НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ
УКРАЇНИ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

# Мова програмування C++ для початківців

Укладач: Ю.О. Міловідов

Київ 2019

# ЗМІСТ

# ВСТУП

На сьогоднішній день C++ є одною з найпопулярніших і поширених мов програмування.

Своїм корінням вона йде в мову Сі, яка була розроблена в 1969-1973 роках в компанії Bell Labs Деннісом Рітчі (Dennis Ritchie). На початку 1980-х років данський програміст Бйорн Страуструп (Bjarne Stroustrup), який в той час працював в компанії Bell Labs, розробив C++ як розширення до мови Сі.

Згодом нова мова стала набирати популярність.

До неї були додані нові можливості, які робили її абсолютно новою мовою програмування.

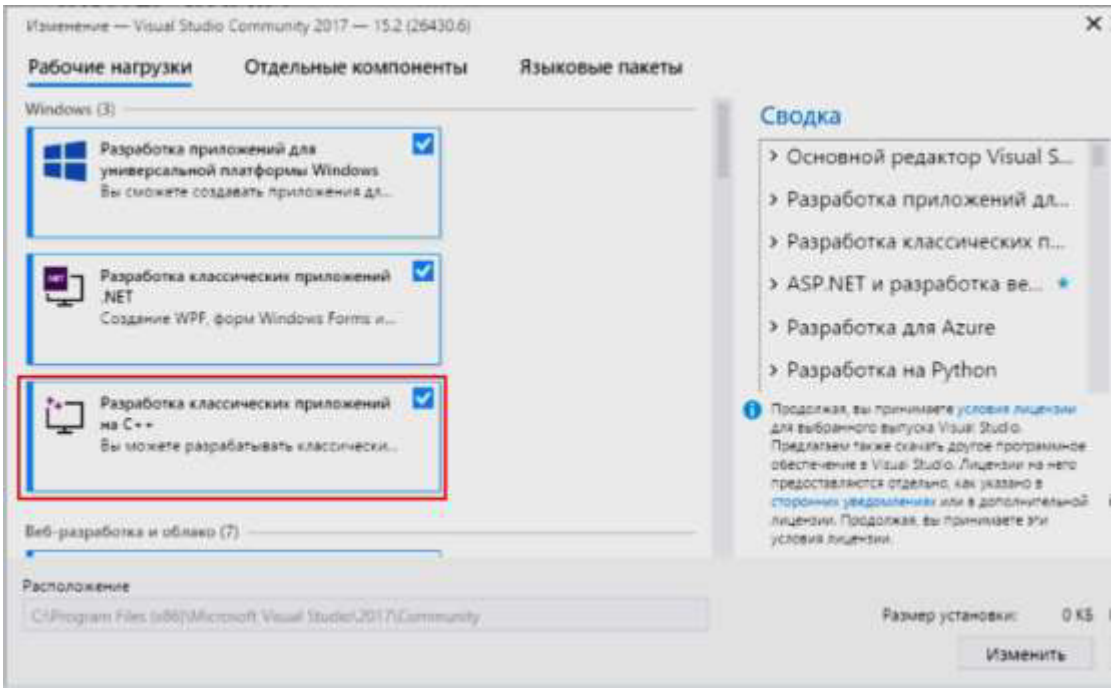
C++ - це мова широкого користування, на якій можна створювати практично будь-які види програм.

# 1 УСТАНОВКА ІНТЕГРОВАНОГО СЕРЕДОВИЩА РОЗРОБКИ

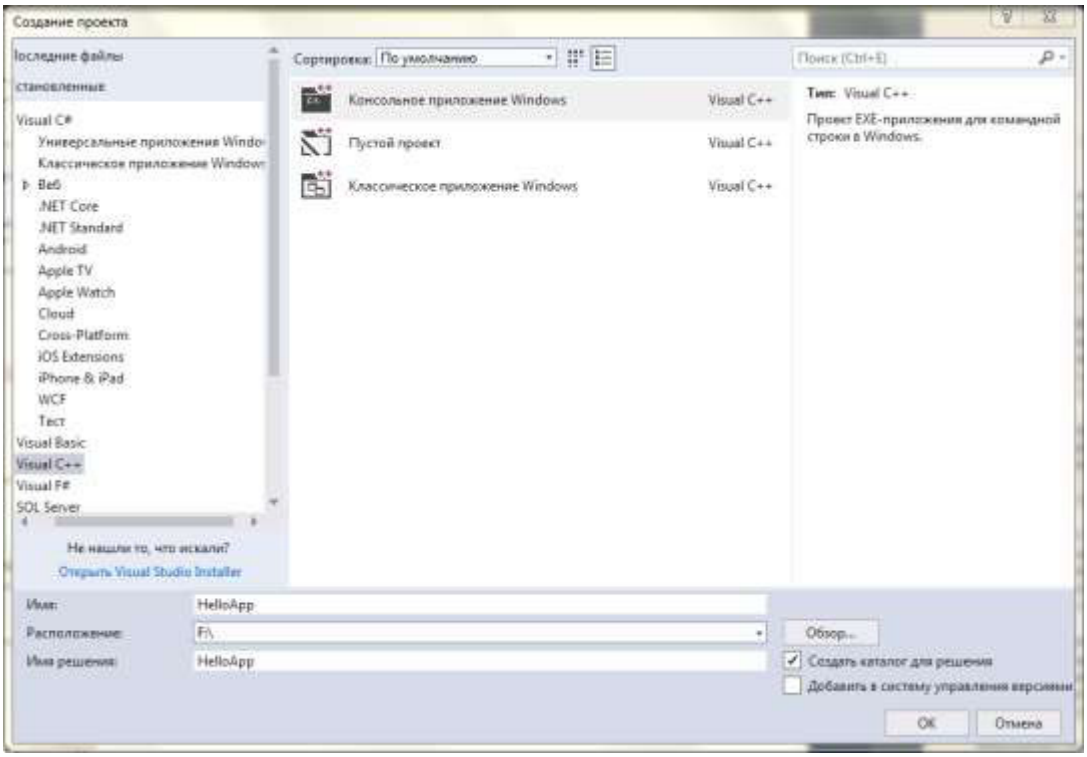
Найперше, що ви повинні зробити, перш ніж приступити до вивчення C++, це переконатися, що у вас є IDE – інтегроване середовище розробки (програма в якій ви будете програмувати).

Для програмування під Windows найбільш популярним середовищем розробки є Visual Studio. Дане середовище можна знайти за посиланням https://www.visualstudio.com/ru/vs/. Зокрема, можна використовувати безкоштовну і повнофункціональну версію Visual Studio 2017 Community.

Після завантаження та запуску установника Visual Studio в ньому необхідно поставити галочку навпроти пункту "Розробка класичних застосувань на C ++":

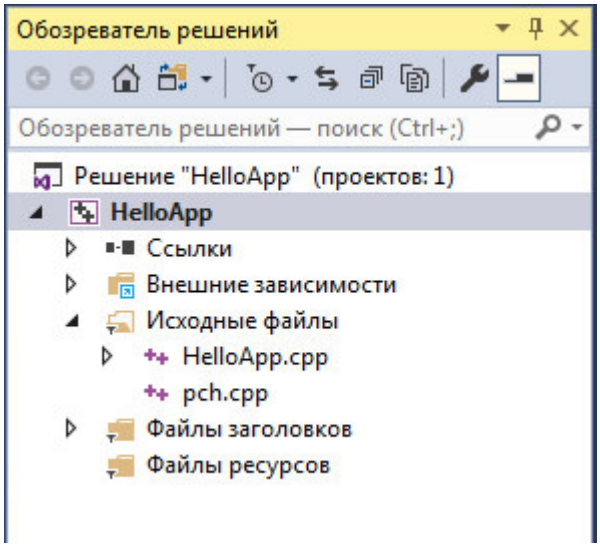



Вибравши всі необхідні пункти, натиснемо ОК для запуску установки. Після установки Visual Studio створимо перший проект. Для цього перейдемо в меню **File** (Файл) -> **New** (Створити) -> **Project** ... (Проект), і нам відкриється вікно створення нового проекту. У ньому перейдемо в лівій частині вікна до мови C ++ і виберемо його підсекцію **Консольное приложение Windows**:




Нехай ім’я нашого проекту буде HelloApp.

Створений  проект матиме наступну структуру:




Тепер власне створимо першу програму і визначимо в файлі HelloApp.cpp найпростіший код, який буде виводити рядок на консоль:

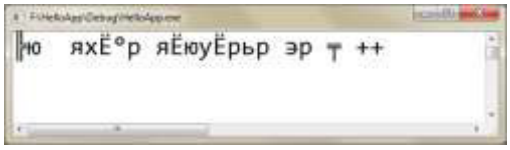
```
#include "pch.h"
#include <iostream>
using namespace std;
int main()
{
    cout << "Hello World!\n";
    cin.get();
}
```

Результат роботи програми:


Трохи модифікуємо програму:

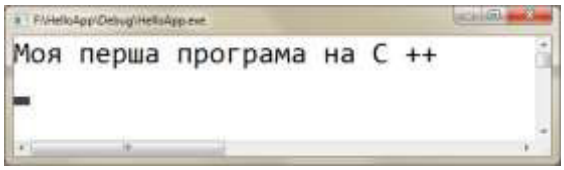
```
int main()
{

    cout << "Моя перша програма на С ++\n";
    cin.get();
}
```

Результат роботи програми:



Для правильного виводу символів кирилиці треба додати ще одну команду:

```
int main()
{
 setlocale(0, "Ukr"); //
    cout << "Моя перша програма на С ++\n";
 cin.get();
}
```




Розглянемо докладно елементи програми. `#include` це директива “препроцесору”, яка повідомляє компілятору помістити код з заголовком `iostream` в нашу програму перед тим як створити виконуваний файл. Підключивши до програми заголовний файл ви отримуєте доступ до безлічі різних функцій, які можете використовувати у своїй програмі. Наприклад, оператору `cout` потрібно `iostream`.

Рядок `using namespace std;` повідомляє компілятору, що потрібно використовувати групу функцій, які є частиною стандартної бібліотеки `std`. У тому числі цей рядок дозволяє програмі використовувати оператори, такі як `cout`. Крапка з комою є частиною синтаксису C++ . Вона повідомляє компілятору, що це кінець команди. Трохи пізніше ви побачите, що крапка з комою використовується для завершення більшості команд в C++ .

Наступний важливий рядок програми `int main().` Цей рядок повідомляє компілятору, що є функція з ім'ям `main`, і що функція повертає ціле число типу `int`. Фігурні дужки `{` і `}` сигналізують про початок `{` і наприкінці `}` функції. Фігурні дужки використовуються і в інших блоках коду, але позначають завжди одне – початок і кінець блоку, відповідно.

