

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт із дисципліни
" Основи алгоритмізації та програмування "
для студентів спеціальності 173 – «Авіоніка»
всіх форм навчання

2020

Методичні вказівки до виконання лабораторних робіт із дисципліни " Основи алгоритмізації та програмування " для студентів спеціальності 173 "Авіоніка" всіх форм навчання. / Укл.: С.І.Арсеньєва. – Запоріжжя: НУ «Запорізька політехніка», 2020. – 26 с.

Укладачі: С.І.Арсеньєва, доцент, к.ф.-м.н.

Рецензент: Е.М. Кулініч, к.т.н., доцент

Відповідальний
за випуск зав. лаб. каф. ЕПА К.І.Пилипенко

Затверджено на засіданні
кафедри
“Електропривод та
автоматизація
промислових установок”
Протокол № 5 від 26.10.2020 р.
Рекомендовано до
видання НМКЕТФ
протокол № 3
від 19.11.2020 р.

ЗМІСТ

ПОРЯДОК ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ	4
ЗАГАЛЬНІ ВІДОМОСТІ	5
ЛАБОРАТОРНА РОБОТА № 1. Базові типи змінних. Елементарне введення-виведення	13
ЛАБОРАТОРНА РОБОТА № 2. Оператори вибору. Оператори циклу.....	16
ЛАБОРАТОРНА РОБОТА № 3. Масиви і покажчики.....	21
ЛІТЕРАТУРА.....	26

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 стандарту.
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, обираємо **Visual C++**, а справа обираємо 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, обираємо значок **C++File.cpp (Файл.cpp)**, а нижче задається ім'я нового файлу. Далі нажимаємо кнопку **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

Програма мовою Сі – це послідовність препроцесорних директив, описів та визначень глобальних змінних та функцій.

Базові типи змінних. Для визначення змінних цілого типу даних використовують ключові слова **char**, **int**, **short**, **long**. Слова **signed**, **unsigned** позначають знакове чи беззнакове ціле число, за умовчанням всі числа є знакові. Для визначення змінних дійсного типу даних використовують ключові слова **float**, **double**, **long double**.

У мові Сі визначені наступні арифметичні типи даних:

Тип даних	Розмір	Діапазон значень
char	1 байт	128 .. +127
unsigned char	1 байт	0 .. 255
short int	2 байти	-32768 .. +32767
unsigned short int	2 байти	0 .. 65535
long int	4 байти	-2147483648 ... 2147483647
unsigned long int	4 байти	0 .. 4294967295
float	4 байти	$3.4 \cdot 10^{-38}$.. $3.4 \cdot 10^{+38}$
double	8 байт	$\pm 1.7 \cdot 10^{-308}$.. $\pm 1.7 \cdot 10^{+308}$
long double	10 байт	$\pm 3.4 \cdot 10^{-4932}$... $\pm 3.4 \cdot 10^{+4932}$

Константи цілого типу за умовчанням мають тип **int**. Для завдання константи типу **long** дописується символ **L|l**, типу **unsigned** – символ **U|u**, в 8-й системі числення (СЧ) на початку константи дописується **0**, в 16-й СЧ – **0x**: **234L**, **456u**, **23456Lu**, **020**, **0x4a**.

Константи дійсного типу за умовчанням мають тип **double**. Для завдання константи типу **float** дописується символ **F|f**, для завдання константи типу **long double** дописується символ **L**: **23.4f**, **4.56E-2L**.

Препроцесорні директиви. – це записані в тексті програми інструкції, які починаються з нового рядка зі знаку **#**. Ці директиви виконуються препроцесором до компіляції програми, доповнюючи вхідний текст програми.

Директива **#include** включає в текст програми зміст зазначеного файлу. Ця директива має дві форми:

```
#include "ім'я_файлу"
```

```
#include <ім'я_файлу>
```

Директива **#define** виконує заміну ідентифікаторів константами або будь-якою послідовністю операторів (виразів), що найчастіше використовуються. Синтаксис директиви **#define** такий:

#define ідентифікатор рядок заміщення

Заміни в тексті можна відмінити за допомогою директиви:

#undef ідентифікатор

Наприклад, перша директива замінить у тексті програми кожний символ **W** на число **80**, а друга директива відмінить цю заміну.

#define W 80

#undef W

Елементарні введення та виведення даних.

Форматоване виведення даних здійснюється функцією:

printf(рядок_формату, список_виведення);

Список_виведення може містити константи, змінні та будь-які вирази, розділені комами. **Рядок_формату** записується в подвійних лапках і може містити: послідовність звичайних символів, що виводяться на екран; спеціальні символи, що керують розміщенням даних на екрані (**ESC**–послідовності); специфікатори формату даних.

Специфікатори формату – це наступна послідовність:

%[довжина поля] [.точність] [1 або L] специфікатор_типу

де **[довжина поля]** – кількість позицій для виведення; **[.точність]** – кількість цифр дробової частини дійсних чисел, чи кількість символів, що виводяться, для рядків; **1** або **L** – позначка для чисел типу **long**, **double**, **long double**.

Для виведення даних використовують такі специфікатори типів:

%c – ціле число типу **char**, **int**, **short** у вигляді символу;

%d – ціле число типу **char**, **int**, **short** у 10-ій системі числення (СЧ);

%u – беззнакове ціле число у 10-ій СЧ;

%o – ціле число у 8-ій СЧ;

%x – ціле число в 16-ій СЧ;

%f – дійсне число типу **float**;

%e – дійсне число типу **float** в експоненціальній формі;

%lf – дійсне число типу **double**;

%le – дійсне число типу **double** в експоненціальній формі;

%Lf – дійсне число типу **long double**;

%Le – дійсне число типу **long double** в експоненціальній формі;

%s – рядок; **%p** – адреса.

