**Міністерство освіти і науки України**

**Національний університет “Львівська Політехніка”**


# Лабораторна робота №6А

**Виконав:**
Студент групи АП-11
Білий Анатолій Іванович

**Прийняв**:

Чайковський І.Б.

Львів – 2024

**Тема роботи:** Загальна структура програми мовою С, дослідження використання функцій уведення та виведення даних.

**Мета роботи:** Дослідження структури і використання функцій уведення та виведення даних у програмах мовою С.

## Попередні відомості

Програма на мові С складається з однієї або більше функцій і хоча б одна з них повинна називатися **main( ).** Опис функції складається з заголовку та тіла. Заголовок у свою чергу містить директиви препроцесора типу **#include** і.т.д, що під'єднують бібліотечні файли та специфікують перетворення тексту програми перед компіляцією; а також ім'я функції. Ознакою імені функції служать круглі дужки. Тіло функції поміщається в фігурні дужки та є набором операторів (команд), кожен із яких закінчується символом " **;** "- крапка з комою. Елементом програми є коментар - частина тексту програми для пояснення окремих операторів, що входять до її складу. Коментар не впливає на виконання операторів і записується таким чином: // **текст коментарю** або так: /* **текст коментарю***/. В першому випадку коментар має бути єдиним у рядку або в кінці рядка. Другий спосіб дозволяє записувати коментар будь-де в тексті програми не розриваючи лексем.

Оголошення змінної задає ім'я та атрибути останньої. Визначення змінної крім задання імені та її атрибутів приводить до виділення для неї пам'яті.

Програма може містити довільне число директив, вказівок компілятору, оголошень та визначень. Їх синтаксис розглядатиметься нижче. Порядок появи цих елементів у програмі є важливий. Загальна структура програми мовою С має вигляд.

## Заголовок

```
#include <  назва бібліотечного файлу>  // директива препроцесора 1
 ......
```

```
#include < назва бібліотечного файлу>  // директива препроцесора N
.....                                  //  інші директиви препроцесора

main ( )
                                  Тіло
        {
           <оголошення змінних>
            < оператор 1 >
             .....
            < оператор N >
         }
```

В наступному прикладі приведена проста програма.

**#include <stdio.h>** // ця інструкція препроцесора вказує компілятору, що до програми необхідно під'єднати інформацію, що міститься в заголовковому файлі stdio.h

```
main()     // визначення головної функції
{
int z;  // оголошення змінних
int w;
int x=1;  // присвоєння змінним їх значеня
int y=2;
z=y+x;   // виконувані оператори
w=y-x;
}
```

У приведеній програмі значення змінних вводяться за допомогою операції присвоєння, що є не зовсім зручним, особливо коли змінних багато. Таке введення даних також вимагає наявності початкового тексту програми та його компіляції при кожній їх зміні. Результат обчислення (змінні z і w) взагалі не візуалізується. Для здійснення цього необхідно застосувати засоби взаємодії з програмою – функції **printf() і scanf()**. Це не єдині функції, якими можна користуватися для введення і виведення даних з допомогою програм на мові С, але вони найбільш універсальні. Вказані функції не входять в опис мови С і реалізація операцій уведення-виведення покладається на розробників компілятора, що дає можливість більш ефективно організовувати

їх на конкретних машинах.

Функції printf() і scanf() працюють подібно - кожна використовує «керуючий рядок» і «список аргументів».

Функція printf(). Інструкції, що передаються функції printf(), якщо необхідно надрукувати деяку змінну, залежать від типу змінної. Наприклад, при виведенні на друк цілого числа застосовується формат %d, а при виведенні символа %c. В таблиці 1 наведені всі формати, що вказуються при звертанні до функції printf .