У C++ об'єкт `cout` використовується для відображення тексту (вимовляється як “Сі аут”). Він використовує символи `<<`, відомі як “оператор зсуву”, щоб вказати, що відправляється до виводу на екран. Результатом виклику функції `cout<<` є відображення тексту на екрані. Послідовність `\n` фактично розглядається як єдиний символ, який позначає новий рядок (ми поговоримо про це пізніше більш детально). Символ `\n` переміщує курсор на екрані на наступний

рядок. Знову ж, зверніть увагу на крапку з комою, її додають у кінець, після кожного оператора C++ .

Наступна команда `cin.get().` Це ще один виклик функції, яка зчитує дані з вхідного потоку даних і очікує натискання клавіші ENTER. Ця команда зберігає консольне вікно від закриття, до тих пір поки не буде натиснута клавіша ENTER. Це дає вам час для того, щоб подивитися результат виконання програми.

Після досягнення кінця головної функції (закриває фігурна дужка), наша програма поверне значення 0 до операційної системи. Це повернуте значення є важливим, оскільки, проаналізувавши його, ОС може судити про те, успішно завершилася наша програма чи ні. Значення, що повертається 0 означає успіх і повертається автоматично (але тільки для типу даних `int`, інші функції, вимагають вручну повертати значення), але якби ми хотіли повернути щось інше, наприклад 1, ми повинні були б зробити це вручну.

```
int main()
{
 setlocale(0, "Ukr");
    cout << "Моя перша програма на C ++\n";
 cin.get();
 return 1;
}
```

Для закріплення матеріалу, наберіть код програми у своїй IDE і запустіть його. Після того, як програма запустилася, і ви побачили результат роботи, поекспериментуйте трохи з оператором `cout`. Це допоможе вам звикнути до мови.

# 2 БАЗОВІ ПОНЯТТЯ МОВИ C++

## 2.1 Оголошення змінних в C++

Для оголошення змінної використовується синтаксис тип <имя>;. Ось деякі приклади оголошення змінних:

```
int num;
char character;
```

```
float num_float;
```

Можливе оголошення декількох змінних одного і того ж типу в одному рядку, для цього кожна з них повинна бути відокремлена комою.

```
int x, y, z, d;
```

Якщо ви дивилися уважно, ви, можливо, бачили, що оголошення змінної завжди закінчується крапкою з комою.

### 2.2 Використання змінних

Отже, тепер ви знаєте, як оголосити змінну. Ось приклад програми, що демонструє використання змінної:

```
int main()
{
 setlocale(0, "Ukr");
 int number;
 cout << "Введіть число: ";
 cin >> number;
 cin.ignore();
 cout << "Ви ввели: " << number << "\n";

 cin.get();
}
```

Давайте розглянемо цю програму і вивчимо її код, рядок за рядком. Ключове слово int говорить про те, що `number` – ціле число. Функція `cin>>` зчитує значення в number, користувач повинен натиснути ENTER після введеного числа. `cin.ignore()` – функція, яка зчитує символ і ігнорує його. Ми організували своє введення в програму, після введення числа, ми натискаємо клавішу ENTER, символ який також передається в потік введення. Нам це не потрібно, тому ми його відкидаємо. Майте на увазі, що змінна була оголошена цілого типу, якщо користувач спробує ввести десяткове число, то воно буде обрізане (тобто десяткова частина числа буде ігноруватися). Спробуйте ввести десяткове число або послідовність символів, коли ви запустите приклад програми, відповідь буде залежати від вхідного значення.

### 2.3 Зміна і порівняння величин

Звичайно, незалежно від того, який тип даних ви використовуєте, змінні не представляють особливого інтересу без можливості зміни їх значення. Далі показані деякі оператори, використовувані спільно зі змінними:

* множення,

- віднімання,

+ додавання,

/ ділення,

= присвоєння,

== рівність,

> більше,

< менше.

!= нерівність

>= більше або дорівнює

<= менше або дорівнює

Ось кілька прикладів:

```
a = 4 * 6; // використання  коментаря і крапки з комою, а дорівнює 24
a = a + 5; // дорівнює сумі початкового значення і п'яти
a == 5     // виконується перевірка, а так само 5 чи ні
```

### 2.4 Типи даних

Кожна змінна має певний тип. І цей тип визначає, які значення може мати змінна, які операції з нею можна робити і скільки байт в пам'яті вона буде займати. У мові С ++ визначені такі базові типи даних:

У таблиці 2.1 описані типи даних С++ :

Таблиця 2.1 – Опис типів даних

| ***Тип*** | ***Назва типу*** | ***Об'єм ОП*** | ***Діапазон значень*** |
|---|---|---|---|
| *char* | Цілий | 8 бітів | -128 … +127 |
| *signed char* | Цілий із знаком | 8 бітів | -128 … +127 |
| *unsigned char* | Беззнаковий цілий | 8 бітів | 0……255 |
| *short* | Цілий короткий | 16 бітів | -32768 …+32767 |

| *Тип* | *Назва типу* | *Об'єм ОП* | *Діапазон значень* |
|---|---|---|---|
| *unsigned short* | Беззнаковий цілий короткий | 16 бітів | 0……65535 |
| *int* | Цілий | 32 біта | -2147483648 … +2147483647 |
| *unsigned int* | Беззнаковий цілий | 32 біта | 0……+429496729 |
| *long* | Цілий довгий | 32 біта | -2147483648 +2147483647 |
| *unsigned long* | Цілий довгий беззнаковий | 32 біта | 0……+429496729 |
| *float* | З плаваючої комою | 32 біта | $-3.4 \cdot 10^{38}$ … $3.4 \cdot 10^{38}$ |
| *double* | З плаваючої комою довгий | 64 біта | $-1.7 \cdot 10^{308}$ … $1.7 \cdot 10^{308}$ |
| *long double* | Довгий із плаваючої комою | 80 бітів | $-3.4 \cdot 10^{4932}$ … $+3.4 \cdot 10^{4932}$ |

Якщо при оголошенні змінної використати ключове слово const, то одержуємо тип, що має ті ж властивості, що й вихідний, за винятком того, що значення змінної типу const не може змінюватися після ініціалізації. Константа повинна ініціалізуватися при оголошенні. Для змінних ініціалізація необов'язкова, але настійно рекомендується.

### 2.5 Приведення типів при обчисленні виразів

При виконанні операцій виконується автоматичне перетворення типів, щоб привести операнди виразів до загального типу або щоб розширити короткі величини до розміру цілих величин, що використовуються у машинних командах. Виконання приведення залежить від специфіки операції і від типу операнда або операндів.

Основні арифметичні перетворення:

1) Операнди типу *float* приводяться до типу *double*.
2) Якщо один операнд *long double*, то другий приводиться до цього ж типу.
3) Якщо один операнд *double*, то другий також приводиться до типу *double*.
4) Будь-які операнди типу *char* і *short* приводяться до типу *int*.
5) Будь-які операнди *unsigned char* або *unsigned short* приводяться до типу *unsigned int*.

6) Якщо один операнд типу *unsigned long*, то другий приводиться до типу *unsigned long*.
7) Якщо один операнд типу *long*, то другий приводиться до типу long.
8) Якщо один операнд типу *unsigned int*, то другий операнд приводиться до цього ж типу.

Таким чином, при обчисленні виразів операнди приводяться до типу того операнда, що має найбільший розмір.

### 2.6 Операції

Операції - це комбінації символів, що визначають дії по перетворенню значень. У мові C++ визначені наступні операції (табл. 2.2):

Таблиця 2.2 - Операції мови *C++*

| **Опе-рація** | **Найменування** | **Приклад** | **Особливості** |
|---|---|---|---|
| **Арифметичні операції** | | | |
| + | Додавання | a + b | |
| − | Віднімання | a − b | |
| * | Множення | a * b | |
| / | Ділення | a / b | Результатом ділення цілих чисел буде ціле число (дробова частина відкидається), результат ділення дійсних чисел – дійсне число |
| % | Остача при діленні цілих чисел | a % b | Знак остачі збігається із знаком дільника |
| − | Унарний мінус | -a | Зміна знака числа |
| **Логічні операції** | | | |
| && | І (логічне множення) | a && b | Результат істина, якщо всі операнди істині |
| ‖ | АБО (логічне додавання) | a ‖ b | Результат істина, якщо хоча б один з операторів - істина |
| ! | НЕ (заперечення) | !a | Результат протилежний значенню операнда |
| **Операції відношення (порівняння)** | | | |
| == | Дорівнює | a == b | |

| Опе-рація | Найменування | Приклад | Особливості |
|---|---|---|---|
| != | Не дорівнює | a != b | |
| > | Більше | a > b | |
| < | Менше | a < b | |
| >= | Не менше | a >= b | |
| <= | Не більше | a <= b | |
| **Бітові операції** | | | |
| & | І (кон'юнкція) | 3 & 2=2 | 1&1=1, 0&1=0, 1&0=0, 0&0=0 |
| \| | АБО (диз'юнкція) | 3 \| 2=3 | 1\|1=1, 0\|1=1, 1\|0=1, 0\|0=0 |
| ^ | виключаюче АБО | 3 ^ 2=1 | 1^1=0, 0^1=1, 1^0=1, 0^0=0 |
| ~ | Доповнення (заперечення бітів) | ~6 = -7 | ~6=~00000110=11111001=-7 |
| **Операції зсуву** | | | |
| << | Зсув вліво на задане число бітів | a << b | Зсув бітів ліворуч у числі a на b позицій. Результат буде того ж типу, що і a. b не може бути від'ємним або більшим, ніж a. |
| >> | Зсув вправо на задане число бітів | a >> b | Зсув бітів праворуч у числі a на b позицій. Результат буде того ж типу, що і a. b не може бути від'ємним або більшим, ніж a. |
| **Операції присвоєння** | | | |
| = | Присвоєння | a = b | Значення b заноситься в комірку оперативної пам'яті з ім'ям a. Приведення даних до одного типу відбувається автоматично |
| += | Присвоєння суми чисел | a += b | a=a+b. Короткі числа перетворюються у довгі із збереженням значення, довгі у короткі – із втратою старших бітів. Дійсні числа перетворюються в цілі з відкиданням дробової частини. |

| Опе-рація | Найменування | Приклад | Особливості |
|---|---|---|---|
| %= | Присвоєння остачі ділення | a %= b | a=a%b. Остача визначається тільки при діленні цілих чисел. |
| −= | Присвоєння різниці чисел | a -= b | a = a - b |
| *= | Присвоєння добутку чисел | a *= b | a = a * b |
| /= | Присвоєння частки від ділення | a /= b | a = a / b |
| >>= | Присвоєння із зсувом вправо | a >>= b | a = a >> b |
| <<= | Присвоєння із зсувом вліво | a <<= b | a = a << b |
| &= | Присвоєння бітової кон’юнкції | a &= b | a = a & b |
| \|= | Присвоєння бітової диз'юнкції | a \|= b | a = a \| b |
| ^= | Присвоєння значення операції «виключаюче АБО» | a ^= b | a = a ^ b |
| **Унарні операції** | | | |
| ++, −− | Префіксне додавання, віднімання | ++a або a=a+1 −−a | Спочатку збільшується (зменшується) значення змінної на 1, потім воно використовується |
| ++, −− | Постфіксний додавання, віднімання | a++, a−− | Спочатку використовується значення змінної, потім воно збільшується на 1 |
| **Тернерна операція** | | | |
| ?: | Умовна операція | <операнд1>? <операнд2>: <операнд3> | Якщо значення <операнд1> істине, то обчислюється <операнд2>, інакше – <операнд3> |