Форматоване введення даних здійснюється функцією:

scanf (рядок_формату, список_введення) ;

Список_введення може містити тільки адреси змінних, раніш визначених у програмі. **Рядок_формату** записується в подвійних лапках і може містити тільки специфікатори типів даних.

Функція **scanf ()** здійснює введення чисел, символів та рядків з клавіатури до натискання клавіші **<Enter>** або до виявлення першого неприпустимого символу у вхідному потоці.

```
int a; float b; char c; char st[20];
scanf("%d %f %c %s",&a,&b,&c,st);
printf("a=%d, b=%f, c=%c st=%s \n",a,b,c,st);
```

Неформатоване введення символів здійснюється функціями **getchar ()**, **getch ()**, **getche ()**, а неформатоване виведення символів – **putchar ()**, **putch ()**.

```
char c1,c2,c3;
c1=getchar();   c2=getch();   c3=getche();
putchar(c1);    putch(c2);
```

Неформатоване введення рядків виконується функцією **gets ()**, а неформатоване виведення рядків – **puts ()**.

```
char st1[20];   gets(st1);   puts(st1);
```

Введення-виведення даних в C++ здійснюється за допомогою операції **cin>>**, а виведення даних – за допомогою операції **cout<<**.

```
int a; cout<<"Input a"<<endl;//виведення на екран повідомлення
cin>>a; // введення з клавіатури значення змінної a
cout<<a; // виведення на екран введенного значення
```

ESC-послідовності (керуючі послідовності). Це послідовності символів, які використовують для запису неграфічних та деяких інших символів в рядках та символічних константах. **ESC**-послідовність складається з нахиленої риски вліво (зворотній слеш), після якого записується буква або знак.

\n	новий рядок;
\t	горизонтальна табуляція;
\a	дзвоник (сигнал);
\'	одиначні лапки;
\"	подвійні лапки;
\\	нахилена вліво риска;
\?	знак питання.

Умовний оператор if...else обирає один з двох варіантів виразів обчислень та має такий синтаксис:

```
if(умова) вираз_1 else вираз_2
```

Якщо **умова** – істина, то виконується **вираз_1**, інакше – **вираз_2**.

Будь-яке значення **умови**, крім 0, вважається істиною. Наприклад:

```
if(x>y) {max=x; min=y;}  
else {max=y; min=x;}
```

Допустима скорочена форма умовного оператору, в якій відсутня конструкція **else вираз_2**. В цьому разі при хибності перевіряємої умови ніяких дій не виконується.

Оператор-перемикач switch дозволяє обрати один з декількох варіантів обчислень та повідомлень. Синтаксис цього оператора такий:

```
switch(перемикаючий_вираз)  
{case константний_вираз_1: оператори_1; break;  
  case константний_вираз_2: оператори_2; break;  
  . . .  
default: оператори_n+1;}
```

Оператор **switch** порівнює значення **перемикаючого виразу**, який може приймати тільки цілі значення, з **константними виразами** та передає керування тому оператору, **константний вираз** якого співпадає зі значенням **перемикаючого виразу**. Якщо такі оператори відсутні, то виконуються оператори, що позначені міткою **default**. Оператор **break** здійснює примусовий вихід з оператора **switch**. Якщо оператор **break** відсутній, то виконуються не тільки оператори відповідної гілки **case**, а й усі інші гілки до кінця оператора **switch**. Оператори **break** та **default** необов'язкові в операторі **switch**. Наприклад:

```
unsigned int a;  
scanf("%u",&a); // введення з клавіатури значення змінної a  
switch(a)  
{case 0: printf ("це нуль"); break;  
  case 1: printf ("це одиниця"); break;  
  case 2: printf ("це двійка"); break;  
  default: printf ("це більше двох");}
```

Оператори циклу. Мова Сі (Cі++) має три оператори, за допомогою яких можна організувати цикли: **for**, **while**, **do...while**.

Цикл з оператором for частіше використовується у випадку, коли відома точна кількість повторів. Цей оператор має синтаксис:

```
for(ініціалізація_циклу; умова; список_виразів) тіло_циклу
```

Ініціалізація_циклу – це послідовність визначень та виразів розділених комами, що виконуються один раз перед початком виконання циклу. Потім перевіряється **умова**. Якщо значення **умови** не дорівнює 0, тобто є істиною, то виконується **тіло_циклу**, а потім **список виразів** та знову перевіряється **умова**. При хибному значенні **умови** цикл завершується. Наприклад:

```
for(int i=0,sum=0,pr=1; i<3; i++) // i=1, 2, 3
{sum+=5; pr*=(i+1);} // sum=5, 10, 15 pr=1, 2, 6
```

Цикл з оператором while використовується в тому випадку, коли невідоме точне число повторів. Синтаксис оператора наступний:

while(умова) тіло_циклу

Спочатку перевіряється **умова**, якщо **умова** є істиною, то виконується **тіло_циклу** і знову перевіряється **умова**. При хибному значенні **умови** цикл завершується. Наприклад:

```
int a=0, sum=0;
while(sum<20)
{a+=2; sum+=a; } // a=2, 4, 6, 8 sum=2, 6, 12, 20
```

Цикл з оператором do...while використовується в тих випадках, коли **тіло_циклу** повинно бути виконане, принаймні, один раз. Синтаксис оператора наступний:

do тіло_циклу while(умова);

Спочатку виконується **тіло_циклу**, потім перевіряється **умова**. Якщо **умова** є істиною, то знову виконується **тіло_циклу**. При хибному значенні **умови** цикл завершується. Наприклад:

```
int m=0,n=0;
do {m+=5; n++;} while(n<3); // m=5, 10, 15 n=1, 2, 3
```

