

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Запорізький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт із дисципліни

"Програмування" (II частина)

для студентів спеціальності 123 «Комп'ютерна інженерія»
денної форми навчання

2019

Методичні вказівки до виконання лабораторних робіт із дисципліни "Програмування " (II частина) для студентів спеціальності 123 "Комп'ютерна інженерія" денної форми навчання / Укл.: Н.В. Луценко, О.В. Польська. – Запоріжжя: ЗНТУ, 2019. – 38 с.

Укладачі:

Н.В. Луценко, ст. викладач
О.В. Польська, ст. викладач

Рецензент:

Р.К.Кудерметов, к.т.н., доцент

Відповідальний за випуск

Н.В. Луценко, ст. викладач

Затверджено
на засіданні кафедри КСМ
Протокол № 7 від 21.01.2019

Затверджено
на засіданні НМК КНТ
Протокол № 6 від 31.01.2019

ЗМІСТ

Порядок виконання лабораторних робіт.....	4
Загальні відомості	5
Лабораторні роботи	17
Лабораторна робота № 1 Елементарне введення у функції	17
Контрольні питання до лабораторної роботи № 1	23
Лабораторна робота № 2 Прийоми використання функцій	24
Контрольні питання до лабораторної роботи № 2	29
Лабораторна робота №3 Робота з файлами	30
Контрольні питання до лабораторної роботи № 3	35
Лабораторна робота № 4 Багатофайлова компіляція.....	36
Контрольні питання до лабораторної роботи № 4	38
Рекомендована література.....	38

ПОРЯДОК ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ

Лабораторні роботи складаються з декількох програм, завдань до них і контрольних питань. Студент повинен розробити, відпрацювати і зберегти у робочому каталозі програму, що наведена, й програму, розроблену згідно заданого варіанта, та проаналізувати результати їхньої роботи. За результатами виконання кожної лабораторної роботи студент складає й захищає письмовий звіт, вимоги до якого викладені нижче.

Зміст звіту

1. Титульний лист, виконаний відповідно до СТП.
2. Мета роботи.
3. Відповіді на контрольні питання.

Поради та рекомендації

Запускаємо середовище програмування **Microsoft Visual C++ 2010 Express Edition**.

При цьому відкривається початкова сторінка, в якій потрібно створити пустий проект. Для цього обирається вкладка **File (файл)** та пункт **New->Project (Создать->Проект)**. В вікні, що з'явилося, обирається **Visual C++**, а справа обирається значок **Empty project (Пустой проект)**. Нижче задається назва проекту та вказується папка, в якій буде зберігатися цей проект.

Для додавання в проект файлу з кодом програми використовується меню **Project (Проект)** та пункт **Add New Item (Добавить новый элемент)**. В вікні, що з'явилося, обирається значок **C++File.cpp (файл.cpp)**, а нижче задається ім'я нового файлу. Далі натискається кнопка **Add (Добавить)**. Відкривається сторінка редактора, в якому пишеться код програми.

Для запуску програми обирається значок ► (**Start Debugging**).

Для відкриття написаної програми обирається вкладка **File (файл)** та пункт **Open->Project (Открыть->Проект)**.

ЗАГАЛЬНІ ВІДОМОСТІ

Функції. Це елементи програми, що можуть використовуватися неодноразово для виконання деякої, заздалегідь визначеної задачі. Кожна програма містить тільки одну головну функцію з ім'ям **main()**, за допомогою якої здійснюється вхід у відкомпільовану програму. Крім головної функції в програмі може бути багато інших функцій.

Усім іменам функцій за умовчанням привласнюється клас пам'яті **extern**, тобто функції глобальні, мають зовнішній вигляд компонування, статичну тривалість існування (до кінця виконання програми) і доступні за певних умов в усіх модулях програми.

Будь-яка функція, окрім головної функції **main()**, має бути визначена або описана до її використання (виклику) в конкретному модулі.

Визначення функції має наступний синтаксис:

```
тип_функції ім'я_функції(список_формальних_параметрів)  
{ тіло функції }
```

де **тип_функції** – це тип значення, що повертається функцією. Якщо функція нічого не повертає, то тип функції позначається **void**;

список_формальних_параметрів (чи сигнатура функції) – це список специфікацій окремих параметрів з вказівкою типів і імен, що записані через кому, або **void** чи порожньо, якщо у функцію не передаються ніякі параметри;

тіло функції – це послідовність описів, визначень змінних та різних операторів, що поміщені у фігурні дужки.

Коли функція викликається, управління передається в початок тіла функції. Повернення в точку виклику функції виконується оператором **return випаз;** або **return;**. У тілі функції може бути декілька операторів **return**, якщо вихід з функції здійснюється в декількох точках.

Наприклад, визначення функції, в яку передаються три параметри типу **float** і яка знаходить суму цих чисел та повертає результат:

```
float sum(float a, float b, float c)  
{return a+b+c;}
```

В цьому прикладі **float** – тип значення, яке повертає функція, **sum** – ім'я функції, за яким у круглих дужках стоїть список формальних параметрів **a, b, c**, а у фігурних дужках – тіло функції.

Якщо функція не повертає ніякого значення, то тип її **void**, а оператор **return** може бути без виразу або бути відсутнім. Наприклад:
void func1() {printf("pi=%lf\n",3.141592);}

Звернення до функції (або *виклик функції*). Це вираз, в якому вказується ім'я функції та список фактичних параметрів в круглих дужках. Результат цього виразу – значення, що повертає функція, тип якого співпадає з типом функції:

ім'я_функції(список фактичних параметрів);

де **список_фактичних_параметрів** – це порожньо або **void**, (якщо в функцію нічого не передається), або фактичні параметри (константи або раніш визначені змінні, змінні з перетворенням типу, вирази).

При звернення до функції *формальні параметри* змінюються на *фактичні параметри* в строгій відповідності до типів параметрів по їх взаємному розташуванню в списках. Наприклад:

```
int max(int x, int y)
{if (x>y) return x; return y;}
void main()
{int a=10,res1,res2; float f=55.9;
 res1=max(a,20); // res1=20
 res2=max(int(f),a+30);} // res2=55
```

Прототип функції. Якщо функція визначена після функції **main()** або в іншому файлі, то до першого звернення до функції вона має бути описана. Опис функції називається прототипом. Прототип – це заголовок функції, наприкінці якого обов'язкова крапка з комою. Прототип потрібен для перевірки відповідності типів та кількості переданих у функцію параметрів і типів повернених значень. У прототипі можна не вказувати імена формальних параметрів. Приклади прототипів:

```
float sum(float a, float b, float c);
int max(int, int);
```

Всі параметри функції, крім покажчика і масиву, передаються за значенням. Тобто, для переданих (фактичних) параметрів в функції створюються копії цих параметрів, під які в стеку виділяється пам'ять. Дії, що виконуються над цими копіями в тілі функції, не впливають на значення фактичних параметрів поза функцією.

При передаче в функцію покажчиків (адрес фактичних параметрів) в функції створюються копії цих адрес, і в тілі функції становиться відомо, де знаходяться фактичні параметри, и, тому, їх значення можна змінити через розіменування покажчика.

Функції і масиви. При передачі масиву у функцію в якості параметру вказується ім'я масиву (показчик-константа), тобто передається адреса початку масиву. Тому усі дії усередині функції проводяться саме з елементами масиву-оригіналу. При цьому список формальних параметрів в заголовку функції має дві рівноправні форми:

```
int func1(int arr[]){...}
```

```
чи int func2(int *mass){...}
```

У обох випадках у функції створюється копія показчика відповідного типу, який діє усередині функції як звичайний показчик-змінна і може змінювати своє значення. Наприклад, **arr++**.

Доступ до елементів масиву усередині функції можна виконувати за допомогою індексів та за допомогою операції розіменування:

```
arr[0], arr[i], *(arr+0), *(arr+i).
```

Для роботи з усіма елементами масиву необхідно знати кількість елементів в масиві. Є два способи визначення кінця масиву:

- 1) в якості одного з параметрів у функцію передається розмір масиву;
- 2) останній елемент масиву задається будь-яким унікальним значенням, яке аналізують у функції.

Приклад. Функція обчислює суму елементів масиву.

