МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Методичні вказівки
до лабораторного практикуму з дисципліни
“ОСНОВИ АЛГОРИТМІЗАЦІЇ ТА ПРОГРАМУВАННЯ”
для студентів спеціальностей
125 – «Кібербезпека та захист інформації»,
освітні програми:
«Системи технічного захисту інформації, автоматизація її обробки»,
«Безпека інформаційних і комунікаційних систем»;
175 – «Інформаційно-вимірювальні технології»,
освітні програми:
«Якість, стандартизація та сертифікація»,
«Інформаційні системи моніторингу і контролю»,
176 – «Мікро- та наносистемна техніка»,
освітня програма:
«Мікро- та наноелектронні прилади і пристрої»
першого (бакалаврського) рівня вищої освіти
денної й заочної форм навчання.

2025

Методичні вказівки до лабораторного практикуму з дисципліни “Основи алгоритмізації та програмування” для студентів спеціальностей 125 – «Кібербезпека та захист інформації», освітні програми: «Системи технічного захисту інформації, автоматизація її обробки», «Безпека інформаційних і комунікаційних систем»; 175 – «Інформаційно-вимірювальні технології», освітні програми: «Якість, стандартизація та сертифікація», «Інформаційні системи моніторингу і контролю», 176 – «Мікро- та наносистемна техніка», освітня програма: «Мікро- та наноелектронні прилади і пристрої» першого (бакалаврського) рівня вищої освіти денної й заочної форм навчання. / Укл.: Неласа Г.В. – Запоріжжя: НУ «Запорізька політехніка», 2025. – 57 с.

Укладач: Ганна НЕЛАСА, к.т.н., доцент
Рецензент: Леонід КАРПУКОВ, д.т.н., професор
Відповідальний за випуск: Андрій КОРОТУН, завідувач кафедри «Інформаційна безпека та наноелектроніка», к.ф.-м..н., професор

Затверджено:
на засіданні кафедри
«Інформаційна безпека та
наноелектроніка»

протокол № 5
від 22 січня 2025р.

Рекомендовано до видання
НМК ФІБЕК
протокол № 7
від 24 лютого 2025 р.

ЗМІСТ

ЛАБОРАТОРНА РОБОТА №1. ПРОГРАМУВАННЯ ЛІНІЙНИХ ТА РОЗГАЛУЖЕНИХ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ

Мета роботи: одержати знання та навички, необхідні для програмування лінійних та розгалужених процесів; придбати та закріпити, на прикладі створення програм, елементарні знання з алгоритмічної мови програмування С++.

1.1 Основні теоретичні відомості

Загальна структура програми на мові С++ наведена на рис. 1.1.

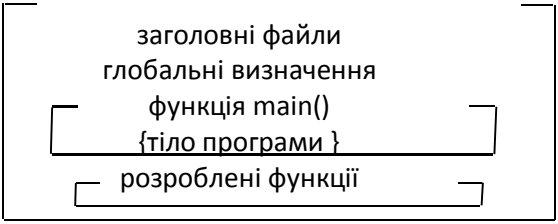


Рисунок 1.1 – Загальна структура програми

Програма зберігається у файлі з розширенням .cpp, заголовні файли у файлах з розширенням .h.

Оголошення змінних

Змінна, це – іменована область пам'яті, в яку під час виконання програми записуються значення відповідно до оголошеного типу.

Змінна оголошується наступним чином:

```
тип ім'я_змінної;
```

Тип змінних визначає об'єм оперативної пам'яті, що буде виділено під змінну, діапазон допустимих значень та можливі операції над змінною. Типи даних, що використовуються у мові С++ наведені у додатку Б.

Наприклад:

```
int a;
float b, c, d;
```

В залежності від місця об'явлення змінної, вона може бути глобальною, або локальною.

```
#include <заголовний файл>
int a;          // глобальна зміна
int main ()  // ім'я основної функції
{
float  b;     // локальна зміна
return 0;
}
```

Основні конструкції операторів мови С++

Для виконання розрахунків за математичними формулами використовується оператор присвоєння "=":

```
ім'я_змінної = вираз;
```

Наприклад:

```
int a=0;     //Присвоєння а значення 0
a=((a+7)/236)*(769-a*9);
```

Математичні операції виконуються відповідно до пріоритету.

В мові С++ додатково використовуються операції присвоєння, що наведені у таблиці 1.1.

Таблиця 1.1 – Операції присвоєння

Оператор	**Опис**	**Приклад**
=	Оператор присвоєння	int a=5;
+=	Пара операторів вказує компілятору присвоїти змінній суму існуючого значення змінної та вказаного	int a=0; a+=5; //a=a+5;
-=	Присвоєння змінній значення, яке буде зменшене на вказане	int a=0; a-=5; //a=a-5;

/=	Присвоєння змінній значення, яке буде поділене на вказане	s/=2*a; //s=s/2*a;
*=	Присвоєння змінній значення, яке буде помножене на вказане	s*=i; //s=s*i;

Введення/виведення даних

Для введення даних з клавіатури використовують операцію „>>”

```
cin >> a;
```

де *cin* – стандартна консоль введення, *a* – ім’я змінної.

Для виведення даних використовують операцію „<<”

```
cout << a << “\n”;
```

де *cout* – стандартна консоль виведення, *a* – ім’я змінної, *“\n”* – стрічний літерал, що позначає перехід на інший рядок.

Функції введення/виведення мови *C* наведені у додатку Д.

Умовний оператор if

Повна форма оператора if наступна:

```
if (умова) оператор1;
else оператор2;
        або
if (умова) {оператор1;
        ……
          операторN;}
else {група операторів}
```

де умова – це логічна операція *(&&, ||, :)*або операції порівняння *(==, !=, >, <, >=, <=)*, або інший вираз, результатом якого є логічний тип.

Якщо значення умови «*true»*, то виконується *оператор1* (ним може бути складовий оператор – блок), наступний після умови. Якщо ж умова приймає значення *«false»*, то виконується *оператор2*, що є наступним після *else*.

Наприклад:

```
int a=5, b=6;
if (a>b) cout<<a;
else cout<<b;
```

Оператор множинного вибору switch

Це вбудований оператор множинного вибору. Основна форма оператора має вигляд:

```
switch (вираз) {
 case constant1:
    послідовність операторів
    break;
 case constant2:
     послідовність операторів
     break;
 ……
 case constantN:
     послідовність операторів
     break;
 default
     послідовність операторів
 }
```

Спочатку обчислюється вираз в дужках за ключовим словом *switch*. Потім продивляється список міток (*case constant1* та ін.) до тих пір, поки не знаходиться мітка, що співпадає зі значенням обчисленого виразу. Далі виконується відповідна послідовність операторів, що слідують за двокрапкою. Якщо ж зі значенням виразу не співпадає жодна з міток оператору *switch*, то виконується послідовність операторів, що слідує за ключовим словом *default*.

Математичні функції мови С++

Для виконання різних математичних дій мова *С++* містить різноманіття математичних функцій. Для їх використання необхідно додати заголовний файл <cmath> або <math.h>.

Для використання математичних констант, зокрема π (M_PI) додатково треба вказати:

```
#define _USE_MATH_DEFINES // for C++
```

Більш докладно див. https://learn.microsoft.com/en-us/cpp/c-runtime-library/math-constants?view=msvc-170

Приклад математичних функцій наведений у таблиці 1.2.

Таблиця 1.2 – Математичні функції

Функція	Призначення
float cos (float arg);	Повертає значення косинуса аргументу *arg*.
double exp(double arg) ;	Повертає значення експоненти від аргументу *arg*
double log(double num);	Повертає значення натурального логарифму для аргументу *num*
double round(double arg);	Повертає значення аргументу *arg,* наближене до цілого. Але тип значення, що повертається є значенням з плаваючою точкою.
double sin(double arg);	Повертає значення синуса аргументу arg.
double sqrt(double num);	Повертає значення квадратного кореня від аргументу *лит.*
double fabs(double num);	Повертає модуль аргументу *num*

1.2 Завдання на лабораторну роботу

1. Ознайомитися зі змістом теоретичних відомостей даних методичних вказівок.
2. Отримати у викладача індивідуальні завдання згідно варіанту.
3. Скласти та виконати програми відповідно до завдання.
4. Оформити звіт та захистити роботу.

1.3 Індивідуальні завдання

Завдання 1. Обчислити та вивести на екран значення a, b, обчислені за формулами, використовуючи математичні функції бібліотеки math.h (cmath). Врахувати аналіз області визначення функцій при введенні вхідних даних. Протестувати виконання програми за допомогою позитивного, негативного та деструктивного тестів.

№	Варіанти до завдання 1.
1	Дано x, y, z. Обчислити a, b. $a = \dfrac{\sqrt{\lvert x-3\rvert} - \sqrt[3]{\lvert y\rvert}}{1 + \dfrac{x^2}{2} + \dfrac{y^2}{4}}$ $b = 5x(arctg\, z + e^{-(x+2)})$
2	Дано x, y, z. Обчислити a, b. $a = \dfrac{6 + e^{y-2}}{3 + 4x^2\lvert y - tg\, z\rvert}$ $b = 7 + \lvert y - x\rvert + \dfrac{(y-x)^2}{2} + \dfrac{\lvert y-x\rvert^3}{3}$
3	Дано x, y, z. Обчислити a, b. $a = (5+y)\dfrac{x + y/(x^2+4)}{e^{-x-2} + 1/(x^2+4)}$ $b = \dfrac{1 + 9\cos(y-2)}{x^4/2 + 4sin^2 z}$
4	Дано x, y, z. Обчислити a, b. $a = y + \dfrac{5x}{y^2 + \left\lvert \dfrac{x^2}{2y + x^3/3} \right\rvert}$ $b = (2 + tg^2)\dfrac{z}{2}$
5	Дано x, y, z. Обчислити a, b. $a = \dfrac{2\cos\left(x - \dfrac{\pi}{6}\right)}{1/2 + 4sin^2 y}$ $b = 12 + \dfrac{z^2}{5 + z^2/2}$

6	Дано x, y, z. Обчислити a, b. $a = \frac{3 + sin^2(x+y)}{5 + \lvert x - 2x/(4 + x^2y^2)\rvert} + x$ $b = 13cos^2(arctg\frac{1}{z})$
7	Дано x, y, z. Обчислити a, b. $a = 2ln\left\lvert(4y - \sqrt{\lvert x\rvert})(x - \frac{4y}{z - x^2/4})\right\rvert$ $b = x - \frac{x^2}{3!} + \frac{x^5}{5!}$
8	Дано x, y, z. Обчислити a, b. $a = \frac{4\sin(x) + 6\ln(x)^3 + 8z}{10\sqrt[3]{xyz}}$ $b = 14 + \frac{(2z+y)^4}{8z^3}$
9	Дано x, y, z. Обчислити a, b. $a = \sqrt[3]{\lvert e^{-2xyz}\rvert} - \sqrt{\ln(8xyz)^3}$ $b = 4xyz + \ln(x + 2y + 3z)$
10	Дано x, y, z. Обчислити a, b. $a = \frac{3x}{2y^2} + \frac{7y}{z^2} + \frac{2z}{3x^2}$ $b = 9x^5 + 4y^4 + z^3$

Завдання 2. Обчислити з використанням оператору умови та вивести на екран значення функції F, де a, b, c – дійсні числа, значення яких вводяться з клавіатури.

