Міністерство освіти та науки України

Інститут «Телекомунікації, радіоелектроніки та радіотехніки»



Звіт до лабораторної роботи № 12

З дисципліни «Програмування частина 2»

Оператори циклу

Виконав студент групи АП-12

Дроф’як Святослав

Перевірив викладач

Пелех Н.В.

Львів 2024

***Мета роботи:*** ознайомитися з особливостями функціонування операторів циклу та навчитись їх використовувати у процесі програмування.

**Хід роботи:**

1. Ознайомитися з теоретичними відомостями.
2. Здійснити виконання усіх прикладів, представлених у теоретичних відомостях, після чого представити скріни їх коду та результати їх виконання у звіті.
3. Виконати нижченаведену програму для обчислення таблиці переведення температури за шкалою Фаренгейта в температуру за шкалою Цельсія. Скрін коду програми та результати її виконання представити у звіті.
4. Скласти програму для створення прямокутного трикутника із зірочок (\*), при цьому трикутник має розміри: n рядків у висоту та n символів у ширину. Значення n вводиться з клавіатури. Скрін коду програми та результати її виконання представити у звіті.
5. Оформити звіт.
6. Обчислити скільки зерен необхідно було би видати винахідникові шахів, якщо за першу клітину шахівниці він попросив видати одну зернину пшениці, а за кожну наступну вдвічі більше за попередні. У шахівниці 64 клітини.
7. Для цілих чисел від 1 до 20 обчислити квадратні, кубічні та корені четвертого порядку. Результати звести у таблицю, використовуючи форматування функції printf().
8. Здійснити табулювання функції, що з певними припущеннями з достатньою точністю моделює імпульс Максвела, який утворюється при ударному збудженні широкосмугової антени. Обчислення провести на проміжку зміни **і** в межах [0-31] з кроком i=1, N=32. Результати вивести у вигляді таблиці. Визначити найбільше та найменше значення функції на цьому проміжку.
9. В обчислювальних задачах при програмуванні ітераційних алгоритмів, що закінчуються при досягненні заданої точності, часто необхідна оцінка «машинного нуля», тобто числового значення, менше за яке неможливо задати точність даного алгоритму. Абсолютне значення «машинного нуля» залежить від розрядної сітки застосовуваного комп’ютера, від прийнятої в конкретному трансляторі точності представлення дійсних чисел і від значень, що використовуються для оцінки точності. Наступна програма оцінює абсолютне значення «машинного нуля» відносно близьких (за модулем) до одиниці змінних типу **float**.

Завдання: змінити програмузастосувавши кожного разу один із трьох циклічних операторів.

Оцінку «машинного нуля» провести також для даних типу **double -**формат виведення %le**, longdouble**формат виведення %Le.

10. Обчислити значення скінченної суми, або добутку згідно свого варіанту. Врахувати, що навіть для невеликих чисел значення факторіала може вийти за гранично допустимі для даного типу даних. Аргумент тригонометричних функцій задавати в межах: 0 £ *X* £ p / 2 .

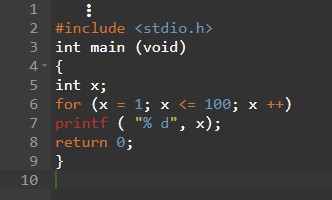
11. Відомо, що одним із методів обчислення багатьох функцій є розкладання їх в ряд Тейлора:

Завдання: для заданого **х**, яке уводиться з клавіатури під час роботи програми, обчислити значення функції **y**за допомогою бібліотечних функцій компілятора (наприклад ex – pow(e,2)), а такожза допомогою вище наведеного явного розкладу її в ряд (ітераційний процес до досягнення заданої точності). Обчислити при цьому також кількість ітерацій, тобто кількість членів ряду в розкладі функції. Точність обчислень, тобто значення члена ряду розкладу функції коли необхідно припиняти ітераційний процес, яке в кожній ітерації обчислюється як різниця між бібліотечним значенням і членом розкладу, **a=0.00001**. Аргумент тригонометричних функцій задавати в межах: 0 £ *X* £ p / 2 .