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

```
int sum(int x[],int N) // N визначає кількість елементів в масиві
{for(int S=0,i=0;i<N;i++)
```

```
    S+=x[i];
```

```
    return S;}
```

```
void main()
```

```
{int a[5]={1,3,5,10,2},c[3]={2,1,6}, res1,res2;
```

```
    res1=sum(a,5); // виклик функції
```

```
    res2=sum(c,3); // виклик функції
```

```
    printf("res1=%d res2=%d\n", res1,res2);} // 21 9
```

Приклад. Функція обчислює кількість символів в слові.

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

```
int len(char *c) // визначення функції
```

```
{int i=0;
```

```
    while(c[i++]!='\0'); // ознака кінця рядка символ '\0'
```

```
    return i-1; } // довжина рядка без урахування символу '\0'
```

```
void main()
```

```
{char *name="TERMINATOR", str[6]="Рядок";;
```

```
    printf("Довжина 1 рядка=%d", len(name)); // 10
```

```
    printf("Довжина 2 рядка=%d", len(str)); } // 5
```

Функції з параметрами за умовчанням. Це функції, що мають в специфікації формальних параметрів значення за умовчанням. При зверненні до такої функції параметри за умовчанням можна не вказувати. Але якщо параметр має значення за умовчанням, то всі параметри, що йдуть за ним, повинні мати теж значення за умовчанням.

Наприклад:

```
int def(int a,int b=3,int c=4)
{return a*b+c;}
void main()
{int am=2,bm=5,cm=6,dm1,dm2,dm3;
 dm1=def(am);           // a=2 b=3 c=4 dm1=2*3+4=10
 dm2=def(am,bm);        // a=2 b=5 c=4 dm2=2*5+4=14
 dm3=def(am,bm,cm);     // a=2 b=5 c=6 dm3=2*5+6=16
 printf("dm1=%d dm2=%d dm3=%d\n",dm1,dm2,dm3);
}
```

Функції із змінним числом параметрів. Це функції, в яких кількість і типи параметрів визначаються тільки під час їх виклику.

Синтаксис визначення прототипу таких функцій :

тип ім'я(специфікація явних параметрів, ...);

Остання кома необов'язкова. Три крапки повідомляють, що подальший контроль відповідності кількості параметрів і їх типів при обробці виклику не треба проводити. Перехід в списку від параметра до параметра здійснюється за допомогою покажчика, тип якого є типом змінної. При збільшенні покажчика на одиницю здійснюється перехід до наступної змінної такого ж типу. У C++ функції із змінним числом параметрів не працюють з типами **char**, **unsigned char**, **float**.

У таких функціях існують два способи визначення кількості параметрів: 1) передача у функцію в списку фактичних параметрів інформації про їх кількість; 2) додавання параметра-індикатора в кінець списку фактичних параметрів.

Приклади прототипів:

```
double sumd(int i,double d...); // i – кількість елементів в списку
double mult(double arg, ...); // аналізується значення 0.0
```

Приклад. У функцію передається в списку явних параметрів кількість елементів, які підсумовуються.

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

```
double sumd(int i,double d...)//1-й параметр - число змінних
{double *ptr=&d; //покажчик на 2-й параметр типу double
  double sum=0.0;
  for(;i;i--) // 1-й параметр - кількість змінних в списку
    sum+=*(ptr++);
  return sum;}
void main()
{double d1=5.7, d2=1.2, s1,s2;
  s1=sumd(2,d1,d2); //сума 2-х чисел
  s2=sumd(4, d1, 1.25, d2, 2.5); //сума 4-х чисел
  printf("s1=%.21f s2=%.21f\n",s2,s3);}
```

Рекурсивна функція – це така функція, яка містить в собі виклик цієї ж функції. Рекурсивний підхід слід уникати, якщо є очевидне ітераційне рішення.

Приклад. Обчислення факторіалу додатного цілого числа

```
int fact(int k)
{if(k<0) return 0;
  if(k==0) return 1;
  return k* fact(k - 1);}
void main()
{int n=5;
  printf("%d!=%d", n, fact(n));} // 5!=120
```

Функція, що підставляється (inline -функція) – це функція, тіло якої підставляється при кожному її виклику. Така функція визначається за допомогою слова **inline**.

Inline-функція обробляється компілятором, як така, що вбудовується, тобто замість кожного виклику цієї функції компілятор спробує підставити тіло функції. При багатократних викликах код програми може значно збільшитися, але швидкість роботи програми підвищиться за рахунок відсутності викликів цієї функції.

Для **inline-функції** прийнято: 1) вона не має бути рекурсивною; 2) функція не повинна містити операторів **for**, **while**, **do**, **switch**, **goto**. У цих випадках компілятор видає попередження, і функція трактуватиметься як звичайна.

Наприклад.

```
inline float mod(float x, float y)
{return sgrt(x*x+y*y);}
void main()
{float f1=4.5, f2=-2.5, f3=-1.5, res1, res2;
```

```

    res1=mod(f1,f2); // 5.1478
//res1=sqrt(f1*f1+f2*f2);
    res2=mod(f1, f3); }// 4.743
//res2=sqrt(f1*f1+f3*f3); }

```

Передача параметрів у функцію main(). В функцію `main()` як і в будь-яку іншу функцію можуть передаватися параметри. Для цього функцію `main()` треба описати у формі:

[тип] main(int argc, char **argv) {...}

чи **[тип] main(int argc, char **argv, char **envp) {...}**

В першому випадку функція `main()` буде мати доступ до слів командного рядка, а в другому - ще і до рядків середовища програми, що визначає деякі особливості операційної системи, файлової системи, шляху пошуку (**path**) і ін.

Параметр **[тип]** може бути `int` чи `void`.

Параметр `int argc` визначає кількість слів у командному рядку.

Параметр `char **argv` – покажчик на масив слів з командного рядка. Розмір масиву дорівнює `argc`. Нульове слово, на яке вказує `argv[0]`, є специфікацією програми, що запускається на виконання.

Параметр `char **envp` – покажчик на масив слів середовища програми.

Перевантажені функції. Це функції, що мають одне спільне ім'я, але відрізняються за сигнатурами (за типами та за кількістю переданих в функцію параметрів). Для забезпечення перевантаження необхідно для кожного імені визначити, скільки різних функцій пов'язано з ним. Розпізнавання таких функцій при виклику виконується тільки за сигнатурою (списком параметрів функції).

Наприклад, функції з ім'ям `sum` обчислюють суму двох або трьох параметрів типу `int`, чи двох параметрів типу `float`.

```

int sum(int x,int y) {return x+y;}
int sum(int x,int y, int z) {return x+y+z;}
float sum(float x, float y) {return x+y;}
void main()
{int a=2,b=5,c=6,d1,d2; float f1=1.5,f2=4.6,f3;
  d1=sum(a,b);           // 2+5=7
  d2=sum(a,b,c);         // 2+5+6=13
  f3=sum(f1,f2);         // 1.5+4.6=6.1
  printf("d1=%d d2=%d f3=%f\n",d1,d2 f3);
}

```

Шаблони функцій. Для автоматизації перевантаження функцій різного виду в C++ введені шаблони функцій. Визначається шаблон сімейства функцій так:

```
template <class type1, class type 2, i т.д.>
```

де **type1**, **type2** і т.д. – формальні параметри шаблону, якими можна позначати тип функції, типи формальних та локальних параметрів в визначенні функції. Наприклад:

```
template <class T>  
T max(T x, T y)  
{if(x>y) return x;  
  return y; }  
void main()  
{int a1=-20,a2=10,a3; float f1=-10.5,f2=-45.8,f3;  
  a3=max(a1,a2);          // T приймає значення int  
  f3=max(f1,f2);          // T приймає значення float
```

Параметри шаблонів мають наступні властивості:

1. Імена параметрів шаблону повинні бути унікальними;
2. Список параметрів шаблону не може бути порожнім;
3. Оголошення кожного параметра повинне починатися із слова **class**;
4. Ім'я параметра шаблону має властивості і права типу, область видимості усе визначення функції;
5. Усі параметри шаблону функції, що використовуються в визначенні функції, повинні обов'язково бути використані в сигнатурі функції.

Показчики на функції. Ім'я функції, за визначенням, є показчик, значенням якого є адреса розміщення функції в пам'яті. Якщо значення цієї адреси присвоїти іншому показчику, то цей показчик можна використовувати для виклику функції.

Показчик на функцію визначається спеціальним чином:

```
тип_функції (*ім'я_показчика) (специфікація_параметрів);  
де тип_функції – тип значення, що повертається функцією;  
специфікація_параметрів – типи формальних параметрів;  
(*ім'я_показчика) – ім'я показчика на функцію в круглих дужках.
```

У визначенні показчика на функцію тип значення, що повертається, і сигнатура функції повинні співпадати з типами і сигнатурами тих функцій, адреси яких передбачається присвоїти показчику.