Оператори циклу можуть мати оператори continue, break, return. Оператор **continue** пропустить залишок тіла циклу та почне виконання наступної ітерації (повторення). Оператор **break** завершить виконання усього циклу. Оператор **return** завершить виконання функції або всієї програми.

```
int n1=0,n2=0,a,sum=0;
while((n1+n2)<20) //перевірка кількості введених значень змінної a
{scanf("%d",&a); // введення 20 значень змінної a
if((a%2)==0)
{ n2++; continue;} // перехід до наступної ітерації, якщо a парне,
if(a==99) break; // закінчення циклу, якщо a=99
if(a==100) return; // закінчення програми, якщо a=100
n1++; sum+=a; } // обчислення суми непарних змінних a
```

Оператор безумовного переходу `goto` передає керування на будь-який позначений міткою оператор в тілі функції. При використанні оператора `goto` не можна входити усередину умовного оператора, перемикача, циклу. Наприклад:

```
if (n==n_max) goto M1; // перехід на M1 при n==n_max
n++;
M1: printf("n=%d", n);
```

Одновимірні масиви. Масив – це сукупність змінних одного типу, що об'єднані під одним ім'ям. При визначенні масиву задають його розмір, значення розміру повинно бути константним цілочисловим виразом. Доступ до елементів масиву може здійснюватися через індекси, причому перший елемент масиву має індекс **0**. Наприклад:

```
int a[3];
a[0]=2; a[1]=6; a[2]=a[0]+a[1]*5; // a[2]=32
```

Показчики. Це цілі беззнакові змінні, що містять адреси пам'яті якого-небудь об'єкту програми. Показчик може містити адресу змінної, масиву, структури, іншого показчика і т.п. Розмір показчика 4 байти: 2 байта – сегмент і 2 байта – зсув в сегменті. Тип показчика – це тип змінної, адреса якої може знаходитися в показчику. Показчик можна ініціалізувати та йому можна привласнювати адресу визначеної змінної, значення раніш визначеного іншого показчика або явно задану адресу пам'яті (з перетворенням типу). Наприклад:

```
int a=1;
int *ptr_a; ptr_a=&a; // адреса змінної a
int *ptr_b; ptr_b=ptr_a; // показчик ptr_a
int *ptr_c=(int*)0xB8000000; // явна адреса пам'яті
```

Ім'я масиву – це показчик-константа, значення якого є адреса нульового елемента масиву. Це значення не можна змінити. Наприклад, якщо визначити `int arr[10]`, то при цьому виділиться пам'ять для 10 елементів масиву, а ім'я `arr` стане показчиком-константою на цей масив. При визначенні розміру показчика `arr` визначається розмір усього масиву. Ім'я масиву можна привласнювати іншим показчикам. З елементами масивів, зв'язаними з показчиками, можна працювати за допомогою індексів і операції розіменування. Наприклад, доступ до останнього елемента масиву можна виконати таким чином:

```
int arr[10]; int *ptr=arr;
arr[9]=2; *(arr+9)=2; *(9+arr)=2; 9[arr]=2;
ptr[9]=2; *(ptr+9)=2; *(9+ptr)=2; 9[ptr]=2;
```

Рядок. Це послідовність елементів типу **char**, останнім елементом якої є **'\0'**. Ініціалізується рядок послідовністю символів у парних подвійних лапках. Адреса першого символу може використовуватися для ініціалізації масиву типу **char**, у цьому випадку ім'я масиву й адреса першого символу стають синонімами. Наприклад:

```
char str1[5]="name"; char str2[]="name";  
char str3[5]={'n','a','m','e','\0'};
```

Адреса першого елемента рядка може використовуватися для ініціалізації покажчика типу **char**, наприклад:

```
char *strptr="name" ;
```

У цьому випадку змінна-покажчик **strptr** (але не покажчик-константа) одержує значення адреси символу **'n'**, а сам рядок розташовується в сегменті даних завантажувального модуля програми.

Масиви покажчиків. Це масиви, елементами якого є покажчики. Прикладом визначення масиву з трьох покажчиків на об'єкти типу **int** є запис: **int*mas[3]={&a,&b,&c}**; Ім'я масиву – **mas**, він містить три покажчика типу **int***. Вираз **(mas+1)** відповідає зсуву у пам'яті на **sizeof(int*)** байтів від початку масиву (тобто на 4 байти).

Масиви покажчиків на рядки частіше використовуються для раціонального розміщення в пам'яті й сортування рядків із неоднаковими розмірами. Наприклад:

```
char *name[3]={"Іванов","Петро","Єгорович"};
```

При такому визначенні в пам'яті виділяється **3*sizeof(char*)=12** байт для покажчиків **name[0]**, **name[1]**, **name[2]**, 7 байт – для рядка **"Іванов"**, 6 – для **"Петро"** і 9 – для **"Єгорович"**.

Багатовимірний масив. Це масив масивів, тобто такий масив, елементами якого є масиви. Ім'я багатовимірного масиву є покажчик-константа на масив покажчиків-констант, елементами якого є покажчики-константи з адресами початку кожного з рядків масиву елементів. Елементи багатовимірного масиву розташовуються в пам'яті за рядками, тобто в порядку зростання найправішого індексу. Доступ до елементів двовимірного масиву здійснюється за індексом або за допомогою покажчиків. Наприклад, звернення до **arr[i][j]** еквівалентно ***(arr[i]+j)**, ***(*(arr+i)+j)** або **((*(arr+i))[j])**.

Ініціалізація масивів. Привласнення початкових значень елементам масивів при їх визначенні, тобто ініціалізація масивів, може здійснюватися за умовчанням та явно.

За умовчанням глобальні і статичні масиви ініціалізуються нулями, автоматичні – довільними даними. Явна ініціалізація буває повна (з явним завданням кількості елементів і початкових значень усіх елементів масиву) та неповна (з явним завданням кількості елементів, але початкові значення задаються не для всіх елементів масиву):

```
int a[5]={2,5,7,9,1}; // явна повна ініціалізація
int b[7]={5,7,9};      // явна неповна ініціалізація
```

У останньому рядку інші чотири елементи масиву **b** будуть мати або нульові значення (для зовнішнього і статичного масивів), або їхні значення будуть невизначені (для автоматичного масиву).