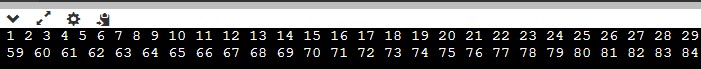
**Завдання 1**

**Реалізація:**

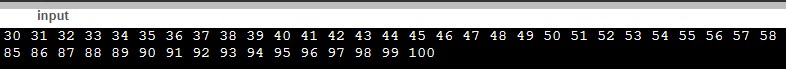
Програмна реалізація мовою С



Код завдання 1, приклад 1

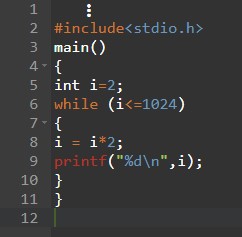


Скріншот виконання коду, приклад 1

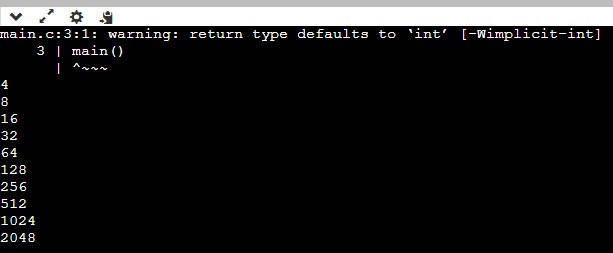


Скріншот виконання коду, приклад 1

Коли програма виконується, вона починає цикл for, який ініціалізує змінну x значенням 1 і продовжує виконувати цикл, поки x не стане більше 100. У кожній ітерації циклу вона виводить поточне значення x, а потім збільшує x на 1. Після завершення циклу програма повертає 0 і закінчується.

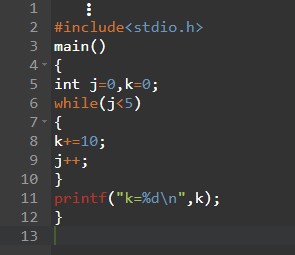


Код завдання 1, приклад 2

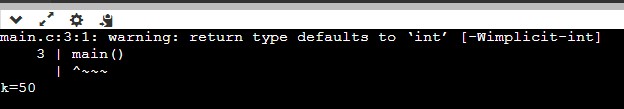


Скріншот виконання коду, приклад 2

Коли програма запускається, вона ініціалізує змінну i значенням 2. Потім вона входить у цикл while, який продовжується до тих пір, поки i не перевищить 1024. У кожній ітерації циклу значення i множиться на 2, і нове значення i виводиться. Цикл завершується, коли i стає більше 1024. В результаті програма виводить степені двійки: **4, 8, 16, … , 2048**.



Код завдання 1, приклад 3

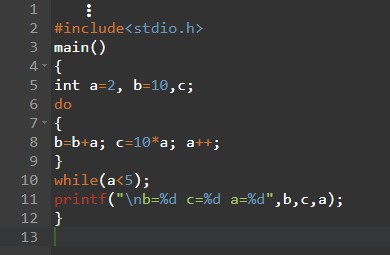


Скріншот виконання коду, приклад 3

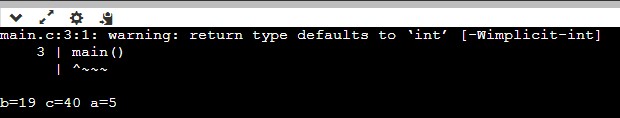
Ця програма на мові програмування C виконує наступні дії:

1. Ініціалізує дві змінні j та k зі значенням 0.
2. Виконує цикл while, який продовжується, поки j менше 5.
3. У кожній ітерації циклу до k додається 10, а j збільшується на 1.

Після завершення циклу, коли j досягне 5, програма виводить значення k, яке буде 50, оскільки 10 було додано п’ять разів.