Приклад.

```
float mul(float x, float y){return x*y;}  
float sum(float x, float y){return x+y;}
```

```

void main()
{float f1=3.0, f2=9.0, f3;
 float(*ptr)(float, float); //покажчик на функцію
 ptr=sum; //покажчику присвоїли ім'я функції sum
 printf("sum=%.1f", (*ptr)(f1,f2)); // sum=12.0
 ptr=mul; //покажчику присвоїли ім'я функції mul
 printf("mul=%.1f", (*ptr)(f1,f2)); }// mul=27.0

```

Функції роботи з файлами. Бібліотечні функції для роботи з файлами поділяються на дві групи: потокові і префіксні. Потокові функції виконують додаткову буферизацію даних.

Функції обміну з файлами працюють в двох режимах: текстовому і двійковому. За замовченням встановлюється текстовий режим. В *текстовому режимі* при запису в файл символ '\n' перетворюється в два символи '\n' та '\0', а при читанні з файлу такі символи вже сприймаються як два символи '\n' и '\0'. Крім того натискання клавіш **Ctrl+Z** (кінець введення рядка) сприймається як символ '\0', який не записується в файл. При обміні з файлом в *двійковому режимі* ніяких перетворень символів не виконується.

В файлах `<fcntl.h>`, `<stdlib.h>` описана спеціальна змінна `_fmode`, може приймати значення `O_BINARY` та `O_TEXT`.

Для кожного файлу, відкритого для доступу через потік, створюється структурна змінна за шаблоном `FILE`, що визначена у файлі `<stdio.h>`. У цій структурі зберігаються різні атрибути і прапори відкритого файлу. Після закриття файлу ця змінна вилучається.

Відкриття і закриття файлу. Для відкриття файлу використовується функція `fopen()`, що повертає покажчик на змінну типу `FILE`, якщо файл успішно відкритий, або `NULL`-покажчик, якщо файл не відкрився. Прототип функції має вигляд

```
FILE *fopen(const char *filename, const char *mode);
```

де **filename** – покажчик на рядок, що задає шлях та ім'я файлу, якщо шлях не задано, то вибирається поточний диск, директорій;

mode – покажчик на рядок символів, що задають режим відкриття.

Деякі комбінації цих символів приведені нижче:

w – потік відкривається тільки для запису у файл, **w+** – потік відкривається для запису у файл і для читання з файлу. Якщо файлу немає, він створюється, якщо файл існує, його старий вміст пропадає;

r – потік відкривається для читання з існуючого файлу, **r+** – потік відкривається для читання і запису в існуючий файл;

a – потік відкривається для додавання (запису даних в кінець файлу),
a+ – потік відкривається для додавання в файл і для читання з файлу.
 Якщо файл не існує, він створюється;
t – потік відкривається в текстовому режимі, задається після **r, w, a**;
b – потік відкривається в двійковому режимі, задається після **r, w, a**.

Закриття файлу виконує функція **fclose()**, що має прототип
int fclose(FILE *fp);

Функція закриває потік, на який вказує **fp**. При успішному закритті файлу функція повертає 0, інакше – **EOF**. Наприклад:

```
FILE *fp=fopen("D:\\User\\myfile.txt","w");
if(fp==NULL){puts("помилка відкриття файлу");return -1;}
...
fclose(fp);
```

Функції запису в файл та читання з файлу. Нижче приводяться функції запису (введення) в файл та читання (виведення) з файлу, які найчастіше застосовуються.

Функція символьного запису в файл

int fputc(int ch, FILE *fp); // прототип

Функція у випадку успіху записує у відкритий файл символ **ch** після його перетворення в тип **char**, інакше – **EOF**. Наприклад:

```
char c1='A'; // визначення та ініціалізація символу
fputc(c1,fp); // запис символу в файл
```

Функція символьного читання з файлу

int fgetc(FILE *fp); // прототип

Функція при успіху повертає прочитаний символ після його перетворення в **int** без врахування знаку. У випадку помилки чи досягнення кінця файлу повертається **EOF**. Наприклад:

```
char c2; // резервування пам'яті під символ
c2=fgetc(fp); // читання символу з файлу
putch(c2); //виведення символу на екран
```

Функція рядкового запису в файл

int *fputs(char *str, FILE *fp); // прототип

Функція переносить байти символів з рядка, на який вказує покажчик **str**. Перенесення закінчується при досягненні символу '**\0**' (кінець рядка), який в файл не записується. У випадку успіху повертає код останнього записаного символу рядка, інакше – **EOF**. Наприклад:

```
char str1[20]="Програмування"; //визначення та ініціалізація рядка
fputs(str1,fp); // запис рядка в файл
```

Функція рядкового читання з файлу

char *fgets(char *str, int N, FILE *fp); // прототип

Функція приймає байти з відкритого файлу і розміщує їх за адресом, що заданий покажчиком **str**. Прийом символів закінчується після передачі **N** байт чи при одержанні символу **'\n'**. У випадку успіху повертає покажчик на прочитаний рядок символів, інакше – **NULL**. Наприклад:

```
char str[21]; // резервування пам'яті під рядок (21 байт)
fgets(str,21,fp); // читання рядка з файлу не більш 21 байтів
puts(str); // виведення рядка на екран
```

Функція форматowanego запису даних в файл

int fprintf(FILE *fp, char *format[,arg]...); // прототип

Функція записує значення даних з перетворенням за вказаним форматом у відкритий файл. У випадку успіху повертає записане число байтів, інакше – повертає **EOF**. Наприклад:

```
int d=7; float f=3.5; char g='B'; //визначення змінних
fprintf (fp,"%d %f %c",d,f,g); // запис змінних в файл
```

Функція форматowanego читання даних з файлу

int fscanf(FILE *fp, char *format[,pointer]...); // прототип

Функція зчитує з відкритого файлу значення даних з перетворенням за вказаним форматом і розміщує їх за вказаними адресами. У випадку успіху повертає кількість оброблених при читанні полів (чисел), інакше – повертає **EOF**. Наприклад:

```
int a; float b; char c; // резервування пам'яті під змінні
fscanf(fp,"%d %f %c",&a,&b,&c); //читання змінних з файлу
printf("%d %f %c\n",a,b,c); //виведення змінних на екран
```

Функція блокового запису даних в файл

int fwrite(void *ptr,int size,int N,FILE* fp); // прототип

Функція виконує запис даних у відкритий файл **N** блоків, розміром **size** байт кожен. Запис виконується у двійковому режимі **"wb"**.

У випадку успіху повертає введене число блоків, інакше – повертає **EOF**.

Наприклад:

```
struct REC{int a; float b[2];};
REC st1={1, 2.2, 3.3}; //визначення структури (блок з 10 байт)
fwrite(&st1,sizeof(REC),1,fp); //запис в файл структури
//визначення масиву структур (три блоки з 10 байт)
REC st2[3]={10, 2.5, 1.5},{30, 4.4, 2.5},{50, 6.6, 3.5}};
fwrite(st2,sizeof(REC),3, fp); //запис в файл масиву структур
```

Функція блокового читання з файлу

int fread(void *ptr, int size, int N, FILE *fp); // прототип
 Функція виконує читання з відкритого файлу **N** блоків, розміром **size** байт кожен. Читання виконується у двійковому режимі "**rb**".

У випадку успіху повертає зчитане число блоків, інакше – повертає **EOF**. Наприклад:

```
REC st3, st4[3]; // резервування пам'яті під блоки даних
fread(&st3, sizeof(REC), 1, fp); // читання з файлу структури
// виведення на екран зчитаної структури
printf("%d %.2f %.2f\n", st3.a, st3.b[0], st3.b[1]);
fread(st4, sizeof(REC), 3, fp); // читання з файлу масиву структур
// виведення на екран зчитаного масиву структур
for(int i=0; i<3; i++)
printf("%d %.2f %.2f\n", st4[i].a, st4[i].b[0], st4[i].b[1]);
```

Довільний доступ до файлу При обміні даними з файлом (запис і читання) місце, куди записується або прочитується інформація, визначається показчиком запису-читання. Раніше розглядалися функції обміну з файлом, які здійснювали послідовний доступ до файлу. У бібліотеці мови Сі є функції, що дозволяють виконати довільний доступ до файлу, тобто доступ в будь-яку точку файлу. Ці функції керують положенням показчика запису-читання.