№	Варіанти до завдання 2.
1	$F = \begin{cases} ax^2 + b - \text{c}, & \text{якщо } x < 0\ b \neq 0 \\ \dfrac{x - 2a}{x - c}, & \text{якщо } x > 0\ b = 0 \\ \dfrac{12x}{c} & \text{в інших випадках} \end{cases}$
2	$F = \begin{cases} \dfrac{1}{ax} - b\text{c}, & \text{якщо } x + 5 < 0\ c = 0 \\ \dfrac{x - 3a}{x}, & \text{якщо } x + 5 > 0\ \text{c} \neq 0 \\ \dfrac{13x}{c - 4} & \text{в інших випадках} \end{cases}$
3	$F = \begin{cases} -\dfrac{2x - 4c}{cx - a}, & \text{якщо } x < 0\ b \neq 0 \\ \dfrac{x - 5a}{x - 3c}, & \text{якщо } x > 0\ b = 0 \\ -\dfrac{2x}{c} + \dfrac{(-c)}{2x} & \text{в інших випадках} \end{cases}$
4	$F = \begin{cases} -ax - 3c, \text{якщо } c < 0\ x \neq 0 \\ \dfrac{x - 6a}{-c}, \text{якщо } c > 0\ x = 0 \\ \dfrac{2bx}{c - a} & \text{в інших випадках} \end{cases}$
5	$F = \begin{cases} a - \dfrac{12x}{10 + b}, \text{якщо } x < 0\ b \neq 0 \\ \dfrac{x - 2a}{x - 7c}, \text{якщо } x > 0\ b = 0 \\ 3x + \dfrac{14}{c} & \text{в інших випадках} \end{cases}$

6	$F = \begin{cases} 3ax^2 + 2b^2x, \text{якщо } c < 0\ b \neq 0 \\ \dfrac{3x + a}{x + 3c}, \text{якщо } c > 0\ b = 0 \\ \dfrac{5x}{c} \text{ в інших випадках} \end{cases}$
7	$F = \begin{cases} -ax^2 - 7b, \text{якщо } x < 5\ c \neq 0 \\ \dfrac{x - a}{x}, \text{якщо } x > 0\ b = 0 \\ \dfrac{-x}{9c} \text{ в інших випадках} \end{cases}$
8	$F = \begin{cases} -ax^2, \text{якщо } c < 0\ a \neq 0 \\ \dfrac{a - 23x}{6cx}, \text{якщо } c > 0\ a = 0 \\ \dfrac{7x - 8a}{6c} \text{ в інших випадках} \end{cases}$
9	$F = \begin{cases} 8ax^2 + 6b^2x, \text{якщо } a < 0\ x \neq 0 \\ x - \dfrac{7a}{x - 2c}, \text{якщо } a > 0\ x = 0 \\ 1 + \dfrac{x}{c} \text{ в інших випадках} \end{cases}$
10	$F = \begin{cases} ax^2 - 13bx + c, \text{якщо } x < 3\ b \neq 0 \\ \dfrac{6x - 7a}{x - c}, \text{якщо } x > 3\ b = 0 \\ \dfrac{12x}{c} \text{ в інших випадках} \end{cases}$

1.4 Контрольні питання

1. Яка загальна структура програми на мові С++?
2. Що називається ідентифікатором?
3. Які типи даних вам відомі?
4. Що таке змінна?
5. Вкажіть операції по складу пріоритету?
6. З якою метою використовують модифікатори типів?
7. Коли використовують оператори вибору та множинного вибору?
8. Як додати коментарі до програми?
9. З якою метою використовують *{}*?
10. Дайте визначення алгоритму та алгоритмізації.
11. Наведіть схему компіляції програми.
12. Які математичні функції ви знаєте і як їх використовують?

ЛАБОРАТОРНА РОБОТА № 2. ПРОГРАМУВАННЯ ЗАДАЧ ЦИКЛІЧНОЇ СТРУКТУРИ. МАСИВИ.

Мета роботи: Одержання знань і навичок щодо використання різних видів циклів та обробки статичних масивів.

2.1 Основні теоретичні відомості

Цикл — це група операторів, що виконуються багаторазово.

Оператор *while* визначає операції, які циклічно виконуються до того моменту, поки вираз, що стоїть після *while*, стане хибним. Цей оператор називається оператором циклу з передумовою: спочатку перевіряються умови, і якщо умови виконуються, то потім виконується тіло оператору. Тому можлива ситуація, коли тіло циклу може бути не виконаним жодного разу. Форма запису оператора наступна:

while (вираз) оператор; або *while (вираз) {група операторів}*
де *вираз* – це умова виконання тіла циклу.

Наприклад:

```
const n=25;
int i=0;
while (i<n) i++;
```

При організації циклу, коли його тіло повинно бути виконане фіксовану кількість разів необхідно реалізувати три операції: ініціювання лічильника, порівняння його з повним значенням межі і збільшення (зменшення) лічильника при кожному проходженні циклу. В *C* є спеціалізований оператор циклу типу *for*, в якому органічно поєднано організацію цих трьох операцій.

В операторі *for* використовуються три вирази, що керують роботою цикла. Вони розділені символом **;** . Початковий вираз обчислюється тільки один раз до початку виконання одного з операторів циклу. Якщо вираз-перевірка буде істинним (не рівним нулю), тіло циклу виконається один раз. Потім обчислюється величина виразу коректування і визначається знову величина виразу-перевірки. Оператор циклу *for* - оператор з передумовою, отже знову

ж може трапитися, що тіло циклу не виконається жодного разу. Він має таку форму:

for (ініціювання; умова_виконання; вирази_корекції) оператор

ініціювання_циклу – послідовність визначень та виразів, розділених комами. Всі вирази, що входять до ініціалізації циклу розраховуються тільки один раз при вході до цикла;

вирази_корекції – розраховуються на кожній ітерації після виконання операторів тіла циклу і до наступної перевірки умови_виконання.

Наприклад:

```
int i;
for (i=0; i<n; i++) cout<<i;
```

Тіло цикла виконується так довго, поки вираз-перевірка не стане хибним (рівним нулю).

Якщо умова_виконання не змінюється або відсутня, то цикл нескінченний.

Наприклад:

```
for ( ; ; );     // нескінченний цикл
for ( ;1; );   // нескінченний цикл
```

Для багатьох обчислювальних задач корисним є використання циклу з постумовою (умовою на виході). В цьому випадку тіло циклу обов`язково виконається як мінімум один раз. В *C* такий оператор циклу реалізується конструкцією *do while*. Взагалі цикл має вигляд:

```
do оператор while (вираз);
```

Проілюструємо його виконання таким прикладом:

```
do
   { ch=getchar();
     putchar(ch);
   }  while (ch!= `\n`);
```

Прикладом використання циклів може стати алгоритми розрахунку сум, добутку та ін.

Алгоритм розрахунку суми

Крок 1. Ініціалізація змінної суми *(s=0)* та змінної циклу *(i=0)*.

Крок 2. Якщо зміна циклу більше кількості ітерацій, перехід на крок 4, інакше перехід на крок 3.

Крок 3. Накопичуємо суму. Збільшуємо оператор циклу *(i++)*, перехід на крок 2.

Крок 4. Закінчення

Приклад: Розрахувати $\sum_{i=0}^{20} i$

```
int s, i;
s=0;
for (i=0; i<20; i++) s+=i;
```

Алгоритм розрахунку нескінчених сум

Крок 1. Ініціалізація змінної суми. Перехід на крок 2.

Крок 2. Розрахувати доданок на теперішній ітерації. Перехід на крок 3.

Крок 3. Накопичуємо суму. Перехід на крок 4.

Крок 4. Якщо доданок менший за завданну точність, то перехід на крок 5. Якщо ні – крок 2.

Крок 5. Кінець.

Приклад: Розрахувати $\sum_{i=1}^{\infty} \frac{1}{i^2}$ із заданою точністю ε

```
float s0 ,s1, t, ε;
int i=1;
s1=1.0;
do {
  i++;
  s0=s1;
  t=1.0/i*i;
  s1+=t;
   }  while (fabs(s1-s0)>ε);
```

Масиви

Масив це – сукупність елементів одного типу. Масив об'являється наступним чином:

тип_елементів ім'я_масиву [розмірність];

де тип_елементів – це деякий стандартний тип (int, float), розмірність – кількість елементівмасиву. Індексація виконується з 0.

При зверненні до елементу масива використовуються індекси:

for (i=0; i<n; i++) cout<<a[i];

Операції з елементами матриці

У повсякденній практиці часто доводиться мати справу з матрицею (двовимірним масивом).