Код завдання 1, приклад 4



Скріншот виконання коду, приклад 4

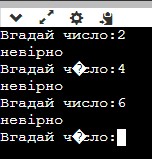
Ця програма на мові C виконує наступне:

1. Ініціалізує змінні a зі значенням 2, b зі значенням 10, і c без початкового значення.
2. Виконує цикл do-while, який продовжується до тих пір, поки a менше 5.
3. У кожній ітерації циклу:
   * До b додається значення a.
   * c присвоюється значення 10 помножене на a.
   * a збільшується на 1.
4. Цикл завершується, коли a стає рівним 5.

Програма виводить остаточні значення b, c, і a.



Код завдання 3, приклад 5

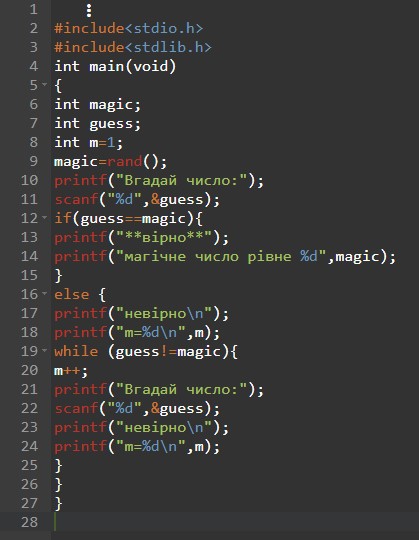


Скріншот виконання коду, приклад 5

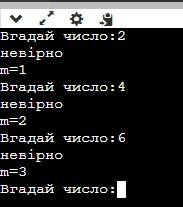
Ця програма на мові C є простим генератором випадкових чисел та грою вгадування числа. Ось що вона робить:

1. Ініціалізує змінну magic та присвоює їй випадкове значення за допомогою функції rand().
2. Запитує у користувача вгадати число та зберігає введене значення у змінній guess.
3. Якщо guess дорівнює magic, програма виводить повідомлення “**вірно**” та показує магічне число.

Якщо guess не дорівнює magic, програма виводить повідомлення “невірно” та продовжує запитувати користувача вгадати число до тих пір, поки не буде вгадано правильне число.



Код завдання 6, приклад 6



Скріншот виконання коду, приклад 6

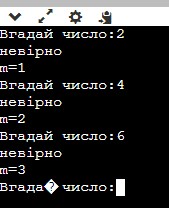
Програма, яку ви надали, є простою грою на вгадування чисел. Ось короткий опис того, що вона робить:

1. Генерує випадкове число і зберігає його у змінній magic.
2. Запитує у користувача вгадати число і зберігає введене значення у змінній guess.
3. Порівнює вгадане число (guess) з магічним числом (magic):
   * Якщо числа співпадають, виводить повідомлення “**вірно**” і показує магічне число.
   * Якщо числа не співпадають, виводить повідомлення “невірно” і показує кількість спроб m.
4. Якщо число не вгадано з першої спроби, програма продовжує запитувати користувача вгадати число до тих пір, поки він не вгадає правильно, збільшуючи лічильник спроб m з кожною невдалою спробою.

Важливо зазначити, що функція rand() генерує псевдовипадкове число, і щоб отримати різні результати при кожному запуску програми, зазвичай використовують функцію srand() з аргументом, заснованим на часі, наприклад srand(time(NULL)). У вашій програмі цього не зроблено, тому magic буде мати одне і те ж значення при кожному запуску програми. Щоб виправити це, вам потрібно додати #include<time.h> на початку програми і викликати srand(time(NULL)) перед rand().



