходить найменше спільне кратне (НСК) двох чисел.
9	Написати функцію, яка перевертає цифри числа (наприклад, 123 → 321).
10	Написати функцію, яка обчислює число Фібоначчі за номером (ітеративний метод).
11	Написати функцію, яка обчислює число Фібоначчі за номером (рекурсивний метод).
12	Реалізувати функцію, яка перевіряє, чи є число паліндромом.
13	Написати функцію, яка обчислює середнє арифметичне масиву чисел.
14	Реалізувати функцію, яка знаходить максимальний елемент у масиві.
15	Реалізувати функцію, яка знаходить мінімальний елемент у масиві.
16	Написати функцію, яка підносить число до степеня (рекурсивний метод).
17	Реалізувати функцію, яка рахує кількість входжень певного символу в рядку.
18	Написати функцію, яка конвертує температуру з Цельсія у Фаренгейти.
19	Написати функцію, яка конвертує температуру з Фаренгейтів у Цельсій.
20	Реалізувати функцію, яка перевіряє, чи є рік високосним.

Порядок оформлення практичної роботи

Для коректного оформлення практичної роботи студенти повинні дотримуватись наступної структури звіту:

1. Титульна сторінка:

– назва закладу освіти;
– факультет та кафедра;
– дисципліна;
– тема практичної роботи;
– прізвище, ім'я, по батькові студента;
– група;
– прізвище викладача;
– дата виконання.

3. Мета роботи - чітке формулювання основної мети практичної роботи (наприклад, «Ознайомитися з функціями та способами передачі параметрів у С, навчитися їх використовувати у програмуванні»).
4. Теоретичні відомості:

– короткий виклад теоретичного матеріалу, який розглядається у межах завдання;
– пояснення основних понять (функція, аргументи, повернення значення тощо);
– приклади використання функцій у С.

Постановка завдання на практичне заняття:

1) опис індивідуального завдання відповідно до варіанту;
2) формулювання вхідних та вихідних даних;
3) очікуваний результат виконання програми.

Хід виконання роботи:

1) опис розробки коду (покрокове пояснення реалізації функції);
2) представлення вихідного коду програми з поясненнями;
3) код має бути структурованим, з коментарями.

Приклад оформлення коду:

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

```
// Функція обчислення факторіала (ітеративний підхід)
unsigned long long factorial(int n) {
    unsigned long long result = 1;
    for (int i = 1; i <= n; ++i) {
        result *= i;
    }
    return result;
}
int main() {
    int number;
    printf("Введіть число: ");
    scanf("%d", &number);
    printf("Факторіал %d = %llu\n", number, factorial(number));
    return 0;
}
```

У разі використання допоміжних функцій необхідно вказати їхню роль, а також вказати можливі помилки при введенні некоректних даних.

Результати роботи повинні містити:

1) вивід програми після тестування;
2) декілька тестових запусків з різними вхідними даними;
3) аналіз отриманих результатів.

Висновки повинні містити:

1) аналіз виконаної роботи;
2) що нового дізнався студент під час виконання практичного завдання;
3) висновки щодо реалізації функцій та передачі параметрів у С.

Список використаних джерел до практичного заняття №3

1. Байда Є.І., Кропачек О.Ю. Рішення задач електромеханіки в прикладних пакетах програм. Харків: НТУ «ХПІ», 2017. 180 с.
2. Jens Gustedt, Modern C. Manning Publications, 2022. 475 p.
3. Richard Reese. Understanding and Using C Pointers. O’Reilly Media, 2023. 250 p.

Практичне заняття 4

Тема: масиви та вказівники у С

Мета практичної роботи: дослідити роботу з масивами, вказівниками та передачею даних через них.

Теоретичні відомості

Масиви та вказівники є фундаментальними концепціями в мові програмування С, які тісно пов'язані між собою та забезпечують ефективну роботу з даними.

Масиви – це структура даних, яка дозволяє зберігати набір елементів одного типу під спільним ім'ям. Елементи масиву розташовані в пам'яті послідовно, що забезпечує швидкий доступ до кожного елемента за його індексом. Оголошення масиву в С має такий синтаксис:

тип_даних ім'я_масиву[розмір];

Наприклад, оголошення масиву з 10 цілих чисел:

```
int numbers[10];
```

В цьому прикладі numbers – це масив із 10 елементів типу int. Доступ до елементів масиву здійснюється за допомогою індексів, починаючи з нуля: numbers[0], numbers[1] і т.д.

Вказівники – це змінні, які містять адреси інших змінних. Вони дозволяють працювати з пам'яттю безпосередньо, що є потужним інструментом для оптимізації програм та управління ресурсами. Оголошення вказівника має такий вигляд:

тип_даних *ім'я_вказівника;

Наприклад, оголошення вказівника на ціле число:

```
int *ptr;
```

Тут ptr – це вказівник, який може зберігати адресу змінної типу int. Щоб присвоїти вказівнику адресу змінної, використовують оператор & (оператор адреси):

```
int value = 5;
ptr = &value;
```

Тепер ptr містить адресу змінної value. Для доступу до значення, на яке вказує вказівник, використовують оператор розіменування *:

```
int value = 5;
```

```
int *ptr = &value;
printf("%d", *ptr); // Виведе 5
```

Взаємозв'язок між масивами та вказівниками проявляється в тому, що ім'я масиву фактично є вказівником на його перший елемент. Тобто, якщо numbers – це масив, то numbers еквівалентно &numbers[0]. Це дозволяє використовувати вказівники для ітерації по масиву:

```
int numbers[10];
int *ptr = numbers; // Те ж саме, що і int *ptr = &numbers[0];
```

Тепер, використовуючи вказівник ptr, можна отримати доступ до елементів масиву:

```
*(ptr + 1) = 10; // Присвоює значення 10 другому елементу масиву
```

Важливо розуміти, що хоча масиви і вказівники мають тісний зв'язок, вони не є повністю взаємозамінними. Наприклад, вказівнику можна присвоїти нову адресу, тоді як ім'я масиву завжди вказує на початок відповідного блоку пам'яті і не може бути змінене.

Передача масивів у функції також здійснюється через вказівники. Коли масив передається у функцію, фактично передається вказівник на його перший елемент, що дозволяє функції модифікувати елементи масиву без створення його копії.

Порядок виконання роботи:

1. Створити програму для обчислення середнього значення масиву з 10 чисел:

1) оголосіть масив із 10 елементів типу int;

2) заповніть масив випадковими числами або введіть їх з клавіатури;

3) використовуючи цикл, обчисліть суму всіх елементів масиву;

4) розділіть отриману суму на кількість елементів масиву, щоб отримати середнє значення.

5) виведіть результат на екран.

Приклад коду:

```
#include <stdio.h>
int main() {
   int numbers[10];
   int sum = 0;
   double average;
```

```
    // Заповнення масиву числами
    for(int i = 0; i < 10; i++) {
        printf("Введіть число %d: ", i + 1);
        scanf("%d", &numbers[i]);
    }
        Обчислення суми елементів масиву:
    for(int i = 0; i < 10; i++) {
        sum += numbers[i];
    }
        Обчислення середнього значення:
    average = (double)sum / 10;
        Виведення результату:
    printf("Середнє значення: %.2f\n", average);
    return 0;
}
```

Реалізація сортування масиву методом вибору:

1. Оголосіть масив із 10 чисел.
2. Реалізуйте алгоритм сортування методом вибору:

– знайдіть мінімальний елемент у невідсортованій частині масиву;
– поміняйте його місцями з першим елементом цієї частини;
– повторюйте процедуру для всіх елементів;
– виведіть відсортований масив.