Матриця – це прямокутна (в частинному випадку - квадратна) таблиця чисел, що має M рядків та N стовпців.

Головна умова правильного розв'язання відповідних задач полягає в розумінні порядку змінювання індексів елементів матриці. Перший індекс завжди визначає номер рядка, другий — номер стовпця.

Рекомендації по написанню програм з використанням обробки масивів та матриць

а) масив (матриця) повинний бути оголошений;

б) при оголошенні масиву (матриці) при задані розмірності зручно використовувати іменовані константи;

в) доступ до елемента масиву (матриці) здійснюється шляхом вказівки індексу (номера) елемента, у якості якого можна використовувати вираз цілого типу, наприклад, цілу константу чи змінну типу *int*;

г) для введення, виведення та обробки масивів зручно використовувати оператори циклів (*for, while, do while*);

д) для роботи з матрицями найчастіше треба використовувати вкладені цикли.

Приклад програми по обробці матриць

Опис матриці та введення даних до неї розглянемо на *елементарному прикладі роботи з матриціями*:

Завдання: В квадратній матриці з правого боку від побічної діагоналі знайти найбільший елемент і відповідний номер рядка.

```
# include <iostream.h>
const int M=5, N=5;
void main ()
{ int a[M][N];
int i, j, maxzn, nom;
cout <<"Введіть матрицю \n";
  for (i=0; i<M; i++)
     for (j=0; j<N; j++)
      cin >>a[i][j];   //Вводимо значення елементів масиву
maxzn=a[0][0];
nom=1;
  for (i=0; i<M; i++)
     for (j=N-i+1; j<N; j++)
          if (maxzn<a[i,j])
          { maxzn=a[i,j];
  nom=i; }
cout << "Початкова матриця \n";
  for (i=0; i<M; i++)
       { for (j=0; j<N; j++)
              cout <<a[i,j];
          cout << "\n";}
cout << "max елемент="<<maxzn<<"номер рядка="<<nom;
}
```

Важливе місце в цій програмі займає організація внутрішнього циклу. При *i=l* змінна j приймає значення *j=5*, при *i=2* індекс *j* буде змінюватись від 4 до 5, і так далі. При *i=5* параметр j змінюється від 1 до 5. Такі значення індексів забезпечують перегляд всіх елементів, які розміщені на побічній діагоналі та з правого боку від неї.

2.2 Завдання на лабораторну роботу

1. Ознайомитися зі змістом теоретичних відомостей даних методичних вказівок.
2. Отримати у викладача індивідуальні завдання згідно варіанту.
3. Скласти та виконати програми відповідно до завдання.
4. Оформити звіт та захистити роботу.

2.3 Індивідуальні завдання

Завдання 1. Обчислити та вивести на екран суму нескінченого ряду точністю *e*=10E-5. Значення змінної x ввести з клавіатури. Обов'язково оптимізувати швидкість обчислень шляхом обчислення кожного наступного доданку на основі значення попереднього.

№	Варіанти до завдання 1.
1	$\frac{\sin(x)}{x} = 1 - \frac{x^2}{3!} + \frac{x^4}{5!} - \frac{x^6}{7!} \dots$
2	$e^{-x^2} = 1 - \frac{x^2}{1!} + \frac{x^4}{2!} - \dots + (-1)^n \frac{x^{2n}}{n!}$
3	$arctg(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \frac{x^9}{9} \dots$
4	$\arcsin(x) = x + \frac{1}{2} \cdot \frac{x^3}{3} + \frac{1 \cdot 3}{2 \cdot 4} \cdot \frac{x^5}{5} + \frac{1 \cdot 3 \cdot 5}{2 \cdot 4 \cdot 6} \cdot \frac{x^7}{7} \dots$
5	$\frac{1}{\sqrt{1 - x^2}} = 1 + \frac{1}{2} \cdot x^2 + \frac{1 \cdot 3}{2 \cdot 4} \cdot x^4 + \frac{1 \cdot 3 \cdot 5}{2 \cdot 4 \cdot 6} \cdot x^6 \dots$
6	$\frac{1}{\sqrt{1 + x}} = 1 - \frac{1}{2} \cdot x + \frac{1 \cdot 3}{2 \cdot 4} \cdot x^2 - \frac{1 \cdot 3 \cdot 5}{2 \cdot 4 \cdot 6} \cdot x^3 \dots$
7	$\sqrt{1 + x} = 1 + \frac{1}{2} \cdot x - \frac{1}{2 \cdot 4} \cdot x^2 + \frac{1 \cdot 3}{2 \cdot 4 \cdot 6} \cdot x^3 - \dots$
8	$\frac{1}{(1 + x)^3} = 1 - \frac{2 \cdot 3}{2} \cdot x + \frac{3 \cdot 4}{2} \cdot x^2 - \frac{4 \cdot 5}{2} \cdot x^3 + \dots$
9	$\frac{1}{(1 + x)^2} = 1 - 2 \cdot x + 3 \cdot x^2 - 4 \cdot x^3 + 5 \cdot x^4 - \dots$
10	$\frac{1}{1 + x} = 1 - x + x^2 - x^3 + x^4 \dots$

Завдання 2. Використовуючи цикл `for(){}` обробити елементи статичного одновимірного масиву (вектору). Розмір масиву *n* ввести з клавіатури. Елементи масиву згенерувати як псевдовипадкові значення.

№	**Варіанти до завдання 2.**
1	В одновимірному масиві, що складається з *n* цілих елементів, обчислити суму від'ємних елементів масиву.
2	В одновимірному масиві, що складається з *n* цілочисельних елементів, обчислити добуток елементів масиву, розташованих між максимальним та мінімальним елементами.
3	В одновимірному масиві, що складається з *n* цілочисельних елементів, обчислити суму додатних елементів масиву.
4	В одновимірному масиві, що складається з *n* цілочисельних елементів, обчислити добуток елементів масиву, розташованих між максимальним та мінімальним за модулем елементами.
5	В одновимірному масиві, що складається з *n* цілочисельних елементів, обчислити добуток елементів масиву з парними номерами.
6	В одновимірному масиві, що складається з *n* цілочисельних елементів, обчислити суму елементів масиву, розташованих між першим і останнім нулевими елементами.
7	В одновимірному масиві, що складається з *n* дійсних елементів, обчислити суму елементів масиву з непарними номерами.
8	В одновимірному масиві, що складається з n дійсних елементів, обчислити суму елементів масиву, розташованих між першим і останнім від'ємними елементами.
9	В одновимірному масиві, що складається з n дійсних елементів, обчислити максимальний елемент масиву.
10	В одновимірному масиві, що складається з *n* дійсних елементів, обчислити суму елементів масиву, розташованих до додатного останнього елемента.

2.4 Контрольні питання

1. Для чого використовують оператори циклу?
2. Розробіть схеми алгоритмів операторів циклів з передумовою, постумовою, ітераційного циклу.
3. Наведіть алгоритм розрахунку добутку.
4. Наведіть алгоритм розрахунку суми.
5. Наведіть приклади використання циклів з постумовою.
6. Що називається масивом?
7. Як описується масив?
8. Що називається індексом масиву?
9. Які обмеження накладаються на індекси?
10. Як відбувається доступ до індексів масиву?
11. Які типи використовуються для визначення розміру масиву?
12. Що таке матриця?
13. Поясніть різницю між термінами «розмір» та «розмірність» масиву.

ЛАБОРАТОРНА РОБОТА № 3. РОБОТА З ПОКАЖЧИКАМИ ТА ДИНАМІЧНИМИ МАСИВАМИ

Мета роботи: Одержання знань і навичок, необхідних для роботи з покажчиками та динамічними масивами.

3.1 Основні теоретичні відомості

В мові *C/C++* ыснують змінні типа ***покажчик***. Значенням змінної типу покажчик буде адреса в пам'яті деякої змінної.

Враховуючи, що для деяких операцій, пов'язаних з покажчиками, потрібно знати об'єм відведеної пам'яті, опис покажчиків має вигляд:

тип * *ім'я-покажчика*;

Наприклад:

int *pi; /* покажчик на змінну цілого типу */
char *pc; /* покажчик на змінну символьного типу */

З покажчиком використовують два оператори: „*" та „&". Оператор „&" – унарний. Він повертає адресу пам'яті, де розташований його операнд.

Унарний оператор * дозволяє звернутись до значення змінної, що розташована за адресою, що задана його операндом.

Операції, що виконуються за допомогою покажчиків, часто називають операціями непрямого доступу, т.ч. ми непрямо отримуємо доступ до змінної через деяку іншу змінну.

Працюючи з покажчиком, постійно використовують операцію & отримання адреси об'єкту. Для неї існують природні обмеження:

– неможливо визначити адресу непоіменованої константи, тобто не приймаються вирази &3.141593;

– неможливо визначити адресу значення, яке отримане при розрахунку скалярних виразів;

– неможливо визначити адресу змінної, що відноситься до класу пам'яті *register*;

В якості типу при визначенні покажчика може бути використаний як основний тип, так і похідний. До похідних типів можуть бути віднесені масиви, функції, покажчики, посилання, класи, структури, об'єднання та ін.

Основні типи як завжди визначаються ключовими словами *int, float, char, unsigned.*

Операції над покажчиками можна згрупувати наступним чином:

- операції розіменування чи доступу за адресою (*);
- перетворення типів (приведення типів);
- присвоєння;
- отримання (взяття) адреси (&);
- адитивні операції (додавання та віднімання константи);
- інкремент (++) та дикремент (--);
- операції відношення (операції порівняння);

Якщо операція & завжди дає однаковий результат – адрес розміщення в пам'яті, то операція розіменування *покажчик залежить не тільки від адреси, але й від типу. Річ в тому, що при доступі до пам'яті за допомогою розіменування покажчика потрібною буде інформація не тільки про розміщення, але й про розміри ділянки пам'яті, що буде використовуватися. Цю додаткову інформацію компілятор отримує з типу покажчика.

Коли покажчик інкрементується він буде вказувати на область пам'яті, що містить наступний елемент базового типу цього покажчика а при кожному дикрементуванні він буде вказувати на попередню область пам'яті базового типу цього покажчика.

