

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Запорізький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних та контрольних робіт
із дисципліни "Програмування" (II частина)
для студентів спеціальності 123 "Комп'ютерна інженерія"
заочної форми навчання

2019

Методичні вказівки до виконання лабораторних та контрольних робіт із дисципліни "Програмування" (II частина) для студентів спеціальності 123 "Комп'ютерна інженерія" заочної форми навчання / Укл.: Н.В. Луценко. – Запоріжжя: ЗНТУ, 2019. – 42 с.

Укладачі:

Н.В. Луценко, 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

Рекомендовано до видання
на засіданні НМК факультету КНТ
Протокол № 6 від 31.01.2019

ЗМІСТ

Загальні відомості	5
Лабораторні роботи	19
Лабораторна робота № 1 Структури, об'єднання	19
Лабораторна робота № 2 Елементарне введення у функції	20
Лабораторна робота № 3 Прийоми використання функцій	22
Лабораторна робота № 4 Робота з файлами	24
Контрольні роботи	27
Завдання № 1 Структури, об'єднання	27
Завдання № 2 Елементарне введення у функції	30
Завдання № 3 Прийоми використання функцій	33
Завдання № 4 Робота з файлами	36
Питання до контрольної роботи	41
Рекомендована література	42

ПОРЯДОК ВИКОНАННЯ ЛАБОРАТОРНИХ ТА КОНТРОЛЬНИХ РОБІТ

Лабораторні роботи складаються з декількох програм, завдань до них. Студент повинен розробити, налагодити та зберегти у робочому каталозі програми в точній відповідності з приведеними в лабораторних роботах текстами програм (вихідні програми), опрацювати і проаналізувати результати їхньої роботи. Розробити для кожної вихідної програми самостійне завдання. Продемонструвати викладачу роботу програм.

Контрольна робота складається з декількох завдань з варіантами. Студент повинен розробити, налагодити і зберегти програми згідно заданого варіанту, опрацювати і проаналізувати результати їхньої роботи та дати відповіді на контрольні питання.

Зміст контрольної роботи

1. Титульний лист з номером варіанту.
2. Тексти завдань згідно варіанту.
3. Тексти програм.
4. Відповіді на контрольні питання.
5. Електронний носій з розробленими програмами.

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'явилася, 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'явилася, 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

Структури. Це сукупність поіменованих об'єктів у загальному випадку різних типів. Кожна структура включає в себе один або декілька об'єктів (змінних, масивів, покажчиків, структур), які називають елементами (полями) структури.

Структурний тип (*шаблон*) визначає, скільки елементів і якого типу входять в структуру. Визначення шаблону структури починається із слова **struct**, за яким у фігурних дужках через крапку з комою розміщуються описи елементів, що входять в структуру. Під шаблон пам'ять не виділяється.

Після визначення структурного типу (шаблону) за його допомогою можна визначити конкретні структури, масиви структур, покажчики на структуру. Наприклад:

```
struct STD{char name[20]; //визначення структурного типу
           int age;
           float r; };
```

```
STD st1, st2[3], *ptrst=&st1; //визначення структур
```

Пам'ять для структур виділяється згідно структурного типу, тобто для структури **st1** виділиться 28 байт (20+4+4), для масиву структур **st2** – 56 байт, для покажчика на структуру **ptrst** – 4 байти.

Для звернення (доступу) до елементів структури використовуються уточнені імена. Наприклад:

```
st1.age=19;    st1.r=4.3;
```

Для звернення до членів масиву структур використовується індекс, який вказують у квадратних дужках, наприклад:

```
cout<<st2[0].name;
st2[1].age=20; st2[2].r=4.7;
```

При зверненні до членів структури через покажчик використовуються операція стрілка або розіменування покажчика з ім'ям елемента структури після крапки. Нижче наведені обидві форми:

```
ptrst→age=18;    (*ptrst).r=4.5;
```

При визначенні структури можлива її ініціалізація, тобто приписування початкових значень елементам структури. Для цього після визначення структури ставиться знак '=' та у фігурних дужках через кому задаються значення елементів згідно структурного типу. Наприклад:

```
STD st3={"Шевченко", 20, 4.9};
STD st4[2]={{"Петров", 20, 4.4}, {"Романов", 19, 4.3}};
```

Можна створювати і видаляти структури та масиви структур динамічної пам'яті. Для цього використовуються функції `malloc()` і `free()` або операцій `new` і `delete`. Нижче приведено приклад створення структури та масиву структур динамічної пам'яті:

```
struct REC{int a[2]; float b;}; //визначення структурного типу
REC *r1,*r2; //визначення покажчиків на структуру
r1=(REC*)malloc(sizeof(REC)); //запит пам'яті під структуру
// r1=new REC; // для операцій new - delete
if(r1==NULL) // перевірка виділення пам'яті
    {puts("Помилка виділення пам'яті"); return -1;}
scanf("%d %d %f",&r1->a[0],&r1->a[1],&r1->r); //введення даних
free(r1); // звільнення пам'яті
//delete rec1; // для операцій new - delete
int n; puts("Введіть розмір масиву структур");
scanf("%d",&n); //введення розміру масиву (розмір – змінна)
r2=(REC*)malloc(n*sizeof(REC)); //запит пам'яті для масиву структур
// r2=new REC[n]; // для операцій new - delete
if(r2==NULL) // перевірка виділення пам'яті
    {puts("Помилка виділення пам'яті"); return -1;}
for(int i=0;i<n;i++) //введення даних
    scanf("%d %d %f",&r2[i].a[0],&r2[i].a[1],&r2[i].r);
    //введення даних
free(rec2); //звільнення пам'яті
//delete []rec2; // для операцій new - delete
```

Об'єднання. Це сукупність поіменованих об'єктів різних типів, в яких однакова початкова адреса. На відміну від структури всі елементи об'єднання мають нульовий зсув. Тип об'єднання (*шаблон*) визначає, скільки елементів і якого типу входять в нього. Визначення типу об'єднання починається зі слова `union`, за яким розміщуються описи елементів, поміщені у фігурні дужки.