Функція установки показчика запису-читання на початок файлу має прототип **void rewind(FILE *fp);**

Наприклад: **rewind(fp);**

Функція установки показчика запису-читання в довільну точку файлу має прототип:

```
int fseek(FILE *fp, long offset, int from_where);
```

Ця функція встановлює показчик запису-читання в потоці **fp** на **offset** байт управо або вліво. Якщо **offset>0**, то показчик зсувається у бік кінця файлу, якщо **offset<0**, то показчик зсувається у бік початку файлу. Параметр **from_where** задає точку відліку для зсуву: **SEEK_SET** – зсув від початку файлу, **SEEK_CUR** – зсув від поточної позиції показчика, **SEEK_END** – зсув від кінця файлу. У разі успіху функція повертає 0, інакше - **EOF**.

Установка показчика запису-читання лівіше початку файлу вважається помилкою, установка показчика правіше за кінець файлу робить значення показчика запису-читання невизначеним.

Наприклад, установка покажчика запису-читання через 10 байт від початку файлу:

```
fseek(fp, 10L, SEEK_SET);
```

Функція визначення поточного положення покажчика запису-читання має прототип:

```
long ftell(FILE *fp);
```

Ця функція повертає значення в байтах (тип long) від початку файлу до поточного положення покажчика запису-читання файлу, відкритого через потік **fp**. У разі помилки функція повертає **-1L**. Наприклад:

```
long cur=ftell(fp);
```

Багатофайловий проект. Частіше програми створюються з декількох файлів, що містять згруповані за призначенням різні функції. При цьому в одному з файлів міститься головна функція **main()**, в інших - другі необхідні функції, дані й бібліотеки функцій.

Найпростіший спосіб збирання файлів у єдину програму, це створити програму з декількох файлів, використовуючи препроцесорну директиву **#include**, за допомогою якої в файл з головною функцією підключаються усі необхідні файли.

При об'єднанні декількох файлів в один проект слід пам'ятати:

- 1) всі функції мають клас пам'яті **extern**, і вони доступні у всіх файлах одного проекту. Але для використання функцій, що визначені в інших файлах чи нижче першого звертання до неї, необхідно їх описати (вставити в код програми прототип цієї функції);
- 2) об'єкти (змінні, масиви, структури і т.д.), що визначені в файлах як глобальні (поза функцій), доступні у всіх файлах одного проекту. Але для використання їх в програмі (функції) раніш визначення, ці об'єкти необхідно описати поза функцій з специфікатором **extern**.

Кожен об'єкт може бути визначений тільки один раз, але може бути неодноразово описаний.

Можна помістити прототипи функцій, оголошення об'єктів в окремий файл з розширенням **.h**, та включати цей файл в початок інших файлів за допомогою оператора **#include**.

ЛАБОРАТОРНІ РОБОТИ

ЛАБОРАТОРНА РОБОТА № 1

Елементарне введення у функції

Мета роботи – вивчити елемент мови C(C++) – функції, навчитися створювати і використовувати їх.

Програма 1_1. В програмі показана робота найпростіших функцій.

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

int mul(int arg1,int arg2) // визначення функції mul
{return arg1*arg2; }

float func(float x,float y) // визначення функції func
{if(x!=0.0||y!=0.0) return 1/(x*x+y*y);
 else {puts("It is error!"); return -1;}
}

double epsilon() // визначення функції epsilon
{double eps=1.0,eps1;
 do{eps*=0.5;
  eps1=eps+1.0;
 }while(eps1>1.0);
 return eps;
}

int main()
{int a=2,b=3,c=5,d=6,k1,k2;
 puts("\nfunction 1:");
 k1=mul(a,b); // 1-й виклик функції mul
 printf("a=%d b=%d k1=%d\n",a,b,k1);
 k2=mul(c,d); // 2-й виклик функції mul
 printf("c =%d d =%d k2=%d\n", c,d,k2);
 float f1=0.5, f2=0.2, f3=0.0,r1,r2;
 puts("\nfunction 2:");
 r1=func(f1,f2); // 1-й виклик функції func
 printf("f1=%.2f f2=%.2f r1=%.2f\n",f1,f2,r1);
 r2=func(f2,f3); // 2-й виклик функції func
 printf("f2=%.2f f3=%.2f r2=%.2f\n",f2,f3,r2);
 puts("\nfunction 3:");
 double E=epsilon(); // виклик функції epsilon
 printf("E=%le\n",E);
 system("pause"); return 0;}
```

Завдання до програми 1_1. Розробіть задані згідно варіанту функції та зробіть в головній програмі по два виклики цих функцій з різними параметрами. Для першого виклику функцій фактичні параметри проініціалізуйте, для другого – введіть з клавіатури.

1. Функція типу **int**, що повертає більше з трьох цілих чисел, та функція типу **float**, що повертає суму двох вхідних параметрів, якщо другий параметр додатний, інакше повертає різницю цих параметрів.
2. Функція типу **int**, що повертає суму остач від ділення першого та другого параметрів на 5, якщо суму більша 4, інакше повертає число 0, та функція типу **double**, що повертає мінімальне з двох вхідних параметрів.
3. Функція типу **float**, що повертає абсолютне значення різниці першого та другого аргументів, та функція типу **int**, що повертає мінімальне з трьох цілих чисел.
4. Функція типу **long**, що повертає добуток двох чисел типу **long**, якщо друге число менше 100, інакше повертає число 100, та функція типу **float**, що повертає середнє арифметичне трьох чисел.
5. Функція типу **int**, що повертає суму трьох вхідних параметрів, якщо значення першого параметру непарне, інакше повертає число 50, та функція типу **float**, що повертає різницю квадратів двох вхідних параметрів.

Програма 1_2. В програмі функція визначає суму елементів масиву. При цьому у функцію передається покажчик на масив та розмір масиву. Функція визначається наприкінці файлу. Компілятору повідомляється про подальше визначенні функції за допомогою прототипу функції.

```
#include <stdio.h>
#include<iostream>
#include <stdlib.h>
#include<time.h>

float summa(float[], int); // визначення прототипу

int main()
{float res1,mas1[]={3.36,2.7,7.51,1.5};
  int i,n;
  n=sizeof(mas1)/sizeof(mas1[0]);
  puts("  Massiv 1");
  for(i=0;i<n;i++)
```

```

    printf("%.2f  ",mas1[i]);
    res1=summa(mas1,n); // 1-й виклик функції
    printf("summa1=%.1f\n",res1);
    float res2,mas2[10];
    srand((unsigned)time(NULL)); //запуск генератора випадк. чисел
    puts("  Massiv 2");
    for(i=0;i<10;i++)
    {mas2[i]=rand()%101*0.1;
      printf("%.2f  ",mas2[i]);
    }
    res2=summa(mas2,10); // 2-й виклик функції
    printf("\nsumma2=%.1f\n",res2);
    system("pause");
    return 0;
}

float somma(float arr[],int size) // визначення функції
{float sum=0.0;
  for(int i=0;i<size;i++)
    sum+=arr[i];
  return sum;
}

```

Завдання до програми 1_2. Розробіть задану згідно варіанту функцію та зробіть в головній програмі три виклики функції з різними параметрами. При першому виклику значення елементів масиву проініціалізуйте, при другому – задайте випадковими числами, при третьому – введіть з клавіатури. Всі масиви повинні мати різний розмір.

1. Функція, в яку передається масив типу **double**, що повертає середнє арифметичне елементів масиву.
2. Функція, в яку передається масив типу **int**, що повертає суму додатних парних елементів масиву.
3. Функція, в яку передається масив типу **float**, що повертає добуток додатних елементів масиву, що менші 10.0.
4. Функція, в яку передається масив типу **long**, що повертає суму від'ємних непарних елементів масиву.
5. Функція, в яку передається масив типу **float**, що повертає добуток від'ємних елементів масиву, що більші -10.0.

Програма 1_3. В програмі функція визначається індекс максимального елементу масиву. При цьому у функцію передається покажчик на масив та розмір масиву.