## 2.7 Пріоритети операцій і порядок обчислень

У мові C++ операції з вищими пріоритетами обчислюються першими. Найвищим пріоритетом є пріоритет, рівний 1. Пріоритети і порядок операцій наведені в табл. 2.3:

Таблиця 2.3 - Пріоритети операцій

| Пріори-тет | Знак операції | Типи операції | Порядок виконання |
|---|---|---|---|
| 1 | () [] . -> | Вираз | Ліворуч праворуч |
| 2 | + - ~ ! * & ++ -- | Унарні | Праворуч ліворуч |
| 3 | * / % | Мультиплікативні | Ліворуч праворуч |
| 4 | + - | Адитивні | |
| 5 | << >> | Зсуву | |
| 6 | < > <= >= | Відношення | |
| 7 | == != | Відношення (рівність) | |
| 8 | & | Порозрядне І | |
| 9 | ^ | Порозрядне виключає АБО | |
| 10 | \| | Порозрядне АБО | |
| 11 | && | Логічне І | |
| 12 | \|\| | Логічне АБО | |
| 13 | ?: | Умовна | |
| 14 | = *= /= %= += -= &= \|= >>= <<= ^= | Просте і складене присвоювання | Праворуч ліворуч |
| 15 | , | Послідовне обчислення | Ліворуч праворуч |

**Завдання 1:**

Невдаха-учень Сашко сів виконувати домашнє завдання і просидів за столом 2 години. З них X хвилин він чухав потилицю і дивився у вікно, Y хвилин шукав у письмовому столі гумку, щоб стерти у підручнику з англійської мови карикатуру на свого товариша, на малювання якої він витратив перед цим Z хвилин. Весь останній час Сашко перекладав англійські слова. Напишіть програму, яка визначає, та виводить на екран, скільки слів він встиг перекласти, якщо на переклад одного слова у нього йшло 5 хвилин?

**Завдання 2:**

Напишіть програму, яка по даті визначає день тижня, на який ця дата доводиться. Для обчислення дня тижня скористайтеся формулою:

$$(d+\left[\frac{13m-1}{5}\right]+Y+\left[\frac{Y}{4}\right]+\left[\frac{c}{4}\right]-2c+777)\%7$$

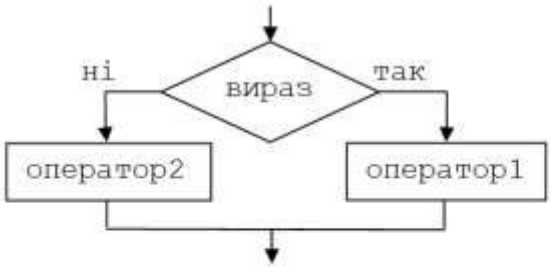
Тут d – число місяця, m – номер місяця, якщо починати рахунок з березня, як це робили у Стародавньому Римі (березень – 1, квітень – 2 , лютий – 12), Y – номер року в сторіччі, c– кількість сторіч, що минуло до початку нинішнього. Квадратні дужки означають, що потрібно узяти цілу частину від значення, що знаходиться в дужках. Обчислене значення визначає день тижня: 1 – понеділок, 2 – вівторок . . . 6 – субота, 0 – неділя.

# 3 УМОВНІ КОНСТРУКЦІЇ

## 3.1 Умовний оператор

Умовний оператор має наступний формат:

```
if (<вираз>) <оператор1>;
             [else <оператор2>;]
```




a) Алгоритмічна конструкція

```
if(вираз)
   оператор1;
else
   оператор2;
```


b) Алгоритмічна конструкція

```
if(вираз)
   оператор1;
```

Виконання оператора `if` починається з обчислення <виразу>. Далі виконання здійснюється за наступною схемою:

якщо <вираз> істинний (відмінний від 0), то виконується <оператор1>;

якщо <вираз> хибний (дорівнює 0), то виконується <оператор2>;

якщо <вираз> хибний і відсутній <оператор2> (у квадратні дужки вкладена необов'язкова конструкція), то виконується наступний за if оператор.

Після виконання оператора if значення передається на наступний оператор програми, якщо послідовність виконання операторів програми не буде примусово порушена використанням операторів переходу.

Наприклад,

```
if (i<j) i++;
    else {j=i-3; i++;}
```

Цей приклад ілюструє також і той факт, що на місці <оператор1>, так само як і на місці <оператор2> можуть знаходитися складні конструкції.

Допускається використання вкладених операторів if: оператор if може бути включений у конструкцію if або в конструкцію else іншого оператора if. Щоб зробити програму більш читабельною, рекомендується групувати оператори і конструкції у вкладених операторах if, використовуючи фігурні дужки. Якщо ж фігурні дужки опущені, то компілятор зв'язує кожне ключове слово else з найближчим до нього оператором if, для якого немає else. Наприклад,

```
char ZNAC;
    int x,y,z;
          …
     if (ZNAC = = '-') x = y - z;
     else  if (ZNAC = = '+') x = y + z;
           else if (ZNAC == '*') x = y * z;
                else if (ZNAC == '/') x = y / z;
                     else ...
```

### 3.2 Оператор перемикач (switch)

Іноді при глибіні вкладеності умовних операторів понад три, така конструкція робить програму складною для сприйняття людиною. Більш зручним у такій ситуації виявляється застосування оператора switch

Загальна структура оператора така:

```
switch(вираз)
{
case константний вираз 1:  оператори 1;
        break;
case константний вираз 2:  оператори 2;
        break;
.......
case константний вираз n:  оператори n;
        break;
default: оператори;
}
```

Виконання оператора починається з обчислення виразу (він повинен бути цілочисельним . Якщо значення виразу співпало з константним виразом блоку `case`, виконуються оператори, починаючи з цього блоку case і до першого оператора `break`. Якщо оператор break відсутній, то виконуватимуться оператори наступного блоку `case`, або `default`. Якщо значення виразу не дорівнює жодному константному виразу блока case, виконуються оператори блоку `default`.

### 3.3 Приклади розв'язання завдань

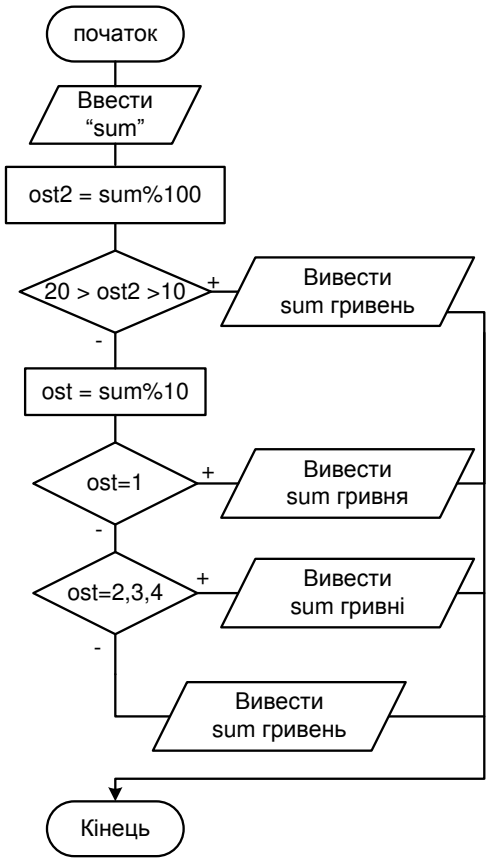
**Завдання**: Написати програму, яка після введеного з клавіатури числа, що позначає грошову одиницю, дописує слово "гривня" в правильній формі. Наприклад, 0 гривень, 11 гривень, 123 гривни і так далі.

Один варіант програми вирішити за допомогою оператору `if`, інший – за допомогою `switch`.

**Алгоритм:**

1) Вводимо число “sum”.
2) Визначаємо остачу від ділення числа “sum” на 100, записуючи його у змінну ost2 (діапазон значень від 10 до 19 буде виводити на екран “гривень”)
3) Якщо 10< ost2<19 виводимо на екран “гривень’.
4) Визначаємо остачу введеного числа, ділячи його на 10, та записуючи результат в змінну ost.
5) За допомогою операції switch визначаємо за остачею (змінною ost) закінчення слова:
6) якщо ost дорівнює 1 – виводимо на екран “гривня”;
7) інакше якщо ost =2,3,4 – виводимо “гривні”;
8) інакше – виводимо “гривень”.

Нижче представлена блок-схема цього алгоритму



Текст програми з використанням оператора if :

```
#include <iostream>
#include <conio.h>
using namespace std;

int main()
{
  setlocale(0, "Russian");
  int sum;
  cin>>sum;
  int ost=sum % 10;
  int ost2=sum % 100;

 if(ost2 > 10 && ost2 < 20)
    cout<<sum<<" гривень";
 else
  if(ost==1)
     cout<<sum<<" гривня";
       else if(ost>=1 && ost<=4)
     cout<<sum<<" гривні";
       else
     cout<<sum<<" гривень";

 _getch();
return 0;
}
```

Текст програми з використанням оператора switch :

```
#include <iostream>
#include <conio.h>
using namespace std;

int main()
{
  setlocale(0, "Ukr");
  int sum;
  cin>>sum;
  int ost=sum % 10;
  int ost2=sum % 100;

 if(ost2 > 10 && ost2 < 20)
    cout<<sum<<" гривень";
 else
  switch(ost)
    {
    case 1: cout<<sum<<" гривня";
        break;
    case 2:
```

```
    case 3:
    case 4: cout<<sum<<" гривни";
        break;
    default: cout<<sum<<" гривень";
     }
 getch();
return 0;
}
```

**Завдання:**

1. Написати програму, яка запрошує у користувача номер дня тижня і виводить одне з повідомлень: "Робочий день", "Субота" АБО "Неділя".