При повній явній ініціалізації одновимірного масиву можливо не задавати розмір масиву. Компілятор визначає кількість елементів масиву за списком ініціалізації:

```
int c[]={5,7,9,8}; // Масив із чотирьох елементів
```

Нижче наведені приклади ініціалізації двовимірних масивів.

```
int m[3][2]={2,4,6,8,1,3}; // явна повна ініціалізація
int n[2][3]={ {1,2}, {3,5}, {2} }; // явна неповна ініціалізація
```

Багатовимірні масиви можна ініціалізувати без вказівки найлівої розмірності. Компілятор визначає кількість елементів за списком ініціалізації:

```
int p[][3]={ {3,2,1}, {3,8,9}, {2,7,4} };
float f[][2]={ {1.2}, {2.2,3.4}, {2.5,6.4}, {2.0} };
```

В цьому прикладі визначені масиви **int p[3][3]** і **float f[4][2]**.

Масиви динамічної пам'яті. Пам'ять для збереження даних може виділятися динамічно в так званій "купі", розмір і положення якої залежать від моделі пам'яті. Це зручно у випадку, коли невідомо, скільки об'єктів буде створено при роботі програми. При цьому розмір масиву можна визначити як змінну.

В мові Сі використовується функція **malloc()** для запиту і виділення пам'яті, кількість якої задається в байтах, і функція **free()** для її звільнення. Прототипи цих функцій описані в заголовному файлі **<stdlib.h>**. Наприклад:

```
int n; scanf("%d",&n); //введення розміру масиву (розмір – змінна)
int *ptri=(int*)malloc(n*sizeof(int)); //запит пам'яті
...
free(ptri); // звільнення пам'яті
```

У мові С++ для цієї мети існують операції **new** і **delete**. Операція **new** автоматично визначає необхідний обсяг пам'яті в байтах.

```
double *ptrd =new double[n]; //запит пам'яті
...
delete[]ptrd; // звільнення пам'яті
```

ЛАБОРАТОРНА РОБОТА № 1

Базові типи змінних.

Елементарне введення-виведення

Мета роботи – ознайомитися з основними типами даних, які використовуються в мові Cі, засвоїти елементарні функції введення-виведення.

Програма 1.1 Програма визначає код символу за кодовою таблицею **ASCII** та визначає розмір типу даних у байтах.

```
#include<stdio.h>
#include<conio.h>
int main()
{puts("My first program"); //виведення повідомлення
  char s='a';
  printf("Symbol %c =( %d)10S =( %#o) 8S ",s,s,s);
  printf("( %#x)16S size char=%d bait\n",s,sizeof(s));
  int a=78;
  printf("a=%d size int=%d bait \n",a,sizeof(a));
  float b=56.23;
  printf("b=%f size float=%d bait \n",b,sizeof(b));
  puts(" (\nPress any key and Enter...\n");
  getchar(); //чекання натискання будь-якої клавіші з <Enter>
  return 0;}
```

Завдання до програми 1.1

Доповніть програму: виведіть на екран своє прізвище та першу літеру Вашого імені з кодами; визначте нові змінні типу **short**, **long**, **double**, **long double** з будь-якими значеннями та виведіть на екран їх значення та розмір в байтах; замініть **getchar()**; на **getch()**;

***Програма 1.2** Програма виконує прості арифметичні дії.

```
#include <stdio.h>
#include <iostream>
int main()
{int a1=9, a2=2, b1, b2, b3;
  float f1=3.6, f2=0.5, r1, r2;
  a2+=a1;          // a2=2+9=11
  b1=a2/4;         // b1=11/4=2
  b2=a2%4;         // b2=11%4=3
  b3=b1>b2;        // b3= (2>3)? ні 0
  r1=f1/f2;        // r1=3.6/0.5=7.2
  r2=(f2>f1)?f1+2:--f2; // r2=(0.5>3.6)? ні (0.5-1)=-0.5
  printf("a2=%d b1=%d b2=%d b3=%d\n",a2,b1,b2,b3);
  printf("r1=%5.2f r2=%5.2f\n",r1,r2);}
```

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

*Завдання до програми 1.2

Розробіть програму, що обчислює задані вирази. Значення змінних задає викладач. Результати поясніть за допомогою коментарів.

1. $a2 = -a1$; $b1 = a1 + 2 \cdot a2^2$; $b2 = !a1$; $b3 = b2++$;
 $r1 = f1 \cdot f2$; $r2 = (f2 \neq f1) ? f1 - 0.5 : f2 + 0.5$;
2. $a2 *= 10$; $b1 = a1^3 + a2$; $b2 = a1 > a2$; $b3 = ++b2$;
 $r1 = 2 \cdot f1 - f2$; $r2 = (f2 == f1) ? f1 \cdot 0.5 : f2 \cdot 0.5$;
3. $a2 /= 5$; $b1 = 5 \cdot a1 + a2^2$; $b2 = a1 > a2$; $b3 = --b2$;
 $r1 = f1 + 4 \cdot f2$; $r2 = (f2 \leq f1) ? f1 - 2.5 : f2 + 5.5$;
4. $a2 += a1 + 5$; $b1 = 3 \cdot a1 + a2^3$; $b2 = a1 < a2$; $b3 = b2--$;
 $r1 = f1 / f2 \cdot 2$; $r2 = (f2 >= f1) ? f1 - 5.5 : f2 + 3.5$;
5. $a2 = -a1 + 5$; $b1 = a1 + a2^2$; $b2 = a1 >= a2$; $b3 = --b2 + 3$;
 $r1 = f1 + f2 / 2$; $r2 = (f2 \neq f1) ? f1 - 0.5 : f2 + 0.5$;

Програма 1.3 Програма за введеними з клавіатури значеннями радіусу та висоти обчислює та виводить на екран об'єм циліндра.

```
#include <stdio.h>
#include <iostream>
int main()
{float h,r,V;
 puts("Input r, h");
 scanf("%f %f", &r,&h);
 V=3.141592f*r*r*h;
 printf("V=%8.3f\n",V);
 system("pause");
 return 0;}
```

Завдання до програми 1.3

Розробіть програму, що виконує задані дії та виводить результат в заданому форматі (перша цифра в специфікаторі формату – загальна кількість позицій для виведення даних, включаючи точку та знак числа).