Наприклад. Після виконання *p1=p1+9 p1* буде посилатися на дев'ятий елемент базового типу покажчика *p1* відносно поточного елемента, на який посилається *p1.*

Додавати покажчики не можна. Але якщо відняти один покажчик з другого (які вказують на один базовий тип) то різниця вкаже на кількість елементів базового типу, що розподіляють ці два покажчика.

Представлення масиву у пам'яті

При об'яві одновимірного масиву

тип ім'я_масиву [n];

в пам'яті виділяється *n*-елементів вказаного типу (*n* – розмірність масиву) та створюється змінна, що є покажчиком на нульовий елемент масиву (ім'я масиву)

int a[10];

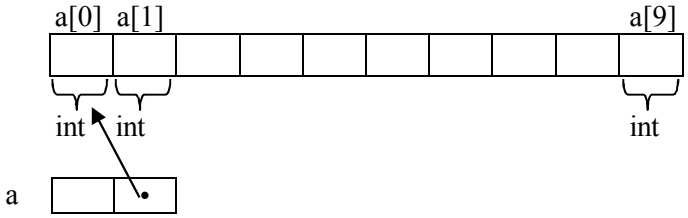


Рисунок 3.1 – Представлення одномірного масива у пам'яті

При об'яві двомірного масиву *int b[5][5]*, розподіл пам'яті відбудеться відповідно до рис. 4.2.

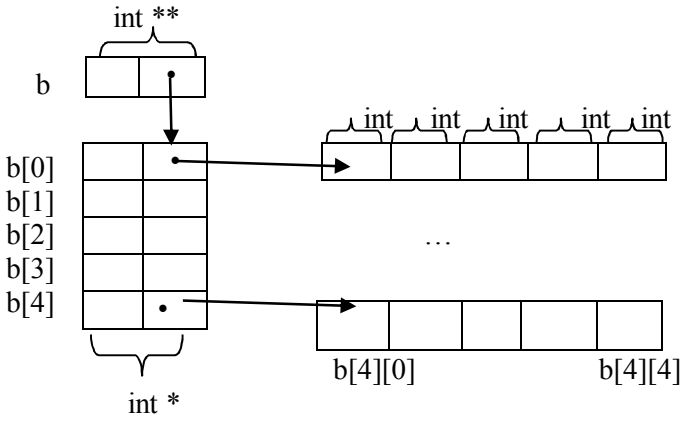


Рисунок 3.2 – Представлення двомірного масиву у пам'яті

Для виділення динамічної пам'яті використовують оператор *new:*

змінна_покажчик = new тип;
змінна_покажчик = new тип [];

Для звільнення динамічної пам'яті використовують оператор *delete:*

delete змінна_покажчик;
delete [розмірність] ім'я_масву;

Для виділення місця в пам'яті під динамічний масив необхідно виконати наступні кроки:

```
int *a;
a = new int [n];
```

Під двовимірний масив:

```
int **b;
b = new int * [n];
for (int i=0; i<n; i++)
b[i] = new int [m];
```

3.2 Завдання на лабораторну роботу

1. Ознайомитися зі змістом теоретичних відомостей даних методичних вказівок.
2. Отримати у викладача індивідуальні завдання згідно варіанту.
3. Скласти та виконати програми відповідно до завдання.
4. Оформити звіт та захистити роботу.

3.3 Індивідуальні завдання

Завдання 1. Виконати завдання 2 лабораторної роботи №2 використовуючи динамічний масив замість статичного.

Завдання 2. Обробити елементи двовимірного динамічного масиву (матриці) відповідно завдання. Розмір масиву ввести з клавіатури. Елементи масиву згенерувати як псевдовипадкові значення.

№	**Варіанти до завдання 2.**
1	Знайти суму елементів головної діагоналі розміром $n \times n$
2	В матриці розміром $n \times m$ знайти мінімальні елементи в кожному стовбці
3	В матриці розміром $n \times m$ знайти мінімальний елемент серед максимальних елементів кожного рядка
4	Знайти суму квадратів зафарбованої частини матриці 
5	Знайти визначник матриць розміром 2×2 та 3×3.
6	Знайти суму кубів зафарбованої частини матриці 
7	Знайти максимальний елемент серед непарних стовбців матиці розміром $n \times m$
8	Знайти максимальний елемент побічної діагонали квадратної матриці розміром $n \times n$
9	Транспонувати матрицю розміром $n \times m$
10	Знайти мінімальний елемент серед парних рядків матиці розміром $n \times m$

3.4 Контрольні запитання

1. Що таке покажчик?
2. Які матричні операції можливо використовувати з покажчиками?
3. Що спільного між покажчиком та масивом?
4. Коли використовується динамічна пам’ять?
5. Як звільнити динамічну пам’ять?
6. Що таке динамічна пам’ять?
7. Представлення двовимірного масиву у пам’яті?
8. Що працює швидше: індекси або покажчики?

ЛАБОРАТОРНА РОБОТА № 4.
МОДУЛЬНЕ ПРОГРАМУВАННЯ. ФУНКЦІЇ.

Мета роботи: Одержання знань і навичок, необхідних для створення користувацьких функцій.

4.1 Основні теоретичні відомості

Функції – це будівельні блоки деякої програми на мові *C++*.
Функції бувають двох типів:
– функції користувачів (включно *main()*);
– стандартні (бібліотечні) функції мови *C* та *C++*.

Форма запису функцій:
тип_результату ім'я_функції (список_параметрів){
//Тіло функції
}

де *тип_результату* – задає тип даних, що повертаються функцією, *список_параметрів* – задає список змінних, що передаються функції при зверненні.

Дуже важливим оператором тіла функції є оператор повернення в точку виклику:

return вираз;
return;

Вираз в операторі *return* визначає значення, що повертається функцією. Саме це значення буде результатом звернення до функції.

Навіть у випадку, коли функція не виконує ніяких дій й не повинна повертати деяке значення, тіло функції буде складатися з фігурних дужок {}.

Приклад:
int max(int a, int b) {
if (a>b) return a;
else return b;}

Суворе узгодження за типами між формальними та фактичними параметрами потребує, щоб у модулі до першого звернення до функції містилося або її визначення, або її опис (прототип), що містить інформацію про її тип (про тип результату, тобто повертаємого значення) та про типи всіх параметрів. Прототип (опис) функції може зовнішньо співпадати зі заголовком її визначення:

тип_функції ім'я_функції(специфікація_формальних_параметрів);

Наприклад:
int max(int a, int b);

У визначенні функції специфікація параметра може містити його значення за замовченням. Це значення використовується у тому випадку, якщо при зверненні до функції відповідний параметр відсутній.

Наприклад:
int max (int a, int b=0);

Функція main()

Функція *main()* не має прототипу. Тобто, можна використовувати різні форми запису функції *main()*. Для мови *C++* мають місце наступні варіанти функції *main()* (дозволяються й інші форми)*:*

1. *int main();*
2. *int main(int argc, char *argv[]);*

Як видно з другої форми запису, функція *main()* підтримує два параметра – *argc* та *argv*. Ці дві змінні будуть зберігати кількість аргументів командного рядка та покажчик на них. Параметр *argc* має цілий тип, та його значення завжди буде не менше за 1, тому що у мові *C++*, першим аргументом завжди є ім'я програми. Параметр *argv* повинен бути оголошений як масив символьних покажчиків, в якому кожний елемент вказує на аргумент командного рядку.

В функцію можна передати покажчик на масив. Функції можуть повертати покажчик на масив в якості результату.

Посилання в мові С++

В мові С++ посиланням є інше ім'я існуючого об'єкту. Для визначення посилання використовується символ *&*, якщо він використовується у такому контексті:

type&ім'я_посилання ініціалі затор

У відповідності із синтаксисом ініціалізатора, наявність якого обов'язкове, визначення посилання може бути таким:

type&ім'я_посилання=вираз; або type&ім'я_посилання(вираз);

При визначенні посилання обов'язковою є його ініціалізація. Однак в опису посилань ініціалізація не обов'язкова. До таких описів посилань відносяться:

а) опис зовнішніх посилань (через специфікатор *extern*):

```
float& ref;                 //Помилка – немає ініціалізації
extern float& ref2;         //Допустимо – ініціалізується
//в іншому блоці;
```

б) опис компонентів класу;

в) описи (специфікації) формальних параметрів функції;

г) опис типу значення, що повертається функцією.

В якості основних причин включення посилань у мові *С++* позначають необхідність збільшити ефективність обміну з функціями через апарат параметрів та доцільність можливості використовувати визивання функції в якості допустимого значення. При використанні посилання в якості формального параметра забезпечується доступ з тіла функції до відповідного фактичного параметра, тобто до ділянки пам'яті, що виділяється під фактичний параметр.

Перевантаження функцій

Мета перевантаження функцій складається в тому, щоб функція із одним ім'ям по-різному виконувалася та повертала різні значення при звертанні до неї із різними за типом та кількістю фактичними параметрами.

Для забезпечення перевантаження функцій необхідно для кожного імені визначити скільки різних функцій пов'язано з ним, тобто скільки варіантів сигнатур є допустимими при зверненні до них. Будемо вважати, що функція вибору максимального значення елемента із масиву має працювати для масивів типу *int, long, float, double*. В цьому випадку треба написати чотири різні варіанти функції з одним ім'ям.

```
int max (int a, int b);
float max (float a, float b);
char max (char a, char b);
```

При виклику перевантаженої функцій компілятор точно визначає, яку з версій функції треба використовувати. Рішення компілятора основане на результатах аналізу кількості та типу аргументів. Визивається функція, що найточніше відповідає "моделі" виклику.