Таблиця 1

| Формат | Тип інформації, що виводиться |
|---|---|
| **%d** | Десяткове ціле число |
| **%c** | Один символ |
| **%s** | Рядок символів |
| **%e** | Число з плаваючою точкою, експонентний запис |
| **%f** | Число з плаваючою точкою, десятковий запис |
| **%g** | Використовується замість записів **%f або %e**, якщо він коротший. |
| **%u** | Десяткове ціле число без знаку |
| **%o** | Вісімкове ціле число без знаку |
| **%x** | Шістнадцяткове ціле число без знаку |

Кожному формату відповідає тип внведеної з їх допомогою інформації. Слід зазначити, що останні чотири формати застосовуються досить рідко.

**Використання функції printf ().**

```
/* друк різноманітної інформації*/
#define PI 3.14159
# include <stdio.h>
main()
{
int a =5;
float b=23. 5;
int c=31000;
printf(“ %d  метрів тканини коштувало %f  гривень.\n”, a,b);
printf(“Значення числа рі рівне %f.\n”, PI);
```

```
printf(“ IBM сумісні комп’ютери набули широкого
розповсюдження.\n)”;
printf(“ %c%d\n”, ‘$’, c);
}
```

**Результат роботи програми:**

5 метрів тканини коштувало 23.500000 гривень.

Значення числа pi рівне 3.14159.

IBM сумісні комп’ютери набули широкого розповсюдження.

$31000

Формат, що вказується при зверненні до функції **printf()** має наступний вигляд:

**printf(Керуючий рядок, аргумент1, аргумент2, ,);**

Аргумент1, аргумент2 і т.д. – це друковані параметри, які можуть бути змінними, константами або навіть виразами, що обчислюються спочатку, перед виведенням на друк.

Керуючий рядок – рядок символів, що показує як повинні бути надруковані параметри. Наприклад у операторі **printf(“ %d метрів тканини коштувало %f гривень.\n”, a,c);** керуючим рядком служить фраза в лапках (враховуючи попередні зауваження, це – рядок символів), а **a** і **b** аргументи або в даному випадку значення двох змінних.

У рядку програми printf(“Значення числа pi рівне %f.\n”, PI); список аргументів містить тільки один елемент – символічну константу PI.

В керуючому рядку міститься інформація двох різних видів: символи, що друкуються текстуально; ідентифікатори даних, що називаються також “специфікаціями перетворення”.

Кожному аргументу із списку, що слідує за керуючим рядком, повинна відповідати одна специфікація перетворення.

Якщо необхідно надрукувати яку-небудь фразу, то необхідності використовувати спефікацію перетворення немає. Тому такий оператор є коректний:

**printf(“ IBM сумісні комп’ютери набули широкого розповсюдження.\n)”;**

Зауважимо, що в операторі **printf(“ %c%d\n”, ‘$’, c);** перший аргумент із друкованого списку є символьною константою, а не змінною.

В функції **printf()** є можливість дещо розширити основне визначення специфікації перетворення, помістивши модифікатори між знаком % і символом, що визначає тип перетворення. В таблиці 2 подано список цих символів. При використанні одночасно декількох модифікаторів вони повинні бути вказані в тому порядку, в якому перечислені в таблиці.

**Розглянемо приклади роботи модифікаторів ширини поля на друк цілого числа.**

```
main()
{
        printf(“/%d/n”, 557);
        printf(“/%2d/n”, 557);
        printf(“/%10d/n”, 557);
        printf(“/%–10d/n”, 557);
}
```

Програма надрукує наступне:

**/557/**
**/557/**
**/ 557/**
**/557 /**

Таблиця 2

| Модифікатор | Значення |
|---|---|
| — | Аргумент буде друкуватися з лівої позиції заданої ширини. Друк аргументів закінчується в правій крайній позиції поля. Приклад: **%–10d** |
| Рядок цифр | Задає мінімальну ширину поля. Більше поле буде використовуватися, якщо друковане число або рядок не вміщуються в початковому полі. Приклад: **%4d** |

| | |
|---|---|
| Рядок цифр | Визначає точність: для типів даних із плаваючою точкою - число друкованих цифр праворуч від десяткової точки; для символьних рядків максимальне число друкованих символів. Приклад: **%4.21** (дві десяткові цифри для поля шириною в чотири символи) |
| L | Відповідний елемент даних має тип long, а не int. Приклад: %ld |