1. Введіть з клавіатури змінні **x, z** (типу **float**), обчисліть вираз $res = (4 \cdot x - z)^2 / 3$ та виведіть на екран результат з трьома цифрами цілої частини і двома – дробової.
2. Введіть з клавіатури змінні **a, b, c** (типу **double**), обчисліть вираз $res = a + 2 \cdot b - c^2$ та виведіть на екран результат з двома цифрами цілої частини і трьома – дробової.
3. Введіть з клавіатури змінні **k, g** (типу **double**), обчисліть вираз

$res = k^2 + (g - k) / 2$ та виведіть на екран результат з чотирма цифрами цілої частини і трьома – дробової.

4. Введіть з клавіатури змінні **v, s, f** (типу **float**), обчисліть вираз $res = s^3 / v - f * v$ та виведіть на екран результат з двома цифрами цілої частини і чотирма – дробової.

5. Введіть з клавіатури змінні **x, z, h** (типу **double**), обчисліть вираз $res = x^3 + z / h^2$ та виведіть на екран результат з трьома цифрами цілої частини і трьома – дробової.

Програма 1.4 Програма за введенням з клавіатури значенням кількості дюймів виводить на екран відповідне значення сантиметрів. Значення цілої змінної **d** вводиться за допомогою операції **cin>>**, а результат виводиться на екран за допомогою операції **cout<<**

```
#include <iostream>
using namespace std;
int main()
{float d, D;
  cout<<"Input the number of inches\n";//введіть кіл-ть дюймів
  cin>>d;
  D=d*2.54f;
  cout<<d<<" inches = "<<D<<" cm\n\n";
  system("pause");
  return 0;}
```

Завдання до програми 1.4

Розробіть програму, яка виконує задані обчислення. Значення змінних введіть та виведіть за допомогою операцій **cin>>** та **cout<<**.

1. Обчисліть об'єм куба $V = r^3$.
2. Обчисліть площу квадрата $S = a^2$.
3. Обчисліть довжину кола $L = 2\pi r$.
4. Обчисліть площу прямокутника $S = a * b$.
5. Обчисліть площу круга $S = \pi r^2$.

Контрольні питання до лабораторної роботи № 1

1. Які типи даних використовуються в мові C++? Скільки байт займає кожен із типів даних у мові Cі?
2. Введіть та виведіть різними способами цілі та дійсні числа.
3. Введіть та виведіть різними способами символ. Поясніть різницю.
4. Введіть та виведіть різними способами рядок. Поясніть різницю.

ЛАБОРАТОРНА РОБОТА № 2

Оператори вибору. Оператори циклу

Мета роботи – ознайомитися з умовними операторами `if...else`, операторами-перемикачами, операторами циклу `while()`, `do...while()`, `for()`, навчитися розробляти найпростіші програми, які розгалужуються, циклічно виконуються.

Програма 2.1 Програма визначає, чи є введене число парним.

```
#include<stdio.h>
#include <iostream>
int main()
{setlocale(LC_ALL,"Russian");
  int n;
  puts("Введите целое число:");
  scanf("%d",&n);
  if(n%2==0) printf("Число %d четное\n\n", n);
  else printf("Число %d нечетное\n\n", n);
  system("pause"); return 0;}
```

Завдання до програми 2.1

Розробіть програму, яка виконує задані обчислення та виводить на екран відповідні повідомлення.

1. Перевірте, чи більше 100 введене з клавіатури ціле число.
2. Перевірте, чи менше 20 введене з клавіатури ціле число.
3. Перевірте, чи закінчується на 3 введене з клавіатури ціле число.
4. Перевірте, чи закінчується на 0 введене з клавіатури ціле число.
5. Перевірте, чи ділиться на 5 введене з клавіатури ціле число.

Програма 2.2 Програма визначає, чи є введене число додатним та більшим чи меншим заданого числа.

```
#include<stdio.h>
#include<conio.h>
int main()
{float x;
  puts("Input value:");
  scanf("%f",&x);
  if(x>0) if(x>100) printf("x %.2f>0 and >100\n",x);
           else     printf("x %.2f>0 and <=100\n",x);
  else    if(x<-100) printf("x %.2f <=0 and <-100\n",x);
           else     printf("x %.2f <=0 and >=-100\n",x);
  puts("\nPress any key...");
  getch();
  return 0;}
```

Завдання до програми 2.2

Розробіть програму, яка виконує задані обчислення, використовуючи вкладений умовний оператор, та виводить на екран чотири відповідні повідомлення.

1. Введіть з клавіатури ціле число, якщо воно парне, перевірте чи є додатним, якщо – непарне, чи більше 25.
2. Введіть з клавіатури два числа, для більшого з них числа перевірте, чи знаходиться воно в діапазоні від 5 до 30.
3. Введіть з клавіатури ціле число, якщо воно більше 50, перевірте, чи кратне воно 4, інакше – чи кратне воно 3.
4. Введіть з клавіатури два цілих числа, для меншого з них числа перевірте, чи є воно непарним.
5. Введіть з клавіатури ціле число, якщо воно додатне, перевірте, чи кратне 5, інакше – чи менше воно за -100.