Розмір пам'яті, що виділена під об'єднання, визначається розміром елемента з максимальною довжиною. Визначення шаблону об'єднання і самого об'єднання, а також доступ до елементів об'єднання виконується аналогічно структурам. Наприклад:

```
union UN{int a[2]; char c[4];}; //визначення типу об'єднання
UN un1,*ptrun=&un1; //визначення об'єднання, покажчика на об'єднання
un1.a[0]=25; //звернення до елемента об'єднання
ptrun->c[3]='e'; //звернення до un1.c[3] через покажчик
```

При визначенні об'єднання можна ініціалізувати тільки перший елемент. Наприклад:

```
UN un2={1,12}; // ініціалізується перший елемент (масив a)
```

Елементами об'єднань можуть бути структури. Наприклад :

```
struct WORD{unsigned short ax;};  
struct BYTE{unsigned char al; unsigned char ah;};  
union REGS{WORD x; BYTE h;}rg;  
rg.x.ax=0x4300; //запис в ax  
printf("\n ah=%#x al=%#x\n", rg.h.ah rg.h.al);
```

Об'єднання можуть бути елементами структур. Наприклад:

```
union UN1{long b; char c[4];}; //виділиться пам'ять 4 байта  
struct STR{char st; int a[2]; UN1 my_u;}str1;  
str1.my_u.b=500; str1.st='z';
```

Бітові поля структур та об'єднань. Це цілі значення (типу **int**, **char**, **long**, **short**), що займають у пам'яті фіксоване число бітів (від 1 до 16 біт). Бітові поля можуть бути тільки елементами структур і об'єднань. Їх використовують для доступу до окремих бітів даних, для більш раціонального використання пам'яті. Розмір бітових полів задається через двокрапку після імені елемента структури або об'єднання. Наприклад:

```
struct SS{int a:5; // для елемента a виділиться 5 бітів  
int b:3;}; // для елемента b виділиться 3 біта  
SS stb,*pb=&stb; //визначення структури, покажчика на структуру
```

Для доступу до бітових полів використовуються уточнені імена:

```
stb.a=31; stb.b=3; (*pb).a=1; pb->b=3;
```

Членами об'єднання можуть бути структури з бітовими полями,:

```
union UB{char c; SS str1;}un3;
```

Для об'єднання **un3** виділиться 1 байт (8 біт) пам'яті.

Функції. Функція – це елемент програми, що може використовуватися неодноразово для виконання деякої, заздалегідь визначеної задачі. Кожна програма містить 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 та мають 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(сигнатура_функції)  
{ тіло функції }
```

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

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

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

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

Наприклад, функція, яка повертає суму трьох чисел:

```
float func1(float a, float b, float c)  
{float S=a+b+c;  
  return S; }
```

В цьому прикладі **float** – тип значення, яке повертає функція, **func1** – 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 func2()  
{printf("pi=%f\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**, або фактичні параметри (константи, раніш визначені змінні або 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) // визначення функції
{if(x>y)return x;
 return y;}
int main()
{int a=10,res1,res2; float f=55.9;
 res1=max(a,20); // перший виклик функції res1=20
 res2=max(int(f),a+30); // другий виклик функції res2=55
 return 0;}
```

Якщо функція визначена після функції `main()` або в іншому файлі, то до першого звернення до функції вона має бути описана.

Опис функції називається прототипом. Це заголовок функції, наприкінці якого обов'язкова крапка з комою. Прототип потрібен для перевірки відповідності типів та кількості переданих у функцію параметрів і типів значень, що повертаються. У прототипі можна не вказувати імена формальних параметрів.. Приклади прототипів :

```
float func1(float a, float b, float c);
int max(int, int y);
int sum(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 здійснюються над цими копіями в тілі функції, не впливають на значення фактичних параметрів поза функцією.

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

Функції і масиви. При передачі масиву у функцію як аргумент вказується ім'я масиву (покажчик-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 покажчика відповідного типу, який діє усередині функції як звичайний покажчик-змінна і може змінювати своє значення. Наприклад, `arr++`.

Доступ до елементів масиву усередині функції може здійснюватися за допомогою індексів (`arr[0]`, `arr[i]`) і за допомогою операції розіменування (`*(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 - кількість елементів в масиві
```

```
{int S=0;
```

```
    for(int i=0;i<N;i++) S+=x[i];
```

```
    return S;}
```

```
int main()
```

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

```
    res1=sum(a,5); // перший виклик функції, res1=21
```

```
    res2=sum(c,3); // другий виклик функції, res2=91
```

```
    printf("res1=%d res2=%d\n",res1,res2);
```

```
    return 0;}
```

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'
```

```
int main()
```

```
{char *str="Рядок";;
```

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

```
    return 0;}
```

Функції з параметрами за умовчанням. Це функції, що мають в специфікації формальних параметра значення за умовчанням. При зверненні до такої функції параметр 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;}
```

```
int 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);
  return 0;}
```

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. 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 - число змінних в списку
    sum+=*(ptr++);
  return sum;}
int 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);
  return 0;}
```

Рекурсивна функція. Ця 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 вбудована, тобто замість кожного виклику цієї функції компілятор пробує 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 sqrt(x*x+y*y);}
int 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);
  return 0;}
```

Передача параметрів у функцію main(). В функцію **main()** як і в будь-яку іншу функцію можуть передаватися параметри. Для цього функцію **main()** варто 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.

Аргумент **int argc** визначає кількість слів у командному рядку. Аргумент **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. Аргумент **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 параметрів). Для забезпечення перевантаження необхідно для кожного імені визначити, скільки різних функцій пов'язано з ним.

Розпізнавання перевантажених функцій при виклику виконується за сигнатурою (списком параметрів функції). Наприклад:

```
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;}
int main()
{int a=2,b=5,c=6,d1,d2; float f1=1.5,f2=4.6,f3;
 d1=sum(a,b);           // виклик 1-ої функції d1=2+5=7
 d2=sum(a,b,c);         // виклик 2-ої функції d2=2+5+6=13
 f3=sum(f1,f2);         // виклик 3-ої функції d3=1.5+4.6=6.1
 printf("d1=%d d2=%d d3=%.1f\n",d1,d2 d3);
 return 0;}
```

Шаблони функцій. Для автоматизації написання перевантажених функцій різного виду (перевантажених) 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) // визначення функції з шаблоном T
{if(x>y) return x; return y; }
int 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);    } // Т приймає значення float
printf("a3=%d f3=%.1f\n",a3,f3);
return 0;}
```

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 визначення функції.

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 доступу: 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' та '\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 структуру 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:

r – потік відкривається для читання з існуючого файлу,
r+ – потік відкривається для читання і запису в існуючий файл;
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;
a – потік відкривається для запису даних в кінець (для додавання) файлу. Якщо файл не існує, він створюється;
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);**

що закриває потік введення-виведення, 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;}
// Програма ...
fclose(fp);
```

Функції запису даних в файл та читання даних з файлу.

Для запису даних у файл та читання даних з файлу відкривається потік з вказівкою імені файлу, після роботи потік закривають:

```
FILE *fp=fopen("abc.txt", "w"); // потік відкрито для запису
if(fp==NULL){puts("помилка відкриття файлу");return -1;}
// Програма ...
fclose(fp); // потік закрито
fp=fopen("abc.txt", "r");// потік відкрито для читання
if(fp==NULL){puts("помилка відкриття файлу");return -1;}
// Програма ...
fclose(fp); // потік закрито
```

Нижче приводяться функції запису даних у файл та читання з файлу.