```

#include <stdio.h>
#include<iostream>
#include <stdlib.h>
#include<time.h>

int max_ind(float[], int); // визначення прототипу

int main()
{float m1[6]={-3.36,132.7,770.51,31.5,-23.8,67.45};
  int i,ind1,ind2;
  puts("  Massiv 1");
  for(i=0;i<6;i++)
    printf("%.2f  ",mas1[i]);
  ind1= max_ind(m1,6); // 1-й виклик функції
  printf("\nindex_ max1=%d max1=%.2f\n",ind1,m1[ind1]);
  float m2[5];
  puts(" Vvedite massiv 2");
  for(i=0;i<5;i++)
    scanf("%f",&m2[i]);
  puts(" Massiv 2");
  for(i=0;i<5;i++)
    printf("%.2f  ",mas2[i]);
  printf("\nindex_ max2=%d max2=%.2f\n",ind2,m2[ind2]);
  system("pause");
  return 0;
}

int max_ind(float arr[],int size) // визначення функції
{int index=0;
  for(int i=0;i<size;i++)
    if(arr[i]>arr[index]) index=i;
  return index;}

```

Завдання до програми 1_3. Розробіть задану згідно варіанту функцію та зробіть в головній програмі два виклики функції з різними параметрами. При першому виклику значення елементів масиву проініціалізуйте, при другому — введіть з клавіатури. Всі масиви повинні мати різний розмір.

1. Функція, в яку передається масив типу **double**, що повертає кількість додатних елементів масиву, що менші 10.5.
2. Функція, в яку передається масив типу **int**, що повертає кількість додатних парних елементів масиву.
3. Функція, в яку передається масив типу **float**, що індекс максимального елементу масиву.

4. Функція, в яку передається масив типу **long**, що повертає кількість від'ємних непарних елементів масиву.
5. Функція, в яку передається масив типу **char**, що повертає індекс мінімального елемента цього масиву.

Програма 1_4. В програмі визначається функція з параметрами за умовчанням.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int bigger(int var,int n=10) // визначення функції
{if(var<n) return n;
 return var;
}
int main()
{int arr=8, barr, arr1, barr1;;
 puts("max=10");
 barr=bigger(arr); // виклик функції, значення n=10
 printf("arr=%d barr=%d\n",arr, barr);
 srand((unsigned)time(NULL)); //запуск генератора випадк. чисел
 puts("\nmax=50");
 arr1=rand()%101;
 barr1=bigger(arr1,50); // виклик функції, значення n=50
 printf("arr1=%d barr1=%d\n",arr1,barr1);
 system("pause");
 return 0;}
```

Завдання до програми 1_4. Розробіть задану згідно варіанту функцію з трьома входними параметрами типу **float** **x**, **a**, **b**, два з яких (**a**,**b**) мають значення за умовчанням. Продемонструйте роботу цієї функції, зробивши в головній програмі три виклики функції з різною кількістю параметрів. Параметри для виклику проініціалізуйте. Запишіть коментарі до отриманих результатів.

1. Функція, якщо **a** додатне, то повертає значення **x*a**, інакше повертає значення **x*b**.
2. Функція, якщо **a** від'ємне, то повертає значення **x*x-a**, інакше значення **x*x-b**.
3. Функція, якщо **a** більше **b**, то повертає значення **x+a**, інакше значення **x+b**.

4. Функція, якщо x більше a , то повертає значення виразу $(x+a*b)$, інакше значення $(x-a*b)$.
5. Функція, якщо x менше b , яка повертає значення виразу $(x*a-b)$, інакше значення $(x*b-a)$.

Програма 1.5 Показана робота функцій зі змінним числом параметрів. В першій функції за допомогою першого параметра передається до неї значення кількості параметрів, що йдуть за першим. Ця функція повертає суму другого і наступних параметрів. Друга функція обчислює добуток аргументів, а ознакою закінчення списку параметрів є деяке визначене значення параметра.

```
#include<stdio.h>
#include<stdlib.h>
#include<iostream>
int varg1(int i,...) // визначення функції varg1
{int *num_arg=&i;
 int sum=0;
 for(;i;i--)
  sum+=*(++num_arg);
 return sum;}
double varg2(double arg,...) // визначення функції varg2
{double mul=1.0;
 double *ptr=&arg;
 if(*ptr==0.0) return 0;
 for(;*ptr!=999.0;ptr++) mul*=*ptr;
 return mul;}
int main()
{int a=11,b=22,c=33,sum;
 double mul,x=2.25,y=4.4,z=5.0,end=999.0;
 sum=varg1(2,a,b); // виклик функції
 printf("sum(a,b)=%d\n",sum);
 int j=3;
 sum=varg1(j,a,b,c); // виклик функції
 printf("sum(a,b,c)=%d\n",sum);
 sum=varg1(6,a,b,c,100,200,700); // виклик функції
 printf("sum(...)=%d\n",sum);
 mul=varg2(x,y,z,end); // виклик функції
 printf("mul(x,y,z)=%.21f\n",mul);
 mul=varg2(x,y,z,1.5,4.2,end); // виклик функції
 printf("mul(...)=%.21f\n",mul);
 system("pause"); return 0;}
```

Завдання до програми 1.5 Розробіть задану згідно варіанту функцію зі змінним числом параметрів та продемонструйте роботу цієї функції, зробивши в головній програмі кілька викликів функції з різними значеннями. Всі параметри, крім розміру, мають тип **double**.

1. Функція обчислює суму першого параметра і добуток параметрів, що йдуть за другим: $F = \text{arg1} + \text{arg3} * \text{arg4} * \dots$, загальна кількість параметрів передається за допомогою другого параметру (тип **int**).
2. Функція обчислює добуток першого параметра і суми параметрів, що йдуть за першим: $F = \text{arg1} * (\text{arg2} + \text{arg3} + \text{arg4} + \dots)$, ознакою закінчення списку параметрів є деяке визначене значення параметра.
3. Функція обчислює суму додатних параметрів типу **double**, загальна кількість параметрів передається за допомогою першого параметру (тип **int**).
4. Функція обчислює добуток додатних параметрів, ознакою закінчення списку параметрів є деяке визначене значення параметра.
5. Функція обчислює суму другого параметра і добуток параметрів, що слідують за другим: $F = \text{arg2} + (\text{arg3} * \text{arg4} * \dots)$, загальна кількість параметрів передається за допомогою першого параметру (тип **int**).

Контрольні питання до лабораторної роботи № 1

1. Приведіть основні складові визначення функції.
2. Які параметри називають фактичними, а які формальними?
3. Що таке прототип функції? Для чого та коли він використовується?
4. Для чого використовується оператор **return** у функції? Скільки таких операторів може мати функція?
5. В чому різниця передачі параметрів за значенням та за покажчиком?
6. Як передати в функцію масив? Приведіть два способи визначення кінця масиву. Привести приклади функцій.
7. Дайте визначення та приведіть приклад *прототипу* функції з параметрами за умовчанням.
8. Дайте визначення та приведіть приклад *прототипу* функції з змінним числом параметрів. Які типи даних не можна використовувати в таких функціях?

ЛАБОРАТОРНА РОБОТА № 2

Прийоми використання функцій

Мета роботи – познайомитися і зрозуміти деякі сучасні засоби мови C++, такі як перевантаження функцій, шаблони функцій і покажчики на функції.

Програма 2.1. В програмі є три функції з однаковими іменами, при компіляції компілятор розпізнає перевантажені функції по їх сигнатурах (у даному випадку за типами параметрів).

```

#include <stdio.h>
#include <stdlib.h>
#include<iostream>
char func1(char arr[],int n)//визначення 1 перевантаженої функції
{char sum=0;
  for(int i=0;i<n;i++)  sum+=arr[i];
  return sum; }
int func1(int arr[],int n)//визначення 2 перевантаженої функції
{int sum=0;
  for(int i=0;i<n;i++)  sum+=arr[i];
  return sum;}
float func1(float arr[],int n)//визначення 3 перевантаж. функції
{float sum=0.0;
  for(int i=0
;i<n;i++)  sum+=arr[i];
  return sum;}
int main()
{char resch,ch[5]={4,20,2,5,10};
  int resint,set[8]={2,3,5,7,9,11,25,1};
  float resfl,mass[6]={2.1,3.1,4.1,5.1,6.1,7.1};
  resch=func1(ch,5);    // виклик 1 перевантаженої функції
  printf("Sum of elements of array char=%d\n", resch);
  resint=func1(set,8);  // виклик 2 перевантаженої функції
  printf("Sum of elements of array int=%d\n", resint);
  resfl=func1(mass,6);  //виклик 3 перевантаженої функції
  printf("Sum of ele
ments of array float=%.2f\n", resfl );
  system("pause");  return 0;}

```

Завдання до програми 2.1 Розробіть задані згідно варіанту перевантажені функції та продемонструйте роботу цих функцій, зробивши кілька викликів функції з різними значеннями. Змінні та масиви проініціалізуйте. Масиви повинні бути різного розміру.