Програма 2.3 Програма в залежності від введенного знаку визначає результат виконання дії за допомогою оператора `switch()` із застосуванням оператора `break`.

```
#include <stdio.h>
#include <conio.h>
int main()
{float a,b;
 char s;
 puts("Input value a, b:");
 scanf("%f %f",&a,&b);
 fflush(stdin);
 puts("Input sing (-, +, *, /):");
 s=getchar();
 switch(s)
 {case '-': printf("%.2f-%.2f=%.2f\n",a,b,a-b); break;
 case '+': printf("%.2f+%.2f=%.2f\n",a,b,a+b); break;
 case '*': printf("%.2f*%.2f=%.2f\n",a,b,a*b); break;
 case '/': printf("%.2f/%.2f=%.2f\n",a,b,a/b); break;
 default: printf("Error! sings (-, +, *, /)\n");
 }
 puts("\nPress any key..."); getch();return 0;}
```

Завдання до програми 2.3

Розробіть програму з використанням оператора `switch()`, яка згідно з введеною з клавіатури першої букви виводить на екран задану повну назви або повідомлення про помилку, якщо введена буква не співпадає з заданими.

1. Назви 5 фільмів.
2. Назви 6 книг.
3. Назви 7 країн Європи.
4. Назви 5 квітів.
5. Назви 6 дерев.

Програма 2.4 Програма обчислює суму ряду цілих парних чисел, доки сума не стане більше 20, використовуючи оператор **while**.

```
#include <stdio.h>
#include <conio.h>
int main()
{int a=0, sum=0, n=0;
  while (sum<20)
  {n+=1;
   a+=2;
   sum+=a;
   printf("n=%d a=%d sum=%d\n",n,a,sum); }
  printf("Loops n=%d\n", n);
  puts("\nPress any key..."); getch();
  return 0;}
```

Завдання до програми 2.4

Розробіть програму, що виконує задані обчислення та виводить на екран результат обчислення, останній член та кількість циклів.

1. Обчисліть суму ряду цілих чисел, кратних 3, доки останнє число не стане більше 21.
2. Обчисліть суму цілих чисел, що вводяться з клавіатури, доки сума не стане більше 100.
3. Обчисліть суму ряду цілих чисел, кратних 4, доки останнє число не стане більше 36.
4. Обчисліть суму цілих чисел, що вводяться з клавіатури, доки сума не стане більше 50.
5. Обчисліть суму ряду цілих чисел, кратних 5. Кількість циклів вводиться з клавіатури.

Програма 2.5 Програма обчислює квадрати послідовності чисел.

```
#include <stdio.h>
#include <conio.h>
int main()
{ int i, b, a=0,kol;
  puts("Input number of cycles ");
  scanf("%d",&kol);
```

```

for(i=0; i<kol; i++)
{if(a>500){puts("b>250000"); break;}
 a+=1; b=a*a;
 printf("a=%3d b=%d\n",a,b);
}
 puts("\nPress any key..."); getch(); return 0;}

```

Завдання до програми 2.5

Розробіть програму, що виконує задані обчислення та виводить на екран результат обчислення за допомогою оператора циклу **for()**. Загальна кількість чисел та числа вводяться з клавіатури. При введенні заданого числа цикл завершується (оператор **break**).

1. Обчисліть середнє арифметичне (тип **float**) тільки цілих додатних чисел. При введенні числа **-11** цикл завершується.
2. Обчисліть суму тільки цілих додатних парних чисел. При введенні числа **55** цикл завершується.
3. Обчисліть середнє арифметичне тільки додатних дійсних чисел. При введенні числа, що більше **100** цикл завершується.
4. Обчисліть середнє арифметичне тільки від'ємних дійсних чисел. При введенні числа менше **-99** цикл завершується.
5. Обчисліть середнє арифметичне (тип **float**) тільки цілих від'ємних непарних чисел. При введенні числа **13** цикл завершується.

Програма 2.6 Програма обчислює та виводить на екран частину таблиці множення, використовуючи вкладений оператор циклу.

```

#include <stdio.h>
#include <conio.h>
int main()
{int i,j;
 for(i=1; i<=10; i++)
 {for(j=1; j<=5; j++)
  printf("%d * %d = %2d\n",i,j,i*j);
  printf("\n"); getch();
 }
 puts("\nPress any key..."); getch();
 return 0;}

```

Завдання до програми 2.6

Обчисліть та виведіть на екран значення заданої функції **F**, використовуючи вкладений оператор циклу, якщо **a**, **b** змінюються у заданих діапазонах з заданим кроком.

1. $F=a^2+b^2$; діапазон зміни **a** від **4** до **20** з кроком **2**;
діапазон зміни **b** від **3** до **21** з кроком **3**.

2. $F=a^2-4b$; діапазон зміни **a** від 2 до 10 з кроком 1;
діапазон зміни **b** від 5 до 20 з кроком 5.
3. $F=a^2-b^2$; діапазон зміни **a** від 0 до 40 з кроком 4;
діапазон зміни **b** від 6 до 20 з кроком 2.
4. $F=2a^2+b$; діапазон зміни **a** від 3 до 30 з кроком 3;
діапазон зміни **b** від 8 до 20 з кроком 4.
5. $F=5a+b^2$; діапазон зміни **a** від 5 до 50 з кроком 5;
діапазон зміни **b** від 4 до 10 з кроком 2.

