**Практична робота №8.** Написання програм з вкладеними розгалуженнями та циклами**.**

**Мета**: навчитися писати програми з вкладеними циклами та розгалуженнями, використанням математичних функцій та виконанням операцій введення - виведення **на мові С++ в консольному режимі.**

**Запитання**

1. Які оператори розгалуження ви знаєте?
2. Які оператори циклу ви знаєте?
3. Яка бібліотека С++ використовується для програмування математичних формул?
4. Як підключити математичні константи до програми С++?

**Завдання.**

1. Запустити середовище програмування С++ .
2. Записати програму, що виконує 3 завдання з пп.3.1-3.3. В першому рядку програми записати в коментарі номер групи та прізвище.
3. Вхідні дані ввести, а результати вивести, використовуючи потокове введення-виведення даних. В першому рядку кожної програми записати

*// МНТ/ЕТ Група № Прізвище Номер ЛР*

* 1. Написати на мові С++ блок введення даних для розрахунку формул в пп. 3.3 – 3.4. Використати потокове виведення даних z, х, y для 3.3 та y, х, а для 3.4.
  2. Програма повинна запитати номер завдання (число 1, 2 або 0 для закінчення) і в залежності від введеного значення виконувати відповідне завдання. Якщо введений 0 – програма припиняє роботу, якщо значення відмінне від 0-2 надається повідомлення про помилку (1 – фрагмент 3.3, 2 – фрагмент 3.4), в інших випадках виконується запитане завдання, після чого знову повторюється запит.
  3. Виконати розрахунок для пункту меню 1:

,

де *х*=0,25; *у*=1,31; *а*=3,5; *b*=0,9

* 1. Виконати розрахунок для пункту меню 2:

 якщо    

Результати надсилати на електронну адресу викладача [**t.i.lumpova@gmail.com**](mailto:t.i.lumpova@gmail.com)у вигляді cpp-файлу з іменем у форматі

**<Номер групи><Номер лабораторної><Прізвище англійською>**

Наприклад, 31-01Ivanov.cpp.

Іншим рішенням є надсилання поштою посилання на текст програми за URL адреси, яку надає C++Shell, вказавши в темі листа, номер групи прізвище студента та номер ПР.

В темі листа вказати, номер групи, прізвище студента та номер ПР як "ПР№3".

**Строк відсилки ЛР для МНТ/ЕТ-41 18.03.2024**

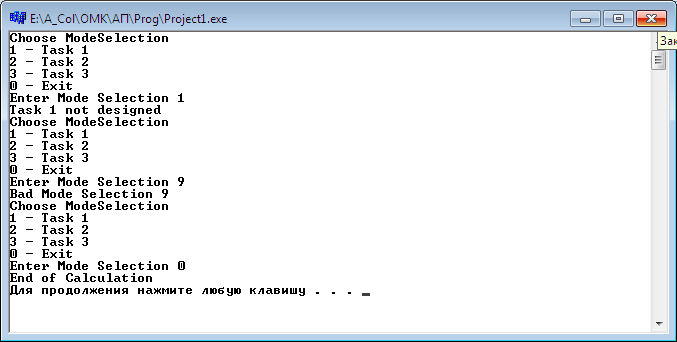
Всі запитання, що виникнуть, надсилайте на електронну адресу викладача, В темі листа вказати, номер групи, прізвище студента та номер ПР як "ПР№5 В темі листа вказати, номер групи, прізвище студента, номер ПР та фразу "Запитання".

**Методичні вказівки щодо організації самостійної роботи студентів**

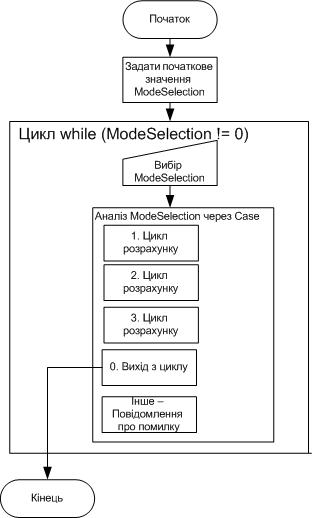
**Теоретичні відомості.**

Такого типу завдання доцільно виконувати використовуючи запит на обрання варіанту розрахунку (ModeSelection) через меню, а сам варіант розрахунку обирати через оператор вибору **case**, і все це виконувати в циклі.

Приклад запиту нижче (оскільки розрахунок тут не зроблено, то видається повідомлення "Task not designed".



Два завдання – це розрахунки в циклі, які в разі обрання **case** для вибору оператору розрахунку потрібно вкладати у відповідну гілку. Схематично це представлено нижче (це не блок-схема, а ілюстрація).