Для візуалізації результатів зручно використовувати графічне представлення. Наведемо приклад побудови графіка синусоїди в консольному вікні, але з використанням графічного інтерфейсу підсистеми Windows.

```
#include <windows.h>
#include <stdlib.h>
#include <math.h>
#include <iostream>

int main(void)
{
  float x;
  HDC hDC = GetDC(GetConsoleWindow());
  HPEN Pen = CreatePen(PS_SOLID, 2, RGB(255, 255, 255));
  SelectObject(hDC, Pen);
  MoveToEx(hDC, 0, 85, NULL);
  LineTo(hDC, 200, 85);
  MoveToEx(hDC, 100, 0, NULL);
```

```
    LineTo(hDC, 100, 170);


    for (x = -8.0f; x <= 8.0f; x += 0.01f) // O(100,85) - center
    {
        MoveToEx(hDC, 10 * x + 100, -10 * sin(x) + 85, NULL);//10 - scale
        LineTo(hDC, 10 * x + 100, -10 * sin(x) + 85);
    }
    std::cin.get();
    system("pause");
    return 0;
}
```

Windows Programming using MFC and API. Graphics Device Interface
https://www.nexus-6.uk/api-graphics/

4.2 Завдання на лабораторну роботу

1. Ознайомитися зі змістом теоретичних відомостей даних методичних вказівок.
2. Отримати у викладача індивідуальні завдання згідно варіанту.
3. Скласти та виконати програми відповідно до завдання.
4. Оформити звіт та захистити роботу.

4.3 Індивідуальні завдання

Завдання 1. Розробити функцію згідно варіанту. В головній програмі зробити три виклику функції:

- з параметрами, що задані при об'явлені змінних в головній програмі;
- з параметрами, що генеруються як псевдовипадкові числа;
- з параметрами, що вводяться з клавіатури.

№	Варіанти до завдання 1.
1	Знайти корені квадратного рівняння $ax^2 + bx + c = 0$. Параметри функції - коефіцієнти рівняння типу *double*.
2	Обчислити кількість цифр заданого цілого числа n.
3	Обчислити значення біному Ньютона $(a + b)^n$. Параметри функції – змінні a, b, n типу *unsigned long long*.
4	Визначити, чи є задане додатне ціле число n простим.
5	Знайти найбільший спільний дільник двох додатних цілих чисел a, b за алгоритмом Евкліда.
6	Обчислити значення функції Ейлера (кількість чисел від 1 до *n*, взаємно простих з *n*) від заданого додатного цілого числа *n*.
7	Обчислити суму та *n*-й елемент арифметичної прогресії з першим елементом a_1 та різницею *d*.
8	Обчислити суму та *n*-й елемент геометричної прогресії зі з першим елементом b_1 та знаменником *q*.
9	Знайти значення полінома $f(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$в точці *x* за схемою Горнера. Параметри n-кількість елементів та $a[n]$ - вектор коефіцієнтів типу *float*.
10	Знайти результат множення двох матриць, що мають розміри $(n \times k)$ та $(k \times m)$. Параметри – розміри матриць (цілі числа) та елементи матриць – числа типу *float*.

Завдання 2. Обчислити та вивести на екран таблицю значень функції $f(x)$ для x, що змінюється від $x1$ до $x2$ з кроком d. Значення змінних $x1$, $x2$, d ввести з клавіатури. Обчислення при заданому значенні параметра синтаксично виконати як окрему функцію, що приймає та повертає значення типу *double*. Передбачити перевірку області допустимих значень для параметрів, що передаються. Для виведення таблиці використати функцію *fprintf().* Вивести графік

функції для $x \in [x1..x2]$ на екран.

№	Варіанти до завдання 2.
1	$f(x) = \sin(x^2) + \cos(x)$
2	$f(x) = \sin(\sin(x))$
3	$f(x) = sin^2(x) + \sin(x^2)$
4	$f(x) = tg(x)/x + \exp(x)$
5	$f(x) = \cos(x^3) + cos^3(x) + x$
6	$f(x) = \exp(x)/x^2 + \ln(\|x\|)$
7	$f(x) = (\sin(x^2) + \sin(x+2))/\exp(x)$
8	$f(x) = \cos(\cos(x^3))/\cos(3x)$
9	$f(x) = \ln(\|x\|) - tg(x^2) + ctg^2(x)$
10	$f(x) = \ln(\|2x\|) - ctg(x^3) + tg(3x)$

4.4 Контрольні питання

1. Що таке функція на мові С(С++)? Для чого вона застосовується?
2. Опишіть механізм передачі параметрів у функцію.
3. Як використовується оператор *return* у функції?
4. Що ми маємо на увазі, коли говоримо про тип функції?
5. Чим відрізняються прототип, визначення і виклик функції?
6. Які можливо передати параметри до функції *main()*?
7. Що таке перевантаження функцій? Для чого воно застосовується?
8. Навести приклади прототипів перевантажених функцій із сигнатурами, що розрізняються за кількістю аргументів і за типом аргументів.

ЛАБОРАТОРНА РОБОТА №5. ПРОГРАМУВАННЯ ЗАДАЧ ПО ОБРОБЦІ ПОСЛІДОВНОСТІ СИМВОЛІВ. РОБОТА З ФАЙЛАМИ

Мета роботи: одержання знань і навиків, необхідних для роботи з символьною інформацією.

5.1 Основні теоретичні відомості

В мовах *C* та *C++* існують достатньо розвинені засоби для обробки символьної інформації, такі як введення/виведення окремих символів та рядків, функції для визначення належності символів та рядків, об'єднання рядків, перетворення їх.

Рядок - це послідовність символів, що заключенні у подвійні лапки. Транслятор додає в кінець кожного рядка нульовий байт *'\0'*, так що програма, що переглядає рядок, може знайти її кінець.

Приклад:

cout<<"This is character string";

Кожна рядкова константа, навіть якщо вона ідентична іншій рядковій константі, зберігається в окремому місці пам'яті.

Рядкова константа – це масив символів. Вона має тип *char[]*. Символьна константа уявляє собою деякий символ в лапках.

Приклад :

'a', 'A', '7', ''.*

Символьні константи вважаються данними типу *int*.

Існує декілька способів визначення рядків: використання рядкових констант, масивів типу *char*, покажчиків на тип *char* та масивів, що складаються з символьних рядків.

Приклади :

1. Ініціалізація масива за допомогою рядкової константи:

char m[]="string";

Даний оператор ініціалізував зовнішній масив m для вказаного рядка. Аналогічно

char m[]={'s','t','r','i','n','g','\0'};

ім'я m - покажчик на нульвий елемент масиву:

*m = =&m [0]; *m = = ‘s’, *(m+1) = = m [1] = = ‘t’.*

2. *а) char *l = “string”;*

б) static char l[] = “string”;

Опис з масивом б) визиває створення масиву з сьоми елементів (кількість символів у слові *string+1* символ, що завершує *‘\0’*). Кожний елемент ініціалізується сумісним символом. Компілятор розглядає ім'я *l* як синонім адреси першого елемента масиву, де *l* - константа покажчика. Можно використовувати операції *l+1* для ідентифікації наступного елемента масиву (*‘t’*). Але не можна використовувати вираз *++l* .

Форма з покажчиком а) також визиває створення в статичній пам'яті сім елементів для зберігання рядка. Окрім того, виділяється ще одна чарунка пам'яти для змінної *l*, яка є покажчиком. Спочатку ця змінна вказує на початок рядка, але її значення може змінюватися. В даному випадку, використовуючи операції збільшення, *++l* можно посилатися на елемент *‘t’*. Таким чином, у випадку а) ініціалізовано одну змінну типу покажчик.

Приведемо приклад явного задання розміру пам'яти.

Приклад :

char l[7] = “string”;

char name [81];

Масив *name* буде заповнюватись у процесі роботи, саме тому резервується 80 символів. Якщо кількість елементів в *name* буде менш ніж 80, то невикористані елементи автоматично ініціалізуються символом нуль.

Введення-виведення рядків

Розглянемо найбільш розповсюдженні бібліотечні функції *gets()* та *puts( )*.

Функція *gets( )* отримує рядок з буфера клавіатури. Вона зчитує символи до тих пір, поки не зустріне символ нового рядка (‘\n’). Функція бере всі символи до знака ‘\n’, приєднує до них нуль-символ та передає рядок програмі, що її визиває.

Приклад:

/ отримання імені * /*

main ( )

{ char name [81]; / виділення пам'яти*/*

printf ("Як вас звуть? \n");
gets(name); / ім'я поміщується до масиву name */*
cout<<"Гарне ім'я"<<name<<"\n";

Функція *puts( )* виводить рядок символів на консоль виводу, у неї є тільки один аргумент - покажчик на рядок.

Приклад:

```
void main ( ) {
char str[]="Приклад";
puts (str);
}
```

Операції над рядками

Для використання наведених далі функцій необхідно додати до програми файл *string.h*:

#include <string.h>

Таблиця 5.2 – Функції для роботи з рядками

№	Функція	Опис
1	*char *strcat (st1, st2);* *char *st1, *st2;*	Використовується для злиття рядків символів. В результаті st2 заповнюється символами st1 та st2.
2	*char *strncat (st1, st2, n);* *char *st1, *st2;* *int n;*	Виконує злиття двох рядків так, що з другого рядка копіює не больш за n символів.
3	*int strcmp (st1, st2);* *char *st1, *st2;*	Порівнює два рядка у лексикографічному порядку. Повертає 0, якщо st1=st2 -1, якщо st1<st2 1, якщо st1>st2
4	*int strncmp (st1, st2, n);* *char *st1, *st2;* *int n;*	Порівнює перші n символів двох рядків.
5	*char *strcpy (st1, st2);* *char *st1, *st2;*	Копіює рядок st2 в st1.

6	*char *strncpy (st1, st2, n);* *char *st1, *st2;* *int n;*	Копіює не більш ніж n символів рядка st2.
7	*int strlen (str);* *char *str;*	Визначає довжину рядка, тобто кількість символів у рядку без завершаючого ‘0’.
8	*char *strchr (str, c);* *char *str;* *int c;*	Знаходить у вказаному рядку перші входження символа с.
9	*char *strrchr (str, c);* *char *str;* *int c;*	Знаходить у рядку останнє входження символа с.