2. Написати програму, яка запрошує у користувача номер місяця і виводить відповідну назву пори року. У випадку, якщо користувач введе недопустиме число, програма повинна вивести повідомлення "Помилка введення даних".

3. Написати програму обчислення площі трикутника за формулою Герона. Сторони *a, b c* вводить користувач. Якщо трикутник зі сторонами *a, b c* не існує, вивести відповідне повідомлення.

$$S = \sqrt{p(p-a)(p-b)(p-c)}, p = \frac{(a+b+c)}{2}$$

4. Написати програму, яка обчислює вік користувача, який вводить свій рік народження і видає рекомендацію що до часу роботи за комп'ютером: якщо вік менше 7, сидіти за комп'ютером не більше 0,5 години, якщо більше 7 і менше 15, то не більше 1 години, в решті випадків – не більше 4 годин в день.

5. Змінить програму **завдання 2 розділу 2** так, щоб можна було вводити дату у більш зручному вигляді: наприклад, щоб ввести дату 9 лютого 2019 року організувати такий діалог:

```
Введіть число    9
Введіть місяць   2
Введіть рік       2019
Субота
```

6. Написати програму, яка обчислює оптимальну вагу для користувача, порівнює її із реальною і видає рекомендацію про необхідність або набрати вагу або схуднути, або залишити все як є, якщо вага оптимальна. Оптимальна вага обчислюється за формулою: `Оптимальна вага = Зріст (см) – 100.`

7. У царівни Несмеяни кругле обличчя, радіус якого R. Царівна має квадратне дзеркало зі стороною A. Коли Несмеяна милується собою, її відображення поміщається у дзеркалі чи ні? Написати програму, яка визначає це.

8. Для кожної з двох подій, часовий проміжок між якими не перевищує 24 години, заданий час, коли вони відбулися у вигляді трьох цілих чисел: година, хвилина та секунда. Визначте, яка подія відбулась раніше?

9. Відомі поточна дата та дата народження користувача програми у вигляді трьох цілих чисел: день, місяць та рік. Визначити повну кількість років користувача.

10. Дані два конверта прямокутної форми з довжинами сторін (a,b) та (c,d). Визначити, чи можна один з конвертів вкласти в інший. Відповіддю повинно бути відповідно, «YES» чи «NO».

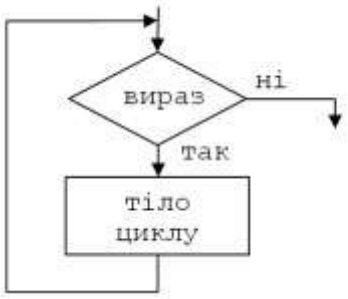
# 4 ЦИКЛИ

Для виконання деяких дій багато разів в залежності від певної умови використовуються цикли. Цикл (оператор ітерації) - це оператор, який призначений для організації багаторазового виконання набору інструкцій. Також циклом може називатися будь-яка послідовність інструкцій, яка багаторазово виконується.

## 4.1 Цикл з передумовою

Цикл, який виконується поки вірна (істинна) деяка умова, яка визначена перед початком циклу. Ця умова перевіряється до виконання тіла циклу, тому

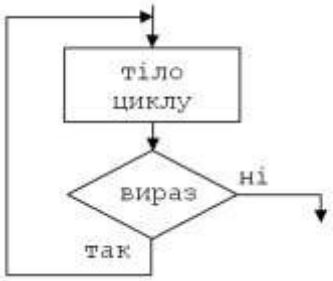
тіло може не виконуватись жодного разу (якщо умова з самого початку хибна). В мові С++ реалізується оператором `while`.




```
a=0;
while(a<10)
{
  cout<<a++;
 }
```

### 4.2 Цикл з постумовою

Цикл, у якому умова перевіряється після виконання тіла циклу. Звідси випливає, що тіло завжди виконується хоча б один раз. Реалізується оператором `do-while`.




```
a=0;
do
{
  cout<<a++;
}while(a<10);
```

Оператори `while, do-while` в більшості випадків використовуються, коли невідомо наперед, скільки разів цикл буде виконуватись і перевіряють необхідність завершення при кожній ітерації.

### 4.3 Цикл з лічильником

Цикл, у якому деяка змінна змінює своє значення від заданого початкового значення до кінцевого значення з деяким кроком, і для кожного значення цієї змінної тіло циклу виконується один раз. У більшості процедурних мов програмування реалізується оператором `for`, в якому вказується: лічильник (змінна циклу), необхідна кількість проходів і крок, з яким змінюється лічильник.

Оператор `for` може містити декілька лічильників, від яких може залежати умова чи зміна лічильника:

```
for (int i=0, int j=0; i+j<100; i++, j++)
{…}
```

У циклі може існувати декілька умов виходу. При хибності будь-якої із заданих умов цикл завершується:

```
for(int a=0, b=5; a<10 && b>2; a++, b--)
{…}
```

Умова виходу із циклу може залежати не тільки від змінної-лічильника, але будь-яких інших факторів (закінчення строк у потоці, тощо):

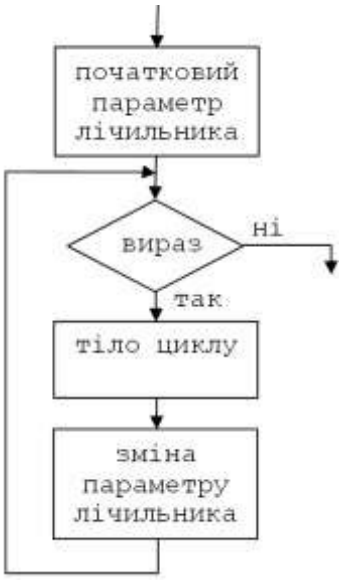
```
for (int i=0; (text=serlz.ReadLine())!=null; i++)
{…}
```

Якщо немає необхідності задавати якусь із частин оператору for її можна пропустити:

```
for(; i< node.Length; i++)
{…}
```

Якщо немає необхідності виконувати якісь операції в тілі циклу, то можна створити пустий оператор `for`:

```
for(int i=0; text[i]!='\0'; i++);
```




**Оператор переривання**. Оператор `break` забезпечує припинення виконання самого внутрішнього із об'єднуючих його операторів `break switch`,

`do`, `for`, `while`. Після виконання оператора break керування передається операторові, що іде за перерваним.

**Оператор продовження** `continue`.

Оператор `continue`, як і оператор `break`, використовується тільки всередині операторів циклу, але на відміну від останнього, виконання програми продовжується не з оператора, що іде за перерваним оператором, а з початку перерваного оператора.

Наприклад,

```
int main()
  { int a,b;
    for (a=1,b=0; a<100; b+=a,a++)
     { if (b % 2) continue;
            ...    // обробка парних сум
     }
    return 0;
  }
```

Коли сума чисел від 1 до a стає непарною, оператор `continue` передає керування на чергову ітерацію циклу `for`, не виконуючи оператори обробки парних сум.

Оператор `continue`, як і оператор `break`, перериває самий внутрішній із об'єднуючих його циклів.

### 4.4 Приклади розв’язання завдань

**Завдання 1**: Користувач вводить число n. Визначити n-те число Фібоначчі. Числа Фібоначчі вираховуються наступним чином: перші два значення дорівнюють 0 та 1, а кожне наступне значення – це сума двох попередніх.

**Алгоритм:**

1. Вводиться значення n.

2. Змінна f0 для першого числа і ініціалізується значенням 0.
3. Змінна f1 для другого числа ініціалізується значенням 1 .
4. Змінна next для наступного числа Фібоначчі, починаючи з другого.
5. Якщо значення n дорівнює 1 (перше число Фібоначчі), змінна next = f0.
6. Якщо значення n дорівнює 2 (друге число Фібоначчі), змінна next = f1, інакше виконуються наступні дії:
7. змінна і ініціалізується значенням 3.
8. До тих пір, поки і менше, або дорівнює n, виконуються наступні дії:

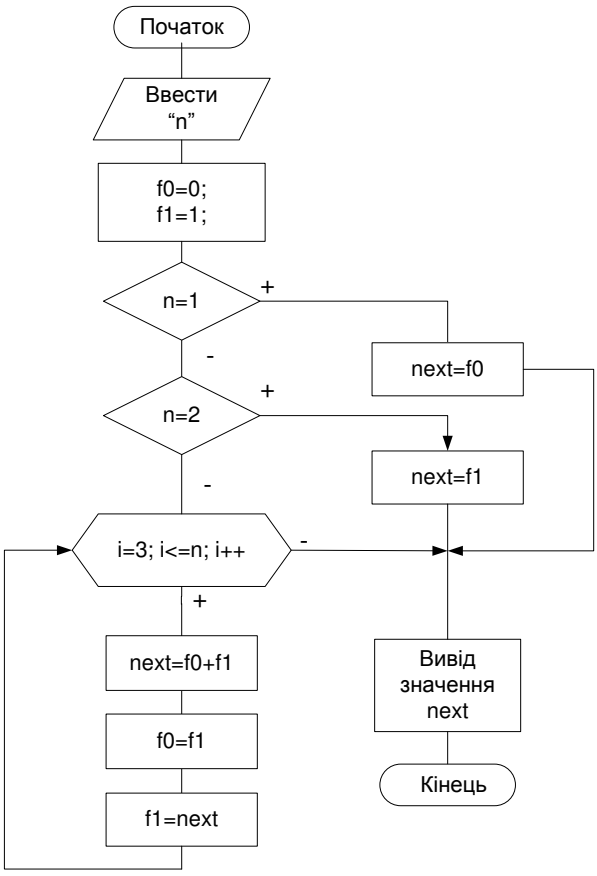
   next=f0 + f1;

   f0=f1;

   f1=next;

   i=i+1;
9. Виводиться значення змінної next, яка містить n-те число Фібоначчі.

Нижче представлена блок-схема цього алгоритму:



Текст програми:

```
#include <iostream>
#include <conio.h>
using namespace std;
int main()
{
  setlocale(0, "Ukr");
  int n, i, f0=0, f1=1, next=0;
  cout<<"Введить n=";
  cin>>n;
  if(n==1)
     next=f0;
  if(n==2)
     next=f1;
  for(i=3; i<=n; i++)
     {   next=f0 + f1;
         f0 = f1;
         f1 = next;
     }
  cout<<next;
_getch();
}
```

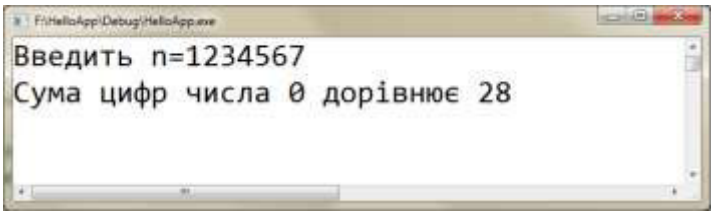
**Завдання 2**: Знайти суму цифр у числі n.