Код завдання 1, приклад 1



Скріншот виконання коду, приклад 1

Ця програма є варіацією гри на вгадування чисел з обмеженням кількості спроб. Ось її основні функції:

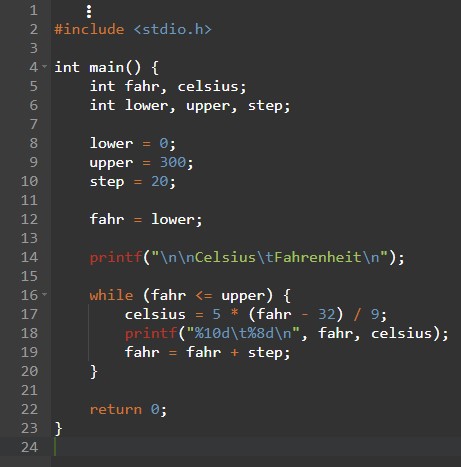
1. Генерує випадкове число і зберігає його у змінній magic.
2. Запитує у користувача вгадати число і зберігає введене значення у змінній guess.
3. Якщо користувач вгадав число (guess == magic), виводить повідомлення “**вірно**” і показує магічне число.
4. Якщо користувач не вгадав число, виводить “невірно” і показує поточну кількість спроб m.
5. Якщо число не вгадано, програма продовжує запитувати користувача вгадати число, збільшуючи лічильник спроб m з кожною невдалою спробою.
6. Якщо кількість спроб перевищує 9 (m > 9), програма виводить магічне число і припиняє гру.

Як і в попередній програмі, для забезпечення різних результатів при кожному запуску, рекомендується використовувати srand(time(NULL)) перед rand(). Також, зверніть увагу, що без використання srand(), число magic буде однаковим при кожному запуску програми.

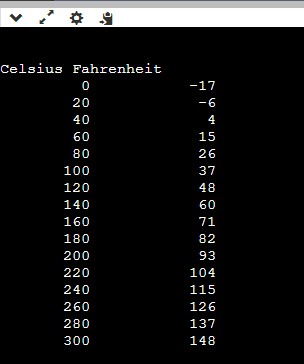
**Завдання 2**

**Реалізація:**

Програмна реалізація мовою С



Код завдання 2, приклад 1



Скріншот виконання коду, приклад 1

Ця програма на мові програмування C виконує конвертацію температур з Фаренгейтів у Цельсії. Ось її основні кроки:

1. Встановлює початкові значення для нижньої (lower), верхньої (upper) меж температур і кроку (step) зміни температури.
2. Ініціалізує змінну fahr з нижньою межею температури.
3. Виводить заголовки стовпців для Цельсіїв та Фаренгейтів.
4. Використовує цикл while для перебору температур від нижньої межі до верхньої, збільшуючи температуру на величину кроку після кожної ітерації.
5. Для кожного значення fahr обчислює відповідне значення у Цельсіях за формулою celsius = 5 \* (fahr - 32) / 9.
6. Виводить результати у вигляді таблиці, де кожен рядок містить значення у Фаренгейтах та відповідне йому значення у Цельсіях.

Формула перетворення використовується для переведення температури з Фаренгейтів у Цельсії і базується на відношенні, де ( \Delta T\_{\text{F}} = \frac{9}{5} \Delta T\_{\text{C}} ) і точка замерзання води у Фаренгейтах (32° F) відповідає 0° C.

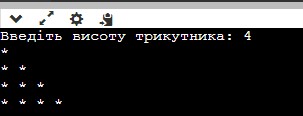
**Завдання 3**

**Реалізація:**

Програмна реалізація мовою С



Код завдання 3, приклад 1



Скріншот виконання коду, приклад 1

Ця програма на мові C створює простий текстовий вивід у формі прямокутного трикутника. Ось що вона робить:

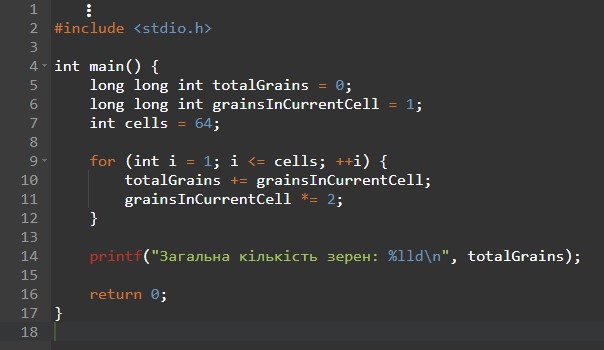
1. Запитує у користувача ввести висоту трикутника і зберігає це значення у змінній height.
2. Використовує два вкладені цикли for:
   * Зовнішній цикл (i) відповідає за кількість рядків.
   * Внутрішній цикл (j) відповідає за кількість зірочок у кожному рядку.
3. Для кожного рядка виводить кількість зірочок, що відповідає номеру рядка, тобто перший рядок має одну зірочку, другий — дві, і так далі, до height.

Після завершення внутрішнього циклу, виводить новий рядок (\n), щоб почати друкувати наступний рядок трикутника.

**Завдання 4**

**Реалізація:**

Програмна реалізація мовою С



Код завдання 4, приклад 1



Скріншот виконання коду, приклад 1

Ця програма використовує концепцію з легенди про винахід шахів, де на кожну наступну клітинку шахової дошки кладуть удвічі більше зерен, ніж на попередню. Ось що робить програма:

1. Ініціалізує змінну totalGrains для зберігання загальної кількості зерен.
2. Встановлює початкову кількість зерен у першій клітинці (grainsInCurrentCell) рівною 1.
3. Використовує цикл for для ітерації через 64 клітинки шахової дошки.
4. Додає кількість зерен у поточній клітинці до загальної кількості (totalGrains).
5. Подвоює кількість зерен у grainsInCurrentCell для наступної клітинки.
6. Після завершення циклу, виводить загальну кількість зерен на дошці.

Програма виведе дуже велике число, яке показує, як швидко зростає кількість зерен через експоненціальне збільшення. Це ілюструє концепцію геометричної прогресії.

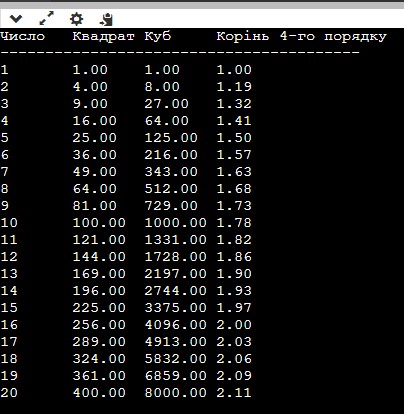
**Завдання 5**

**Реалізація:**

Програмна реалізація мовою С



Код завдання 1, приклад 1



Скріншот виконання коду, приклад 1

Ця програма виводить таблицю значень для чисел від 1 до 20, включно з їх квадратами, кубами та коренями четвертого порядку. Ось детальніше:

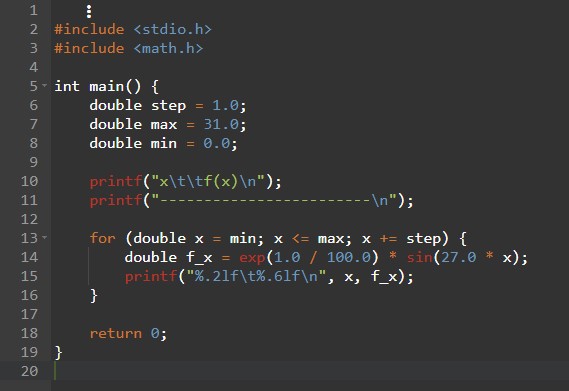
1. Виводить заголовки стовпців для числа, його квадрата, куба та кореня четвертого порядку.
2. Використовує цикл for для ітерації від 1 до 20.
3. Обчислює квадрат (square), куб (cube) та корінь четвертого порядку (fourthRoot) для кожного числа n.
4. Виводить результати у вигляді таблиці, де кожен рядок містить число та його обчислені значення, округлені до двох десяткових знаків.