**Розглянемо приклади форматів, що відповідають даним з плаваючою точкою.**

```
main()
{
      printf("/%f/\n", 6543,21);
      printf("/%e/\n", 6543,21);
      printf("/%4,2f/\n", 6543,21);
      printf("/%3.1f/\n", 6543,21);
      printf("/%10.3f/\n", 6543,21);
      printf("/%10.3e/\n", 6543,21);
 }
```

Програма надрукує наступне:

**/6543.210000/**
**/6.543210e+03/**
**/6543.21/**
**/6543.2/**
**/  6543.210/**
**/ 6.543e+03/**

Розглянемо застосування функції **printf()** для роботи із рядками.

```
#define riadok "Чудова погода"
main()
{
      printf("/%2s/\n", riadok);
      printf("/%15.s/\n", riadok);
      printf("/%15.5s/\n", riadok);
      printf("/%–15.5s/\n", riadok);
```

```
 }
```

Програма надрукує наступне:

**/ Чудова погода/**

**/ /**

**/ Чудова /**

**/ Чудова /**

Розглянемо застосування функції **printf()** для перетворення даних.

```
main()
{
      printf("%d\n", 557);
      printf("%o\n", 557);
      printf("%x\n", 557);
      printf("%d\n", –557);
      printf("%u\n", –557);
 }
```

Програма надрукує наступне:

**557**

**1055**

**22d**

**–557**

**64979**

Розглянемо застосування функції **printf()** для знаходження коду символів таблиці ASCII. Оператор printf("%c%d\n", 'B','B'); надрукує наступне **B66**.

B – це буква, 66 – десятковий код ASCII символа B. Для отримання вісімкового коду ASCII символя B слід застосувати специфікацію %o, а шістнадцяткового %x.

**Функція scanf().** Подібно до функції **printf()** для функції **scanf()** вказуються керуючий рядок і слідуючий за нею список аргументів. Основна відмінність цих двох функцій полягає в особливостях даного списку. Функція **printf()** використовує імена змінних, константи та вирази, тоді як функція **scanf() –** тільки вказівники на змінні. Якщо необхідно ввести деяке значення

та присвоїти його змінній одного з основних типів, то перед іменем змінної необхідно написати символ &. Якщо необхідно ввести значення рядкової змінної, використовувати символ & необов'язково.

Робота функції. Виконувана програма зупиняється і система переходить у режим очікування введення даного. В цей час монітор комп'ютера темний і миготить курсор. Користувач набирає на клавіатурі значення змінної та натискає клавішу вводу **Enter.** В результаті виконання цієї функції змінній буде присвоєно певне значення, що введене з клавіатури. Для кращого розуміння функціонування створюваної програми перед функцією введення даних варто застосовувати функцію виведення на монітор, зокрема **printf()**.

Розглянемо програму, що демонструє використання фунцій **printf() і scanf().**

```
#include<stdio.h>
#include<conio.h>
main()
{
int vik;
char name[30];
clrscr();
printf("Vash vik?\n");
scanf("%d",&vik);
printf("Uvedit vashe imja\n");
scanf("%s",name);
printf("Pryvit %s jakomu %d rokiv",name,vik);
}
```

Функція **scanf()** використовує практично той самий набір символів специфікації перетворення, що й функція **printf().** Головні відмінності для функції **scanf()** наступні:

1. Відсутня специфікація %g.
2. Специфікації %f і %e еквівалентні. Обидві специфікації допускають наявність (або відсутність) знаку, рядка цифр із десятковою точкою або без неї і поля показника степені.

3. Для читання цілих чисел типу short застосовується специфікація **%h**.