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';
fputc(c1, fp); // запис символу у 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=fgetc(fp); // читання символу з файлу  
printf("%c",c2); // виведення символу на екран
```

Функція запису рядка в файл

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

Функція записує у відкритий файл всі байти символів з рядка, на який вказує покажчик **s** до досягнення символу кінця рядка '**\0**', який в файл не записується. У випадку успіху повертає код останнього записаного у файл символу рядка, інакше – **EOF**. Наприклад:

```
char string[20]="Програмування";  
fputs(string,fp); // запис рядка у файл
```

Функція читання рядка з файлу

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

Функція зчитує символи з відкритого файлу і розміщує їх за адресом, що задана покажчиком **str**. Зчитування символів закінчується після передачі **N** байт чи при одержанні символу '**\n**'. У випадку успіху повертає покажчик на прочитаний рядок символів, інакше – **NULL**. Під зчитаний рядок треба зарезервувати пам'ять. Наприклад:

```
char str[20]; // резервування пам'яті  
fgets(str,20,fp); // читання з файлу рядка не більш 20 символів  
puts(str); // виведення рядка на екран
```

Функція форматowanego запису даних у файл

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

Функція перетворює дані за вказаним форматом у символи та записує їх у відкритий файл. У випадку успіху повертає записане число байтів, інакше – повертає **EOF**. Наприклад:

```
float f=3.5; char g='B';  
fprintf(fp,"%f %c",f,g); // запис даних у файл  
int d[5]={7,8,5,3,9};  
for(int i=0;i<5;i++)  
    fprintf(fp,"%d",d[i]); //запис масиву файл
```

Функція форматованого читання даних з файлу

int fscanf(FILE *fp, char *format[, pointer]...); // прототип
 Функція зчитує дані з відкритого файлу з перетворенням їх в числа за вказаним форматом, та розміщує за вказаними адресами. У випадку успіху повертає число успішно оброблених при виведенні полів, інакше – повертає **EOF**. Наприклад:

```
float b; char c;           // резервування пам'яті
fscanf(fp, "%f %c", &b, &c); // читання даних з файлу
printf("b=%f c=%c\n", b, c); // виведення даних на екран
int a[5]; // резервування пам'яті
for(int j=0; j<5; j++)
    {fscanf(fp, "%d", &a[j]); // читання масиву з файлу
    printf("%d\n", a[j]); } // виведення масиву на екран
```

Функція запис блоків даних в файл

int fwrite(void *ptr, int size, int n, FILE* fp); //прототип
 Функція виконує введення у відкритий файл **n** блоків, розміром **size** байт кожен. У випадку успіху повертає введенне число блоків, інакше – повертає **EOF**. Наприклад:

```
struct REC{int a; float b[2];};
REC st1={1,2,3.3};
fwrite(&st1, sizeof(REC), 1, fp); // запис 1-ого блоку у файл
REC st2[3]={ {10,20,1.5}, {30,40,2.5}, {50,60,3.5} };
fwrite(st2, sizeof(REC), 3, fp); // запис 3-х блоків у файл
```

Функція читання блоків даних з файлу

int fread(void *ptr, int size, int N, FILE *fp); // прототип
 Функція виконує виведення з відкритого файлу **N** блоків, розміром **size** байт кожен. У випадку успіху повертає виведене число блоків, інакше – повертає **EOF**. Наприклад:

```
struct REC {int a[2]; float b;};
REC st3, st4[3]; // резервування пам'яті
fread(&st3, sizeof(REC), 1, fp); //читання 1-ого блоку з файлу
// виведення на екран зчитаної структури
printf("%d %d %.2f\n", st3.a[0], st3.a[1], st3.b);
fread(st4, sizeof(REC), 3, fp); //читання 3-х блоків з файлу
for(int i=0; i<3; i++) // виведення на екран масиву структур
printf("%d %d %.2f\n", st4[i].a[0], st4[i].a[1], st4[i].b);
```

Довільний доступ до файлу При обміні даними з файлом (записі і читанні) місце, куди записується або звідки 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 вправо 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 показчика запису-читання через 10 байт від початку файлу:

fseek(fp, 10, SEEK_SET);

Функція визначення поточного положення показчика запису-читання має прототип:

long ftell(FILE *fp);

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

long cur=ftell(fp);

ЛАБОРАТОРНІ РОБОТИ

Лабораторна робота № 1 Структури, об'єднання

Мета роботи – вивчити способи визначення та ініціалізації структур, показчиків на структуру, масивів структур та об'єднань.

Програма 1.1. В програмі визначаються проста структура, масив структур та показчик на структуру. Елементом всіх структур привласнюються певні значення.

```
#include <stdio.h>
#include<iostream>
struct REC{char *name;    // визначення структурного типу
           int age;
           float r; };

int main()
{REC rec1; // визначення структури rec1 типу REC
  rec1.name="Petrov";
  rec1.age=19;  rec1.r=4.5;
  printf("%-8s %d %.2f\n",rec1.name,rec1.age,rec1.r);
//Визначення масиву з 3-х структур типу REC і ініціалізація першої з них
  REC str[3]={"Sidorov",19,4.17};
  str[1].name="Semenov";
  str[1].age=20;  str[1].r=4.33;
  str[2].name="Orlov";
  str[2].age=21;  str[2].r=3.95;
  for(int i=0;i<3;i++)
    printf("%-8s %d %.2f\n",str[i].name,str[i].age,str[i].r);
  REC *ptrr;    // визначення показчика на структуру типу REC
  ptrr=new REC; // запит пам'яті для структури типу REC
  if(ptrr==NULL){puts("Error"); return -1;}
  ptrr->name="Ivanov";
  ptrr->age=18;  (*ptrr).r=4.2;
  printf("%-8s %d %.2f\n",(*ptrr).name,ptrr->age,ptrr->r);
  delete ptrr; // звільнення пам'яті
  system("pause"); return 0; }
```

Вказівка до програми 1.1: Визначте структурний тип, членами якого є масив з 2-х чисел типу **char** і масив з 3-х чисел типу **int**. Визначте просту структуру, масив з 4-х структур та показчик на структуру, під яку запросить пам'ять. Перші три структури з масиву структур проініціалізуйте, всім іншим членам цих структур

привласніть які-небудь значення. Визначте розмір пам'яті, що займають кожна з структур. Виведіть всі структури на екран.

Програма 1.2. В програмі демонструється використання об'єднання **union** для дослідження внутрішнього зображення дійсних чисел. Число типу **float** виводиться у вигляді цілого числа типу **long**, а потім аналізуються його двійкові розряди.