Продовження таблиці 5.2

10	*char *strpbrk (st1, st2);* *char *st1, *st2;*	Знаходить у рядку st1 деякий із множини символів, що входять до рядка st2.
11	*char *strspn (st1, st2);* *char *st1, *st2;*	Визначає довжину відрізка рядка st1, що містить символи зі множини символів, що входять до рядка st2.
12	*char *strtok (st1, st2);* *char *st1, *st2;*	Виділяє з рядка st1 лексеми, разподілені любим із множини символів, що входять до рядка st2.

Приклад:

```
/* використання функцій strcat( ) и strcpy( )*/
main ( )
{
static char name [80];
static char cpname [80];
static char add [] = “додано до кількості користувачів системи”;
printf(“Вкажіть ваше прізвище.\n”);
gets(name);
strcpy(cpname, name);                    /*копія прізвища*/
strcat(name, add);
puts(name);
```

```
puts(cpname);
puts(add);
}
```

Результати роботи програми:

Вкажіть ваше прізвище.
Іванов
Іванов додано до кількості користувачів системи
Іванов
додано до кількості користувачів системи

Перевірка та перетворення символів

Файл *ctype.h* містить декілька макровизначень, що перевіряють, до якого класу належать символи.

Визначення:

```
int c;
```

Таблиця 5.3 – Функції перевірки належності символів

Функція	Призначення
isalpha (c)	перевіряє, чи с є символом літерою
isdigit (c)	перевіряє, чи с є символом цифрою
islower (c)	перевіряє, чи с є символом рядковою літерою
isspace (c)	перевіряє, чи с є символом пустим символом (пробіл, табуляція чи новий рядок)
issuper (c)	перевіряє, чи с є символом прописною літерою
isascii (c)	перевіряє, чи с є символом кодом A S C I I
iscntrl (c)	перевіряє, чи с є символом управляючим символом
ispunct (c)	перевіряє, чи с є символом знаком пунктуації
isalnum (c)	перевіряє, чи с є символом літерою чи цифрою

Приклад :

```
isalpha ('s')! = 0
isalpha ('#') = = 0
```

Функція *isalpha(c)* повертає ненульове значення (істина), якщо *c* є символом літери, та нуль (не істина) - у противному випадку. Перетворення символьних рядків:

atoi( ), atof( )

Функція *atoi( )* перетворює рядок в ціле.

Функція *atof( )* перетворює рядок в число із плаваючою точкою.

Система може мати зворотні функції: *itoa( )* перетворює ціле в символьний рядок, а функція *ftoa( )* число із плаваючою точкою в символьний рядок.

Приклад :

```
        /*включення atoi( )*/
#include <stdio.h>
main( )
{
static char number [10];
int value;
puts("Введіть ціле число");
gets(number);
value = atoi (number);
printf("Число= %d\n",value);
}
```

Функція *atoi( )*, ігноруючи початкові і кінцеві пропуски, обробляє цифри та алгебраїчний знак, якщо він є. Обробка ведеться до тих пір, поки символ, що обробляється, є цифрою чи знаком.

Функція *atof( )* виконує ті ж самі дії для чисел із плаваючою точкою. Вона повертає тип double, саме тому повинна бути описана як double у програмі, що її використовує.

Кожному символу ставиться у відповідність ASCII-код. Таблиця ASCII-кодів наведена у додатку Д.

Робота з рядками в стилі C++. Тип string.

Основні дії з рядками типу string виконуються в ньому за допомогою операцій та методів, а довжина рядка змінюється динамічно відповідно до потреб.

```
#include<string>
```

string S1, S2, S3;
S3=S1+S2

Синтаксис операцій та їх дії очевидні. Розміри рядків встановлюються автоматично так, щоб об'єкт міг містити значення, що йому присвоюється.

Таблиця 5.4 – Операції над рядками типу string

Операція	**Дія**
=	присвоєння
+	конкатенація
==	рівність
!=	нерівність
<	менше
<=	менше або дорівнює
>	більше
>=	більше або дорівнює
[ ]	індексація
<<	виведення
>>	введення
+=	конкатенація з присвоєнням

Таблиця 5.5 – Операції над рядками типу string

Функція	**Дія**
at()	повертає посилання на символ, вказаний як параметр
assingn()	привласнення частини одного рядка іншого. (+=)
append()	додає частину одного рядка до іншого
insert()	вставляє частину одного рядка до іншого
erase()	видаляє частину рядка
repace()	заміна частини рядка
substr()	видалення частини рядка
c_str ()	перетворення типу у рядок старого стилю
copy()	копіювання частину рядка

find()	пошук підрядка
size()	визначення довжини рядка
compare()	порівняння рядків

5.2 Завдання на лабораторну роботу

1. Ознайомитися з теоретичними відомостями даної лабораторної роботи.
2. Скласти та виконати програми відповідно до варіанту.
3. Оформити звіт та захистити роботу.

5.3 Індивідуальні завдання

№	Варіанти завдань.
1	1. Написати програму, яка зчитує з текстового файлу слова та виводить кожне на екран у зворотному порядку. 2. Написати програму, яка зчитує текст із файлу та виводить на екран речення, що містять максимальну кількість знаків пунктуації.
2	1. Написати програму, яка зчитує текст із файлу та виводить на екран лише речення, що містять введене з клавіатури слово. 2. Напишіть програму, яка обчислює середнє арифметичне дійсних чисел, що знаходяться у файлі *numbers.txt*.
3	1. Написати програму, яка зчитує текст із файлу і виводить на екран лише рядки, що містять двоцифрові цілі числа. 2. Написати програму, яка зчитує текст із файлу і виводить на екран лише речення, що не містять коми.
4	1. Написати програму, яка зчитує англійський текст із файлу і виводить на екран слова, що починаються з голосних літер. 2. Дано файл *f.txt,* компоненти якого є додатними цілими числами. Записати у файл *g.txt* усі парні числа файлу *f.txt*, а файл *h.txt* – непарні. Порядок слідування чисел зберігається.
5	1. Написати програму, яка зчитує текст із файлу і виводить на екран лише цитати, тобто речення в лапках. 2. Дано файл *f.txt,* компоненти якого є дійсними числами. Знайти суму найбільшого та найменшого із значень

	компонентів.
6	1. Написати програму, яка зчитує англійський текст із файлу та виводить на екран слова тексту, що починаються та закінчуються на голосні літери. 2. Написати програму, яка знаходить максимальне та мінімальне значення у файлі дійсних numbers.txt.
№	**Варіанти завдань.**
7	1. Написати програму, яка зчитує текст із файлу і виводить на екран лише рядки, які не містять двоцифрових цілих чисел. 2. Написати програму, яка створить новий файл output.txt , що містить усі прості числа файлу input.txt.
8	1. Написати програму, яка зчитує текст із файлу і виводить на екран лише речення, що складаються із заданої кількості слів. 2. Написати програму, яка зчитує текст із файлу та визначає, скільки в ньому слів, що складаються з не більше ніж чотирьох літер.
9	1. Написати програму, яка зчитує текст із файлу, знаходить найдовше слово і визначає, скільки разів воно зустрілося в тексті. 2. Даний файл *f.txt*, компоненти якого є цілими числами. Отримати у файлі *g.txt* усі компоненти файлу *f.txt*, які є точними квадратами.
10	1. Написати програму, яка зчитує англійський текст із файлу і виводить на екран для кожної літери алфавіту її кількість та ймовірніть. 2. Написати програму, яка зчитує текст із файлу і виводить на екран лише речення, що починаються з тире, перед яким можуть бути лише символи пробілу.

5.4 Контрольні запитання

1. Що таке символ?
2. Що таке ASCII-код символу?
3. Уявлення рядка у пам’яті.
4. Наведіть функції для роботи з рядками.
5. Яка бібліотека містить функції для роботи з рядками?

ЛАБОРАТОРНА РОБОТА № 6.
СТРУКТУРИ. ОБ'ЄДНАННЯ. БІТОВІ ПОЛЯ СТРУКТУР І ОБ'ЄДНАНЬ

Мета роботи: Отримати практичні навички при використанні операцій обробки структур та об'єднань. Навчитись зберігати дані структур у типізованих файлах.

6.1 Основні теоретичні відомості

Структура - це об'єднання однієї або більше змінних, можливо, різних типів, в одну групу, що для простоти роботи має одне ім'я.

Структури використовують для організації складних даних в великих програмах. Тому що вони в багатьох ситуаціях дозволяють групувати в єдине ціле поріднені між собою змінні та працювати з ними як з єдиним цілим, а не з окремими складовими. Структури також дозволяють створювати нові типи даних.

Використання покажчиків зі структурами бажане по трьом причинам: їх легше використовувати; структура не може використовуватись в якості аргументу функції, а покажчик на структуру може; багато типових структур даних є структурами які містять покажчики на інші структури.

Загальна форма об'явлення структури наступна:

```
struct ім'я_структури {
тип 1 поле 1;
тип 2 поле 2;
……
тип n поле n;
} список_змінних;
```

Наприклад. Об'явимо новий тип *stud*

```
struct stud {
char name[80];
int age;
double b;
} st1, st2;
stud Ivan,Petr;  //Об'являємо змінні Ivan, Petr типа stud
```

На рис. 8.1 наведена уява змінної st1 у пам’яті.

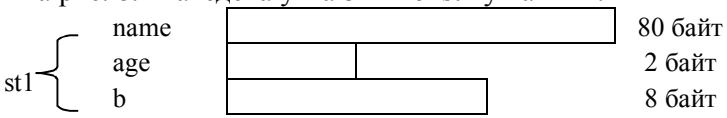


Рисунок 8.1 – Ділянка пам’яті, що виділена під змінну st1

Отже, під змінну st1 в пам’яті виділяється 86 байт

При визначенні структур можлива їх ініціалізація, тобто задання початкових значень їх елементів. Наприклад:

stud st1={“Ivanov”, 20, 4.6};

Для звертання до елемента структури використовують оператор *‘.’*

ім‘я_структури.ім’я_елемента_структури

Наприклад:

st1.name=”Ivan”;

st1.age=20;

gets(st2.name);

for (int t=0; st1.name[t]; t++) cout<<st1.name[t];

В *C++* вміст однієї структури можна присвоїти другій, якщо обидві ці структури мають один тип.