**Функції getchar() і putchar()**

Функція **getchar()** отримує один символ, що поступає з пульта термінала і передає його програмі, що виконується в даний момент. Функція **putchar()** отримує один символ, що поступає з програми та пересилає його для виведення на екран.

Функція **getchar()** аргументів не має, тобто при її виклику в круглих дужках не поміщається жодна величина. Вона отримує черговий поступаючий символ і сама повертає його значення виконуваній програмі.

Функція **putchar()** має один аргумент. При її виклику необхідно в дужках вказати символ, який необхідно вивести на друк. Аргументом може бути одиночний символ (включаючи знаки, що представляються керуючими послідовностями, змінна або функція значенням якої є один символ. Зразки звертання до фунції **putchar()**:

putchar(‘S’); – символьна константа, що поміщена в апострофи
putchar(‘\n’);
putchar(‘\007’);
putchar(ch); – змінна типу char
putchar(getchar());

Наступна програма демонструє виведення на друк групи символів із припиненням роботи програми при введенні певного символу.

```
#include<stdio.h>
#include<conio.h>
#define STOP '*'
main()
{
char ch;
clrscr();
ch=getchar();
m1: if( ch !=STOP)
    {
      putchar(ch);
```

```
ch=getchar(); goto m1; } }
```

## ЗАВДАННЯ.

1. Виконати усі приклади, що наведені в теоретичних відомостях.
2. У звіті зазаначити формати, що використовуються функціями **printf()** і **scanf().**
3. Створиит програму, в якій задати числа, що оголошені як типи int, float, char, long та вивчити вплив модифікаторів специфікації перетворення для функції **printf().**
4. Надрукувати в рядок 10 будь-яких символів таблиці ASCII та відповідні їм коди в десятковій, вісімковій, шістнадцятковій системах.
5. Створити програму для розв'язання задачі купівлі товарів за формулою - вартість купівлі дорівнює: ціна товару помножена кількість. Знайти суму купівлі при номенклатурі товарів не менше 5. Вхідні дані задавати: а). під час оголошення змінних, б). введенням із клавіатури використовуючи функцію **scanf().** Результати оформити у вигляді таблиці.
6. Створити програму обчислення довжини кола та площі круга за радіусом, який задавати введеням із клавіатури.
7. Створити програму обчислення коренів квадратного рівняння. Задачу виконати у вигляді діалогу з уведенням набору коефіцієнтів за допомогою клавіатури.
8. Модифікувати програму виведення на друк групи символів із застосуванням функцій **getchar() і putchar()** так, щоб символом припиненням роботи програми при уведенні був звуковий сигнал - функція **sound(частота)**.
9. Обчислити периметр трикутника, його площу та радіус вписаного кола за заданими координатами його вершин A(1; 1), B(2k; 2k-1), C(-2k; k+2), де к – номер варіанта.

   Формули для обчислення:

   - відстань між точками $(x_1, y_1),(x_2, y_2)$:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}\ ;$$

- півпериметр трикутника: $p = (a + b + c)/2$, де *a, b, c* сторони трикутника;
- площа трикутника: $s = \sqrt{p(p-a)(p-b)(p-c)}$ ;
- радіус вписаного кола: $r = \frac{s}{p}$