```
#include <stdio.h>
#include<iostream>
//визначення типу UN об'єднання
union UN{float F;
        unsigned int L;};
int main()
{UN1; // визначення об'єднання un1 типу UN
  puts("Enter a valid number");// введіть дійствительное число
  scanf("%f",&un1.F);
  printf("F=%.3f  L=%ld=%#lx \n", un1.F, un1.L, un1.L);
  system("pause");
  return 0; }
```

Вказівка до програми 1.2: Визначте об'єднання, членами якого є масив із трьох цілих чисел і масив із шести символів. Введіть з клавіатури елементи об'єднання як цілі числа, а виведіть їх на екран в вигляді символів та з кодом символів в 16-річній системі числення.

Лабораторна робота № 2

Елементарне введення у функції

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

Програма 2.1. В програмі показана робота найпростіших функцій.

```
#include <stdio.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;}
}
```

```

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);
 system("pause"); return 0;}

```

Завдання до програми 2.1 Розробіть функцію типу `int`, що повертає більше з трьох цілих чисел та функцію типу `float`, що повертає суму двох вхідних параметрів, якщо другий параметр додатний, інакше повертає різницю цих параметрів. Продемонструйте роботу цих функцій, зробивши в головній програмі кілька викликів функцій з різними параметрами.

Програма 2.2. 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("\nsuma2=%.1f\n",res2);
system("pause");
return 0;
}
float suma(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 2.2 Розробіть функцію, в яку передаються масив типу `int` та його розмір, і яка повертає максимальний елемент масиву. Продемонструйте роботу цієї функції, зробивши в головній програмі кілька викликів функції з різними параметрами.

Лабораторна робота № 3

Прийоми використання функцій

Програма 3.1. 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<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 elements of array float=%.2f\n", resfl );
  system("pause"); return 0;}

```

Завдання до програми 3.1 Розробіть перевантажені функції типу **float**, що повертають середнє арифметичне двох, трьох або чотирьох чисел. Продемонструйте роботу цих функцій, зробивши кілька викликів функції з різними значеннями.

Програма 3.2. 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<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=%.2lf\n",sg1);
sg2=sigma(6,darr,21.5); //виклик функції, А замінюється на double
printf("sg2=%.2lf\n",sg2);
system("pause"); return 0;}
```

Завдання до програми 3.2 Розробіть шаблон для функцій типу `int`, `float`, `double`, що повертають максимальний елемент масиву. Продемонструйте роботу цих функцій, зробивши в головній програмі кілька викликів функцій з різними значеннями.

Лабораторна робота № 4

Робота з файлами

Мета роботи – вивчити основні функції роботи з файлами.

Програма 4.1. Показано роботу функції відкриття файлу для запису та читання, запис символів в файл та читання символів з файлу. Після завершення роботи програми перевірте у своєму каталозі наявність і вміст файлу `test1.txt`.

```
#include <stdio.h>
#include<iostream>
int main()
{FILE *fout;
  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
  FILE *fin;
  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 4.1 Розробіть програму, що відкриває файл з іменем, що складається з перших чотирьох букв Вашого прізвища з розширенням **txt**. Запишіть у вигляді символів рядок, який проініціалізуєте Вашим прізвищем, у відкритий файл за допомогою функції **fputc()**. Перевірте правильність запису, для цього зчитайте з файлу записані символи за допомогою функції **fgetc()** та виведіть їх на екран.

Програма 4.2. У програмі використовуються функції запису рядків у файл та читання рядків з файлу.

```
#include <stdio.h>
#include<iostream>
int main()
{FILE *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
  FILE *fin;
  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 4.2 Розробіть програму, що відкриває файл з іменем, що складається з Ваших ініціалів. Запишіть масив з 4-х рядків, який проініціалізуєте назвами дисциплін, з яких треба скласти іспити. 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.

Програма 4.3. У програмі використовуються функції файлових запису та читання форматованих даних.

```
#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 4.3 Розробіть програму, що відкриває файл з іменем, що складається з перших п'яти літер Вашого імені. Запишіть за допомогою функції **fprintf()** значення лічильника та випадкові числа, що не більші за поточне значення лічильника. Кінцеве значення лічильника задається з клавіатури. Перевірте правильність запису, для цього прочитайте з файлу записані числа за допомогою функції **fscanf()** та виведіть їх на екран.

КОНТРОЛЬНІ РОБОТИ

Завдання № 1 Структури, об'єднання

Варіант 1_1

1. Визначте шаблон структури, членами якого є символ, змінна типу **long** та масив з 3-х чисел типу **float**. За допомогою цього шаблону визначте структуру, масив з 4-х структур та покажчик на структуру, під яку запросить пам'ять. Перші три структури з масиву структур проініціалізуйте, всім іншим членам цих структур привласніть які-небудь значення. Виведіть елементи всіх структур на екран. Визначте розміри пам'яті, що займає структура, масив структур та покажчик на структуру, які запишіть у вигляді коментаря.
2. Визначте об'єднання, елементами якого є масиви з трьох чисел типу **short** та з шести символів. Значення елементів введіть з клавіатури у вигляді чисел типу **short** в 16-річному коді, а виведіть на екран у вигляді символів та з кодами символів в 16-річній системі числення.

Варіант 1_2

1. Визначте шаблон структури, членами якого є змінні типу **double**, **int** та масив з 3-х символів. За допомогою цього шаблону визначте структуру, масив з 3-х структур та покажчик на структуру, під яку запросить пам'ять. Перші дві структури з масиву структур проініціалізуйте, всім іншим членам цих структур привласніть які-небудь значення. Виведіть елементи всіх структур на екран. Визначте розміри пам'яті, що займає структура, масив структур та покажчик на структуру, які запишіть у вигляді коментаря.
2. Визначте об'єднання, елементами якого є масиви з трьох чисел типу **long** та з шести чисел типу **short**. Значення елементів об'єднання введіть з клавіатури як 16-річні числа типу **long**, а виведіть на екран як числа типу **short** в 16-річному та 10-річному кодах.

Варіант 1_3

1. Визначте шаблон структури, членами якого є масив з 3-х чисел типу **int**, змінна типу **float** та символ. За допомогою цього шаблону визначте структуру, масив з 3-х структур та покажчик на структуру, під яку запросить пам'ять. Перші дві структури з масиву структур проініціалізуйте, всім іншим членам цих структур привласніть які-небудь значення. Виведіть елементи всіх структур на екран. Визначте

розміри пам'яті, що займає структура, масив структур та покажчик на структуру, які запишіть у вигляді коментаря.

2. Визначте об'єднання, елементами якого є масиви з трьох чисел типу **short** та з шести символів. Значення елементів об'єднання введіть з клавіатури у вигляді символів, а виведіть на екран у вигляді чисел типу **short** в 16-річному та 10-річному кодах.