Приклад коду:

```
#include <stdio.h>
void selectionSort(int arr[], int size) {
    int i, j, minIndex, temp;
    for (i = 0; i < size - 1; i++) {
        minIndex = i;
        for (j = i + 1; j < size; j++) {
            if (arr[j] < arr[minIndex]) {
                minIndex = j;
```

```
            }
        }
        // Обмін значень
        temp = arr[minIndex];
        arr[minIndex] = arr[i];
        arr[i] = temp;
    }
}
int main() {
    int numbers[10] = {29, 10, 14, 37, 13, 5, 22, 41, 8, 11};
    printf("Початковий масив:\n");
    for (int i = 0; i < 10; i++) {
        printf("%d ", numbers[i]);
    }
    printf("\n");
    selectionSort(numbers, 10);
    printf("Відсортований масив:\n");
    for (int i = 0; i < 10; i++) {
        printf("%d ", numbers[i]);
    }
    printf("\n");
    return 0;
}
```

Приклад оформлення роботи

Практична робота №4

Тема: Масиви та вказівники у С

Мета роботи: Дослідити роботу з масивами, вказівниками та передачею даних через них.

Теоретичні відомості

1. Масиви – це структури даних, що дозволяють зберігати набір елементів одного типу.

2. Вказівники – змінні, які містять адресу інших змінних.

3. Взаємозв’язок між масивами та вказівниками дозволяє ефективно передавати дані у функції.

Порядок виконання роботи

1. Реалізовано програму для обчислення середнього значення елементів масиву.

2. Реалізовано алгоритм сортування масиву методом вибору.

Програми та їх результати

1. Код для знаходження середнього значення масиву.

2. Код для сортування методом вибору.

3. Виведення результатів програм.

Висновки: (Що було вивчено, які навички набуті, які висновки зроблені).

Список використаних джерел до практичного заняття №4

1. Kernighan, B.W., Ritchie, D.M. The C Programming Language, 2nd ed., Prentice Hall, 2021. 274 p.

2. Kochan, S.G. Programming in C, 4th ed., Addison-Wesley, 2022. 560 p.

3. Gookin, D. Beginning Programming with C For Dummies, 3rd ed., Wiley, 2023. 416 p.

4. Глєбов І. І. Сучасне програмування на С, Київ: Видавничий дім «Києво-Могилянська академія», 2023. 312 с.

Практичне заняття 5

Тема: динамічне виділення пам’яті та робота зі структурами у С

Мета практичної роботи: навчитися працювати з динамічним виділенням пам’яті (malloc, free) та структурами у мові програмування С. Закріпити навички створення структурованих даних та керування пам’яттю у динамічному середовищі.

Теоретичні відомості

У мові С існують дві основні категорії пам’яті: статична та динамічна. Статична пам’ять виділяється під час компіляції і залишається доступною протягом усього часу виконання програми. Динамічна пам’ять, навпаки, виділяється та звільняється під час виконання програми за допомогою спеціальних функцій.

Для роботи з динамічною пам’яттю використовуються наступні функції:

- malloc(size_t size): виділяє пам’ять зазначеного розміру, повертає вказівник на неї або NULL у разі невдачі;
- free(void *ptr): звільняє раніше виділену пам’ять.

Приклад роботи з malloc та free:

```
#include <stdio.h>
#include <stdlib.h>
int main() {
   int *ptr = (int*) malloc(sizeof(int));
   if (ptr == NULL) {
      printf("Помилка виділення пам’яті\n");
      return 1;
   }
   *ptr = 10;
   printf("Значення: %d\n", *ptr);
   free(ptr);
   return 0;
}
```

Структури у С дозволяють об’єднувати дані різних типів в єдиний логічний об’єкт.

Оголошення структури:

```
struct Student {
    char name[50];
    int age;
    float average_grade;
};
```

Приклад оформлення програми:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct Student {
    char name[50];
    int age;
    float average_grade;
};
void addStudent(struct Student **students, int *count) {
    *students = realloc(*students, (*count + 1) * sizeof(struct Student));
    if (*students == NULL) {
        printf("Помилка виділення пам’яті\n");
        return;
    }
    printf("Введіть ім'я студента: ");
    scanf("%s", (*students)[*count].name);
    printf("Вік: ");
    scanf("%d", &(*students)[*count].age);
    printf("Середній бал: ");
    scanf("%f", &(*students)[*count].average_grade);
    (*count)++;
}
void deleteStudent(struct Student **students, int *count) {
    if (*count == 0) {
        printf("Немає записів для видалення\n");
```

```
        return;
    }
    (*count)--;
    *students = realloc(*students, (*count) * sizeof(struct Student));
}
void displayStudents(struct Student *students, int count) {
    printf("\nСписок студентів:\n");
    for (int i = 0; i < count; i++) {
        printf("Ім'я: %s, Вік: %d, Середній бал: %.2f\n", students[i].name, students[i].age,
students[i].average_grade);
    }
}
int main() {
    struct Student *students = NULL;
    int count = 0;
    int choice;
    do {
        printf("\n1. Додати студента\n2. Видалити останнього студента\n3. Вивести
список\n4. Вийти\nВиберіть дію: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1: addStudent(&students, &count); break;
            case 2: deleteStudent(&students, &count); break;
            case 3: displayStudents(students, count); break;
            case 4: break;
            default: printf("Невірний вибір!\n");
        }
    } while (choice != 4);
    free(students);
    return 0;
}
```

Порядок виконання роботи

1. Ознайомитися з теоретичними відомостями щодо динамічного виділення пам’яті та структур у С.
2. Виконати компіляцію та запуск демонстраційного прикладу роботи malloc та free.
3. Реалізувати програму для роботи з базою студентів згідно з поданим прикладом.
4. Виконати тестування програми: додати студентів, видалити деякі записи, перевірити коректність виводу.
5. Оформити звіт відповідно до вимог.

Порядок оформлення звіту

Звіт повинен містити:

1) тему, мету роботи;
2) теоретичні відомості;
3) код розробленої програми;
4) скриншоти роботи програми (додавання, видалення, вивід списку студентів);
5) висновки щодо виконаної роботи;
6) список використаних джерел.

Список використаних джерел до практичного заняття №5

1. Керніган Б., Рітчі Д. Мова програмування С. Київ: Видавництво XYZ, 2022. 320 с.
2. Прата С. С Primer Plus. Нью - Йорк: Pearson, 2023. 1024 с.
3. Кунцевич В. Програмування мовою С. Львів: Видавництво ABC, 2024. 400 с.
4. Керниган Б., Річі Д. Мова програмування С. Київ: Наукова думка, 2024. 350с.
5. Таненбаум А. Сучасні операційні системи. Львів: Видавництво Львівського університету, 2024. 500 с.
6. King K. C Programming: A Modern Approach. New York: Addison-Wesley, 2024. 750 p.
7. Stroustrup B. The C++ Programming Language. Boston: Pearson Education, 2024. 1200 p.

Практичне заняття 6

Тема: робота з файлами в C та Python

Мета практичної роботи: ознайомитися з основними операціями роботи з файлами в мовах програмування C та Python. Навчитися працювати з текстовими та бінарними файлами, виконувати обробку помилок та використовувати механізми серіалізації даних.

Теоретичні відомості

Розглянемо файлові операції у C. Основні функції для роботи з файлами в C містяться у бібліотеці stdio.h:

- fopen("filename", "mode") – відкриття файлу у вказаному режимі (r, w, a, rb, wb, тощо);
- fclose(file) – закриття файлу;
- fgets(buffer, size, file) – читання рядка з файлу;
- fputs(string, file) – запис рядка у файл;
- fread(buffer, size, count, file) – читання з бінарного файлу;
- fwrite(buffer, size, count, file) – запис у бінарний файл.

Обробка помилок здійснюється за допомогою перевірки результатів виклику функцій (наприклад, if (file == NULL)).

Розглянемо файлові операції у Python. Основні операції роботи з файлами у Python здійснюються за допомогою функції open():

- open("filename", "mode") – відкриття файлу у вказаному режимі (r, w, a, rb, wb, тощо);
- file.read(size) – зчитування вмісту файлу;
- file.write(string) – запис у файл;
- file.close() – закриття файлу.

Рекомендується використовувати менеджер контексту (with open(...) as file:), що забезпечує автоматичне закриття файлу.

Розглянемо серіалізацію даних у Python. Серіалізація дозволяє зберігати складні структури даних у файлах у зручному для передачі форматі. Основні бібліотеки:

- pickle – для бінарної серіалізації;

– json – для роботи з текстовими JSON-файлами.

Приклад роботи з pickle:

```
import pickle
# Збереження об'єкта у файл
with open("data.pkl", "wb") as file:
    pickle.dump({"key": "value"}, file)
# Завантаження об'єкта
with open("data.pkl", "rb") as file:
    data = pickle.load(file)
    print(data)
```

Приклад роботи з json:

```
import json
# Запис у JSON-файл
with open("data.json", "w") as file:
    json.dump({"key": "value"}, file)
# Зчитування з JSON-файлу
with open("data.json", "r") as file:
    data = json.load(file)
    print(data)
```

Звіт про виконану роботу та порядок оформлення

1. Навести код програм, реалізованих під час практичної роботи.
2. Описати використані методи та функції роботи з файлами.
3. Навести приклади обробки помилок у C та Python.
4. Продемонструвати результат роботи програм та зробити висновки щодо можливостей роботи з файлами у двох мовах.

Список використаних джерел до практичного заняття №6

1. Документація бібліотеки json. URL: https://docs.python.org/3/library/json.html (дата звернення 15.05.2025).
2. Документація бібліотеки pickle. URL: https://docs.python.org/3/library/pickle.html (дата звернення 15.05.2025).
3. Маттес Е. Python Crash Course. 3rd edition. ithub.ua, 2023. 552 с.
4. Офіційна документація Python. URL: https://docs.python.org/3/tutorial/inputoutput.html (дата звернення 15.05.2025).

Практичне заняття 7

Тема: робота зі структурами даних та об'єктно-орієнтоване програмування (ООП) у Python

Мета практичної роботи:

1. Ознайомитися з поняттями структур даних у мові С, зокрема зі структурами (struct) та об'єднаннями (union).
2. Розглянути основи об'єктно-орієнтованого програмування у Python, включаючи класи, об'єкти, методи та наслідування.
3. Порівняти реалізацію структур даних у С та Python.
4. Дослідити використання структур у мікроконтролерах для зберігання даних.

Теоретичні відомості

Розглянемо структури в С (struct, union). У мові програмування С структура (struct) дозволяє об'єднати різнотипні дані під одним іменем. Це корисно для групування пов'язаних даних, таких як характеристики об'єкта.

Наприклад:

```
struct Car {
   char brand[50];
   char model[50];
   int year;
};
```

Як бачимо, тут структура Car містить три поля: brand, model та year.

Об'єднання (union) схоже на структуру, але всі його члени зберігаються в одній області пам'яті, і в кожен момент часу може бути використано лише одне поле. Це корисно, коли потрібно зберігати різні типи даних в одному місці, але використовувати лише один з них одночасно.

```
union Data {
   int i;
   float f;
   char str[20];
};
```

В цьому прикладі об'єднання Data може зберігати або ціле число, або число з плаваючою комою, або рядок, але лише одне з них у конкретний момент часу.

ООП у Python: класи, об'єкти, методи, наслідування.

Об'єктно-орієнтоване програмування (ООП) є парадигмою програмування, яка базується на концепції «об'єктів» – сутностей, що об'єднують дані та поведінку. У Python ООП реалізується через класи та об'єкти.

Клас – це шаблон або схема для створення об'єктів. Він визначає атрибути та методи, які будуть властиві об'єктам цього класу.

Об'єкт – це конкретний екземпляр класу з власними значеннями атрибутів.

Методи – це функції, визначені всередині класу, які описують поведінку об'єктів.

Наслідування дозволяє створювати нові класи на основі існуючих, успадковуючи їх атрибути та методи.

Розглянемо приклад класу в Python:

```
class Car:
    def __init__(self, brand, model, year):
        self.brand = brand
        self.model = model
        self.year = year
    def info(self):
        return f"{self.brand} {self.model}, {self.year}"
```

тут ми визначаємо клас Car з конструктором __init__ та методом info.

Наведемо порівняння структур у С та класів у Python.

Хоча структури в С та класи в Python мають схожість у групуванні даних, між ними є суттєві відмінності:

1. Функціональність. Структури в С призначені лише для зберігання даних, тоді як класи в Python можуть містити як дані, так і методи для обробки цих даних.
2. Інкапсуляція. У Python класи підтримують механізм інкапсуляції, що дозволяє приховувати внутрішні деталі реалізації. У С структури не мають вбудованої підтримки інкапсуляції.

3. Наслідування. Python підтримує наслідування, що дозволяє створювати ієрархії класів. У C структури не підтримують наслідування.

Розглянемо використання структур у мікроконтролерах для зберігання даних. У програмуванні мікроконтролерів структури широко використовуються для організації та зберігання даних. Вони дозволяють ефективно групувати пов'язані дані, що спрощує доступ до них та покращує читабельність коду.

Наприклад, можна визначити структуру для зберігання стану різних сенсорів:

```
struct SensorData {
   int temperature;
   int humidity;
   int pressure;
};
```

Використання структур у мікроконтролерах також сприяє оптимізації використання пам'яті, що є критичним у вбудованих системах з обмеженими ресурсами.

У програмуванні мікроконтролерів структури широко використовуються для організації та зберігання даних. Вони дозволяють ефективно групувати пов'язані дані, що спрощує доступ до них та покращує читабельність коду. Наприклад, можна визначити структуру для зберігання стану різних сенсорів:

```
struct SensorData {
   int temperature;
   int humidity;
   int pressure;
};
```

Порядок виконання роботи

1. Ознайомитись зі структурами в C:

– створіть структуру Car з полями brand, model та year;

– ініціалізуйте змінну цієї структури та виведіть її значення.

2. Ознайомлення з об'єднаннями – створіть union Data і проаналізуйте поведінку пам'яті при використанні різних полів.

3. Реалізація класів у Python:

– створіть клас Car з методами __init__ та info;

– створіть декілька об'єктів цього класу та викличте метод info.

4. Порівняння структур у C та класів у Python – визначити спільні риси та відмінності між цими концепціями.

5. Застосування структур у мікроконтролерах – створіть структуру SensorData і реалізуйте її використання у коді для AVR або Arduino.

Порядок оформлення роботи

1. Титульна сторінка.
2. Мета роботи.
3. Теоретичні відомості.
4. Виконання практичних завдань з кодами та їх поясненнями.
5. Висновки.
6. Список використаних джерел.

Список використаних джерел до практичного заняття №7

1. Загородній В. І., Підгорний А. В. Програмування на С для вбудованих систем. – Київ: Ліра-К, 2022. 356 с.

2. Кравчук О. А. Об'єктно-орієнтоване програмування на Python: Навчальний посібник. – Київ: Видавництво НТУУ "КПІ", 2023. 280 с.

3. Семенов І. П. Основи мікроконтролерного програмування на С: Практичний посібник. – Львів: Видавництво ЛНУ, 2021, 312 с.

4. Сидоренко В. Ю. Вбудовані системи: Структури даних та алгоритми. – Харків: Факт, 2023. 295 с.

5. Kernighan B., Ritchie D. The C Programming Language. – Pearson, 2022. 272 p.

6. Lutz M. Learning Python. – 6th ed. – O'Reilly Media, 2023. 750 p.

7. Patterson D. A., Hennessy J. L. Computer Organization and Design RISC-V Edition. – Morgan Kaufmann, 2021. 760 p.

8.Gookin D. C Programming For Dummies. – 3rd ed. – Wiley, 2022. 468 p.

Практичне заняття 8

Тема: умовні оператори та оператори циклів у Python

Мета роботи: ознайомитися з умовними операторами та операторами циклів у мові програмування Python, навчитися застосовувати їх для розв'язання практичних завдань, розробити програми з використанням умовних конструкцій та циклів.

Теоретичні відомості

Умовні оператори та оператори циклів є фундаментальними елементами будь-якої мови програмування, включаючи Python. Вони дозволяють керувати потоком виконання програми залежно від певних умов або повторювати виконання блоку коду до досягнення заданої умови.

Розглянемо умовні оператори. У Python для реалізації умовних переходів використовується оператор if, який дозволяє виконувати певний блок коду, якщо задана умова є істинною (True).

Синтаксис оператора if наступний:

```
if умова:
    # блок коду, що виконується, якщо умова істинна.
```

Для обробки альтернативних сценаріїв використовуються оператори else та elif (скорочення від «else if»):

```
if умова:
    # блок коду, що виконується, якщо умова істинна
elif інша_умова:
    # блок коду, що виконується, якщо інша_умова істинна
else:
    # блок коду, що виконується, якщо жодна з умов не є істинною
```

Приклад:

```
x = 10
if x > 0:
    print("x є додатним числом")
elif x == 0:
    print("x дорівнює нулю")
```

```
else:
    print("x є від'ємним числом")
```

Розглянемо оператори циклів. Цикли дозволяють повторювати виконання певного блоку коду кілька разів. У Python існують два основних типи циклів: for та while.

```
Цикл for елемент in послідовність:
    # блок коду, що виконується для кожного елемента
```

Цей код виведе числа від 0 до 4.

Розглянемо цикл while. Цикл while повторює виконання блоку коду, поки задана умова є істинною.

Синтаксис циклу while:

```
while умова:
    # блок коду, що виконується, поки умова істинна
```

Приклад:

```
count = 0
while count < 5:
    print(count)
    count += 1
```

Цей код також виведе числа від 0 до 4.

Порядок виконання роботи

1. Ознайомитися з теоретичними відомостями щодо умовних операторів та операторів циклів у Python.
2. Розглянути наведені приклади програм з використанням умовних операторів та циклів.
3. Виконати практичні завдання відповідно до свого варіанту (див. нижче).
4. Перевірити правильність роботи розроблених програм.
5. Оформити звіт про виконану роботу.

Варіанти завдань

1. Написати програму, яка перевіряє, чи є введене число досконалим (сума його дільників дорівнює самому числу).
2. Розробити програму для знаходження суми всіх парних чисел у списку.

3. Створити програму, яка знаходить найбільший та найменший елемент у списку.
4. Написати програму для перевірки, чи є рядок анаграмою іншого рядка.
5. Розробити програму, яка обчислює добуток елементів списку.
6. Написати програму, яка обчислює кількість слів у рядку.
7. Розробити програму для знаходження всіх простих чисел у заданому діапазоні.
8. Створити програму, яка реалізує сортування методом бульбашки.
9. Написати програму, яка обчислює кількість унікальних символів у рядку.
10. Розробити програму для знаходження найбільшої спільної підстрічки у двох рядках.

Порядок оформлення роботи

1. Титульна сторінка з зазначенням назви роботи, ПІБ студента, групи та дати виконання.
2. Мета роботи.
3. Теоретичні відомості.
4. Код виконаних програмних завдань.
5. Висновки.
6. Список використаних джерел.

Список використаних джерел до практичного заняття №8

1. Eric Matthes. Python Crash Course. No Starch Press, 2023. 544 p.
2. Al Sweigart. Automate the Boring Stuff with Python. No Starch Press, 2022. 592 p.
3. Mark Lutz. Learning Python. O'Reilly Media, 2021. 1648 p.
4. Luciano Ramalho. Fluent Python. O'Reilly Media, 2022. 1014 p.
5. David Beazley. Python Cookbook. O'Reilly Media, 2021. 706 p.

Практичне заняття 9

Тема: таймери та переривання в мікроконтролерах

Мета роботи: ознайомитися з принципами роботи таймерів та системою переривань мікроконтролерів. Навчитися програмувати таймери для реалізації подій у реальному часі та використовувати переривання для обробки зовнішніх подій.

Теоретичні відомості

Розглянемо загальні поняття про таймери. Таймери є внутрішніми периферійними пристроями мікроконтролера, які відраховують кількість тактів тактового генератора. Їх можна використовувати для:

- генерації затримок;
- періодичних подій (наприклад, блимання світлодіода);
- вимірювання часу між подіями.

Таймери мають регістри:

- CNT (лічильник) – зберігає поточне значення лічильника;
- PSC (прескейлер) – визначає, наскільки потрібно зменшити частоту годинника;
- ARR (auto-reload register) – задає граничне значення, при якому генерується подія переповнення.

Розглянемо теоретичні основи переривань. Переривання (interrupts) – це механізм, який дозволяє мікроконтролеру призупинити виконання основної програми для обробки зовнішніх або внутрішніх подій.

Типи переривань:

- зовнішні (наприклад, натискання кнопки);
- внутрішні (наприклад, переповнення таймера);

Кожне переривання має:

- джерело (подія);
- обробник (ISR – interrupt service routine);
- прапорець запиту (interrupt flag);
- механізм дозволу/заборони.

Завдяки комбінації таймерів та переривань, можливо точно керувати подіями в мікроконтролерах:

- точне блимання світлодіода,
- реагування на натискання кнопки без постійного опитування порту,
- генерація ШІМ сигналів, тощо.

Порядок виконання роботи

Завдання 1: Таймер для блимання світлодіода (ATmega32, Timer0)

Схема підключення – світлодіод підключено до PD0 через резистор 220 Ом.

Розглянемо код на C (AVR-GCC):

```
#include <avr/io.h>
#include <avr/interrupt.h>
void timer0_init()
{
   // Встановити режим Normal
   TCCR0 = (1 << CS02) | (1 << CS00); // Частота = F_CPU/1024
   TIMSK |= (1 << TOIE0);            // Дозвіл переривання по переповненню
   TCNT0 = 0;                        // Початкове значення лічильника
   sei();                            // Глобальний дозвіл переривань
}
ISR(TIMER0_OVF_vect)
{
   static uint16_t counter = 0;
   counter++;
   if (counter >= 61) // приблизно 500 мс при 8 МГц і діленні на 1024
   {
      PORTD ^= (1 << PD0); // Перемикання стану світлодіода
      counter = 0;
   }
}
int main(void)
{
```