**Для довідки**

**Математичні функції (**заголовний файл **math.h)**

| **Прототип функції** | **Ім’я** | **Призначення** |
| --- | --- | --- |
| Double sin (double \_х); | **sin (x)** | синус x (в радіанах) — **sin x** |
| Double cos (double \_x); | **cos (x)** | косинус x (в радіанах) — **cos х** |
| Double tan (double \_x); | **tan (x)** | тангенс х (в радіанах) — **tg х** |
| Double asin (double \_x); | **asin (x)** | арксинус х — **arcsin х** |
| Double acos (double \_x); | **acos (x)** | арккосинус х — **arcos х** |
| Double atan (double \_x); | **atan (x)** | арктангенс х — **arctg х** |
| Double atan2 (double \_y, Double\_x); | **atan2 (y,x)** | арктангенс у/х — **arctg (у/х)** |
| Double sinh (double \_x); | **sinh (x)** | синус гіперболічний х — **sh х** |
| Double cosh (double \_x); | **cosh (x)** | косинус гіперболічний х — **ch х** |
| Double tanh (double \_x); | **tanh (x)** | тангенс гіперболічний х — **th х** |
| Double log (double \_x); | **log (x)** | натуральний логарифм х — **ln х** |
| Double log10 (double \_x); | **log10 (x)** | десятковий логарифм х — **log х** |
| Double exp (double \_x); | **exp (x)** | піднесення е до степеня х — **ех** |
| Double pow (double \_x, double\_y); | **pow (x,y)** | піднесення х до степеня у — **ху** |
| Double pow 10 (int \_p) | **pow10 (p)** | повертає **10р** |
| Double sqrt (double \_х); | **sqrt (x)** | корінь iз x, x > 0 |
| Double hypot (double\_x, double\_y); | **hypot (x,y)** | корінь із (х2+у2) |
| Double fabs (double \_\_x); | **fabs (x)** | абсолютне значення х — |х| типу **double** |
| int abs (int \_x); | **abs (x)** | абсолютне значення х — |х| типу **int** |
| long labs (long \_x); | **labs (x)** | абсолютне значення х — |х| типу **long** |
| Double fmod (double \_\_x, double\_y); | **fmod (x,y)** | залишок від ділення х на у |
| Double ceil (double \_\_x); | **ceil (x)** | округлення до більшого |
| Double floor (double \_x); | **floor (x)** | повертає найближче ціле, не більше за х |
| Double modf (double \_x, double); | **modf(x,&p)** | виділяє цілу й дробову частинні числа |
| Double atof(const char\* \_s); | **atof (s)** | перетворює рядок символів у число з плаваючою крапкою |

**ЗАСТОРОЖЕННЯ ЩОДО ВИКОРИСТАННЯ ЦИКЛІВ**

**Нескінченні цикли**

Якщо умова циклу завжди приймає значення true, то і сам цикл буде виконуватися нескінченно. Це називається **нескінченним циклом**. Наприклад:

**#include <iostream>**

**using namespace std; *// простір імен***

**int main()**

**{**

**int count = 0;**

**while (count < 10) // ця умова ніколи не буде false**

**// тому цей рядок буде виконуватися постійно**

**cout << count << " ";**

**return 0; // а цей рядок ніколи не виконається**

**}**

Оскільки змінна count не збільшується на одиницю в програмі, то умова count < 10 завжди буде true. Відповідно, цикл ніколи не завершиться і програма буде постійно виводити 0 0 0 0 0 ... .

Можна навмисно оголосити нескінченний цикл наступним чином:

**while (1) // чи while (true)**

**{**

**// Цей цикл буде виконуватися постійно**

**}**

Єдиний спосіб вийти з нескінченного циклу — використати один з наступних операторів: [**return**](https://acode.com.ua/urok-15-funktsiyi-i-operator-return/), break, exit, [**goto**](https://acode.com.ua/urok-69-operator-goto/) або згенерувати виняток. Програми, які працюють до тих пір, поки користувач не вирішить зупинити їх, іноді навмисно використовують нескінченні цикли разом з операторами return, break або exit для завершення циклу. Поширена така практика в серверних веб-додатках, які працюють безперервно і постійно обслуговують веб-запити.

## Цикл for

**for (оголошення змінних; умова; інкремент/декремент лічильника)**

**тіло циклу;**

Або, конвертуючи for в еквівалентний **цикл while**:

**{ // зверніть увагу, що цикл знаходиться в блоці**

**// оголошення змінних;**

**while (умова)**

**{**

**тіло циклу;**

**інкремент/декремент лічильника;**

**}**

**} // змінні, оголошені всередині циклу, виходять з**

**// області видимості тут**

Змінні, визначені всередині циклу for, мають спеціальний **тип області видимості:** **область видимості циклу**. Такі змінні існують тільки всередині циклу і недоступні за його межами.

## Помилка неврахованої одиниці

Однією з найчастіших проблем з якою стикаються початківці в циклах for (а також і в інших типах циклів) є **помилка неврахованої одиниці**. Вона виникає, коли цикл повторюється на 1 раз більше або на 1 раз менше від потрібної кількості ітерацій. Це зазвичай відбувається через те, що в умові використовується некоректний оператор порівняння (наприклад, > замість >= або навпаки). Як правило, ці помилки важко відстежити, так як компілятор не буде скаржитися на них і програма працюватиме нормально, але її результати будуть неправильні.