Варіант 1_4

1. Визначте шаблон структури, членами якого є рядок, змінні типу **long** та типу **float**. За допомогою цього шаблону визначте структуру, масив з 3-х структур та покажчик на структуру, під яку запросить пам'ять. Перші дві структури з масиву структур проініціалізуйте, всім іншим членам цих структур привласніть які-небудь значення. Виведіть елементи всіх структур на екран. Визначте розміри пам'яті, що займає структура, масив структур та покажчик на структуру, які запишіть у вигляді коментаря.

2. Визначте об'єднання, елементами якого є масиви з трьох чисел типу **float** і з трьох чисел типу **long**. Значення елементів об'єднання введіть з клавіатури у вигляді чисел типу **float**, а виведіть на екран у вигляді чисел типу **long** в 16-річній та 10-річній системах числення

Варіант 1_5

1. Визначте шаблон структури, членами якого є масив з 2-х чисел типу **float**, символ та змінна типу **int**. За допомогою цього шаблону визначте структуру, масив з 4-х структур та покажчик на структуру, під яку запросить пам'ять. Перші три структури з масиву структур проініціалізуйте, всім іншим членам цих структур привласніть які-небудь значення. Виведіть елементи всіх структур на екран. Визначте розміри пам'яті, що займає структура, масив структур та покажчик на структуру, які запишіть у вигляді коментаря.

2. Визначте об'єднання, елементами якого є масиви з двох чисел типу **double** та з восьми чисел типу **long**. Значення елементів об'єднання введіть з клавіатури як числа типу **double**, а виведіть на екран як числа типу **long** в 16-річній та 10-річній системах числення.

Варіант 1_6

1. Визначте шаблон структури, членами якого є змінна типу **long**, масив з 3-х чисел типу **int** та масив з 2-х символів. За допомогою цього шаблону визначте структуру, масив з 3-х структур та покажчик

на структуру, під яку запросить пам'ять. Перші дві структури з масиву структур проініціалізуйте, всім іншим членам цих структур привласніть які-небудь значення. Виведіть елементи всіх структур на екран. Визначте розміри пам'яті, що займає структура, масив структур та покажчик на структуру, які запишіть у вигляді коментаря.

2. Визначте об'єднання, елементами якого є масиви з двох чисел типу **float** та з восьми символів. Значення елементів об'єднання введіть з клавіатури як числа типу **float**, а виведіть на екран як символи з кодом символів в 16-річній та 10-річній системах числення.

Варіант 1_7

1. Визначте шаблон структури, членами якого є масив з 2-х чисел типу **int**, та масив з 2-х чисел типу **float**. За допомогою цього шаблону визначте структуру, масив з 4-х структур та покажчик на структуру, під яку запросить пам'ять. Перші три структури з масиву структур проініціалізуйте, всім іншим членам цих структур привласніть які-небудь значення. Виведіть елементи всіх структур на екран. Визначте розміри пам'яті, що займає структура, масив структур та покажчик на структуру, які запишіть у вигляді коментаря.

2. Визначте об'єднання, елементами якого є масиви з двох чисел типу **long** і з восьми символів. Значення елементів введіть з клавіатури як числа типу **long** в 16-річній системі числення, а виведіть на екран як символи з кодами символів в 16-річній системі числення.

Варіант 1_8

1. Визначте шаблон структури, членами якого є масив з 3-х символів та масив з 2-х чисел типу **int**. За допомогою цього шаблону визначте структуру, масив з 3-х структур та покажчик на структуру, під яку запросить пам'ять. Перші дві структури з масиву структур проініціалізуйте, всім іншим членам цих структур привласніть які-небудь значення. Виведіть елементи всіх структур на екран. Визначте розміри пам'яті, що займає структура, масив структур та покажчик на структуру, які запишіть у вигляді коментаря.

2. Визначте об'єднання, елементами якого є масиви з восьми символів та з двох чисел типу **long**. Значення елементів введіть з клавіатури як символи, а виведіть на екран як числа типу **long** в 16-річному та 10-ому кодах.

Варіант 1_9

1. Визначте шаблон структури, членами якого є змінна типу **float**, масив з 4-х чисел типу **int**. За допомогою цього шаблону визначте структуру, масив з 3-х структур та покажчик на структуру, під яку запросить пам'ять. Перші дві структури з масиву структур проініціалізуйте, всім іншим членам цих структур привласніть які-небудь значення. Виведіть елементи всіх структур на екран. Визначте розміри пам'яті, що займає структура, масив структур та покажчик на структуру, які запишіть у вигляді коментаря.
2. Визначте об'єднання, елементами якого є масиви із 2-х чисел типу **double** та 4 -х чисел типу **float**. Значення елементів об'єднання введіть з клавіатури як числа типу **double**, а виведіть як числа типу **float**.

Варіант 1_10

1. Визначте шаблон структури, членами якого є масив з 2-х чисел типу **double** та масив з 3-х чисел з типу **int**. За допомогою цього шаблону визначте структуру, масив з 3-х структур та покажчик на структуру, під яку запросить пам'ять. Перші дві структури з масиву структур проініціалізуйте, всім іншим членам цих структур привласніть які-небудь значення. Виведіть елементи всіх структур на екран. Визначте розміри пам'яті, що займає структура, масив структур та покажчик на структуру, які запишіть у вигляді коментаря.
2. Визначте об'єднання, елементами якого є масиви із 6-х чисел типу **float** та 3-х чисел типу **double**. Значення елементів об'єднання введіть з клавіатури як числа типу **float**, а виведіть як числа типу **double**.

Завдання № 2

Елементарне введення у функції

Варіант 2_1

1. Розробіть функцію типу **float**, що повертає абсолютне значення результату ділення першого аргументу на другий. Передбачить перевірку ділення на нуль. Зробіть в головній програмі два виклики функції з різними параметрами. На екран виведіть результати обчислень.
2. Розробіть функцію, в яку передається масив типу **float**, що повертає суму додатних елементів цього масиву. Зробіть в головній програмі два виклики функції з масивами різного розміру. Один масив

проініціалізуйте, другий – введіть зклавіатури. На екран виведіть значення елементів масиву і результати обчислень.

Варіант 2_2

1. Розробіть функцію типу **int**, що повертає більше з трьох чисел за абсолютним значенням. Зробіть в головній програмі два виклики функції з різними параметрами. На екран виведіть результати обчислень.
2. Розробіть функцію, в яку передається масив типу **char**, що повертає максимальний елемент цього масиву. Зробіть в головній програмі два виклики функції з масивами різного розміру. Один масив проініціалізуйте, другий – введіть зклавіатури. На екран виведіть значення елементів масиву і результати обчислень.