| № варіант | Завдання 10 |
|---|---|
| 1 | $\frac{(a+b)^2 - (a^2 + 2ab)}{b^2}$ , при a=1000, b=0.0001 |
| 2 | $\frac{(a-b)^3 - (a^3)}{b^3 - 3ab^2 - 3a^2b}$ , при a=1000, b=0.0001 |
| 3 | $\frac{(a+b)^3 - (a^3 + 3a^2b)}{3ab^2 + b^3}$ , при a=1000, b=0.0001 |
| 4 | $\frac{(a+b)^3 - (a^3)}{3ab^2 + b^3 + 3a^2b}$ , при a=1000, b=0.0001 |
| 5 | $\frac{(a-b)^3 - (a^3 - 3a^2b)}{b^3 - 3ab^2}$ , при a=1000, b=0.0001 |
| 6 | $\frac{(a-b)^3 - (a^3 - 3ab^2)}{b^3 - 3a^2b}$ , при a=1000, b=0.0001 |
| 7 | $\frac{(a-b)^3 - (a^3)}{b^3 - 3ab^2 - 3a^2b}$ , при a=1000, b=0.0001 |
| 8 | $\frac{(a+b)^4 - (a^4 + 4a^3b + 6a^2b^2)}{4ab^3 + b^4}$ ,при a=100, b=0.001 |
| 9 | $\frac{(a+b)^4 - (a^4 + 4a^3b)}{6a^2b^2 + 4ab^3 + b^4}$ , при a=100, b=0.001 |
| 10 | $\frac{(a-b)^4 - (a^4 - 4a^3b + 6a^2b^2)}{b^4 - 4ab^3}$ , при a=100, b=0.001 |
| 11 | $\frac{(a-b)^4 - (a^4 - 4a^3b)}{6a^2b^2 - 4ab^3 + b^4}$ , при a=100, b=0.001 |
| 12 | $\frac{(a+b)^2 - (a^2 + 2ab)}{b^2}$ , при a=1000, b=0.0001 |
| 13 | $\frac{(a-b)^2 - (a^2 - 2ab)}{b^2}$ , при a=1000, b=0.0001 |

| № варіант | Завдання 10 |
|---|---|
| 14 | $\frac{(a+b)^3-(a^3+3a^2b)}{3ab^2+b^3}$, при a=1000, b=0.0001 |

1. Для обчислення степеню можна використати багатократну операцію множення.
2. При виконанні завдання необхідно використати допоміжні змінні для зберігання проміжних результатів: наприклад: c=a*a*a; d=3*a*a*b; e=3*a*b*b; f=b*b*b;

Приклад 1

```
#include<stdio.h>
void main(){
    int z,w;
    int x = 1;
    int y = 2;
    z = x + y;
    w = y - x;
    printf("z = %d, w = %d",z,w);
}
```

---

z = 3, w = 1

Приклад 2

```
#define PI 3,14159
#include<stdio.h>
#include<windows.h>
void main(){
    SetConsoleCP(65001);
    SetConsoleOutputCP(65001);
    int a = 5;
    float b = 23.5;
    int c = 31000;
    printf("%d метрів тканини коштувало %.2f гривень\n",a,b);
    printf("Значення числа PI %f\n",a,b);
    printf("IBM сумісні комп'ютери набули широкого розповсюдження.\n");
    printf("%c%d",'&',c);
}
```

```
-----------------------------------------------------------------------------------------------
// 5 метрів тканини коштувало 23.50 гривень
// Значення числа PI 0.000000
// IBM сумісні комп'ютери набули широкого розповсюдження.
// &31000
```

Приклад 3

```
#include<stdio.h>
void main(){
    printf("/%d/\n",557);
    printf("/%10d/\n",557);
    printf("/%-10d/\n",557);
}
-----------------------------------------------------------------------------------------------
// /557/
// /       557/
// /557       /
```

Приклад 4

```
#include<stdio.h>
void main(){
    printf("%d\n",557);
    printf("%o\n",557);
    printf("%x\n",557);
    printf("%d\n",-557);
}
-----------------------------------------------------------------------------------------------
// 557
// 1055
// 22d
// -557
```

Приклад 5

```
#include<stdio.h>
#include<windows.h>
void main(){
```

```
    SetConsoleCP(65001);
    SetConsoleOutputCP(65001);
    int vik;
    char name[30];
    printf("Ваш вік?  ");
    scanf("%d",&vik);
    printf("Введіть ваше ім'я  ");
    scanf("%s",&name);
    printf("Привіт %s якому %d років",name,vik);
}
```

---

```
Ваш вік?  18
Введіть ваше ім'я  Анатолій
Привіт Анатолій якому 18 років
```