1. Функції типу **int**, **long**, **float**, що повертають суму тільки значень чотирьох чисел.
2. Функції, що повертають кількість додатних чисел в масивах типу **int**, **float**, **double**.
3. Функція типу **float**, що повертають середнє арифметичне двох, трьох або чотирьох чисел.
4. Функції типу **int**, **float**, **double**, що повертають мінімальний елемент масиву.
5. Функції типу **int**, **long**, **float**, що повертають максимальне з трьох чисел.

Програма 2.2. В програмі створюється шаблон функції для оцінки дисперсії результатів вимірів еталонної величини, заданих як масив цілих і дійсних чисел. При компіляції програми формуються дві функції, відповідно для масиву типу **int** й масиву типу **double**.

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
template <class A> //визначення шаблону функції
double sigma(int m,A arr[],A e) //визначення функції
{A tmp,sum=0;
  for(int i=0;i<m;i++)
    {tmp=(arr[i]-e);
     sum+=tmp*tmp; }
  return (double)sum/m;}
int main()
{int iarr[]={20,21,21,20,18,19,19,20,21,21,22,21};
 double darr[6]={21.3,21.4,20.9,20.8,20.9,21.1};
 double sg1, sg2;
 int sz=sizeof(iarr)/sizeof(iarr[0]);
 sg1=sigma(sz,iarr,21); //виклик функції, A замінюється на int
 printf("sg1=%.21f\n",sg1);
 sg2=sigma(6,darr,21.5); //виклик функції, A замінюється на double
 printf("sg2=%.21f\n",sg2);
 system("pause");
 return 0;}
```

Завдання до програми 2.2 Розробіть згідно варіанту шаблон для функцій та продемонструйте роботу цих функцій, зробивши в головній програмі кілька викликів функції з різними значеннями. Масиви повинні мати різний розмір.

1. Функції типу **int**, **float**, **double**, що повертають максимальний елемент масиву.
2. Функції типу **int**, **float**, **long**, що повертають мінімальне з трьох чисел.
3. Функції, що повертають кількість від'ємних чисел в масивах типу **int**, **float**, **double**.
4. Функції типу **int**, **long**, **char**, що повертають суму тільки парних значень з чотирьох чисел.
5. Функції, що повертають індекс мінімального елемента в масивах типу **int**, **float**, **double**.

Програма 2.3. В цій програмі реалізовано приклад застосування шаблону функції. Функція здійснює обмін значень вмісту двох покажчиків. Компілятор створить відповідну функцію в залежності від переданих до неї параметрів.

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
template <class P> // визначення шаблону функції
void swap(P *, P *); // визначення прототипу функції
int main()
{char c1='A', c2='B';
 int i1=10, i2=20;
 puts("before swap()");
 printf("&c1=%p c1=%c  &c2=%p c2=%c\n", &c1, c1, &c2, c2);
 printf("&i1=%p i1=%d  &i2=%p i2=%d\n", &i1, i1, &i2, i2);
 swap(&c1, &c2); //виклик функції
 swap(&i1, &i2); //виклик функції
 puts("after swap()");
 printf("&c1=%p c1=%c  &c2=%p c2=%c\n", &c1, c1, &c2, c2);
 printf("&i1=%p i1=%d  &i2=%p i2=%d\n", &i1, i1, &i2, i2);
 system("pause");
 return 0;}
template <class T>
void swap(T *arg1, T *arg2) //визначення функції
{T tmp=*arg1;
 *arg1=*arg2;
 *arg2=tmp;
}
```

Завдання до програми 2.3 Доповніть програму, застосувавши наведений в програмі шаблон функції для обміну значень між двома об'єктами, заданими згідно варіанту. Виведіть на екран адреси переданих покажчиків, вміст цих покажчиків та значення об'єктів.

1. Визначте та проініціалізуйте дві структури одного типу, поміняйте значення цих структур.
2. Визначте і проініціалізуйте два масиви типу **int**, визначте покажчики на ці масиви, поміняйте значення цих покажчиків.
3. Визначте та проініціалізуйте два об'єднання одного типу, поміняйте значення цих об'єднань.
4. Визначте та проініціалізуйте дві змінні типу **float**, визначте два покажчики на ці змінні, поміняйте значення цих покажчиків.
5. Визначте та проініціалізуйте два покажчика на рядки, поміняйте значення цих покажчиків.

Програма 2.4 В програмі демонструється передача аргументів у функцію **main (int argc, char **argv, char**envp)**. Для роботи програми її необхідно запустити з командного рядка, наприклад, у середовищі **FAR, NC, VC** і ін.

```
#include <stdio.h>
#include <stdlib.h>
#include<iostream>
int main(int argc, char **argv, char **envp)
{for(int i=0;i<argc;i++)
    printf("argv[%d]=%s\n",i,argv[i]);
    system("pause");
    while(*envp) puts(*envp++);
    system("pause");
    return 0; }
```

Завдання до програми 2.4 Розробіть програму з використанням передачі аргументів в головну функцію з командного рядка. В функцію **int main(int argc, char**argv)** з командного рядка передаються заданий згідно варіанту пароль, число та інші повідомлення. Якщо пароль правильний, то на екран виводиться вся інформація з командного рядка, прізвище і ім'я студента та таблиця множення на задане число, інакше виводиться повідомлення про неправильний пароль.

1. Місяць народження студента.
2. Назва міста, в якому проживає студент.

3. Ім'я улюбленої кінозірки.
4. Назва улюбленої спортивної ігри.
5. Улюблена пора року.

Програма 2.5. В цій програмі при розробці "універсальної" функції сумування використовується в якості параметру покажчик на функцію. При цьому застосовується тип покажчика на функцію. Функція сумує спочатку куби елементів масиву, а потім - їх квадрати.

```
#include <stdlib.h>
```

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

```
float cube(float y) {return y*y*y;}
```

```
float sqr(float e) {return e*e;}
```

```
typedef float (*PF)(float); //визначення типу покажчика на функцію
```

```
float summa(PF pfl, float a[], int N) //визначення функції
```

```
{float sum=0.0;
```

```
//обчислення суми квадратів (кубів) елементів масиву
```

```
//за допомогою покажчика на задану функцію
```

```
for(int i=0;i<N;i++)
```

```
sum+=(*pfl)(a[i]);
```

```
return sum;}
```

```
int main()
```

```
{float res1, res2, res3, A[3]={1.5,2.0,3.0};
```

```
res1=summa(cube,A,3); // виклик функції
```

```
printf("res1=%.3f\n",res1);
```

```
res2=summa(sqr,A,3); //виклик функції з покажчиком на функцію
```

```
printf("res2=%.3f\n",res2);
```

```
system("pause");
```

```
return 0;}
```

Завдання до програми 2.5 Розробіть програму використання покажчика на функцію, як параметра функції, згідно заданому варіанту. Зробіть в головній програмі кілька викликів функції з різними значеннями. Всі параметри, крім розміру масивів, мають тип **float**.

1. Визначте дві функції, що обчислюють максимальне та мінімальне з трьох чисел. Потім визначте функцію, що помножує суму заданих трьох чисел на максимальне (мінімальне) число, яке обчислюється за допомогою покажчика на задану функцію.
2. Визначте дві функції, що обчислюють максимальний та мінімальний елемент масиву. Потім визначте функцію, що обчислює суму

елементів масиву без обліку максимального (мінімального) числа, яке обчислюється за допомогою покажчика на задану функцію.

3. Визначте дві функції, що обчислюють суму та добуток трьох чисел. Потім визначте функцію, що помножає максимальне з трьох чисел на суму чи добуток, які обчислюються за допомогою покажчика на задану функцію.

4. Визначте дві функції, що обчислюють максимальний (мінімальний) елемент масиву. Потім визначте функцію, що суму перших 3-х елементів масиву помножує на максимальний (мінімальний) елемент, який обчислюється за допомогою покажчика на задані функції.

5. Визначте дві функції, що обчислюють суму (добуток) перших 2-х елементів масиву. Потім визначте функцію, що суму чи добуток, які обчислюються за допомогою покажчика на задані функції, ділить на максимальний елемент масиву.