Контрольні питання до лабораторної роботи № 2

1. Які оператори вибору Ви знаєте? Для чого вони призначені?
2. Яке значення може прийняти вираз умови в операторі **if (умова)**? Що є істиною, а що хибністю?
3. Які повідомлення виведуться в фрагменті програми? Поясніть.

```
int x=5,y=6;
if(x==3) puts("AA1");
else puts("BB2");
if(x=10, x<y) puts("CC3");
else puts("DD4");
if(x<y, x=-4) puts("EE5");
else puts("FF6");
```
4. Які значення може приймати **вираз** в операторі **switch(вираз)**?
5. Яке призначення оператора **break**, мітки **default** у операторі-перемикачі? Чи завжди вони потрібні? Що станеться при їхвилученні?
6. Які оператори циклу Ви знаєте? Чим вони відрізняються?
7. Поясніть призначення операторів **continue**, **break**, **return** та наведіть приклади їх використання в операторах циклу.
8. Замініть оператор **for()** у фрагменті програми на оператори **while()**, **do...while()**.

```
int i, a=2,res;
for(i=0;i<3;i++)
{res=i*i-a*a; a+=2; printf ("%d\n",res);}
```
9. Як виконується оператор **for** із роздільником **кома**? Який буде результат роботи наступного фрагменту програми? Поясніть.

```
int a,b,c;
for(a=0,b=0, c=5; c>10,a<3; a++,b+=c)
{c+=3; printf("a=%d b=%d c=%d\n",a,b,c);}
printf("Результат a=%d b=%d c=%d\n",a,b,c);
```

ЛАБОРАТОРНА РОБОТА № 3

Масиви і покажчики

Мета роботи – ознайомитися з формами визначення одновимірних масивів у мові Сі, використанням покажчиків при роботі з масивами, навчитися формувати числові і символічні масиви.

Програма 3.1 У програмі визначається та заповнюється випадковими значеннями одновимірний масив. Розглядається розміщення масиву в пам'яті та визначається його розмір. Визначається мінімальний за значенням елемент масиву.

```
#include<iostream>
#include<time.h> //для запуску генератора випадкових чисел
#include<stdlib.h> //для підключення генератора випадкових чисел
#include <stdio.h>
int main()
{int arr[10], min, i;
  srand( (unsigned) time(NULL)); // запуск генератора випадк. чисел
  for(i=0; i<10; i++)
    {arr[i]=rand()%101-50; // значення від -50 до +50
    printf("arr[%d]=%3d  %p\n",i,arr[i],&arr[i]); }
  printf("Size massiv arr: %d bait\n",sizeof(arr));
  min=arr[0];
  for(i=0; i<10; i++)
    if(arr[i]<min) min=arr[i];
  printf("min = %d\n", min);
  system("pause");  return 0;}
```

Завдання до програми 3.1

Розробіть програму, в якій з клавіатури вводиться масив цілих чисел заданого розміру, виконуються задані обчислення. На екран виводяться значення елементів масиву і результати обчислень.

1. Обчисліть середнє арифметичне тільки ненульових елементів. Розмір масиву 6.
2. Обчисліть середнє арифметичне елементів без обліку мінімального елемента масиву. Розмір масиву 8.
3. Обчисліть середнє арифметичне тільки додатних елементів. Розмір масиву 5.
4. Обчисліть середнє арифметичне тільки парних елементів. Розмір масиву 6.
5. Обчисліть суму квадратів від'ємних елементів. Розмір масиву 7.

Програма 3.2 В програмі визначається значення максимального елементу масиву, використовуючи метод доступу через розіменування покажчика.

```
#include<stdio.h>
#include<conio.h>
int main()
{int arr[10]={5,3,2,-4,6,7,110,-17,0,13}, max, i=0;
  int *px= arr; // визначення покажчика типу int на масив
  max=*px; //привласнення max значення нульового елемента масиву
  while(i<10)
    {if(*(px+i)>max) max=*(px+i);
      i++; }
  printf("max=%d\n", max);
  puts("\nPress any key..."); getch();
  return 0;}
```

Завдання до програми 3.2

Розробіть програму, в якій визначається масив цілих чисел заданого розміру. Перший елемент масиву задається, всі інші елементи заповнюються випадковими значеннями, потім виконуються задані обчислення, використовуючи метод доступу до елементів масиву через розіменування покажчика. На екран виводяться значення елементів масиву і результати обчислень.

1. Визначте мінімальний елемент масиву, що кратний 5. Перший елемент масиву дорівнює 500. Розмір масиву 8.
2. Визначте номер мінімального додатного елемента масиву. Перший елемент масиву дорівнює 1000. Розмір масиву 6.
3. Визначте номер максимального від'ємного елемента масиву. Перший елемент масиву дорівнює -2000. Розмір масиву 7.
4. Визначте максимальний парний елемент масиву. Перший елемент масиву дорівнює -100. Розмір масиву 8.
5. Визначте мінімальний непарний елемент масиву. Перший елемент масиву дорівнює 201. Розмір масиву 7.

Програма 3.3 Програма обчислює суму значень елементів двовимірної масиву в рядках та стовпцях.

```
#include <stdio.h>
#include <conio.h>
int arr[4][3]={ {2,3,4},{7},{1,9},{8,4,10}},s1[4],s2[3];
int main()
{int i,j;
```

```

puts("Array arr[4][3] ");
for(i=0;i<4;i++)
{for(j=0;j<3;j++)
    printf("%5d",arr[i][j]);
    printf("\n");}
printf("\n Sum elements strings\n");
for(i=0;i<4;i++)
{for(j=0;j<3;j++)    s1[i]+=*(arr+i+j);
    printf("s1[%d]=%d\n",i,s1[i]);}
printf("\n Sum elements columns\n");
for(j=0;j<3;j++)
{for(i=0;i<4;i++)    s2[j]+=(arr[i]+j);
    printf("s2[%d]=%d\n",j,s2[j]);}
puts("\nPress any key..."); getch(); return 0;}

```

Завдання до програми 3.3

Розробіть програму, в якій вводиться із клавіатури і виводиться на екран двовимірний масив цілих чисел заданого розміру та виконуються задані обчислення.

1. Обчисліть суму додатних непарних елементів в стовбцях масиву. Розмір масиву 5 на 4.
2. Обчисліть кількість додатних парних елементів в рядках масиву. Розмір масиву 3 на 6.
3. Обчисліть кількість непарних елементів в стовбцях масиву. Розмір масиву 5 на 3.
4. Обчисліть суму від'ємних елементів в рядках. Розмір масиву 3 на 5.
5. Обчисліть добуток елементів в діапазоні від 1 до 5 в стовбцях. Розмір масиву 4 на 5.