Варіант 2_3

1. Розробіть функцію типу **double**, що повертає результат від ділення першого аргументу на другий, якщо другий аргумент більше нуля, інакше функція повертає добуток аргументів. Зробіть в головній програмі два виклики функції з різними параметрами. На екран виведіть результати обчислень.
2. Розробіть функцію, в яку передається масив типу **int**, що повертає індекс мінімального елементу цього масиву. Зробіть в головній програмі два виклики функції з масивами різного розміру. Один масив проініціалізуйте, другий – введіть зклавіатури. На екран виведіть значення елементів масиву і результати обчислень.

Варіант 2_4

1. Розробіть функцію типу **int**, що повертає суму першого та другого аргументів, якщо суму більша 5, інакше повертає число 0. Зробіть в головній програмі два виклики функції з різними параметрами. На екран виведіть результати обчислень.
2. Розробіть функцію, в яку передається масив типу **float**, що повертає суму елементів масиву в діапазоні від 5.0 до 100. Зробіть в головній програмі два виклики функції з масивами різного розміру. Один масив проініціалізуйте, другий – введіть зклавіатури. На екран виведіть значення елементів масиву і результати обчислень.

Варіант 2_5

1. Розробіть функцію типу **float**, що повертає різницю першого та другого аргументів, якщо другий аргумент додатний, інакше функція

повертає суму аргументів. Зробіть в головній програмі два виклики функції з різними параметрами. На екран виведіть результати обчислень.

2. Розробіть функцію, в яку передається масив типу **int**, що повертає кількість непарних елементів масиву. Зробіть в головній програмі два виклики функції з масивами різного розміру. Один масив проініціалізуйте, другий – введіть з клавіатури.

Варіант 2_6

1. Розробіть функцію типу **char**, що повертає суму двох вхідних параметрів, якщо значення першого параметру парне, інакше повертає число **45**. Зробіть в головній програмі два виклики функції з різними параметрами. На екран виведіть результати обчислень.

2. Розробіть функцію, в яку передається масив типу **float**, що повертає добуток додатних елементів масиву. Зробіть в головній програмі два виклики функції з масивами різного розміру. Один масив проініціалізуйте, другий – введіть з клавіатури. На екран виведіть значення елементів масиву і результати обчислень.

Варіант 2_7

1. Розробіть функцію типу **double**, що повертає куб вхідного параметру, якщо значення параметра додатне, інакше повертає квадрат вхідного параметру. Зробіть в головній програмі два виклики функції з різними параметрами. На екран виведіть результати обчислень.

2. Розробіть функцію, в яку передається масив типу **double**, що повертає суму квадратів елементів масиву. Зробіть в головній програмі два виклики функції з масивами різного розміру. Один масив проініціалізуйте, другий – введіть з клавіатури. На екран виведіть значення елементів масиву і результати обчислень.

Варіант 2_8

1. Розробіть функцію типу **float**, що повертає суму двох вхідних параметрів, якщо значення першого параметру від'ємне, інакше повертає різницю цих параметрів. На екран виведіть результати обчислень.

2. Розробіть функцію, в яку передається масив типу **float**, що повертає середнє арифметичне елементів масиву. Зробіть в головній програмі два виклики функції з масивами різного розміру. Один масив

проініціалізуйте, другий – введіть клавіатури. На екран виведіть значення елементів масиву і результати обчислень.

Варіант 2_9

1. Розробіть функцію типу **long**, що повертає абсолютне значення параметру, якщо значення параметру парне, інакше повертає число **50000**. Зробіть в головній програмі два виклики функції з різними параметрами. На екран виведіть результати обчислень.

2. Розробіть функцію, в яку передається масив типу **long**, що повертає число елементів, які більші за задане число. Зробіть в головній програмі два виклики функції з масивами різного розміру. Один масив проініціалізуйте, другий – введіть клавіатури. Зробіть в головній програмі два виклики функції з масивами різного розміру. Один масив проініціалізуйте, другий – введіть клавіатури. На екран виведіть значення елементів масиву і результати обчислень.

Варіант 2_10

1. Розробіть функцію типу **int**, що повертає суму трьох вхідних параметрів, якщо значення другого параметру непарне, інакше повертає число **55**. Зробіть в головній програмі два виклики функції з різними параметрами. На екран виведіть результати обчислень.

2. Розробіть функцію, в яку передається масив типу **int**, що повертає число елементів, які дорівнюють нулю. Зробіть в головній програмі два виклики функції з масивами різного розміру. Один масив проініціалізуйте, другий – введіть клавіатури. На екран виведіть значення елементів масиву і результати обчислень.

Завдання № 3

Прийоми використання функцій

Варіант 3_1

1. Розробіть перевантажені функції, в які передаються масиви типів **int**, **float**, **double**, що виводять на екран значення елементів масивів. Зробіть в головній програмі три виклики функцій з масивами різного типу і різного розміру. На екран виведіть результати обчислень.

2. Розробіть шаблон функції, в яку передається масив та розмір і повертається сума додатних значень елементів масиву. Зробіть в головній програмі три виклики функцій з масивами різного розміру

типів **int**, **float**, **double**. На екран виведіть значення елементів масиву і результати обчислень.

Варіант 3_2

1. Розробіть перевантажені функції, в які передаються один, два або три параметри типу **float**, що повертають подвійне значення параметру, суму двох або трьох параметрів. Зробіть в головній програмі три виклики функцій з різною кількістю параметрів. На екран виведіть результати обчислень.
2. Розробіть шаблон функції, в яку передається масив та розмір і повертається максимальне значення елемента масиву. Зробіть в головній програмі три виклики функцій з масивами різного розміру типів **int**, **float**, **double**. На екран виведіть значення елементів масиву і результати обчислень.

Варіант 3_3

1. Розробіть перевантажені функції, в які передаються масиви типів **int**, **float**, що повертають суми від'ємних елементів масивів. Зробіть в головній програмі два виклики функцій з масивами різного типу і різного розміру. На екран виведіть результати обчислень.
2. Розробіть шаблон функції, в яку передається масив та розмір і повертається індекс максимального елемента масиву. Зробіть в головній програмі три виклики функцій з масивами різного розміру типів **int**, **float**, **double**. На екран виведіть значення елементів масиву і результати обчислень.

Варіант 3_4

1. Розробіть перевантажені функції, в які передаються один, два або три параметри типу **int** та повертають квадрат параметру, добуток двох або трьох параметрів. Зробіть в головній програмі три виклики функцій з різною кількістю параметрів. На екран виведіть результати обчислень.
2. Розробіть шаблон функції, в яку передається масив та розмір і повертається добуток додатних елементів масиву. Зробіть в головній програмі три виклики функцій з масивами різного розміру типів **int**, **float**, **double**. На екран виведіть значення елементів масиву і результати обчислень.