Наприклад:

struct tt{

int a, b;} var1, var2;

var1=var2;

В *C++* покажчики на структури можна використовувати таким самим чином, як і покажчики на змінні різного типу.

*ім‘я_структурного типу *ім’я покажчика_на_структуру;*

Наприклад:

*stud *pStud;*

pStud=&st1;

В *C++* для того, щоб отримати доступ до членів структури за допомогою покажчика, використовується оператор *„–>”*.

pStud–>age;

Інша можливість звернення до елементу структури за допомогою адресуючого її покажчика – це розіменування покажчика та формування уточнюючого ім’я такого виду:

*(*ім’я_покажчика).ім’я_елемента_структури*

Покажчик на структуру може входити до визначення того ж структурного типу. Саме так у визначенні формата структуру *record* введені два покажчики:

*record *prior* – покажчик на попередній елемент у двохзв'язному списку структур;

*record *next* – покажчик на наступний елемент у двозв'язному списку структур.

Об'єднання.

Об'єднання це тип, де всі змінні розділяють одну ділянку пам'яті. Загальна форма запису об'єднання наступна:

union ім'я_типу (
тип 1 поле 1;
тип 2 поле 2;
- - - - - - - - - -
тип N поле N;
*} перелік*_змінних;

Наприклад:
union { long L; int il, i2; char c[4]; } UNI;

Відповідно до структур звернення до елемента об'єднання можливо через им'я об'єднання або через покажчик:

ім'я_объеднання.ім'я_елемента
покажчик_на_об'єднання->і'мя_елемента
*(*покажчик_на_об'єднання).і'мя_елемента*

6.2 Завдання на лабораторну роботу

1. Ознайомитися з теоретичними відомостями.

2. Скласти та виконати програми відповідно до індивідуального завдання згідно варіанту. Дані повинні зберігатись у структурі або об'єднанні. Реалізувати введення та виведення заданих інформаційних масивів у типізований файл.

3. Оформити та захистити звіт.

6.3 Індивідуальні завдання

№	Варіанти завдань.
1	Дані про студента складаються з його ім'я та прізвища, року народження, та номера групи, в якій він вчиться. Є інформаційний масив f, в якому містяться дані про студентів факультету та їх оцінки за останню сесію. а) з'ясувати, скільки учнів школи мають оцінки не нижче 75 балів; б) середній бал кожного студента; в) зібрати в інформаційному масиві g дані про 25% найкращих студентів школи, тобто студентів, що не мають оцінки нижче 75 балів та за сумою балів входять до 25% найкращих студентів.
2	Дані про автомобіль складаються з його марки, ціни, номера та прізвища власника. Дано інформаційний масив *f*, в якому записані дані про декілька автомобілів. Знайти: а) прізвища власників та номера авто заданої марки; б) кількість автомобілів кожної марки. в) дані щодо найдорожчого та найдешевшого автомобілів.
3	Дано інформаційний масив *f*, в якому зібрані дані про книги (прізвище автора, назва, рік друку та кількість сторінок). а) знайти назви книг даного автору, що надруковані з 2000 р. б) визначити чи є книга с назвою «Алгоритмізація та програмування». Якщо є, то видати прізвище автора та рік друку. Якщо таких книг декілька, то видати всі дані за цими книгами; в) знайти всі книги, які містять не менше заданої кількості сторінок та вивести інформацію про них на екран.
4	Є інформаційний масив *f*, що містить інформацію про працівників підприємства: табельний номер, прізвище та ініціали співробітника, оклад та номер телефону. а) вивести всю інформацію про співробітників у вигляді таблиці; б) знайти працівників, оклад яких не перевищує заданого значення; в) знайти телефон співробітника за його ПІБ.

5	Є інформаційний масив *f*, в якому містяться різні дати. Кожна дата – це три змінних: число, місяць та рік. Знайти: а) рік з найменшим номером; б) усі зимові дати; в) саму пізню дату.
6	Дано інформаційний масив *f*, в якому збережено дані про кубики: розмір кожного кубику (довжина ребра в міліметрах), його колір (зелений, білий, червоний чи синій) та матеріал (дерево, метал, пластик). Знайти: а) кількість кубиків кожного з перелічених кольорів; б) сумарний об'єм кубиків зроблених із заданого матеріалу; в) кількість дерев'яних кубиків з ребром не меншим заданого значення.
7	Дано інформаційний масив *f*, в якому збережено дані про навчальні дисципліни, що містять назву дисципліни, кількість кредитів, номер семестру та код спеціальності. Знайти: а) дисципліни, що не перебільшують задану кількість кредитів; б) дисципліни, що читаються в певному семестрі; в) дисципліни, що читаються на певній спеціальності.
8	Дано інформаційний масив f, в якому збережено дані про олімпіадні змагання: назва команди, склад команди, кількість вирішених задач, рейтинг в списку переможців. Знайти: а) команди, які зайняти призові місця; б) команди, які вирішили не менше половини задач; в) інформацію про команду за її назвою;
9	Дано інформаційний масив f, в якому збережено дані про замовлення в інтернет-магазині: номер замовлення, дата замовлення, адреса доставки, сума оплати, статус (замовлено, доставлено, сплачено). Знайти: а) всі замовлення з вказаною датою; б) інформацію про конкретне замовлення за його номером; в) інформацію про всі замовлення, що мають певний статус;
10	Дано інформаційний масив f, в якому збережено дані про екскурсійні програми: назва програми, країна подорожі, дата початку та дата закінчення, ціна на одну персону. Знайти: а) екскурсійні програми до заданої країни, що починаються після вказаної дати;

	б) екскурсійні програми, ціна за які не перевищує заданої суми; в) екскурсійні програми, які владуються в заданий діапазон дат;

6.4 Контрольні питання

1. Чим відрізняється визначення шаблона структури від визначення структури?
2. Запишіть (у виді прикладів) способи доступу до членів структури, до елементів масиву - члена структури, до членів структури, що є елементом масиву.
3. Що означає код *(*str).alpha* і як його ще можна записати?
4. Як ініціалізувати елементи структури?
5. Напишіть код, що визначає розмір масиву структур.
6. Чи правильний запис *?*

   struct x{int x,y;double z[2];}x;
7. Схематично показати розподіл пам'яті для структури

   struct{int alfa; int a : 5; int b : 3; char ch;}record;
8. Наведіть приклад використання структури в іншій структурі. Привласніть значення одному елементу цієї структури.
9. Що таке об'єднання (*union*)? Як створити шаблон об'єднання і саме об'єднання?
10. Схематично показати розподіл пам'яті для об'єднання

    union {long word;char delta; int number;char gamma[4];}UN;
11. Як ініціалізувати об'єднання?
12. Чим структура відрізняється від об'єднання?
13. Для чого можна використовувати бітові поля?
14. Як задається розмір бітового поля при описі його в структурі?
15. Запишіть визначення структури для збереження імені, що включає до 40 символів, поля для збереження інформації про родинний стан як одного з чотирьох значень і поля для збереження відділу як одного з восьми значень.
16. Чи може до бітового поля застосовуватися операція "одержати адресу" &?

ЛАБОРАТОРНА РОБОТА № 7.
АЛГОРИТМИ ПОШУКУ ТА СОРТУВАНЯ

Мета роботи: Отримати практичні навички при використанні операцій обробки структур та об'єднань. Навчитись зберігати дані структур у типізованих файлах.

7.1 Основні теоретичні відомості

Одна з найбільш часто застосовуваних у програмуванні дій - пошук. Існує декілька основних варіантів пошуку, і для них створюється багато різних алгоритмів. При організації пошуку розрізняють пошук у впорядкованому та невпорядкованому масивах.

Лінійний пошук.

Якщо немає жодної додаткової інформації про дані, що розшукуються, то очевидний підхід - просто послідовний перегляд масиву із збільшенням крок за кроком тієї його частини, де шуканий елемент не знайдено. Такий метод має назву лінійний пошук.

```
#include <iostream>
#include <locale>
#include "windows.h"
using namespace std;

int main(){
	SetConsoleCP(1251);
SetConsoleOutputCP(1251);
	const int n = 10;
	int x, i = 0;
	int a[n] = { 1, 8, 5, 6, 3, 7, 6, 3, 9, 0 };
	cout << "Input x=>";
	cin >> x;
	while ((a[i]!= x) && (i < n)) i++;
	if (a[i]!= x) cout << "Елемент відсутній";
	else cout << "Результат a[" << i << "]";
	return 0;
}
```

Очевидно, що інших способів пришвидшення пошуку не існує, якщо, звісно, немає ще якоїсь інформації про дані, серед яких іде пошук. Добре відомо, що пошук можна зробити значно ефективнішим, якщо дані будуть впорядковані.

Бінарний пошук.

Наступний алгоритм, заснований на знанні того, що масив a на вході вже впорядкований. Основна ідея - обрати випадково деякий елемент at та порівняти його з аргументом пошуку x. Якщо at = x, то пошук закінчується, Якщо at < x, то ми робимо висновок, що всі елементи з індексами, меншими або рівними т, можна виключити з подальшого пошуку; якщо at > x, то можна виключити індекси більші або рівні т. Це приводить нас до алгоритму *бінарного пошуку* (пошуку діленням навпіл, діхотомії).

Алгоритм бінарного пошуку.

Крок 1. Задаємо початкові значення лівої (left - лівий індекс) та правої (right - правий індекс) межі пошуку. Якщо пошук проводиться по всьому масиву, то початкові значення це 0 та загальна кількість елементів масиву n.

Крок 2. Обираємо випадковий елемент у масиві a[m]. Як варіант, середній (middle) елемент.

Крок 3. Якщо a[m] дорівнює шуканому елементу або лівий індекс більше правого, переходимо на крок 6, інакше, якщо a[m] < шуканого елемента, переходимо на крок 4, інакше (якщо a[m]> шуканого елемента), переходимо на крок 5.