```
    DDRD |= (1 << PD0);  // PD0 як вихід
    timer0_init();
    while (1)
    {
        // основна програма порожня – все в перериваннях
    }
}
```

Завдання 2: Переривання від кнопки (INT0)

Схема підключення: – кнопка до INT0 (PD2) і GND, з підтягувальним резистором (внутрішній або зовнішній); світлодіод до PD1.

Код на C (AVR-GCC):

```
#include <avr/io.h>
#include <avr/interrupt.h>
void ext_int0_init()
{
    MCUCR |= (1 << ISC01);   // Переривання по спаду (falling edge)
    GICR |= (1 << INT0);     // Дозвіл переривання INT0
    sei();                   // Глобальний дозвіл переривань
}
ISR(INT0_vect)
{
    PORTD ^= (1 << PD1);     // Перемикаємо стан світлодіода
}
int main(void)
{
    DDRD |= (1 << PD1);      // PD1 як вихід
    DDRD &= ~(1 << PD2);     // PD2 як вхід (INT0)
    PORTD |= (1 << PD2);     // Внутрішній підтягувальний резистор
    ext_int0_init();
    while (1)
    {
```

```
        // Очікуємо натискання кнопки
    }
}
```

Рекомендації для тестування в Proteus

1. Створити схему з ATmega32, кварцом 8 МГц (або 4 МГц).
2. Підключити кнопку до PD2 (INT0) з підтягуванням.
3. Підключити світлодіоди до PD0 (таймер) і PD1 (кнопка).
4. Завантажити .hex файл у мікроконтролер у Proteus.

Порядок оформлення звіту

1. Тема, мета роботи.
2. Схеми підключення світлодіода та кнопки.
3. Лістинг коду з коментарями.
4. Результати експериментів.
5. Скріншоти (якщо можливо):

- роботи на Atmega32 або інші середовища.

6. Висновки:

– що вдалося реалізувати;
– які труднощі виникли;
– практична цінність використання таймерів та переривань.

Список використаних джерел до практичного заняття №9

1. Mazidi, M. A., Naimi, S., & Naimi, S. The AVR Microcontroller and Embedded Systems Using Assembly and C. Pearson, 2022. 944 p.

2. Кубрак, В.І. Мікроконтролери STM32: програмування і приклади. Київ: Ліра-К, 2023. 312 с.

3. Valvano, J. Embedded Systems: Real-Time Interfacing to ARM Cortex-M Microcontrollers. CreateSpace Independent Publishing, 2021. 570 p.

4. Попов, С.В. Основи мікроконтролерного програмування. Харків: ХНУРЕ,2022. 267 с.

Практичне заняття 10

Тема: робота з бітовими операціями в С

Мета практичної роботи:

1) ознайомитися з основними бітовими операторами мови програмування С;

2) навчитися використовувати побітові операції для встановлення, скидання, перемикання та перевірки окремих бітів;

3) закріпити навички програмування через роботу з регістрами вводу/виводу (імітаційно або з використанням мікроконтролера).

Теоретичні відомості

Бітові операції дозволяють працювати з окремими бітами змінних. Це особливо важливо при програмуванні мікроконтролерів, де керування пристроями часто здійснюється через регістри, представлені у вигляді 8- або 16-бітних чисел (таблиця 3).

Таблиця 3 – Основні бітові операції

Оператор	Назва	Приклад	Результат
`&`	Побітове І	`a & b`	1 лише якщо обидва біти = 1
`` ` ``	`` ` ``	Побітове АБО	`` `a ``
`^`	Побітове XOR	`a ^ b`	1 якщо біти різні
`~`	Побітове НЕ	`~a`	Інвертує всі біти
`<<`	Зсув вліво	`a << n`	Зсуває біти вліво
`>>`	Зсув вправо	`a >> n`	Зсуває біти вправо

Порядок виконання роботи

Завдання 1. Встановлення, скидання, перевірка бітів.

Створіть програму, яка:

1) ініціалізує змінну unsigned char reg = 0x00;
2) встановлює 3-й біт у 1 (reg |= (1 << 3);)
3) скидає 1-й біт у 0 (reg &= ~(1 << 1);)
4) перемикає 2-й біт (reg ^= (1 << 2);)
5) виводить значення змінної після кожної операції у двійковій формі.

Завдання 2. Перевірка стану біта.

Напишіть функцію int isBitSet(unsigned char value, int bit) яка повертає 1, якщо вказаний біт встановлено, інакше – 0.

Завдання 3. Імітація регістра пристрою.

Опишіть регістр стану з такими прапорами:

- BIT0 – Стан підключення;
- BIT1 – Помилка;
- BIT2 – Готовність.

Напишіть код, що:

- встановлює прапор «Готовність»;
- виводить стан прапора «Помилка».

Порядок оформлення звіту

У звіті повинно бути:

1) код програм;
2) результати виводу на екран;
3) коментарі до кожної операції;
4) висновки щодо ефективності використання побітових операцій;
5) список використаних джерел.

Список використаних джерел до практичного заняття №10

1. Кочан С. Програмування мовою С. 4-те вид. Харків: Видавництво «Фактор», 2023. 552 с.

2. Малевич І. П. Системне програмування мовою С. Львів: ЛНУ ім. Івана Франка, 2022. 318 с.

3. Прат С. С. Повний курс програмування. Київ: Видавництво «Наука і техніка», 2021. 832 с.

4. Васильєв Є. А. Бітові операції в С та їх застосування. Харків: ТехноПрес, 2023. 184 с.

5. Perry G. C Programming Absolute Beginner’s Guide. 4th Edition. Indianapolis: Que Publishing, 2022. 384 p.

Практичне заняття 11

Тема: організація обміну даними через UART, I2C, SPI

Мета практичної роботи: ознайомитися з принципами роботи послідовних інтерфейсів UART, I2C та SPI; навчитись налаштовувати UART для обміну з ПК та реалізовувати обмін даними між мікроконтролерами через I2C.

Теоретичні відомості

Розглянемо UART (Universal Asynchronous Receiver Transmitter):

1) призначення: асинхронна передача даних точка-точка;
2) параметри: швидкість (baudrate), кількість бітів даних, біти парності, стоп-біти;
3) приклад використання: передача даних на комп'ютер (через USB-UART міст).

Розглянемо I2C (Inter-Integrated Circuit):

1) протокол з двома лініями: SDA (дані), SCL (тактування);
2) один ведучий (master), кілька ведених (slave);
3) 7-бітна або 10-бітна адресація;
4) старт/стоп умови, ACK/NACK;
5) швидкість передачі: стандартна (100 кГц), швидка (400 кГц), високошвидкісна (1 МГц).

Розглянемо SPI (Serial Peripheral Interface):

1) чотири лінії: MOSI, MISO, SCLK, SS;
2) паралельна передача: full-duplex;
3) ведучий керує тактуванням (SCLK);
4) більш швидкий, ніж I2C, але потребує більше ліній.

Порядок виконання роботи

Завдання 1. Налаштування UART для обміну даними з ПК.

1. Підключіть мікроконтролер до комп'ютера через USB-UART адаптер.
2. У середовищі розробки (наприклад, Atmel Studio або Arduino IDE) напишіть програму для передавання рядка "Hello, PC!" через UART.
3. Встановіть відповідну швидкість (9600 бод або 115200 бод).

4. Відкрийте серійний порт у терміналі (наприклад, Tera Term, PuTTY) і перевірте прийом.

Завдання 2. Реалізація обміну даними між двома мікроконтролерами через I2C.

1. Визначте один мікроконтролер як master, інший як slave.
2. Підключіть їх через SDA та SCL з резисторами підтягування (4,7 кОм).
3. У master-програмі реалізуйте надсилання байта (наприклад, 0x55).
4. У slave-програмі реалізуйте прийом і виведення байта через UART.
5. Перевірте правильність передачі.

Приклад програм (на Arduino)

UART (Arduino)