Варіант 3_5

1. Розробіть перевантажені функції, в які передаються три параметри типів **int**, **float**, **double**, що повертають максимальне значення параметру. Зробіть в головній програмі три виклики функцій з параметрами різного типу. На екран виведіть результати обчислень.
2. Розробіть шаблон функції, в яку передається масив та розмір і повертається мінімальне значення елемента масиву. Зробіть в головній програмі три виклики функцій з масивами різного розміру типів **int**, **float**, **double**. На екран виведіть значення елементів масиву і результати обчислень.

Варіант 3_6

1. Розробіть перевантажені функції, в які передаються два параметри типів **int** або **float** і три параметри типів **int** або **float** та повертають суму цих параметрів. Зробіть в головній програмі чотирьох викликів функцій з різною кількістю параметрів і різного типу. На екран виведіть результати обчислень.
2. Розробіть шаблон функції, в яку передається масив та розмір і повертається індекс мінімального елемента масиву. Зробіть в головній програмі три виклики функцій з масивами різного розміру типів **int**, **float**, **double**. На екран виведіть значення елементів масиву і результати обчислень.

Варіант 3_7

1. Розробіть перевантажені функції, в які передаються масиви типів **int**, **float**, що повертають кількість елементів, які менші за 8. Зробіть в головній програмі два виклики функцій з масивами різного типу і різного розміру. На екран виведіть результати обчислень.
2. Розробіть шаблон функції, в яку передається масив та розмір і повертається кількість додатних елементів масиву. Зробіть в головній програмі три виклики функцій з масивами різного розміру типів **int**, **float**, **double**. На екран виведіть значення елементів масиву і результати обчислень.

Варіант 3_8

1. Розробіть перевантажені функції, в які передаються один, два або три параметри типу **int** та повертається куб параметру, подвійний добуток двох параметрів або – трьох параметрів. Зробіть в головній

програмі три виклики функцій з різною кількістю параметрів. На екран виведіть результати обчислень.

2. Розробіть шаблон функції, в яку передається масив та розмір і повертається середнє арифметичне елементів масиву. Зробіть в головній програмі три виклики функцій з масивами різного розміру типів **int**, **float**, **double**. На екран виведіть значення елементів масиву і результати обчислень.

Варіант 3_9

1. Розробіть перевантажені функції, в які передаються масиви типів **int**, **float**, що повертають кількість від'ємних елементів масивів. Зробіть в головній програмі два виклики функцій з масивами різного типу і різного розміру. На екран виведіть результати обчислень.

2. Розробіть шаблон функції, в яку передається масив та розмір і повертається кількість елементів, що більші **1**. Зробіть в головній програмі три виклики функцій з масивами різного розміру типів **int**, **float**, **double**. На екран виведіть значення елементів масиву і результати обчислень.

Варіант 3_10

1. Розробіть перевантажені функції, в які передаються масиви типів **int**, **float**, що повертають кількість елементів, які більше **10**. Зробіть в головній програмі два виклики функцій з масивами різного типу і різного розміру. На екран виведіть результати обчислень.

2. Розробіть шаблон функції, в яку передається масив та розмір і повертається кількість елементів, що менші **20**. Зробіть в головній програмі три виклики функцій з масивами різного розміру типів **int**, **float**, **double**. На екран виведіть значення елементів масиву і результати обчислень.

Завдання № 4 Робота з файлами

Варіант 4_1

1. Розробіть програму, що відкриває файл з іменем, що складається з перших п'яти літер Вашого прізвища. Запишіть у відкритий файл за допомогою функції **fputs()** масив з 3-х рядків, який проініціалізуйте Вашими прізвищем, назвою спеціальності та назвою групи. Перевірте

правильність запису, прочитавши з файлу записані рядки за допомогою функції **fgets()** та виведіть їх на екран.

2. Розробіть програму, що відкриває файл з іменем, що складається з перших п'яти літер Вашого імені. Запишіть у відкритий файл за допомогою функції **fprintf()** у вигляді таблиці двовимірний масив розміром 8 на 5 з випадковими числами типу **int**, що не більші за 100. Перевірте правильність запису, прочитавши з файлу записані числа за допомогою функції **fscanf()** та виведіть їх на екран.

Варіант 4_2

1. Розробіть програму, що відкриває файл з іменем, що складається з перших п'яти літер Вашого прізвища. Запишіть у відкритий файл за допомогою функції **fputs()** значення лічильника та випадкові числа (тип **float**), що не більші за поточне значення лічильника. Кінцеве значення лічильника задається з клавіатури. . Перевірте правильність запису, прочитавши з файлу записані рядки за допомогою функції **fgets()** та виведіть їх на екран.

2. Розробіть програму, що відкриває файл з іменем, що складається з перших п'яти літер Вашого імені. Запишіть у відкритий файл за допомогою функції **fprintf()** значення лічильника та випадкові числа (тип **float**), що не більші за поточне значення лічильника. Кінцеве значення лічильника задається з клавіатури. Перевірте правильність запису, прочитавши з файлу записані числа за допомогою функції **fscanf()** та виведіть їх на екран.

Варіант 4_3

1. Розробіть програму, що відкриває файл з іменем, що складається з перших п'яти літер Вашого прізвища. Запишіть у відкритий файл за допомогою функції **fputs()** масив з 3-х рядків, які проініціалізуйте Вашими прізвищем, іменем та по батькові. . Перевірте правильність запису, прочитавши з файлу записані рядки за допомогою функції **fgets()** та виведіть їх на екран.

2. Розробіть програму, що відкриває файл з іменем, що складається з перших п'яти літер Вашого імені. Запишіть у відкритий файл за допомогою функції **fprintf()** попередньо визначені масиви з 8 чисел типу **char**, **int**, **float**. Перевірте правильність запису, прочитавши з файлу записані числа за допомогою функції **fscanf()** та виведіть їх на екран.

Варіант 4_4

1. Розробіть програму, що відкриває файл з іменем, що складається з перших п'яти літер Вашого прізвища. Запишіть у відкритий файл за допомогою функції **fputs()** масив з 5 рядків, який проініціалізуєте назвами країн Америки. .Перевірте правильність запису, прочитавши з файлу записані рядки за допомогою функції **fgets()** та виведіть їх на екран.
2. Розробіть програму, що відкриває файл з іменем, що складається з перших п'яти літер Вашого імені. Запишіть у відкритий файл за допомогою функції **fprintf()** попередньо визначений двовимірний масив розміром **4** на **5** чисел типу **float**. Перевірте правильність запису, прочитавши з файлу записані числа за допомогою функції **fscanf()** та виведіть їх на екран.