Контрольні питання до лабораторної роботи № 2

1. Що таке перевантаження функцій? Для чого воно застосовується?
2. Наведіть приклади прототипів перевантажених функцій із сигнатурами, що відрізняються за кількістю параметрів та за типом параметрів.

3. Чи є помилки при визначенні прототипів наступних перевантажених функцій? Поясніть.

```
float func1(int, float);
float func1(int, int);
int func1(int, float);
```

4. Дайте визначення шаблону функції та параметру шаблону.

5. Приведіть правила визначення і використання шаблонів функції.

6. Чи є помилки в наступному запису? Поясніть.

```
template <class D, C>
D func2(int, C);
```

7. Що таке покажчик на функцію? Приведіть основні правила його визначення.

8. Визначте покажчики на функції, що мають наступні прототипи:

```
float F1(int n, float arr[]);
float *F2(int n, char *st);
```

9. Як та які параметри передаються до функції `main()`?

ЛАБОРАТОРНА РОБОТА №3

Робота з файлами

Мета роботи – вивчити основні функції роботи з файлами.

Програма 3.1. Показано роботу функції відкриття файлу для запису та запису в нього окремих символів. Після виконання програми перевірте наявність та зміст файлу `testfile.txt` у своєму каталозі.

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
int main()
{FILE *fout, *fin;
  fout=fopen("test1.txt", "w"); // відкриття потоку для запису
  if (fout==NULL)
    {puts("Cannot open output file");
     return 1;}
  char ch1; // визначення символу
  puts("Type any symbol or Ctrl-Z");
  while((ch1=getc(stdin)) != EOF) //введення символу з клавіатури
    fputc(ch1, fout); // запис символу в файл
  fclose(fout); // закриття потоку
  fin=fopen("test1.txt", "r"); // відкриття потоку для читання
  if (fin==NULL)
    {puts("Cannot open input file");
     return 1; }
  char ch2; // резервування пам'яті під символ
  puts("\nSymbol in file");
  while((ch2=fgetc(fin)) != EOF) // читання символу з файлу
    putc(ch2, stdout); // виведення символу на екран
  fclose(fin); // закриття потоку
  system("pause");
  return 0;}
```

Завдання до програми 3.1 Розробіть програму, що відкриває файл з іменем, що складається з перших чотирьох букв Вашого прізвища з розширенням `txt`. Запишіть задані згідно варіанту символи у відкритий файл за допомогою функції `fputc()`. Перевірте правильність запису, для цього прочитайте з файлу записані символи за допомогою функції `fgetc()` та виведіть їх на екран.

1. Запишіть у вигляді символів рядок, який проініціалізуєте Вашим прізвищем.

2. Запишіть 10 символів, що вводяться з клавіатури за допомогою функції `getche()`.
3. Запишіть перші вісім букв латиського алфавіту.
4. Запишіть символи, що вводяться з клавіатури за допомогою функції `scanf()` до першого введення символу 'q'.
5. Запишіть символьний масив, який проініціалізуєте Вашим ім'ям.

Програма 3.2. У програмі використовуються функції файлового рядкового введення-виведення.

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
int main()
{FILE *fin, *fout;
  const int N=30;
  char filename[12]="test2.txt", string[N], str[N+1];
  fout=fopen(filename,"w"); // відкриття потоку для запису
  if(fout==NULL)
    {puts("Cannot open output file");
     return 1; }
  puts("Input string <= N simbols");
  gets(string); // введення рядка з клавіатури
  fputs(string,fout); // запис рядка в файл
  fputs("\n",fout); // запис "\n" в файл
  fclose(fout); // закриття потоку
  fin=fopen(filename,"r"); // відкриття потоку для читання
  if(fin==NULL)
    {puts("Cannot open input file");
     return 2; }
  puts("\nSimbols of string in file:");
  fgets(str,N+1,fin); // читання рядка з файлу не більш N байтів
  puts(str); // виведення рядка на екран
  fclose(fin); // закриття потоку
  system("pause");
  return 0;}
```

Завдання до програми 3.2 Розробіть програму, що відкриває файл з іменем, що складається з Ваших ініціалів. Запишіть задані згідно варіанту рядки у відкритий файл за допомогою функції `fputs()`. Перевірте правильність запису, для цього прочитайте з

файлу записані рядки за допомогою функції **fgets()** та виведіть їх на екран.

1. Визначте двовимірний масив розміром 5 на 10 чисел типу **char**, введіть з клавіатури 5 рядків символів. Запишіть ці рядки у файл.
2. Визначте масив рядків та проініціалізуйте його назвами дисциплін, з яких треба скласти іспити. Запишіть цей масив у файл.
3. Визначте двовимірний масив розміром 4 на 5 чисел типу **char**, проініціалізуйте знаками операцій. Останні елементи рядків – символ **'\0'**. Запишіть цей масив у файл.
4. Визначте масив рядків та проініціалізуйте Вашими прізвищем, ім'ям та по батькові. Запишіть цей масив у файл.
5. Запишіть рядки, що вводять з клавіатури. Кількість рядків задається.

Програма 3.3. У програмі використовуються функції форматованого файлового введення-виведення.

```
#include <stdio.h>
#include<iostream>
#include <stdlib.h>
#include <time.h>
int main()
{FILE *fp;
  char *filename="numbers.txt";
  float num,sqnum,num1,sqnum1;
  int i,i1, N=10;
  if((fp=fopen(filename,"w"))==NULL)
    {printf("Cannot open output file\n");
     return 1; }
  srand((unsigned)time(NULL)); // запуск генератора випадк. чисел
  puts("Numbers:");
  for(i=0;i<N;i++)
  {num=rand()%101*0.01;
   sqnum=num*num;
   printf("%d %.2f %.4f\n",i+1,num,sqnum); //запис даних
   fprintf(fp,"%d %.2f %.4f\n",i+1,num,sqnum); //запис даних
  }
  fclose(fp); // закриття потоку
  if((fp=fopen(filename,"r"))==NULL)
    {printf("Cannot open input file\n");
     return 3;}
  puts("\nNumbers in the file:");
```

```

for (i=0; i<N; i++)
    { fscanf(fp, "%d %f %f", &i1, &num1, &sqnum1); //читання даних
      printf("%3d %.2f %.4f\n", i1, num1, sqnum1); }
fclose(fp); // закриття потоку
system("pause");
return 0; }

```

Завдання до програми 3.3 Розробіть програму, що відкриває файл з іменем, що складається з перших п'яти літер Вашого імені. Запишіть задані згідно варіанту числа у відкритий файл за допомогою функції **fprintf()**. Перевірте правильність запису, для цього прочитайте з файлу записані числа за допомогою функції **fscanf()** та виведіть їх на екран.

1. Визначте та проініціалізуйте двовимірний масив чисел типу **int** розміром 5 на 4. Запишіть у файл числа та остачі від ділення цих чисел на число 10 (перший рядок складається з першого числа, першої остачі, другого числа, другої остачі, третього числа, третьої остачі і т.д.).
2. Визначте та проініціалізуйте масив з 8-х чисел типу **float**. Запишіть номер, значення, квадрат та куб значення елементів масиву.
3. Запишіть значення лічильника та випадкові числа типу **float**, що не більші за поточне значення лічильника. Кінцеве значення лічильника задається з клавіатури.
4. Визначте двовимірний масив розміром 4 на 5 чисел типу **float**, заповніть випадковими числами, що не більші за 20.00. Запишіть масив у файл за рядками.
5. Визначте та проініціалізуйте масиви з 6 чисел типу **char**, **int**, **float**. Запишіть ці масиви у файл за рядками (перше число **char**, перше число **int**, перше число **float** і т.д.).