Таблиця допомагає легко порівняти ці різні математичні операції для послідовності чисел.

**Завдання 6**

**Реалізація:**

Програмна реалізація мовою С



Код завдання 1, приклад 1



Скріншот виконання коду, приклад 1

Ця програма виводить таблицю значень функції ( f(x) = e^{1/100} \cdot \sin(27x) ) для ( x ) від 0 до 31 з кроком 1. Ось що вона робить:

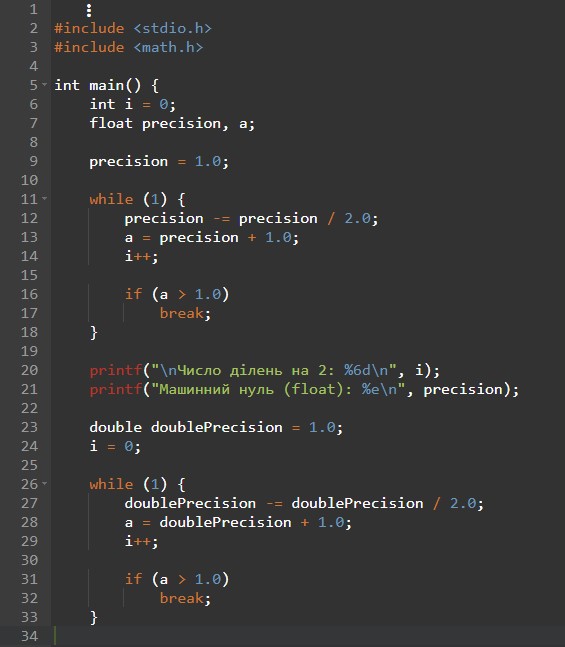
1. Встановлює початкові значення для min (мінімальне значення x), max (максимальне значення x) та step (крок збільшення x).
2. Виводить заголовки стовпців для x та f(x).
3. Використовує цикл for для ітерації значення x від min до max з кроком step.
4. Для кожного значення x, обчислює f\_x за формулою ( f(x) = e^{1/100} \cdot \sin(27x) ).
5. Виводить значення x та відповідне f(x) у вигляді таблиці, де кожен рядок містить значення x та обчислене f(x).

Таблиця допомагає візуалізувати поведінку функції на заданому інтервалі.

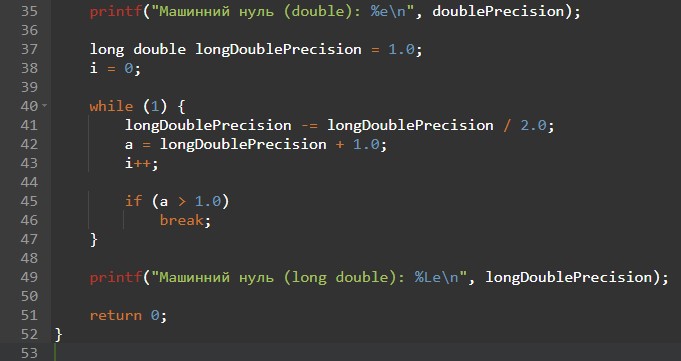
**Завдання 7**

**Реалізація:**

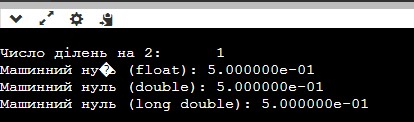
Програмна реалізація мовою С



Код завдання 1, приклад 1



Код завдання 1, приклад 2



Скріншот виконання коду, приклад 1

Ця програма визначає машинний епсилон (або машинний нуль) для різних типів з плаваючою комою в C: float, double та long double. Машинний епсилон – це найменше число, яке, будучи доданим до 1, дає результат більший за 1. Ось кроки програми:

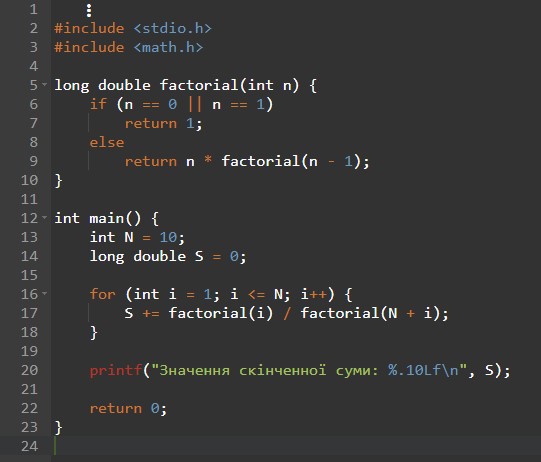
1. Ініціалізує precision як float зі значенням 1.0 і зменшує його вдвічі в циклі while, доки a (яке є precision + 1.0) не стане більшим за 1.
2. Виводить кількість ітерацій, необхідних для досягнення цієї умови, яка представляє кількість ділень на 2, і виводить машинний нуль для float.
3. Повторює цей процес для double та long double.

В результаті програма виводить машинний нуль для кожного типу, що є важливим для розуміння точності обчислень з плаваючою комою в комп’ютерних програмах.

**Завдання 8**

**Реалізація:**

Програмна реалізація мовою С



Код завдання 1, приклад 1



Скріншот виконання коду, приклад 1

Ця програма обчислює суму ряду, де кожен член ряду є відношенням факторіалу числа до факторіалу суми цього числа та заданого числа N. Ось що робить програма:

1. Визначає функцію factorial, яка рекурсивно обчислює факторіал числа n.
2. У функції main, ініціалізує N як 10 і S як 0.
3. Використовує цикл for для обчислення суми S, додаючи до неї відношення factorial(i) / factorial(N + i) для i від 1 до N.
4. Виводить обчислену суму S з точністю до 10 знаків після коми.

Факторіал числа n - це добуток всіх натуральних чисел від 1 до n. Наприклад, факторіал 5 (5!) дорівнює 5 \* 4 \* 3 \* 2 \* 1 = 120. [Функція factorial використовує рекурсію для обчислення факторіалу, що означає, що вона викликає сама себе зі зменшеним на 1 аргументом до тих пір, поки n не стане 0 або 1](https://www.geeksforgeeks.org/c-program-for-factorial-of-a-number/).

## Контрольні питання

1. Призначення операторів циклу.

**Призначення операторів циклу** - використовуються для багаторазового виконання блоку коду до тих пір, поки задана умова є істинною або поки не буде досягнуто певної кількості ітерацій.

1. Конструкція оператора циклу while.

**Конструкція оператора циклу while**:

while(умова) {

// код, який повторюється

}

Цикл while виконує блок коду, поки умова істинна.

1. Конструкція оператора циклу do- while.

**Конструкція оператора циклу do-while**:

do {

// код, який повторюється

} while(умова);

Цикл do-while виконує блок коду один раз, а потім продовжує виконання, поки умова істинна.

1. Конструкція оператора циклу for.

**Конструкція оператора циклу for**:

for(ініціалізація; умова; інкремент/декремент) {

// код, який повторюється

}

Цикл for використовується для виконання блоку коду певну кількість разів.

1. Поясніть призначення виразів у конструкції циклу for.

**1. Призначення виразів у конструкції циклу for**:

* **Ініціалізація**: встановлює початкове значення лічильника циклу.
* **Умова**: визначає умову продовження виконання циклу.
* **Інкремент/Декремент**: змінює значення лічильника після кожної ітерації циклу.

Ці елементи дозволяють точно контролювати кількість ітерацій циклу for.

Висновок:

Лабораторна робота дозволила мені освоїти роботу з циклами в мові C. Виконання різноманітних завдань, таких як робота з температурними конвертаціями та створення трикутника, допомогло покращити мої навички програмування. Також, завдання з оцінкою "машинного нуля" розширило моє розуміння впливу циклічних операторів на виконання програми.

Початок форми