При написанні циклів for пам’ятайте, що цикл буде виконуватися до тих пір, поки умова є істинною. Рекомендується тестувати цикли, використовуючи різні значення для перевірки працездатності циклу. Хорошою практикою є перевірка циклів за допомогою вхідних даних (чисел, символів та іншого), які змушують цикл виконатися 0, 1 і 2 рази. Якщо цикл працює справно, значить все ОК.

**Правило: Тестуйте свої цикли, використовуючи вхідні дані, які змушують цикл виконатися 0, 1 і 2 рази.**

## Пропущені вирази в циклі

Також в циклах можна пропускати один або відразу всі вирази. Наприклад:

**#include <iostream>**

**using namespace std; *// простір імен***

**int main()**

**{**

**int count = 0;**

**for (; count < 10; )**

**{**

**cout << count << " ";**

**++count;**

**}**

**return 0;**

**}**

Результат:

0 1 2 3 4 5 6 7 8 9

Ініціалізацію лічильника ми прописали поза тілом циклу, а інкремент лічильника — всередині тіла циклу. У самому операторі for ми вказали лише умову. Іноді бувають випадки, коли не потрібно оголошувати лічильник циклу (тому що у нас вже є один) або збільшувати його (так як ми збільшуємо його якимось іншим способом).

Хоча це і не часто можна спостерігати, але в операторі for можна взагалі нічого не вказувати. Варто зазначити, що подібне призведе до нескінченного циклу:

**for (;;)  
    тіло циклу;**

Вищенаведений приклад еквівалентний наступному:

**while (true)  
    тіло циклу;**

## Оголошення змінних в циклі for

Хоча в циклах for зазвичай використовується тільки один лічильник, іноді можуть виникати ситуації, коли потрібно працювати відразу з декількома змінними. Для цього використовується **оператор Кома**. Наприклад:

**#include <iostream>**

**using namespace std; *// простір імен***

**int main()**

**{**

**int aaa, bbb;**

**for (aaa = 0, bbb = 9; aaa < 10; ++aaa, --bbb)**

**cout << aaa << " " << bbb << std::endl;**

**return 0;**

**}**

Цей цикл присвоює значення двом раніше оголошеним змінним: aaa = 0 і bbb = 9. Тільки з кожною ітерацією змінна aaa збільшується на одиницю, а bbb — зменшується на одиницю.

Результат виконання програми:

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

Примітка: Вищенаведений цикл можна переписати наступним чином:

**#include <iostream>**

**using namespace std; *// простір імен***

**int main()**

**{**

**for (int aaa = 0, bbb = 9; aaa < 10; ++aaa, --bbb)**

**cout << aaa << " " << bbb << std::endl;**

**return 0;**

**}**

В такому випадку кома в оголошенні змінних є частиною синтаксису, а не використанням оператора Кома. Але ефект ідентичний.

**Оператор Кома** (або ще ***“оператор Comma”***) дозволяє обчислювати декілька виразів (в той час як допускається тільки один).

|  |  |  |  |
| --- | --- | --- | --- |
| **Оператор** | **Символ** | **Приклад** | **Операція** |
| Кома | , | x, y | Обчислюється x, потім обчислюється y, а потім повертається значення y |

Вираз, в якому знаходиться цей оператор, буде мати значення правого операнду, наприклад:

**int x = 0;**

**int y = 2;**

**int z = (++x, ++y); // інкремент x і y**

Змінній z присвоюється результат обчислення ++у (правого операнду), що дорівнює 3.

Майже в кожному випадку, оператор, в якому є оператор Comma, краще записувати у вигляді окремих інструкцій. Вищенаведений код коректніше буде записати наступним чином:

**int x = 0;**

**int y = 2;**

**++x;**

**++y;**

**int z = y;**

Оператор Comma має найнижчий **пріоритет** з усіх операторів (навіть нижче, ніж в оператора присвоювання). Через це, наступні два рядки коду виконують не одне і те ж:

|  |
| --- |
| z = (a, b); // спочатку обчислюється вираз (a, b), який дорівнює значенню b, а потім результат присвоюється змінній z  z = a, b; // обчислюється як "(z = a), b", тому змінній z присвоюється значення a, а змінна b ігнорується |

Більшість програмістів не використовують оператор Comma взагалі (хіба що тільки в циклах for).

Кома, яка використовується у викликах функцій, не є оператором Comma:

**int sum = add(x, y); // ця кома не є оператором Comma**

Аналогічно, при оголошенні декількох змінних в одному рядку, кома використовується як роздільник, а не як оператор:

**int x(3), y(5); // ця кома не є оператором Comma**

**Правило: Уникайте використання оператора Comma (винятком є використання в циклах for).**