```
void setup() {
  Serial.begin(9600);
void loop() {
  Serial.println("Hello, PC!");
  delay(1000);
}
```

I2C Master (Arduino)

```
#include <Wire.h>
void setup() {
  Wire.begin();
}
void loop() {
  Wire.beginTransmission(8); // адреса slave
  Wire.write(0x55);
  Wire.endTransmission();
  delay(1000);
}
```

I2C Slave (Arduino)

```
#include <Wire.h>
void setup() {
```

```
  Wire.begin(8); // адреса slave
  Wire.onReceive(receiveEvent);
  Serial.begin(9600);
}
void loop() {}
void receiveEvent(int bytes) {
  int x = Wire.read();
  Serial.println(x, HEX);
}
```

Порядок оформлення звіту

1. Тема, мета роботи.
2. Короткий опис теоретичних основ.
3. Схеми з’єднання UART, I2C.
4. Лістинг програмного коду.
5. Скриншоти результатів (наприклад, дані в терміналі).
6. Висновки: що вдалось реалізувати, труднощі, особисті зауваження.

Список використаних джерел до практичного заняття №11

1. Сивачук, І. В. Мікроконтролери і периферія. Київ: Ліра-К, 2023. 324 с.
2. Гуржій, А. М. Програмування мікроконтролерів AVR. Харків: ХНУРЕ, 2021. 278 с.
3. Mazidi, M. A., Naimi, S., & Naimi, S. The AVR Microcontroller and Embedded Systems Using Assembly and C. Pearson, London, 2022. 880 p.
4. Valvano, J. W. Embedded Systems: Introduction to ARM Cortex-M Microcontrollers. CreateSpace, USA, 2021. 684 p.
5. Паламарчук, С. М. Програмування периферій мікроконтролерів. Львів: Львівська політехніка, 2024. 312 с.

Практичне заняття 12

Тема: безпровідні протоколи передачі даних (Bluetooth, Wi-Fi)

Мета практичної роботи: ознайомитися з принципами роботи безпровідних протоколів передачі даних Bluetooth і Wi-Fi. Набути практичних навичок підключення Bluetooth-модуля HC-05/HC-06 до мікроконтролера ARDUINO UNO, організації передачі команд з мобільного телефону, а також обробки отриманих даних на мікроконтролері.

Теоретичні відомості

Розглянемо загальна характеристика протоколів:

– Bluetooth – безпровідна технологія передачі даних на короткі відстані (до 10 м), що працює на частоті 2,4 ГГц. Призначена для створення персональних мереж (PAN). Найчастіше використовується для зв'язку між мікроконтролерами та мобільними пристроями;

– Wi-Fi – технологія бездротової локальної мережі (WLAN), що дозволяє пристроям підключатися до мережі Інтернет або один до одного на відстанях до 100 м.

Розглянемо модулі Bluetooth HC-05 та HC-06:

– HC-05 – універсальний модуль з можливістю роботи у режимі Master/Slave;

– HC-06 – спрощена версія, працює лише в режимі Slave;

– стандартний інтерфейс UART (TX, RX) дозволяє легко підключити модуль до більшості мікроконтролерів.

Розглянемо підключення до мікроконтролера:

– RX HC-05 до TX мікроконтролера (через подільник напруги до 3.3 В);

– TX HC-05 до RX мікроконтролера;

– VCC – 5 В;

– GND – земля.

Обробка команд з мобільного телефону:

– на телефоні використовується термінальний додаток (наприклад, Bluetooth Terminal, Serial Bluetooth Terminal);

– команди передаються у вигляді текстових символів через Bluetooth.

Порядок виконання роботи

1. Підключити модуль HC-05 або HC-06 до мікроконтролера (Arduino Uno або інший).
2. Написати програму на мові C для прийому команд з Bluetooth і керування світлодіодом або іншим пристроєм.
3. Завантажити програму в мікроконтролер.
4. На мобільному телефоні встановити та запустити додаток Bluetooth Terminal.
5. Встановити з'єднання з модулем HC-05 (пароль – 1234).
6. Надіслати команди (наприклад, "ON" – увімкнути світлодіод, "OFF" – вимкнути).
7. Перевірити коректність прийому та обробки команд.

Приклад програми (Arduino C)

```
char command = 0;
void setup() {
  Serial.begin(9600); // Підключення до HC-05
  pinMode(13, OUTPUT); // Світлодіод на піні 13
}
void loop() {
  if (Serial.available() > 0) {
    command = Serial.read();
    if (command == '1') {
      digitalWrite(13, HIGH); // Увімкнути світлодіод
    }
    else if (command == '0') {
      digitalWrite(13, LOW); // Вимкнути світлодіод
    }
  }
}
```

Порядок оформлення звіту

1. Тема, мета роботи.
2. Схема підключення HC-05 до мікроконтролера.
3. Список обладнання та інструментів.

4. Лістинг програми з поясненням.
5. Скріншоти з мобільного додатку при передачі команд.
6. Фото з підключення та результатами роботи.
7. Висновки щодо реалізації Bluetooth-зв’язку.

Список використаних джерел до практичного заняття №12

1. Mazidi, M. A., Naimi, S., & Naimi, S. The AVR Microcontroller and Embedded Systems Using Assembly and C. London: Pearson, 2022. 864 p.

2. Бєляєв, О. В. Системи бездротового зв’язку для вбудованих пристроїв. Київ: Артбукс, 2023. 288 с.

3. Lobur, J., & Savytskyi, M. Embedded Systems and Wireless Communication. Warsaw: PWN, 2022. 402 p.

4. Руденко, В. А. Мікроконтролери та сенсорні системи на практиці. Львів: Новий Світ, 2021. 312 с.

5. Monk, S. Programming Arduino: Getting Started with Sketches. New York: McGraw-Hill, 2021. 224 p.

Практичне заняття 13

Тема: налаштування середовища розробки та перша програма для мікроконтролера AVR

Мета практичної роботи: ознайомитися з інтерфейсом середовища розробки Atmel Studio та симулятором Proteus, навчитись створювати перший проект для AVR-мікроконтролера, зібрати схему в Proteus та реалізувати керування світлодіодом.

Теоретичні відомості

Розглянемо середовище розробки Atmel Studio.

1. Безкоштовне IDE для AVR та ARM мікроконтролерів.
2. Підтримує написання коду мовою C/C++ або Assembler.
3. Основні компоненти: проект, target MCU, toolchain, компілятор.
4. Процес: створення нового проєкту → написання коду → компіляція → генерація *.hex файлу.

Розглянемо мікроконтролери AVR (на прикладі ATmega16/ATmega32).

1. Виробник: Microchip (раніше Atmel).
2. Вбудована Flash-пам'ять, SRAM, EEPROM, таймери, порти вводу/виводу, переривання.
3. Працюють з тактовими генераторами до 16 МГц.

Розглянемо Proteus для моделювання схем.

1. Симулятор електронних схем.
2. Підтримка мікроконтролерів, LCD, кнопок, світлодіодів, UART тощо.
3. Імпортування .hex файлу у мікроконтролер (наприклад, ATmega32).
4. Візуалізація роботи програм через світлодіоди, дисплеї тощо.

Порядок виконання роботи

Завдання. Увімкнути світлодіод на PA0 мікроконтролера ATmega32.

Налаштування Atmel Studio:

1. Створіть новий проект для ATmega32.
2. Виберіть C мову програмування.
3. Напишіть програму для встановлення порту A.0 в логічну 1.

Код програми:

```
#define F_CPU 8000000UL
#include <avr/io.h>
#include <util/delay.h>
int main(void) {
   DDRA |= (1 << PA0);  // встановлення PA0 як вихід
   while (1) {
      PORTA |= (1 << PA0); // вмикаємо світлодіод
      _delay_ms(500);
      PORTA &= ~(1 << PA0); // вимикаємо світлодіод
      _delay_ms(500);
   }
}
```

Компіляція проєкту: – згенеруйте .hex файл через Build → Build Solution.

Налаштування Proteus.

1) Створіть схему з ATmega32, світлодіодом та резистором.
2) Підключіть LED до порту PA0 через резистор (330 Ом).
3) Додайте джерело живлення (VCC, GND).
4) У властивостях ATmega32 вкажіть шлях до .hex файлу.

Запуск симуляції: – запустіть симуляцію і перевірте, що світлодіод блимає кожні 0,5 с.

Порядок оформлення звіту

1. Тема, мета роботи.
2. Короткий опис теоретичних основ.
3. Скріншоти:

– інтерфейсу Atmel Studio з кодом;

– схеми в Proteus;

– результату роботи (наприклад, засвічення світлодіода).

4. Лістинг програми.
5. Висновки: що зроблено, які труднощі виникли, що вдалося.

Список використаних джерел до практичного заняття №13

1. Гуржій, А. М. Програмування мікроконтролерів AVR. Харків: ХНУРЕ, 2021. 278 с.

2. Mazidi, M. A., Naimi, S., & Naimi, S. The AVR Microcontroller and Embedded Systems Using Assembly and C. Pearson, London. 2022, 880 p.

3. Паламарчук, С. М. Системне проектування мікропроцесорних пристроїв. Львів: Видавництво ЛП, 2023. 342 с.

4. Iovine, J. PIC Microcontroller Project Book. McGraw-Hill, New York, 2021. 212 p.

5. Чиркін, А. І. Емуляція мікропроцесорних систем в Proteus. Київ: КНУ, 2022. 198 с.

Практичне заняття 14

Тема: таймери та переривання в мікроконтролерах

Мета практичної роботи: ознайомитися з принципами роботи таймерів і переривань у мікроконтролерах, навчитися налаштовувати таймери, обробляти переривання та встановлювати пріоритети переривань, використовуючи мову асемблера.

Теоретичні відомості

Таймери – це внутрішні лічильники мікроконтролера, які можуть працювати в режимах підрахунку тактів, генерації затримок, частот, широтно-імпульсної модуляції (ШІМ) тощо.

Базовими параметрами таймера є:

- джерело тактування (внутрішнє або зовнішнє);
- розрядність (8/16/32 біт);
- режим роботи (однократний, періодичний, PWM);
- подільник частоти (prescaler).

Переривання – це механізм, який дозволяє мікроконтролеру призупинити основний потік виконання програми і перейти до обробника переривань, якщо виникла певна подія. Таймери часто використовуються для генерації періодичних переривань.

Розглянемо класифікації переривань.

1. Апаратні (hardware) – ініціюються зовнішніми чи внутрішніми подіями.
2. Програмні (software) – ініціюються інструкцією в програмі.

Пріоритет переривань дозволяє системі визначити порядок обробки, коли одночасно виникає декілька переривань. В більшості мікроконтролерів (наприклад, AVR чи STM32) існують регістри пріоритету переривань.

Основні регістри таймерів (на прикладі AVR ATmega32):

- TCCRn – регістр керування таймером n;
- TCNTn – лічильник таймера;
- OCRn – регістр порівняння;
- TIMSK – маска переривань таймера;
- TIFR – реєстр прапорців переривань.

Порядок виконання роботи

1. Вивчити принцип роботи таймера та переривань у мікроконтролері.
2. Обрати один із 25 варіантів завдань, які подано нижче.
3. Реалізувати програму на мові асемблера згідно з варіантом.
4. Зібрати та протестувати програму в симуляторі або на стенді.
5. Підготувати звіт за наведеним шаблоном.

Приклади типових програм на асемблері (AVR, ATmega32).

Генерація переривання кожні 1 с таймером 1 (16-бітний):