Варіант 4_5

1. Розробіть програму, що відкриває файл з іменем, що складається з перших п'яти літер Вашого прізвища. Запишіть у відкритий файл за допомогою функції **fputs()** масив з 4-х рядків, який проініціалізуєте назвами дисциплін, з яких треба скласти заліки. . Перевірте правильність запису, прочитавши з файлу записані рядки за допомогою функції **fgets()** та виведіть їх на екран.
2. Розробіть програму, що відкриває файл з іменем, що складається з перших п'яти літер Вашого імені. Запишіть у відкритий файл за допомогою функції **fprintf()** двовимірний масив розміром **6** на **5** випадковими числами типу **float**, що не більші за **0.5**. Перевірте правильність запису, прочитавши з файлу записані числа за допомогою функції **fscanf()** та виведіть їх на екран.

Варіант 4_6

1. Розробіть програму, що відкриває файл з іменем, що складається з перших п'яти літер Вашого прізвища. Запишіть у відкритий файл за допомогою функції **fputs()** рядки, що вводяться з клавіатури. Кількість рядків вводиться з клавіатури. . Перевірте правильність запису, для цього прочитайте з файлу записані рядки за допомогою функції **fgets()** та виведіть їх на екран.
2. Розробіть програму, що відкриває файл з іменем, що складається з перших п'яти літер Вашого імені. Запишіть у відкритий файл за допомогою функції **fprintf()** рядки, що складаються з номера рядка, випадкових чисел типу **float**, які не більші за **0.1**, та значень

квадратів цих чисел. Кінцеве значення кількості рядків задається за допомогою директиви **#define**. Перевірте правильність запису, прочитавши з файлу записані числа за допомогою функції **fscanf()** та виведіть їх на екран.

Варіант 4_7

1. Розробіть програму, що відкриває файл з іменем, що складається з перших п'яти літер Вашого прізвища. Запишіть у відкритий файл за допомогою функції **fputs()** масив з 4-х рядків, який проініціалізуєте назвами підручників. . Перевірте правильність запису, для цього прочитайте з файлу записані рядки за допомогою функції **fgets()** та виведіть їх на екран.
2. Розробіть програму, що відкриває файл з іменем, що складається з перших п'яти літер Вашого імені. Запишіть у відкритий файл за допомогою функції **fprintf()** вісім рядків, що складаються з однієї літери латинського алфавіту, випадкових чисел типу **int**, які не більші за **50**, та залишків від ділення цих чисел на число **10**. Перевірте правильність запису, прочитавши з файлу записані числа за допомогою функції **fscanf()** та виведіть їх на екран.

Варіант 4_8

1. Розробіть програму, що відкриває файл з іменем, що складається з перших п'яти літер Вашого прізвища. Запишіть у відкритий файл за допомогою функції **fputs()** масив з 5-и рядків, який проініціалізуєте прізвищами та ініціалами студентів. Перевірте правильність запису, прочитавши з файлу записані рядки за допомогою функції **fgets()** та виведіть їх на екран.
2. Розробіть програму, що відкриває файл з іменем, що складається з перших п'яти літер Вашого імені. Запишіть у відкритий файл за допомогою функції **fprintf()** шість рядків, що складаються з двох випадкових чисел типу **int**, які не більші за **10**, та трьох випадкових чисел типу **float**, що не більші за **0.1**. Перевірте правильність запису, прочитавши з файлу записані числа за допомогою функції **fscanf()** та виведіть їх на екран.

Варіант 4_9

1. Розробіть програму, що відкриває файл з іменем, що складається з перших п'яти літер Вашого прізвища. Запишіть у відкритий файл за допомогою функції **fputs()** масив з 6-и рядків, який проініціалізуєте

назвами країн Європи. Перевірте правильність запису, прочитавши файлу записані рядки за допомогою функції **fgets()** та виведіть їх на екран.

2. Розробіть програму, що відкриває файл з іменем, що складається з перших п'яти літер Вашого імені. Запишіть у відкритий файл за допомогою функції **fprintf()** рядки, що складаються з номера рядка, попередньо визначених значень масивів з 10 елементів типів **int** та **float**. Перевірте правильність запису, прочитавши з файлу записані числа за допомогою функції **fscanf()** та виведіть їх на екран.

Варіант 4_10

1. Розробіть програму, що відкриває файл з іменем, що складається з перших п'яти літер Вашого прізвища. Запишіть у відкритий файл за допомогою функції **fputs()** масив з 5-и рядків, який проініціалізуєте назвами міст Запорізької області. . Перевірте правильність запису, для цього прочитайте з файлу записані рядки за допомогою функції **fgets()** та виведіть їх на екран.

2. Розробіть програму, що відкриває файл з іменем, що складається з перших п'яти літер Вашого імені. Запишіть у відкритий файл за допомогою функції **fprintf()** рядки, що складаються з номера рядка та значень елементів рядків типу **float** з трьома знаками після точки з попередньо визначеного двовимірного масиву розміром 6 на 4. Перевірте правильність запису, прочитавши з файлу записані числа за допомогою функції **fscanf()** та виведіть їх на екран.

ПИТАННЯ ДО КОНТРОЛЬНОЇ РОБОТИ

1. Дайте визначення структури, типу структури (шаблону).
2. Визначте структуру, масив структур, покажчик на структуру. Запишіть способи доступу до елементів цих структур.
3. Як ініціалізуються структура, масив структур? Приведіть приклади.
4. Схематично покажіть розподіл пам'яті для структури **str1**:

```
struct STR{int ms[2];
           float fs;
           char *cs;}str1;
```
5. Що таке об'єднання (**union**)? Як визначити шаблон об'єднання і саме об'єднання? Чим об'єднання відрізняється від структури?
6. Як можна ініціалізувати об'єднання? Приведіть приклад.
7. Схематично покажіть розподіл пам'яті для об'єднання **un1**:

```
union UN{long w;
          char c[2];
          double d;}un1;
```
8. Із чого складається визначення функції?
9. Які параметри називають фактичними, а які формальними?
10. Що таке прототип функції? Для чого та коли він використовується?
11. Як передати в функцію масив та визначити останній елемент в масиві? Привести приклади прототипів функцій.
12. Дайте визначення функції з параметрами за умовчанням. Приведіть приклад прототипу такої функції.
13. Які функції називають перевантаженими? Наведіть приклади прототипів перевантажених функцій із сигнатурами, що розрізняються за кількістю аргументів і за типом аргументів.
14. Дайте визначення шаблону функції та параметру шаблону. Приведіть правила оголошення і використання шаблонів функції.
15. Як відкрити і як закрити потік для обміну даними? Які існують режими відкриття файлу?
16. Як записати у файл (прочитати з файлу) символ, рядок? Приведіть приклади.
17. Як здійснюється файлове введення-виведення числових даних? Приведіть приклади.
18. Як здійснюється блокове введення-виведення даних? Приведіть приклади.

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. C/C++. 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. C/C++. Структурное программирование. Практикум / Т.А. Павловская, Ю.А. Щупак. – СПб.: Питер, 2002. – 240с.
9. Культин Н.Б. С /C++ 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.