**Алгоритм:**

1. Вводиться значення n.
2. Початкове значення суми sum=0;
3. Отримуємо останню цифру числа n як остачу от ділення на 10 і додаємо її до суми.
4. Відкідаємо останню цифру числа n шляхом ділення цього числа на 10
5. Повторюємо пункт 3 і 4 до тих пір, поки n більше 0.

Текст програми:

```
int main()
{
 setlocale(0, "Ukr");
 int n, sum = 0;
 cout << "Введить n=";
 cin >> n; // ввід цілого числа n
 while (n > 0) // до тих пір, поки n більше 0
 {
// останню цифру числа як остачу от ділення на 10  додаємо до суми.
 sum += n % 10;
// відкідаємо останню цифру числа n шляхом ділення цього числа на 10
 n /= 10;
}
 cout << "Сума цифр числа " << n << " дорівнює " << sum;
 _getch();
}
```




**Завдання 3:** Створити програму, що вираховує для заданого x суму: $1+\frac{x}{1!}+\frac{x^2}{2!}+\frac{x^3}{3!}+\ldots$ (кількість членів задається користувачем)

**Алгоритм:**

1. Вводяться два числа: x та n.
2. Змінна sum ( сума ряду) приймається рівною  нулю (sum = 0).
3. Змінна i приймається рівною одиниці (i = 1).
4. Змінні powx (ступень x) та fact (факторіал)  приймаються рівними одиниці (powx=1, fact=1).
5. До тих пір, поки i менше, або дорівнює n, виконуються наступні дії:

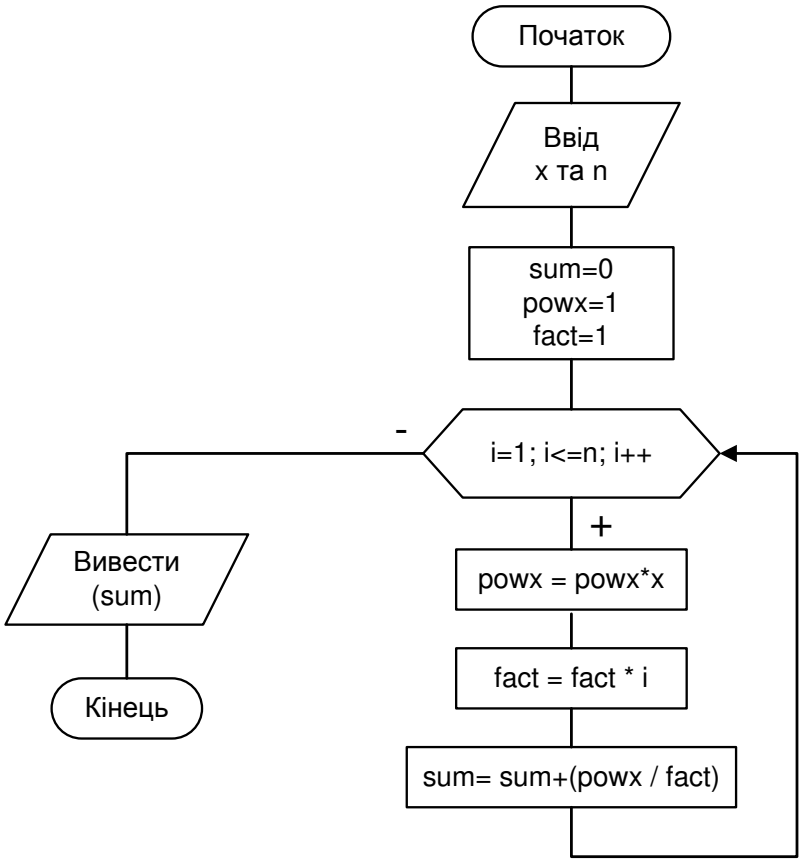
sum = sum + powx / fact;

powx = powx * x;

fact = fact * i;

6. Виводиться значення змінної sum.

Нижче представлена блок-схема цього алгоритму:




Текст програми:

```
#include <iostream>
#include <conio.h>
using namespace std;
int main()
{
  setlocale(0, "Ukr");
    float x, n, sum=0, powx=1, fact=1;
     cout<<"Введіть число x =";
     cin>>x;
     cout<<"Введіть число n =";
```

```
    cin>>n;
    for (int i=1; i<=n; i++)
    {
    sum += powx/fact;
        powx *= x;
        fact *= i;
    }
    cout<<"\nСума ряду="<<sum;
  _getch();
}
```

**Завдання:**

1. Коли Василині Премудрій виповнилося 18 років, Чахлик Невмерущий вирішив взяти її заміж. Василина запитала Чахлика, скільки у нього скринь із золотом. Чахлик сказав, що в нього зараз n скринь і кожний рік додається ще по m скринь. Василина пообіцяла, що вийде заміж тоді, коли у Чахлика буде k повних скринь із золотом. Скільки років буде у цей час нареченій?

2. Папуга навчився висмикувати у дідуся Василя волосся, яке ще залишилось у того на голові. Почавши з однієї волосини, він кожен день збільшував порцію удвічі. Через скільки днів дідусеві не знадобиться гребінець, якщо на початку у нього на голові було аж N волосин?

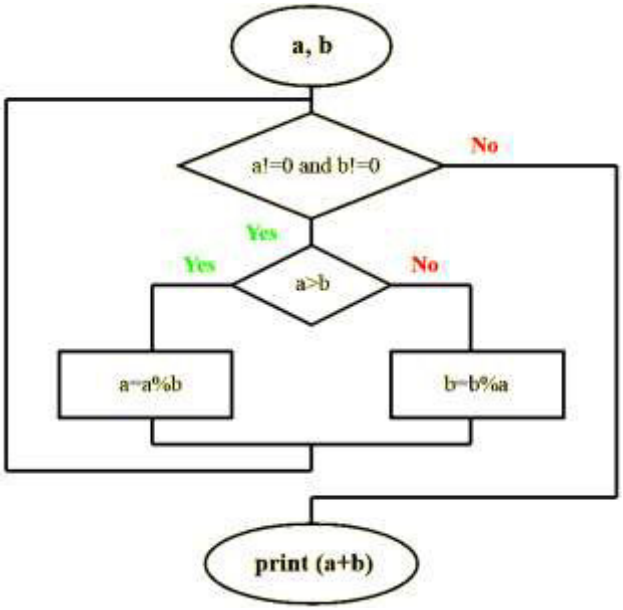
3. На дверях ліфта висіло загрозливе попередження про те, що двері зачиняються самі в той самий момент, коли зайвий пасажир переступить поріг ліфта. Який за рахунком пасажир постраждає, якщо ліфт витримує вагу не більше S кг, а вага пасажирів, що стоять у черзі до ліфта, дорівнює відповідно, a1, a2, a3, … , an ?

4. Курочка Ряба знесла яєчко, а мишка розбила його. Після цього Ряба знесла на K яєчок більше, але мишка знову їх розбила. Ряба піднатужилася і знесла знову на K яєчок більше, ніж в попередній раз, але безсовісна мишка розтрощила і ці яйця. Так продовжувалося N разів, поки Ряба не здалася. Із

скількох яєць Дід і Баба змогли б кінеці-кінців зробити собі яєчню?

5. В понеділок Васько позичив у Миколки 2 цукерки і з задоволенням їх з'їв. У вівторок позичив у два рази більше цукерок, після чого віддав половину боргу, а решту цукерок знову із задоволенням з'їв. Кожний наступний день він позичав у два рази більше цукерок, ніж у попередній день, віддавши з них цілу частину від половини боргу, а решту цукерок із задоволенням з'їдав. Скільки цукерок з'їсть Васько через N тижнів? Скільки у нього при цьому буде складати борг?

6. Створити програму, що обчислює найбільший спільний дільник двох натуральних чисел за алгоритмом Евкліда.



# 5 МАСИВИ

## 5.1 Одновимірні масиви

Масив представляє набір однотипних даних. Визначення масиву виглядає наступним чином:

тип_змінної назва_масиву [довжина_масиву]

Після типу змінної йде назва масиву, а потім в квадратних дужках його розмір. Наприклад, визначимо масив з 4 чисел:

```
int numbers[4];
```

Даний масив має чотири числа, але всі ці числа мають невизначене значення. Однак ми можемо виконати ініціалізацію і привласнити цим числам деякі початкові значення через фігурні дужки:

```
int numbers[4] = { 1,2,3,4 };
```

Значення в фігурних дужках ще називають ініціалізатор. Якщо ініціалізаторів менше, ніж елементів в масиві, то ініціалізатор використовуються для перших елементів, а інші заповнюються нулями. Якщо список ініціалізаторів більше, ніж елементів в масиві, то при компіляції виникне помилка.

Якщо розмір масиву не вказано явно, то він виводиться з кількості ініціалізаторів:

```
int numbers[] = { 1,2,3,4,6,9 };
```

Свої особливості має ініціалізація символьних масивів. Ми можемо передати символьному масиву як набір ініціалізаторів, так і рядок:

```
char s1[] = { 'h', 'e', 'l', 'l', 'o' };
char s2[] = "world";
```

Причому в другому випадку масив s2 матиме не 5 елементів, а 6, оскільки при ініціалізації рядком в символьний масив автоматично додається нульовий символ '\ 0'.

При цьому не допускається присвоєння одному масиву іншого масиву:

```
int nums1[] = { 1,2,3,4,5 };
int nums2[] = nums1; // помилка
nums2 = nums1; // помилка
```

Після визначення масиву ми можемо звернутися до його окремих елементах за індексом. Індекси починаються з нуля, тому для звернення до першого елементу необхідно використовувати індекс 0. Звернувшись до елементу за індексом, ми можемо отримати його значення, або змінити його:

```
int n, sum = 0;
int numbers[4] = { 1,2,3,4 };
int first_number = numbers[0];
cout << first_number << std::endl; // 1
numbers[0] = 34; // змінюємо елемент
cout << numbers[0] << ; // 34
```

Число елементів масиву також можна визначати через константу:

```
const int N = 4;
int numbers[N] = { 1,2,3,4 };
```

Використовуючи цикли, можна пробігтися по всьому масиву і через індекси звернутися до її елементів:

```
int numbers[4] = { 1,2,3,4 };
int size = sizeof(numbers) / sizeof(numbers[0]);
for (int i = 0; i < size; i++)
      cout << numbers[i] << " ";
```

Щоб пройтися по масиву в циклі, спочатку треба знайти довжину масиву. Для знаходження довжини застосовується оператор sizeof. По суті довжина масиву дорівнює сукупної довжині його елементів. Всі елементи представляють один і той же тип і займають один і той же розмір в пам'яті. Тому за допомогою виразу sizeof (numbers) знаходимо довжину всього масиву в байтах, а за допомогою виразу sizeof (numbers [0]) - довжину одного елемента в байтах. Розділивши два значення, можна отримати кількість елементів в масиві. А далі

за допомогою циклу for перебираємо всі елементи, поки лічильник і не стане рівним довжині масиву. У підсумку на консоль будуть виведені всі елементи масиву:


Але також є і ще одна форма циклу for, яка призначена спеціально для робота з колекціями, в тому числі з масивами. Ця форма має наступне формальне визначення:

```
for (тип змінна : колекція)
{
      інструкції;
}
```

Використовуємо цю форму для перебору масиву:

```
int numbers[4] = { 1,2,3,4 };
for (int number : numbers)
      cout << number << " ";
```

При переборі масиву кожен перебирати елемент буде міститися в змінну number, значення якої в циклі виводиться на консоль.