Програма 3.4 В програмі створюється динамічний масив для змінних типу `int`. Розмір масиву та значення елементів масиву вводяться з клавіатури. По закінченні роботи з масивом пам'ять звільняється.

```

#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
int main()
{int n,i;
    int *ptr; //визначення покажчика на масив чисел типу int
    puts("Input size array");
    scanf("%d",&n);
    //запит пам'яті
    ptr=new int[n]; // ptr=(int*)malloc(n*sizeof(int));

```

```

if(ptr==NULL) //перевірка виділення пам'яті
    {puts("Error"); return -1;}
puts("\nInput array");
for(i=0;i<n;i++) scanf("%d",&ptr[i]);
puts("\nArray");
for(i=0;i<n;i++) printf("%d ",*(ptr+i));
delete [] ptr; // free(ptr); //звільнення пам'яті
puts("\nPress any key..."); getch();
return 0;}

```

Завдання до програми 3.4

Розробіть програму, в якій створюється динамічний масив для змінних заданого типу з використанням операції **new** або функції **malloc**. Розмір масиву вводиться з клавіатури. Масив заповнюється випадковими числами у заданому діапазоні та виводиться на екран. Потім звільняється пам'ять.

1. Створіть динамічний масив для змінних типу **short** з використанням операції **new**. Масив заповніть випадковими числами у діапазоні від **-5000** до **+5000**.
2. Створіть динамічний масив для змінних типу **float** з використанням функції **malloc**. Масив заповніть випадковими числами у діапазоні від **-50.5** до **+50.5**.
3. Створіть динамічний масив для змінних типу **double** з використанням операції **new**. Масив заповніть випадковими числами у діапазоні від **0.00** до **32.00**.
4. Створіть динамічний масив для змінних типу **char** з використанням функції **malloc**. Масив заповніть випадковими числами у діапазоні від **60** до **125**. Масив виведіть на екран десятковому та символьному вигляді.
5. Створіть динамічний масив для змінних типу **double** з використанням операції **new**. Масив заповніть випадковими числами у діапазоні від **-15.000** до **15.000**.

Контрольні питання до лабораторної роботи № 3

1. Що таке масив? Що Ви розумієте під ініціалізацією масиву?
2. Що таке глобальний, автоматичний та статичний масиви? Як вони неявно ініціалізуються?
3. Як можна задавати розмір масиву?
4. Що таке покажчик? Який розмір покажчика? Що визначає тип покажчика? Що показує різниця покажчиків?
5. Що таке покажчик-константа? Яка різниця між ім'ям масиву й звичайним покажчиком?
6. Що таке багатовимірний масив? Як розташовуються в пам'яті елементи багатовимірних масивів?
7. Що таке масив покажчиків? Наведіть приклад з явною повною ініціалізацією такого масиву.
8. Що таке динамічний масив? Чим відрізняється від звичайного?
9. Визначить та пояснить результат роботи програми:

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

```
void main()
```

```
{char a[5]={50,30,20,60,80};    char *pa=a;
 printf("%d %d %d \n",*pa, *(pa+2), *(pa+2));
 printf("%d \n",*++pa-1);
 printf("%d %d\n", sizeof(a), sizeof(a[3]));
 printf("%d %d\n", sizeof(pa), sizeof(*pa));
 char m[2][3]={1,5,7,2,6,8};
 char *pm=m[0];
 printf("%d %d %d %d\n",**(m+1),*(*(m+1)+1),*pm+2,* (pm+2));
 printf("%d %d\n", sizeof(m), sizeof(m[0]));
 printf("%d %d\n", sizeof(m[0][0]), sizeof(pm));
}
```

ЛІТЕРАТУРА

1. В.П. Пінчук, Л.І. Лозовська. Програмування мовою C/C++ з прикладами та вправами. Навчальний посібник. - Запоріжжя, ЗНТУ, 2008. – 197 с.
2. Подбельский В.В. Язык Си++: Учеб. пособие. – М.: Финансы и статистика, 2007. – 560 с.
3. Подбельский В.В. Практикум по программированию на языке Си: Учебн. пособие. – М.: Финансы и статистика, 2004. – 576 с.
4. Подбельский В.В., Фомин С.С. Программирование на языке Си: Учебн. пособие. – 2-е изд., – М.: Финансы и статистика, 2009. – 600с.
5. Павловская Т.А. C/C++. Программирование на языке высокого уровня. СПб: Питер, 2009. – 464 с.
6. Березин Б.И., Березин С.Б. Начальный курс С и С++. – М.: ДИАЛОГ-МИФИ, 2007. – 288 с.
7. Фридман А.Л. Основы объектно-ориентированного программирования на языке Си++. – М.: Горячая линия. – Телеком. Радио и связь, 1999. – 208 с.
8. Уильям Топп, Уильям Форд. Структуры данных в С++. / Пер. с англ.. – М.: ЗАО “Издательство БИНОМ”, 1999. – 816с.: ил.
9. Крячков А.В., Сухина И.В., Томшин В.К. Программирование на С и С++. Практикум: Учебное пособие для вузов/ А.В.Крячков, И.В.Сухина, В.К.Томшин.- 2-е изд., исправ.-М.: Горячая линия.- Телеком, 2000.-344с.
10. Крупник А. Изучаем Си.- СПб.: Питер.-2001. – 256с.
11. C/C++. Структурное программирование. Практикум/ Т.А. Павловская, Ю.А.Щупак. – СПб.: Питер, 2002. – 240с.
12. Культин Н.Б. С /C++ в задачах и примерах.- СПб.: БХВ: Петербург, 2001. – 288с.
13. Х.М. Дейтел, П.Дж. Дейтел. Как программировать на С++. – 5-е изд. / Пер. с англ. – М.: ООО "Бином-Пресс", 2008. – 1456 с.
14. Шилдт, Герберт. Полный справочник по С++. – 4-е изд. / Пер. с англ. – М.: Вильямс, 2008. – 800 с.