Програма 3.4. Показана робота блокових функцій запису-читання. Для зручності визначена функція запису і функція читання. Файл створюється в бінарному режимі.

```

#include <stdio.h>
#include <stdlib.h>
#include <iostream>
struct BOOK {char title[40];
             char author[20];
             int year;
};

```

```

int main()
{FILE *out,*in;
BOOK book1, book2;
int flag=0;
// відкриття потоку для запису в двійковому режимі
out=fopen("biblio.txt","ab");
if(out==NULL)
{perror("Cannot open output file: "); return 1;}
do
{puts("Enter name, author, year of publication books");
gets(book1.title);
gets(book1.author);
scanf("%d",&book1.year); fflush(stdin);
fwrite(&book1,sizeof(BOOK),1,out); // запис даних
puts("Продовжити введення? Так-1, Немає-0");
scanf("%d", &flag);
fflush(stdin);
}while(flag);
fclose(out); // закриття потоку
// відкриття потоку для читання в двійковому режимі
in=fopen("biblio.txt","rb");
if(in==NULL)
{perror("Cannot open output file: ");
return 2; }
puts("Your bibliography of these books:");
while(fread(&book2,sizeof(BOOK),1,in)) // читання даних
{puts(book2.title);
puts(book2.author);
printf("%d\n",book2.year);
puts("\nPress any key for next\n");
system("pause");
}
fclose(in); // закриття потоку
system("pause");
return 0;}

```

Завдання до програми 3.4 Розробіть програму, що відкриває файл з іменем, що складається з перших літер Вашого прізвища, імені, по батькові. Запишіть структури типа, заданого згідно варіанту, у відкритий файл за допомогою функції **fwrite()** з можливістю для додавання. Перевірте правильність запису, для цього прочитайте з

файлу записані елементи структури за допомогою функції **fread()** та виведіть їх на екран.

1. Структурний тип містить дані про студента (прізвище, рік народження, номер курсу).
2. Структурний тип містить дані про працівника: прізвище, рік народження, стаж роботи.
3. Структурний тип містить дані про вищі учебні заклади: місто знаходження, назва, кількість факультетів, кількість місць для вступу.
4. Структурний тип містить дані про студента: прізвище, рейтинги першого та другого семестрів.
5. Структурний тип містить дані погодні умови доби: дата, тиск і температура.

Контрольні питання до лабораторної роботи № 3

1. Як відкрити і як закрити потік для обміну даними? Що означає рядок у програмі **FILE *ptr**?
2. Які існують режими відкриття файлу?
3. За допомогою яких функцій можна записати у файл (прочитати з файлу) символ? Приведіть приклади.
4. За допомогою яких функцій можна записати у файл (прочитати з файлу) рядок символів? Приведіть приклади.
5. За допомогою яких функцій здійснюється файлові запис-читання форматованих числових даних? Приведіть приклади.
6. За допомогою яких функцій здійснюється файлові запис-читання блоків числових даних? Приведіть приклади.
7. Коли та які дані виводить функція **perror("Помилка: ");** ?
8. Для чого призначені функції **rewind()**, **fseek()**, **ftell()**? Приведіть приклади їх використання.

ЛАБОРАТОРНА РОБОТА № 4

Багатофайлова компіляція

Мета роботи – вивчити прийоми створення програм, що складаються з декількох файлів.

Програма 4.1. Створіть із декількох файлів програму способом багатофайлової компіляції. Спочатку проект складається з трьох файлів із вихідними текстами функцій і одного заголовного файлу. Після доробки продемонструйте викладачеві уміння створювати багатофайловий проект і програму.

```
//файл complex.h
struct CMP {float real; float imag;};
CMP sum(CMP, CMP);
CMP sub(CMP, CMP);

//файл sum.cpp
#include "complex.h"
CMP sum(CMP com1, CMP com2)
{CMP tmpc;
  tmpc.real=com1.real+com2.real;
  tmpc.imag=com1.imag+com2.imag;
  return tmpc;}

//файл sub.cpp
#include "complex.h"
CMP sub(CMP com1, CMP com2)
{CMP tmpc;
  tmpc.real=com1.real-com2.real;
  tmpc.imag=com1.imag-com2.imag;
  return tmpc;}

//файл test.cpp
#include <stdio.h>
#include <stdlib.h>
#include<iostream>
#include "complex.h"
int main()
{CMP a, b, r1,r2;
  a.real=4;  a.imag=5;
  b.real=1;  b.imag=3;
  r1=sum(a,b);
  printf("%.2f%+.2fi\n",r1.real,r1.imag);
  r2=sub(a,b);
  printf("%.2f%+.2fi\n",r2.real,r2.imag);
  system("pause");  return 0;}
```

Завдання до програми 4.1 За допомогою проекту створіть багатофайлову програму із приведених вище файлів і випробуйте його. Розробіть ще три файли для роботи з комплексними числами. Перший файл повинний містити функцію множення комплексних чисел, другий – функцію ділення комплексних чисел, а третій – функцію виведення на екран комплексних чисел. Додайте усі ці файли в новий проект та перевірте всі операції з комплексними числами.

Для довідки:

$$\begin{aligned}c_1 &= a_1 + ib_1, \quad c_2 = a_2 + ib_2 \\c_1 \cdot c_2 &= (a_1 a_2 - b_1 b_2) + i(a_1 b_2 + a_2 b_1) \\ \frac{c_1}{c_2} &= \frac{a_1 a_2 + b_1 b_2}{a_2^2 + b_2^2} + i \frac{a_2 b_1 - a_1 b_2}{a_2^2 + b_2^2}.\end{aligned}$$

Програма 6.2. Створіть із декількох файлів програму, використовуючи спосіб багатофайлової компіляції.

Програма повинна обчислювати визначений інтеграл від функції на заданому інтервалі методом трапецій.

Метод трапецій:

$$\int_a^b f(x) dx \approx \frac{h}{2} (y_0 + 2y_1 + \dots + 2y_{n-1} + y_n),$$

де $h = (b-a)/n$; $y_0 = f(a)$, $y_1 = f(a+h)$, ..., $y_n = f(b)$.

Функція для інтегрування і підінтегральна функція повинні бути оформлені у вигляді окремих модулів (файлів).

Прототипи підінтегральної функції й функції для інтегрування повинні мати вигляд:

```
double func(double x); // x – аргумент функції
typedef double (*PF)(double); // визначення типу покажчика на функцію
double integ(PF pf, double a, double b, int n);
```

де a, b – границі інтервалу $[a, b]$;

n – кількість ділянок, на які розбивається інтервал $[a, b]$;

pf – покажчик на підінтегральну функцію.

Обчисліть інтеграл від функції $f(x) = x^2$ на інтервалі $[1.5, 4.0]$ при $n = 200$.

Обчисліть інтеграл від функції $f(x) = 3x^2 - 0.5x + 1$ на інтервалі $[1.0, 2.0]$ при $n = 100$.

Контрольні питання до лабораторної роботи № 4

1. Як створити програму з декількох файлів без використання проекту?
2. У чому призначення заголовних файлів?
3. У чому відмінність описів змінних від їхніх визначень?
4. У чому відмінність описів функцій від їхніх визначень?
5. Як у модулі (файлі) одержати доступ до змінної, визначеної в іншому модулі (файлі)?
6. Як у модулі (файлі) одержати доступ до функції, визначеної в іншому модулі (файлі)?

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Подбельский В.В. Язык Си++: Учеб. пособие. – М.: Финансы и статистика, 2007. – 560 с.
2. Подбельский В.В. Практикум по программированию на языке Си: Учебн. пособие. – М.: Финансы и статистика, 2004. – 576 с.
3. Подбельский В.В., Фомин С.С. Программирование на языке Си: Учебн. пособие. – 2-е изд., – М.: Финансы и статистика, 2009. – 600с.
4. Павловская Т.А. С/С++. Программирование на языке высокого уровня. – СПб: Питер, 2009. – 464 с
5. Крячков А.В., Сухинина И.В., Томшин В.К. Программирование на С и С++. Практикум: Учебное пособие для вузов / А.В. Крячков, И.В. Сухинина, В.К. Томшин. – 2-е изд., исправ. – М.: Горячая линия.-Телеком, 2000. – 344с.
6. Крупник А. Изучаем Си. – СПб.: Питер, – 2001. – 256с.
7. Стивен Прата. Язык программирования С++. Лекции и упражнения. – 6-е изд. / Пер. с англ. – М.: Вильямс, 2012. – 478 с.
8. С/С++. Структурное программирование. Практикум / Т.А. Павловская, Ю.А. Щупак. – СПб.: Питер, 2002. – 240с.
9. Культин Н.Б. С /С++ в задачах и примерах. – СПб.: БХВ: Петербург, 2001. – 288с.
10. Х.М. Дейтел, П.Дж. Дейтел. Как программировать на С++. – 5-е изд. / Пер. с англ. – М.: ООО "Бином-Пресс", 2008.– 1456 с.
11. Шилдт, Герберт. Полный справочник по С++. – 4-е изд. / Пер. с англ. – М.: Вильямс, 2008. – 800 с.
12. Брайан Керниган, Деннис Ритчи. Язык программирования С. – 2-ое изд. / Пер. с англ. – М.: Вильямс, 2009. – 532 с.