Якщо нам невідомий тип об'єктів в масиві, то ми можемо використовувати специфікатор auto для визначення типу:

```
int numbers[4] = { 1,2,3,4 };
for (auto number : numbers)
      cout << number << " ";
```

### 5.2 Багатовимірні масиви

Крім одновимірних масивів в C ++ є багатовимірні. Елементи таких масивів самі в свою чергу є масивами, в яких також елементи можуть бути масивами. Наприклад, визначимо двомірний масив чисел:

```
int numbers[3][2];
```

Такий масив складається з трьох елементів, при цьому кожен елемент являє масив з двох елементів. Ініціалізируем подібний масив:

```
int numbers[3][2] = { {1, 2}, {4, 5}, {7, 8} };
```

Масиву виділяється пам'ять, необхідна для розміщення всіх його елементів. Елементи масиву з першого до останнього розміщуються в послідовних комірках пам'яті (по зростанню адрес). Між елементами масиву в пам'яті розриви відсутні.

Для того, щоб створити двовимірний масив  цілих чисел 10 х 12 на ім’я A, потрібно написати:

```
int A[10][12];
```

Наприклад, наступна програма завантажує масив 4 х 5 добутком індексів цього масиву, а потім виводить масив на екран у форматі рядків та стовпчиків.

```
int twod[4][5];
int i, j;
for (i = 0; i < 4; i++)
      for (j = 0; j < 5; j++)
            twod[i][j] = i * j;
for (i = 0; i < 4; i++)
{
      for (j = 0; j < 5; j++)
            cout << twod[i][j] << " ";
      cout << endl;
}
```

Програма виведе на екран наступну таблицю:

### 5.3 Обробка елементів масиву

Обробка елементів масиву виконується в циклі, індекси елементів циклу є параметрами циклу. Для обробки багатовимірних масивів використовуються вкладені цикли.

**Приклади:**

1) ввід з клавіатури елементів масиву A[M][N]:

```
for (i = 0; i < N; i++)
   for (j = 0; j < M; j++) {
      cout << "A[" << i << "][" << j << "]=";
      cin >> A[i][j];
}
```

2) заповнення випадковими числами:

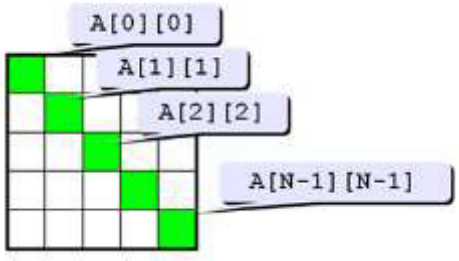
```
for (i = 0; i < N; i++)
  for (j = 0; j < M; j++)
       A[i][j] = rand() % 25 - 10
```

3) вивід на екран:

```
for (i = 0; i < N; i++)
   {
       for (j = 0; j < M; j++)
            cout << A[i][j] << "  ";
       cout << "\n";
   }
```

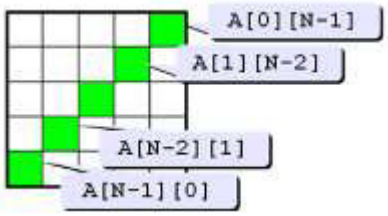
4) вивід на екран головної діагоналі квадратної матриці з N рядків і N стовпців:

```
for (i = 0; i < N; i++)
     cout << A[i][i] << "  ";
```



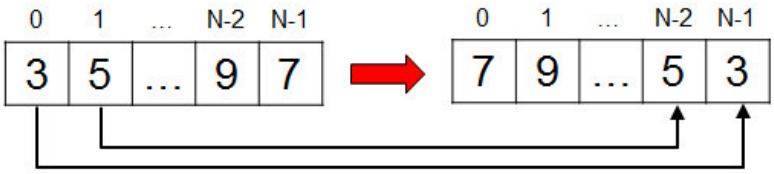
5) вивід на екран другу діагональ квадратної матриці з N рядків і N стовпців:

```
for (i = 0; i < N; i++)
      cout << A[i][N - 1 - i] << "  ";
```




**Реверс масиву**

Завдання: переставити елементи масиву в зворотному порядку (виконати інверсію).




**Алгоритм:**

поміняти місцями `A[0]` и `A[N-1]`, `A[1]` и `A[N-2]`, …

**Псевдокод:**

```
for (i = 0; i < N/2; i++) {
          // поміняти місцями A[i] и A[N-1-i]
```

**Програма:**

```
const int N = 10;
int A[N], i, c;
// заповнити масив
// вивести вихідний масив
for (i = 0; i < N / 2; i++) {
      c = A[i];
      A[i] = A[N - 1 - i];
      A[N - 1 - i] = c;
}
// вивести отриманий масив
```

**Циклічний зсув**

Завдання: зсунути елементи масиву вліво на 1 клітинку, перший елемент стає на місце останнього.


**Програма:**

```
const int N = 10;
int A[N], i, c;
// заповнити масив
// вивести вихідний масив
c = A[0];
for (i = 0; i < N - 1; i++)
     A[i] = A[i + 1];
A[N - 1] = c;
// вивести отриманий масив
```

**Сортування**

Сортування - це розстановка елементів масиву в заданому порядку (по зростанню, зменшенню, по останній цифрі, ...).

Завдання: переставити елементи масиву в порядку зростання.

Прості і зрозумілі, але неефективні для великих масивів алгоритми:

- метод бульбашки
- метод вибору

Складні, але ефективні алгоритми:

- «Швидке сортування» (Quick Sort)
- сортування «купою» (Heap Sort)
- сортування злиттям

- пірамідальна сортування

**Обмінне сортування (метод бульбашки)**

Алгоритм отримав свою назву від того, що процес сортування нагадує поведінку бульбашок повітря у резервуарі з водою. Оскільки для роботи з елементами масиву він використовує лише порівняння, це сортування на основі порівнянь.

Хоча, алгоритм є одним із найпростіших алгоритмів сортування, його ефективність є досить низькою, і він погано підходить для сортування великих масивів.

Код програми:

```
int main()
  {
    const int N = 8;
    int A[N] = { 8, 23, 5, 65, 44, 33, 1, 6 };
    int i, j, flag;
    do {
          flag = 0;   // скинути прапорець
          for (j = N - 2; j >= 0; j--)
                if (A[j] > A[j + 1]) {
                      int c = A[j];
                      A[j] = A[j + 1];
                      A[j + 1] = c;
                      flag = 1;   // встановити прапорець
                }
    } while (flag);  // вихід при flag = 0

    for (int i = 0; i < N; i++)
          cout << A[i] << ' ';
    _getch();
}
```

**Сортування вибором**

При сортуванні масиву a[0], a[1], …, a[n-1] методом простого вибору серед всіх елементів знаходиться елемент з якнайменшим значенням a[i], і a[0] a[i] обмінюються значеннями. Потім цей процес повторюється для елементів a[1], a[2], …, a[n-2] …

Код програми:

```
int main()
 {
  const int N = 8;
  int a[N] = { 8, 23, 5, 65, 44, 33, 1, 6 };;
  int i, j;
  for (i = 0; i < N - 1; i++) // з першого до передостаннього!
  {
    for (j = i + 1; j < N; j++) // з наступного після і-го до останнього
       if (a[j] < a[i])
          {
          int t = a[i]; // міняємо місцями
          a[i] = a[j];  // a[i] та a[j];
          a[j] = t;     // елементи
          }
  }
for (i = 0; i < N; i++)
               cout << a[i] << ' ';

          _getch();

     }
```

### 5.4 Приклади розв'язання завдань

**Завдання 1**: Згенерувати і вивести на екран масив з 20 випадкових чисел від 0 до 99. Знайти індекси мінімального і максимального елементів масиву.