```
.include "m32def.inc"
.org 0x00
   rjmp reset
.org INT_TIMER1_COMPA
   rjmp timer1_isr
reset:
   ldi r16, high(15624)
   sts OCR1AH, r16
   ldi r16, low(15624)
   sts OCR1AL, r16
   ldi r16, (1<<WGM12)  ; режим CTC
   sts TCCR1B, r16
   ldi r16, (1<<OCIE1A)
   sts TIMSK, r16
   ldi r16, (1<<CS12)  ; подільник 256
   ori TCCR1B, r16
   sei ; дозволити переривання
loop:
   rjmp loop
timer1_isr:
   sbi PORTC, 0  ; приклад дії: ввімкнути світлодіод
   reti
```

Приклад оформлення звіту

1. Тема.

2. Мета.

3. Варіант завдання №.

4. Принцип роботи програми (пояснення).

5. Лістинг програми з коментарями.

6. Знімок екрана із симулятора / плати (якщо можливо).

7. Висновки.

8. Список використаних джерел.

Список використаних джерел до практичного заняття №14

1. Mazidi, M. A., Naimi, S., & Naimi, S. The AVR Microcontroller and Embedded Systems Using Assembly and C. Pearson, London, 2022. 880 p.

2. Dhananjay V. Gadre. Programming and Customizing the AVR Microcontroller. McGraw-Hill, New York, 2023. 520 p.

3. Жуков А. В. Мікроконтролери AVR у системах реального часу. Київ: Наукова думка, 2023. 312 с.

4. Соколов П. В. Системне програмування мікропроцесорних пристроїв. Харків: ХНУРЕ, 2022. 278 с.

5. Павленко О. Ю. Мікроконтролери AVR: програмування мовою асемблера. Львів: Львівська політехніка, 2024. 295 с.

Варіанти завдань

1. Реалізуйте переривання від таймера 0 кожні 100 мс у режимі overflow для керування блиманням світлодіода.

2. Налаштуйте таймер 1 у режимі CTC, щоб генерувати переривання кожні 500 мс, і в обробнику змінюйте стан порту.

3. Створіть програму, що формує PWM-сигнал 1 кГц з 75% заповненням на вихідному порту.

4. Побудуйте програму, яка підраховує кількість переповнень таймера 0 і відображає її на 8-бітному порту.

5. Реалізуйте переривання від зовнішнього сигналу (INT0) та викличте зміну частоти PWM-сигналу на виході.

6. Проведіть вимірювання тривалості натискання кнопки за допомогою таймера в режимі захоплення (Input Capture).

7. Налаштуйте два таймери: один генерує ШІМ-сигнал, другий викликає переривання кожні 2 секунди для його вимкнення.

8. Реалізуйте програму, в якій світлодіод блиматиме із затримкою 1 с, використовуючи пріоритет переривань між таймером та зовнішньою подією.

9. Побудуйте програму затримки 10 мс із використанням таймера та обробки переривання без циклів затримки (delay).

10. Реалізуйте частотний лічильник, що підраховує кількість імпульсів за 1 секунду за допомогою таймера.

11. Створіть програму, що виконує циклічне опитування стану кнопки, але блокує натискання менш ніж 50 мс (антидребіжання) через переривання таймера.

12. Налаштуйте переривання від двох таймерів із різними пріоритетами, що керують різними LED.

13. Використайте таймер для генерації частоти 2 кГц із 50% заповненням, змінюваної через зовнішнє переривання.

14. Напишіть програму, яка включає LED на 3 секунди, а потім вимикає, використовуючи переривання таймера.

15. Налаштуйте таймер 2 на режим ШІМ, що змінює яскравість LED із заданим циклом.

16. Виміряйте частоту зовнішнього сигналу, використовуючи таймер у режимі лічильника (counter mode).

17. Реалізуйте генератор імпульсів із змінною частотою (100–1000 Гц), що перемикається кнопкою з перериванням.

18. Запрограмуйте таймер на роботу у режимі CTC із частотою 10 Гц, та керуйте двома різними виходами з чергуванням.

19. Створіть програму, де переривання таймера керує оновленням годинника з виведенням секунд.

20. Налаштуйте таймер для виклику переривання кожні 250 мс і реалізуйте циклічну зміну стану 3-х світлодіодів.

21. Використайте комбінацію зовнішнього та таймерного переривання для реалізації подвійної реакції: на таймінг і на зовнішній вхід.

22. Змініть режим таймера на ходу (з overflow на CTC) під час виконання програми за зовнішнім сигналом.

23. Використайте таймер для реалізації годинника реального часу, який відображає хвилини та секунди через UART або дисплей.

24. Змініть пріоритет переривань у коді та порівняйте вплив на поведінку програми, описавши результати у звіті.

25. Створіть багатозадачну реакцію на таймерне переривання, яке викликає одночасно PWM, затримку та лічильник подій.

Практичне заняття 15

Тема: фінальний проєкт. Розробка комплексної системи

Мета практичної роботи: узагальнити знання, набуті впродовж курсу, та створити завершену мікропроцесорну систему, яка інтегрує різні технології та підходи: введення/виведення, обробка сигналів, інтерфейси зв'язку, таймери, переривання та сенсори.

Теоретичні відомості

Розробка технічного завдання (ТЗ):

1) вибір предметної області (автоматизація, контроль середовища, безпека, моніторинг);

2) визначення мети системи, функцій, вимог до обладнання, сценаріїв використання;

3) Приклад. Проєкт «Розумний термостат». Функції: – вимірювання температури, виведення значень на екран, керування реле нагріву, віддалене керування.

Створення функціональної схеми системи:

1) компоненти: мікроконтролер (ATmega32, STM32, Arduino), сенсори (температури, вологості, газу), виконавчі пристрої (реле, серводвигуни, світлодіоди);

2) інтерфейси: UART, I2C, SPI, Wi-Fi, Bluetooth;

3) пристрій виводу: LCD, OLED, світлодіодні індикатор;

4) схематичне представлення системи (у Proteus, Fritzing або від руки).

Програмування та тестування:

1) написання програмного забезпечення з підтримкою багатомодульної структури.

2) реалізація:

– зчитування сенсорів;
– обробки даних;
– керування виконавчими пристроями;
– взаємодії з користувачем (кнопки, дисплей);
– обміну через UART або Bluetooth.

Порядок виконання роботи

1. Обрати тему проєкту, наприклад:
    1) розумний термостат;
    2) система сигналізації;
    3) система керування освітленням;
    4) моніторинг якості повітря;
    5) автоматизований полив рослин.
2. Скласти технічне завдання:
    1) призначення;
    2) основні функції;
    3) умови експлуатації;
    4) перелік компонентів;
    5) Очікувані результати.
3. Розробити схему системи (у середовищі Proteus або вручну).
4. Написати програму згідно з ТЗ, протестувати її в середовищі моделювання або на реальній платі.
5. Оформити звіт, що включає:
    - опис задачі;
    - блок-схему;
    - схему підключення;
    - код;
    - результати роботи (фото, скріншоти, відео – за наявності).

Типовий приклад – «Розумний термостат на ATmega32»

Обладнання:

1) ATmega32.
2) датчик температури (DS18B20).
3) LCD 1602.
4) реле для керування обігрівачем.
5) кнопки для зміни порогів температури.

Функціонал:

1) зчитування температури;

2) вивід значення на екран;
3) включення/вимкнення реле при перевищенні заданого значення;
4) налаштування порогу кнопками.

Короткий фрагмент коду:

```
#include <avr/io.h>
#include <util/delay.h>
void relay_on() {
   PORTB |= (1 << PB0);
}
void relay_off() {
   PORTB &= ~(1 << PB0);
}
 int main(void) {
   DDRB |= (1 << PB0); // PB0 як вихід (реле)
   while (1) {
      int temp = get_temperature(); // псевдофункція
      if (temp > 25) {
         relay_off();
      } else {
         relay_on();
      }
      _delay_ms(1000);
   }
}
```

Порядок оформлення звіту

1. Тема, мета, технічне завдання.
2. Структурна схема або блок-схема системи.
3. Електрична схема підключення компонентів.
4. Програмний код з коментарями.
5. Результати моделювання / тестування.
6. Фото або скріншоти (обов’язково).

7. Висновки.

Список використаних джерел до практичного заняття №15

1. Гуржій, А. М. Проектування мікропроцесорних систем. Харків: ХНУРЕ, 2022. 312 с.

2. Mazidi, M. A., Naimi, S., & Naimi, S. The AVR Microcontroller and Embedded Systems Using Assembly and C. Pearson, London, 2022. 880 p.

3. Лобур, М. В. Системи управління на мікроконтролерах. Львів: Видавництво ЛП, 2021. 288 с.

4. Валованюк, С. Ю. Мікроконтролери в реальному часі. Київ: Кондор, 2023. 245 с.

5. Жуков, В. І. Embedded Systems: концепції, інтерфейси, проєкти. Львів: ЛП, 2024. 310 с.

ДЛЯ НОТАТОК

О - 73

Основи програмування мікропроцесорів та телекомунікаційних засобів. Методичні вказівки до виконання практичних занять для здобувачів першого (бакалаврського) рівня вищої освіти освітніх програм «Електроніка», «Автомобільна електроніка» та «Комп'ютеризовані телекомунікаційні мережі» галузі знань 17 Електроніка, автоматизація та електронні комунікації, спеціальностей 171 Електроніка та 172 Електронні комунікації та радіотехніка, всіх форм навчання, уклад. М. В. Хвищун, М.М.Євсюк. Луцьк: ЛНТУ, 2025. 62 с.

Комп'ютерний набір Микола ХВИЩУН
Редактор Микола ЄВСЮК

Підп. до друку «__»_____2025 р.
Формат 60x84/16. Папір офс.
Гарн. Таймс. Ум. друк. арк. 1,84.
Тираж 50 прим.

Відділ іміджу та промоції
Луцького національного технічного університету
43018 м. Луцьк, вул. Львівська, 75
Друк – ВІП ЛНТУ