Приклад 6

```
#define STOP '*'
#include<stdio.h>
void main(){
    char ch;
    ch = getchar();
    if (ch != STOP){
    putchar(ch);
    main();}}
```

---

```
fgh234
fgh234
*
```

Завдання 4

```
#include<stdio.h>
void main(){
   char ch[10]= "ABCDEFG123";
   for(int i; i < 10; i++){
      printf("%c - %d - %-3o - %x\n",ch[i],ch[i],ch[i],ch[i]);
   }
```

```
}
```

---

```
 A - 65 - 101 - 41
 B - 66 - 102 - 42
 C - 67 - 103 - 43
 D - 68 - 104 - 44
 E - 69 - 105 - 45
 F - 70 - 106 - 46
 G - 71 - 107 - 47
 1 - 49 - 61  - 31
 2 - 50 - 62  - 32
 3 - 51 - 63  - 33
```

Завдання 6

```
 #include<stdio.h>
#define PI 3.14159
void main(){
    float r,P,S;
    scanf("%f",&r);
    if (r == 0){
        return 0;}
    P = 2*PI*r;
    S = PI*r*r;
    printf("P = %.2f\nS = %.2f\n",P,S);
    main();
}
```

---

```
 2
 P = 12.57
 S = 12.57
```

Завдання 7

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

```
    SetConsoleCP(65001);
    SetConsoleOutputCP(65001);
    float a,b,c,D,x,x1,x2;
    printf("Введіть коефіцієнт при x*2: ");
    scanf("%f",&a);
    if (a==0){
        return 0; }
    printf("Введіть коефіцієнт при x: ");
    scanf("%f",&b);
    printf("Введіть вільний коефіцієнт: ");
    scanf("%f",&c) ;
    D = b*b-(4*a*c);
    if (D<0){
        printf("Квадратне рівняння немає дійсних коренів\n\n");
        main(); }
    else if(D==0){
        x = -b/2*a;
        printf("Рівняння має лише один корінь X = %.2f\n\n",x);
        main(); }
    else if(D>0){
        x1 = (-b-sqrt(D))/(2*a);
        x2 = (-b+sqrt(D))/(2*a);
        printf("Перший корінь X1 = %.2f\n",x1);
        printf("Другий корінь X2 = %.2f\n\n",x2);
        main(); }
}
```

---

```
Введіть коефіцієнт при x*2: 1
Введіть коефіцієнт при x: 1
Введіть вільний коефіцієнт: 1
Квадратне рівняння немає дійсних коренів
Введіть коефіцієнт при x*2: 1
Введіть коефіцієнт при x: 4
Введіть вільний коефіцієнт: 1
Перший корінь X1 = -3.73
Другий корінь X2 = -0.27
```

Введіть коефіцієнт при x*2: 2
Введіть коефіцієнт при x: 4
Введіть вільний коефіцієнт: 2
Рівняння має лише один корінь X = -4.00

Завдання 8

```
#include <stdio.h>
#include <windows.h>
#define STOP '*'
int main(){
   SetConsoleCP(65001);
   SetConsoleOutputCP(65001);
   char ch = getchar();
   if (ch != STOP){
      putchar(ch);
       main();
   }
   else {
       Beep(400,900);
       return 0;}}
```

---

1
1
a
a
*

Завдання 9

```
#include <stdio.h>
#include <windows.h>
#include <math.h>
int main(){
   SetConsoleCP(65001);
   SetConsoleOutputCP(65001);
   float Ax,Ay,Bx,By,Cx,Cy;
   Ax = 1;
```

```
    Ay = 1;
    Bx = 16;
    By = 15;
    Cx = -16;
    Cy = 10;
    float a,b,c,P,p,S,r;
    a = sqrt((Ax-Bx)*(Ax-Bx)+(Ay-By)*(Ay-By));
    b = sqrt((Bx-Cx)*(Bx-Cx)+(By-Cy)*(By-Cy));
    c = sqrt((Cx-Ax)*(Cx-Ax)+(Cy-Ay)*(Cy-Ay));
    P = a+b+c;
    p = P/2;
    S = sqrt(p*(p-a)*(p-b)*(p-c));
    r = S/p;
    printf("Периметр трикутника P = %.2f\n",P);
    printf("Площа трикутника S = %.2f\n",S);
    printf("Раліус вписаного колаr = %.2f",r);
}
```