**Алгоритм:**

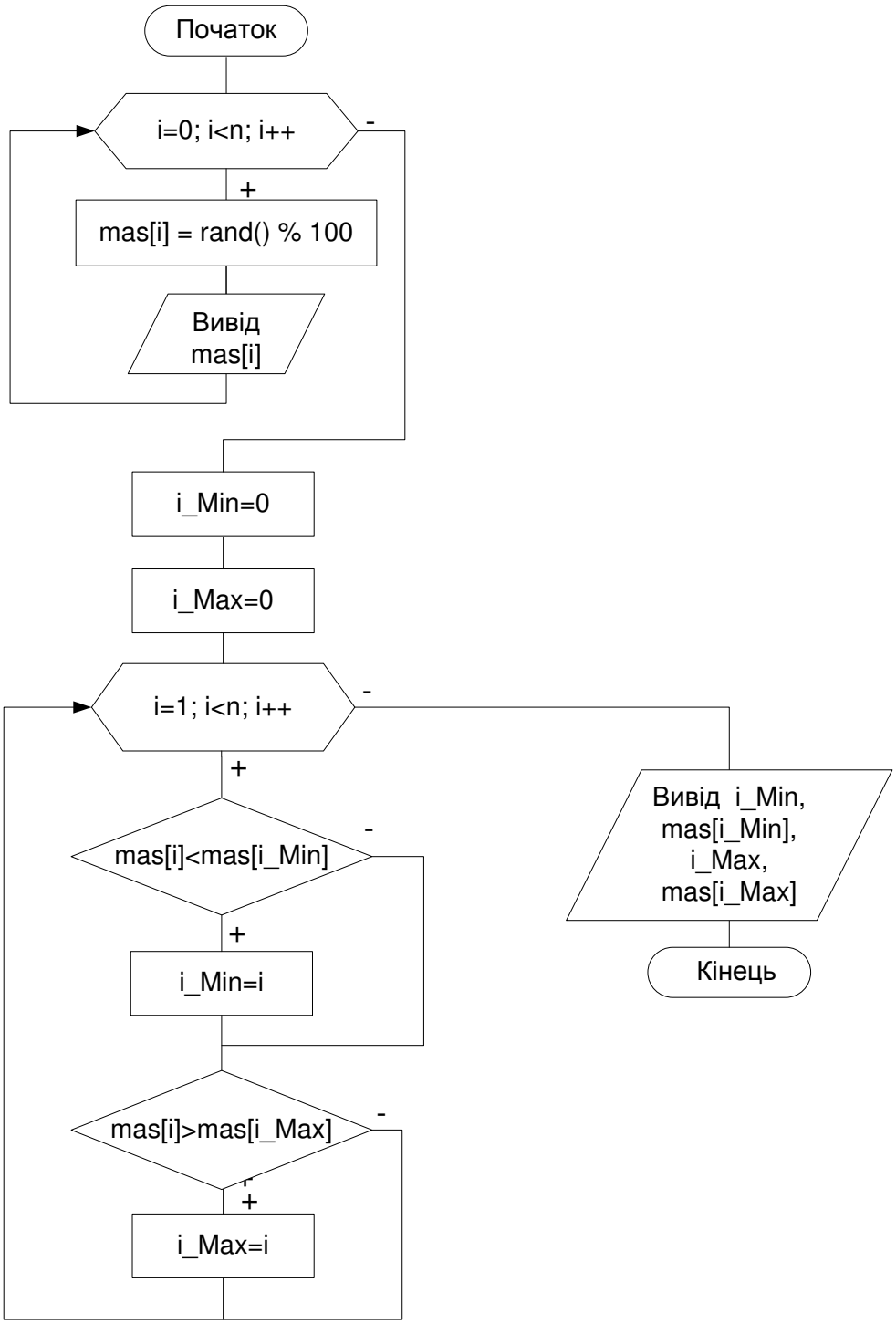
1. Оголошуємо масив mas[n], де n – цілочисельна константа.
2. За допомогою циклу for (i=0; i<n; i++) заповнюємо елементи масиву довільними числами від 0 до 99 (mas[i]=rand() % 100) та виводимо його на екран.
3. Приймаємо початкові значення індексів мінімального (i_Min) та максимального (i_Max) елементів масиву такими, що дорівнюють нулю. (i_Min=0; i_Max=0).
4. Порівнюємо елемент масиву з індексом i_Min з іншими, починаючи з другого і до останнього; якщо елемент масиву з індексом і (mas[i]) менше, ніж елемент з індексом i_Min (mas[i_Min]), то змінюємо значення i_Min на i;

5. Порівнюємо елемент масиву з індексом i_Max з іншими, починаючи з другого і до останнього; якщо елемент масиву з індексом i (mas[i]) більше, ніж елемент з індексом i_Max (mas[i_Max]), то змінюємо значення i_Max на i;

6. Виводимо на екран значення:

7. i_Min, mas[i_Min], i_Max та mas[i_Max].

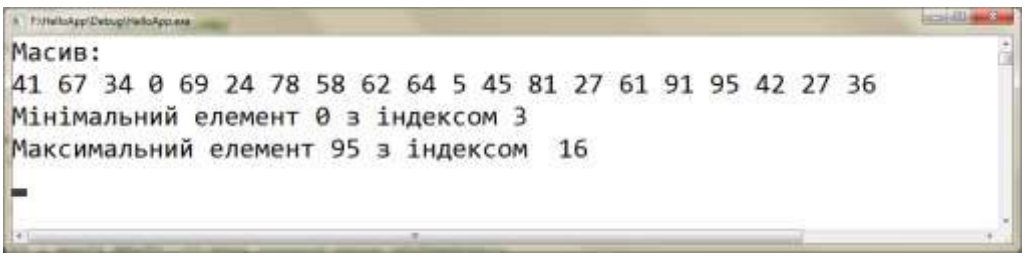
Нижче представлена блок-схема цього алгоритму:



Текст програми:

```
#include <iostream>
#include <conio.h>
using namespace std;
int main()
{
    setlocale(0, "Ukr");
     const int n=20;
     int mas[n];    // оголошення масиву
     int i, i_Min=0, i_Max=0;
     cout<<"Масив: \n";
     for(i=0; i<n; i++)
          {
           mas[i]=rand() % 100;  // генерація випадкових чисел
           cout<<mas[i]<<" ";       // вивід елемент із масиву
         }
     for (i=1; i<n; i++)
     {
         if(mas[i]<mas[i_Min])  // якщо елемент менше мінімального
             i_Min=i;             // то змінюємо індекс i_Min=i
         if(mas[i]>mas[i_Max])  // якщо елемент більше максимального
             i_Max=i;               // то змінюємо індекс i_Max=i
     }
     cout<<"\nМінімальний елемент "<<mas[i_Min];
     cout<<" з індексом "<<i_Min<<endl;
     cout<<"Максимальний елемент "<<mas[i_Max];
     cout<<" з індексом  "<<i_Max<<endl;
    _getch();
}
```

Результат роботи програми:

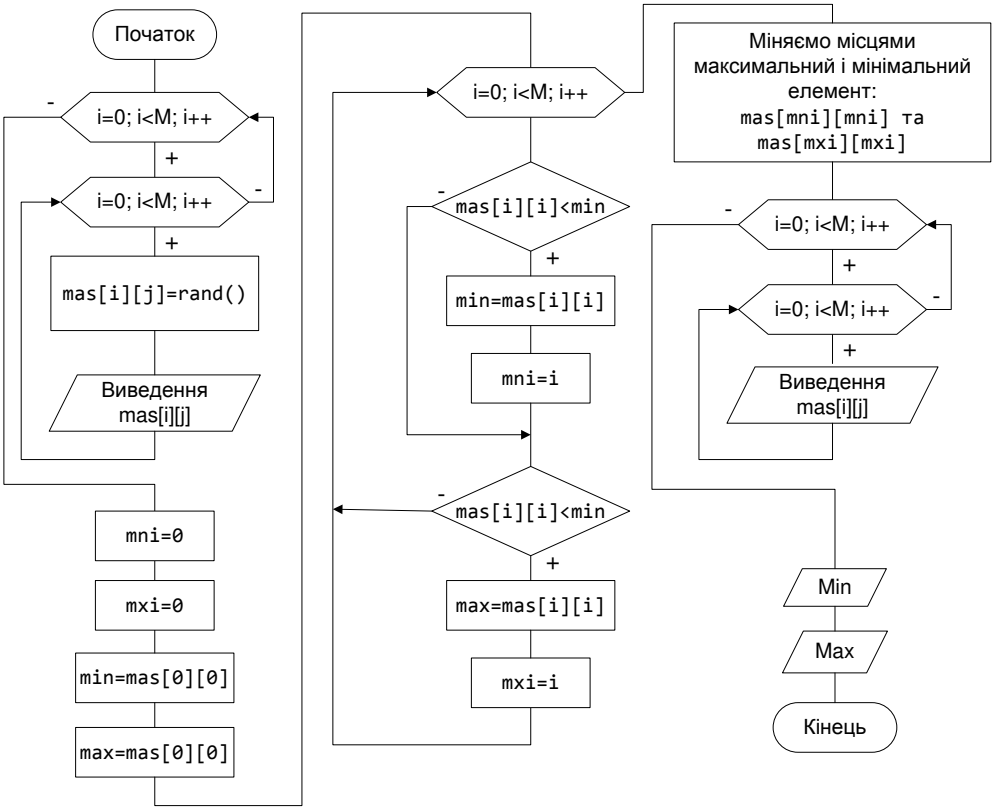


**Завдання 2**: Згенерувати квадратну матрицю, яка складається з випадкових елементів. Поміняти місцями максимальний і мінімальний елементи її головної діагоналі. Вивести на екран обидві матриці.

**Алгоритм:**

1. Оголошуємо двовимірний масив.
2. Заповнюємо двовимірний масив випадковими числами.
3. Припускаємо, що максимальний та мінімальний елементи це перше значення двовимірного масиву (з індексами 0,0).
4. Після цього виділяємо головну діагональ (a[i][i]) і знаходимо максимальний та мінімальний елемент головної діагоналі. Запам'ятовуємо і записуємо індекси цих елементів масиву.
5. Міняємо місцями мінімальний і максимальний елемент масиву використовуючи запам’ятовані індекси.
6. Виводимо на екран змінений масив і максимальний та мінімальний елементи головної діагоналі.

Нижче представлена блок-схема цього алгоритму:

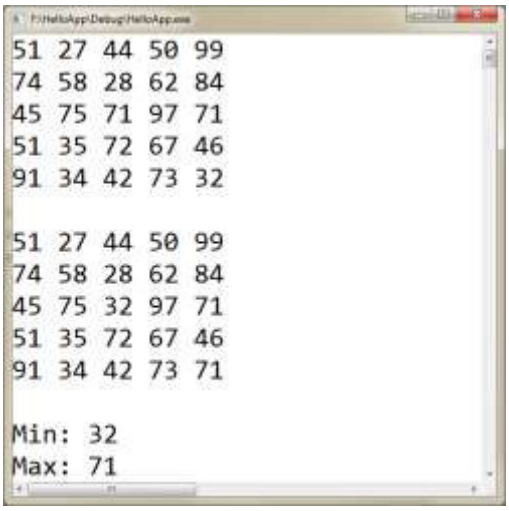


Текст програми:

```
#include <iostream>
#include <conio.h>
using namespace std;

int main()
{
const int N = 5, M = 5;
int mas[N][M];
int max,min;
int mxi,mni;
int i,j;

for (i = 0; i < N; i++) {
   for (j = 0; j < M; j++)
    {
     mas[i][j]=rand()%90+10;
     cout<<mas[i][j]<<" ";
    }
  cout<<"\n";
  }
cout<<"\n";
min=mas[0][0];
max=mas[0][0];
mni=0; mxi=0;
for (i = 0; i < N; i++)
{
          if (mas[i][i]<min) {
               min=mas[i][i];
               mni=i;
          }
          if (mas[i][i]>max) {
               max=mas[i][i];
               mxi=i;
          }
```

```
}
int temp=mas[mni][mni];      // міняєма місцями
mas[mni][mni]=mas[mxi][mxi]; // мінімальний
mas[mxi][mxi]=temp;         // і максимальний елемент
for (i = 0; i < N; i++) {
   for (j = 0; j < M; j++)
      cout<<mas[i][j]<<" ";
  cout<<"\n";
  }

cout<<"\nMin: "<<min;
cout<<"\nMax: "<<max;
_getch();
}
```

Результат виконання програми:




**Завдання**

1. Ввести с клавіатури масив з 5 елементів, знайти середнє арифметичне всіх елементів масиву.
2. Ввести с клавіатури масив з 5 елементів, знайти мінімальний з них.
3. Заповнити масив з 10 елементів випадковими числами в інтервалі [-10..10] і знайти в ньому максимальний і мінімальний елементи та їх номери.
4. Заповнити масив з 10 елементів випадковими числами в інтервалі [-10..10] і знайти в ньому два максимальних елемента та їх номери.