Крок 4. Змінити значення лівої межі на m+1. Перехід на крок 2.

Крок 5. Змінити значення правої межі на m-1. Перехід на крок 2.

Крок 6. Виведення результатів.

```
#include <iostream>
#include <locale>
#include "windows.h"
using namespace std;
int main(){
      SetConsoleCP(1251);
SetConsoleOutputCP(1251);
const int n = 10;
```

```
	int a[n] = { 1, 8, 5, 6, 3, 7, 6, 3, 9, 0 };
	int x, l, r, m;
	bool found;
	cout << "Input x=>";
	cin >> x;
	l = 0;
	r = n - 1;
	found = false;
	while ((l <= r) && (!found))
	{
		m = (l + r) / 2;
		if (a[m] == x) found = true;
		else {
			if (a[m] < x) l = m + 1;
			else r = m - 1;
		}
	}
	if (found) cout << "Результат a[" << m << "]";
	else cout << "Елемента немає";

	return 0;
}
```

Алоритми сортування.

Розрізняють методи сортування масивів і сортування файлів. Методи сортування запринципом дії поділяються на три категорії:

– сортировки з допомогою включень (вставками);

– сортировки з допомогою вибору;

– сортування за допомогою обміну.

Сортування бульбашкою

Алгоритм працює шляхом таким порівняння кожних двох сусідніх елементи. Якщо один з елементів не відповідає критерію сортування (в залежності від шуканого порядку по зменшенню або по зростанню), то ці два елементи міняються місцями. Прохід по списку продовжується доти, доки всі дані не будуть відсортованими. Алгоритм нагадує поведінку бульбашок повітря у резервуарі з водою, від чого і отримав свою назву.

Сортування методом прямого включення

Елементи масиву, починаючи з другого, послідовно перебираються по одному і включаються на потрібне місце у вже впорядковану частину масиву, що розташована зліва від поточного елемента *a[i]*. Позиція включення заздалегідь не відома. Визначимо її порівнюючи в циклі по черзі *a[i]*. з *a[i-1]*. *a[i-2]*. до тих пір, поки не буде знайдений лівий кінець масиву.

7.2 Завдання на лабораторну роботу

Завдання 1. Відсортувати інформаційний масив із лабораторної роботи № 6 за ключовим полем методом бульбашки.

Завдання 2. Реалізувати, проаналізувати та оцінити складність алгоритму сортування відповідно варіанту за швидкодією (практично та теоретично) використовуючи сортування векторів типу double розміром n, змінюючи значення n від 100 до 1000000000. Навести схему алгоритму.

7.3 Індивідуальні завдання

№	Варіанти завдань.
1	Сортування вибором
2	Сортування вставками
3	Пірамідальне сортування
4	Швидке сортування
5	Сортування злиттям
6	Сортування підрахунком
7	Сортування Шелла
8	Сортування за розрядами
9	Сортування перестановкою
10	Плавне сортування (англ. Smoothsort)

7.4 Контрольні питання

1. Назвіть основні принципи побудови алгоритмів сортування.
2. Поясніть назву алгоритму «сортування бульбашкою»
3. Як пояснити теоретичну оцінку складності алгоритму O()?

4. Чим відрізняються методи сортування масивів і сортування файлів.

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Кормен, Томас Г. Алгоритми доступно / пер. з англ. Кирила Яценка, Київ: К.І.С., 2021. – 194 с.

2. Щербаков О.В. Основи об'єктно-орієнтованого програмування навчальний посібник / О.В. Щербаков, Ю.Е. Парфьонов, В.М. Федорченко. – Харків : ХНЕУ ім. С. Кузнеця, 2019. – 237 с.

3. Алхімова С.М. Об’єктно-орієнтоване програмування: підручник. У 2-х ч. Об’єктно-орієнтований підхід до розробки програмного забезпечення / С.М. Алхімова. - Київ: КГП ім. Ігоря Сікорського, Вид-во «Політехніка», 2019.

4. Bjorn Andrist C++ High Performance: Master the art of optimizing the functioning of your C++ code, 2nd Edition / Bjorn Andrist – Packt Publishing, 2020. – 540 p.

5. Bill Weinman C++20 STL Cookbook: Leverage the latest features of the STL to solve real-world problems 1st Edition / Bill Weinman – Packt Publishing, 2022. – 450 p.

6. Marius Bancila Template Metaprogramming with C++: Learn everything about C++ templates and unlock the power of template metaprograming 1st Edition / Marius Bancila – Packt Publishing, 2022. – 480 p.

11. Гришанович Т.О., Основи об’єктно-орієнтованого програмування : навч. посібник / Гришанович Т.О., Глинчук Л.Я. – ВНУ імені Лесі Українки, 2022. – 120 с.

15. Алгоритм_сортування. [Електронний ресурс] https://uk.wikipedia.org/wiki/Алгоритм_сортування

Додаток А.
Титульний аркуш

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»

Кафедра інформаційної безпеки та наноелектроніки

Лабораторна робота № 1
з дисципліни «Основи алгоритмізації та програмування»
на тему «Програмування лінійних та
розгалужених обчислювальних процесів»

Виконав(ла)
ст. гр. БК-815 В.О. Знайко

Прийняв
к.т.н, доц. Г.В. Неласа

Запоріжжя, 2025

Додаток Б.
Основні типи даних. Модифікатори типів

Всі типи даних, що використовуються у С++, поділяються на:
- вбудовані (базові),
- похідні (покажчики та посилання),
- користувацькі (в т.ч. класові).

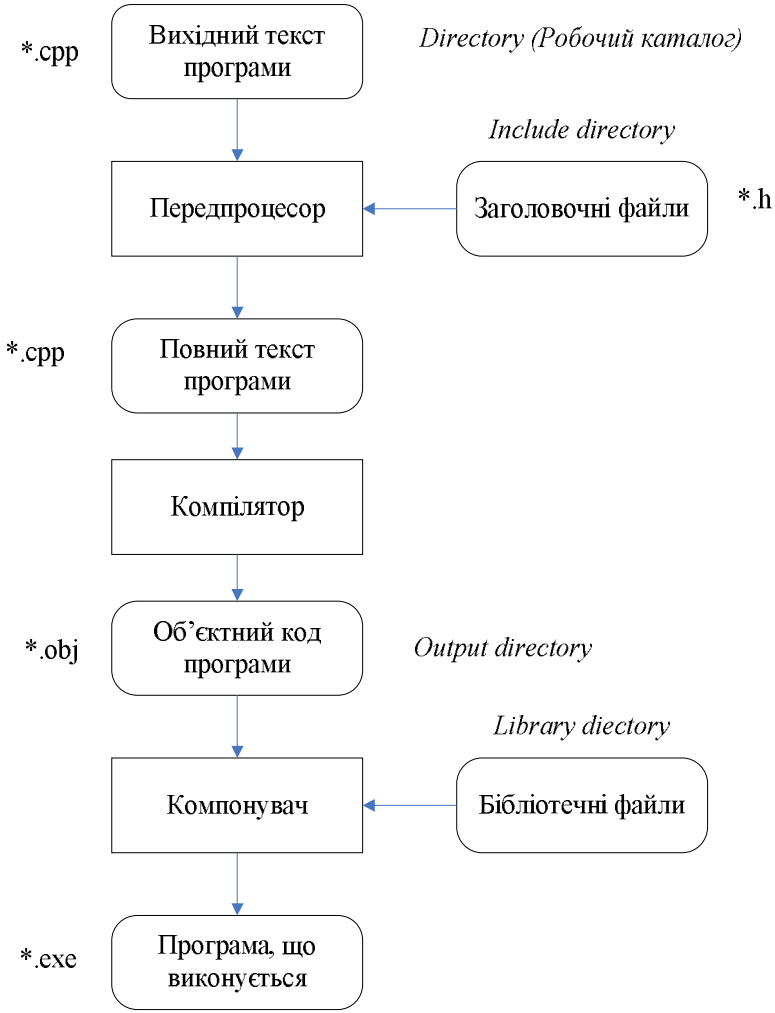
До базових типів відносять: *int, char, bool, float, double.*

Модифікатори типів: *signed, unsigned, long, short.*

Таблиця Б.1 – Діапазони базових типів мови С++

Тип даних	**Розмір, біт**	**Діапазон**
bool	8	true, false
signed char	8	-128..127
unsigned char	8	0..255
signed short int	16	-32768..32767
unsigned short int	16	0..65 535
signed long int	32	-2 147 483 648...2 147 483 647
unsigned long int	32	0..4 294 967 295
signed long long int	64	$-2^{63}..2^{63}-1$
unsigned long long int	64	$0..2^{64}-1$
float	32	6 значущих цифр
double	64	10 значущих цифр
long double	80	10 значущих цифр

Додаток В.
Схема компіляції програми



Додаток Д.
Таблиця ASCII – кодів символів

Таблиця Д.1 – Символи ASCII з кодами 0-127

Код	Символ	Код	Символ	Код	Символ	Код	Символ
0	NUL	32	SP	64	@	96	`
1	SOH	33	!	65	A	97	a
2	STX	34	“	66	B	98	b
3	ETX	35	#	67	C	99	c
4	EOT	36	$	68	D	100	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	‘	71	G	103	g
8	BS	40	(	72	H	104	h
9	HT	41	)	73	I	105	i
10	LF	42	*	74	J	106	j
11	VT	43	+	75	K	107	k
12	FF	44	,	76	L	108	l
13	CR	45	-	77	M	109	m
14	SO	46	.	78	N	110	n
15	SI	47	/	79	O	111	o
16	DLE	48	0	80	P	112	p
17	DC1	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	S	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	v
23	ETB	55	7	87	W	119	w
24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y
26	SUB	58	:	90	Z	122	z
27	ESC	59	;	91	[	123	{
28	FS	60	<	92	\	124	|
29	GS	61	=	93	]	125	}
30	RS	62	>	94	^	126	~
31	US	63	?	95	_	127	Del