---

```
 Периметр трикутника P = 72.14
 Площа трикутника S = 186.50
 Радіус вписаного кола r = 5.17
```

Завдання 10 варіант №1

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

void main() {
    float a = 1000;
    float b = 0.0001;

    float ch = pow(2, a+b) - (a*a +2*a*b);
    float zn = b*b;

    float res = ch / zn;
    printf("Результат: %f\n", res);
}
```

---

```
Результат: inf
```

**Контрольні запитання.**

1. Структура програми на мови С.

Структура програми на мові програмування С включає наступні елементи:

Підключення бібліотек: У першому рядку програми зазвичай включаються необхідні бібліотеки за допомогою директиви #include.

Оголошення глобальних змінних і функцій: Після підключення бібліотек можуть бути оголошені глобальні змінні і функції, які будуть використовуватися в усій програмі.

Функція main: Головна функція main є точкою входу у програму. У ній зазвичай розміщується основний код програми.

Інші функції: Якщо програма складається з більшої кількості функцій, їх оголошення та визначення розміщуються після main або в окремих файлах.

Коментарі: Коментарі допомагають пояснити структуру програми та її окремі частини.

Правила форматування: Дотримання правил форматування полегшує читання та розуміння коду.

Завершення програми: У кінці програми може бути вказано команду або вираз для завершення програми, наприклад, return 0; у функції main.

2. Ідеологія організації операцій уведення-виведення в мові С.

деологія уведення-виведення в мові С базується на буферизації та використанні стандартних потоків: stdin (введення), stdout (виведення) і stderr (помилки). Функції **printf** і **scanf** використовуються для роботи зі стандартним введенням-виведенням, а функції **fopen**, **fclose**, **fread**, **fwrite** для роботи з файлами. Операції форматування та обробки помилок також важливі частини цієї ідеології.

3. Синтаксис функцій printf() і scanf().

printf і scanf використовуються для виведення та введення даних у мові С. Синтаксис printf виглядає як printf("формат", аргументи);, де "формат" вказує, як вивести дані, а аргументи - значення для виведення. Синтаксис scanf виглядає як scanf("формат", &змінні);, де "формат" вказує, як зчитати дані, а &змінні - адреса змінної для збереження введених даних.

4. Основні типи форматів при звертанні до функцій printf() і scanf().

%d - цілі числа

%f - числа з плаваючою точкою

%c - символи

%s - рядки

%p - покажчики

%x, %X - шістнадцяткові числа

%o - вісімкові числа

5. Модифікатори форматів при звертанні до функцій printf() і scanf().

Модифікатори форматів у функціях printf і scanf вказують на розмір або

характеристику аргумента. Наприклад, "%ld" вказує на використання long для цілого числа, а "%lf" - на використання double для числа з плаваючою точкою.

6. Відмінності при застосуванні функцій printf() і scanf()

   printf використовується для виведення даних, а scanf - для їх зчитування. printf приймає рядок для виведення та аргументи, scanf - рядок для зчитування та адреси змінних для збереження даних. printf повертає кількість виведених символів, а scanf - кількість успішно зчитаних значень.

7. Застосування функцій getchar() і putchar().

   Функція **getchar** зчитує один символ зі стандартного вводу, а **putchar** виводить один символ на стандартний вивід.

**Висновок:** під час виконання цієї лабораторної роботи я дослідив структури і використання функцій уведення та виведення даних у програмах мовою С.