5. Заповнити масив з 10 елементів випадковими числами в інтервалі [-10..10] і виконати інверсію (перевернути) окремо для 1-ої і 2-ий половин масиву.

6. Заповнити масив з 12 елементів випадковими числами в інтервалі [-12..12] і виконати інверсію для кожної третини масиву.

7. Заповнити масив з 10 елементів випадковими числами в інтервалі [-10..10] і виконати циклічний зсув вправо.

8. Заповнити масив з 12 елементів випадковими числами в інтервалі [-12..12] і виконати циклічний зсув вправо на 4 елементи.

9. Заповнити масив з 10 елементів випадковими числами в інтервалі [0..100] і впорядкувати його по останній цифрі.

10. Ввести матрицю розмірності $m \times n$. Знайти рядок, сума елементів якого максимальна. Вивести вихідну матрицю і знайдений рядок.

11. Ввести матрицю розмірності $m \times n$. Знайти суму елементів у кожному стовпці. Результат записати в одновимірний масив. Вивести вихідну матрицю і отриманий одновимірний масив.

12. Ввести матрицю розмірності $m \times n$. Підрахувати суму елементів непарних рядків, записати результат в одновимірний масив.

13. Ввести квадратну матрицю. Знайти суму елементів тих рядків, у яких на головній діагоналі знаходяться від’ємні числа.

14. Ввести квадратну матрицю. Поміняти місцями максимальний і мінімальний елементи її головної діагоналі. Вивести на екран обидві матриці.

# 6 ВПРАВИ ДО САМОСТІЙНОГО ВИКОНАННЯ

## 6.1 Цикли

1. Написати програму, яка виводить ваше ім'я та прізвище N раз

Дано два числа. X (початок) Y (кінець) На проміжку XY треба скласти таблицю квадратів цілих чисел і вивести квадрат кожного числа на екран
3 = 9

4 = 16

....

2. Дано два числа X (початок) Y (кінець) На проміжку XY треба вивести таблицю кубів цілих чисел, які діляться на A
3. Написати програму, яка підсумовує ряд чисел від 1 до N
4. Написати програму, що виводить ряд 1,1,2,3,5,8,13 ... N, (число Фіббоначі) N задається користувачем.
5. Є ряд:

$$1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\ldots$$

   Написати програму, яка обчислює такий ряд від одного до N, N задається користувачем.
6. Написати програму таблиці ступенів двійки від нульової до десятої
7. Написати програму обчислення факторіала числа.
8. Написати програму, яка виводить таблицю вартості товару з кроком 100 У таблиці з N значень

| Грам | Ціна |
|---|---|
| 100 | 2.35 |
| 200 | 4.70 |
| 300 | 7.05 |
| . . . . . | |

9. Написати програму, що виводить на екран таблицю Піфагора (вкладені цикли)

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 2 | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 |
| 3 | 3 | 6 | 9 | 12 | 15 | 18 | 21 | 24 | 27 | 30 |
| 4 | 4 | 8 | 12 | 16 | 20 | 24 | 28 | 32 | 36 | 40 |
| 5 | 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 | 50 |
| 6 | 6 | 12 | 18 | 24 | 32 | 36 | 42 | 48 | 54 | 60 |
| 7 | 7 | 14 | 21 | 28 | 35 | 42 | 49 | 56 | 63 | 70 |
| 8 | 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 | 72 | 80 |
| 9 | 9 | 18 | 27 | 36 | 45 | 54 | 63 | 72 | 81 | 90 |

10. Порахувати суму ряду. Параметр n задається користувачем.

$$1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}+\ldots+\frac{1}{2n+1}(-1)^n$$

11. Користувачеві пропонується ввести ціле число N раз, порахувати середнє арифметичне за введенням користувачем числам.
12. Написати програму, що перевіряє чи є введене користувачем число простим.
13. Скласти таблицю цілих чисел і вивести на екран значення $y=2x^2+78x-33$ в діапазоні x=-8 до x= 4
14. Написати програму, яка обчислює НСД (найбільший спільний дільник) двох цілих чисел (алгоритм Евкліда)
15. Написати програму, яка обчислює НСК (найменше спільне кратне) двох цілих чисел.
16. Знайти суму всіх цілих непарних чисел, у зазначеному користувачем діапазоні.
17. Написати програму виведення на екран перших ста простих чисел.
18. Написати програму виведення на екран перших ста простих чисел, пропускаючи кожне друге
19. Дана дріб виду $\frac{x}{y}$. x і y вводяться користувачем. Перевірити чи можна скоротити дріб і вивести на екран чи скорочується вона чи ні.

20. Користувач вводить ціле число. Підрахувати суму його цифр.
21. Визначити розрядність введеного користувачем цілого числа.
22. Користувач вводить ціле шести розрядне число. Порахувати суму з перших трьох цифр і останніх трьох цифр, які його складають

23. Порахувати кількість тризначних чисел, сума цифр яких не перевищує 10

### 6.2 Одновимірні масиви

24. Є масив з відомою кількістю значень. Кількість значень вказати константою. Потрібно виконати наступне: вивести масив на екран у зворотному порядку
25. Є масив з відомою кількістю значень. Кількість значень вказати константою. Знайти найменший елемент і найменший елемент за модулем.
26. Знайти і вивести на екран такі елементи, у яких обидва сусідніх елемента як і сам він діляться без остачі на одне і те ж число
27. Виконати сортування масиву по зростанню методом бульбашки і вивести вхідний масив і оброблений на екран.
28. Після заповнення масиву додати до кожного елементу суму чисел, які є його сусідами.
29. Знайти суму і добуток всіх елементів масиву.
30. Визначити скільки в масиві елементів, рівних сумі всіх елементів масиву.
31. Поміняти місцями найбільший елемент з найменшим.
32. Користувач задає інтервал a-b Вивести всі елементи масиву, які не потрапляють в цей інтервал. a і b - це індекси елементів масиву.
33. Знайти середнє арифметичне всіх елементів масиву і підрахувати кількість чисел всередині масиву які не перевищують знайдене значення.
34. Порахувати відсоток входження кожного з чисел в масив і вивести результати на екран. При виведенні на екран числа не повинні повторюватися.

Наприклад: 1 1 2 3 3 5 5 1 4 2

1 - 30%

2 - 20%

3 - 20%

4 - 10%

5 - 20%

35. Стисніть заданий масив відкиданням нульових елементів.
36. Запропонуйте користувачеві ввести число N. Зробить зсув всіх елементи масиву вправо (вліво) на N елементів.

### 6.3 Двовимірні масиви

Довжина масиву задається константою. (Якщо ви вже злегка «просунутий», то використовуйте масиви, кількість значень яких задається безпосередньо під час роботи програми і отже виконуйте обчислення в залежності від вибору користувача)

37. Підсумуйте всі елементи двовимірного масиву.
38. Заповніть діагоналі масиву нулями.
39. Виведете на екран індекси тих елементів масиву, в яких знаходяться від'ємні числа.
40. Заповніть масив такою послідовністю, в якій через кожні 5 елементів записуване значення збільшується на 1. Заповнювати масив зліва направо Кількість рядків і стовпців задавати з клавіатури.

```
0   0   0   0   0   1
1   1   1   1   2   2
2   2   2   3   3   3
3   3   4   4   4   4
```

41. Заповнити масив схожим чином, але тільки вже по стовпцях:

```
0   1   2   3
0   1   2   3
0   1   2   4
0   1   3   4
0   2   3   4
1   2   3   4
```

42. Знайти максимальний і мінімальний елемент для кожного рядка і для кожного стовпця. Масив і результати показати на екрані.
43. У масиві [N, N] обнулити всі елементи вище головної діагоналі. Вивести на екран вхідний і отриманий масиви

```
3  4  2  4  5
6  2  1  3  6
7  8  2  7  9
1  5  3  4  2
4  7  9  3  6
------------
3  0  0  0  0
6  2  0  0  0
7  8  2  0  0
1  5  3  4  0
4  7  9  3  6
```

44. Користувач вводить число. Визначити суму всіх елементів масиву, які більше заданого користувачем числа.
45. Знайти кількість мінімальних і максимальних елементів масиву на головних діагоналях.

## 6.4 Різні задачи на кмітливість

1. Є дві змінні, значеннями яких є цілі числа. Написати функцію, яка змінює місцями значення цих змінних. Використовувати допоміжні змінні заборонено.
2. Знайти кількість одиниць у двійковому запису числа N
3. Послідовність з 2n цифр називають «щасливим квитком», якщо сума перших n цифр дорівнює сумі останніх n цифр. Знайти кількість «щасливих» квитків заданої довжини n.
4. Написати програму визначення і-го числа Фібоначчі, використовуючи рекурсивний алгоритм.
5. Нехай є словник. Знайти в ньому всі анаграми (слова, що складені з однакових літер)
6. Послідовність цілих чисел a1, a2, … an задана зв’язним списком та вказівником на його перший елемент. Написати функцію, яка виводить список на екран.
7. Задана послідовність цілих додатних чисел. Написати програму для запам’ятування послідовності у вигляді зв’язного списку
8. Скільки чисел між 1000 і 10000 складається з непарних цифр, скільки з різних цифр?

# СПИСОК ЛІТЕРАТУРИ

1. Караванова Т.П. «Інформатика. Збірник вправ та задач з алгоритмізації та програмування. Базовий курс» Шепетівка, «ПП Шестопалов», 2008. – 151с

2. Герберт Шилдт. Полный справочник по C++ The Complete Reference. — 4-е изд. — М.: Вильямс, 2006

3. Бьерн Страуструп. Язык программирования C++. 1104 стр. Бином, Невский Диалект, 2008 г.

4. Основи програмування та алгоритмічні мови: лабораторний практикум / уклад. О.В. Чебанюк, Ю.О. Міловідов. – К.: Університет економіки та права «КРОК», 2013. – 108с.

5. Н. Культин «C/C++ в задачах и примерах» Санкт-Петербург, - 3-е изд «БХВ-Петербург» 2007. – 288с.

6. Ю.О. Міловідов. «Алгоритми і структури даних», навчальний посібник К. НУБіП України, 2018. – 200 с.»

7. .

8. Т.В. Ковалюк «Основи програмування» Київ, Видавнича група BHV, 2005. – 384с.

9. Т.А. Павловская «C/C++ Программирование на языке высокого уровня», Санкт-Петербург, «Питер», 2007. – 461с.