

Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Механіко-технологічний факультет

ЗВІТ
ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ № 9
з навчальної дисципліни
Базові методології та технології програмування
**“ РЕАЛІЗАЦІЯ ПРОГРАМНИХ МОДУЛІВ РОЗГАЛУЖЕНИХ ТА
ІТЕРАЦІЙНИХ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ ”**

ВИКОНАЛА
студентка академічної групи КІ-22-2
Тітарова А. А.

ПЕРЕВІРИВ
викладач кафедри кібербезпеки
та програмного забезпечення
Собінов Олександр Георгійович

Тема: Реалізація програмних модулів розгалужених та ітераційних обчислювальних процесів

Мета роботи полягає у набутті ґрунтовних вмінь і практичних навичок реалізації технології модульного програмування, застосування операторів C/C++ арифметичних, логічних, побітових операцій, умови, циклів та вибору під час розроблення статичних бібліотек, заголовкових файлів та програмних засобів у кросплатформовому середовищі Code:Blocks.

Завдання:

ВАРІАНТ 21

— ЗАДАЧА 9.1 —

Вхід: статистичні дані чисельності населення області: загальна чисельність населення області станом на 1 січня ц.р. та кількість громадян, які станом на 1 число поточного місяця ц.р. знялись з реєстрації (вибули), зареєструвались на постійне місце проживання (прибули), народились, вмерли.

Вихід: чисельність населення області станом на 1 число поточного місяця та інформація про зменшення/збільшення кількості мешканців області у період з 1 січня ц.р. до 1 числа поточного місяця ц.р.

— ЗАДАЧА 9.2 —

Вхід: зафіксовані в Кропивницькому показники швидкості переміщення повітряних мас (вітру, м/сек) щогодини.

Вихід: найменша швидкість вітру (в балах Бофорта), зафіксована продовж доби.

БАЛ БОФОРТА	ШВИДКІСТЬ ВІТРУ, м/сек	ДІЯ ВІТРУ
0	< 0.3	Відсутність вітру. Дим піднімається прямовисно. Листя дерев нерухомі.
1	0.3 – 1.5	Дим «пливе». Флюгер не обертається.
2	1.6 – 9.4	Рух повітря відчувається обличчям. Шелестить листя. Флюгеро обертається
3	9.4 – 5.4	Тріпоче листя, хитаються дрібні гілки. Майорять прапори.
4	5.5 – 7.9	Хитаються тонкі гілки дерев. Вітер піднімає пил та шматки паперу.
5	8.0 – 10.7	Хитаються великі гілки. На воді з'являються хвилі.
6	10.8 – 13.8	Хитаються великі гілки
7	13.9 – 17.1	Хитаються невеликі стовбури дерев. На морі здіймаються хвилі, пінається
8	17.2 – 20.7	Ламаються гілки дерев. і важко йти проти вітру.
9	20.8 – 24.4	Невеликі руйнування. Зриває покрівлі, руйнує димарі.
10	24.5 – 28.4	Значні руйнування. Деревя виринаються з корінням
11	28.5 – 32.6	Великі руйнування
12	≥ 32.7	Приводить до спустошень

— ЗАДАЧА 9.3 —

Вхід: натуральне число N від 0 до 500700.

Вихід: якщо біт D_7 числа N рівний 0, кількість двійкових нулів у ньому, інакше — кількість двійкових нулів*.

*під час підрахунку кількості бінарних 0 або 1 рекомендовано використати тернарний оператор « ? : ».

— ЗАДАЧА 9.4 —

За введеним користувачем символом “d” викликається `s_calculation()`, “g” — функція задачі 9.1, “h” — функція задачі 9.2, “j” — функція задачі 9.3; якщо користувач вводить інші символи, вони ігноруються, при чому видається звуковий сигнал про помилкове введення. Після цього, якщо користувач за запитом додатка вводить символ “k”, “K” або “к”, відбувається вихід з програми, інакше — виконання програми повторюється.



У випадку, якщо параметром i або результатом функції є дані нестандартного типу (наприклад, складового), то такий **тип варто реалізувати у заголовковому файлі**.

Задача 9.1

```
#include <iostream>

using namespace std;

int main() {
    int population, deaths, births, arrivals, departures;

    cout << "Enter the population of the region on January 1: ";
    cin >> population;

    cout << "Enter the number of deaths from January 1 to present day: ";
    cin >> deaths;

    cout << "Enter the number of births from January 1 to present day: ";
    cin >> births;

    cout << "Enter the number of arrivals from January 1 to present day: ";
    cin >> arrivals;

    cout << "Enter the number of departures from January 1 to present day: ";
    cin >> departures;

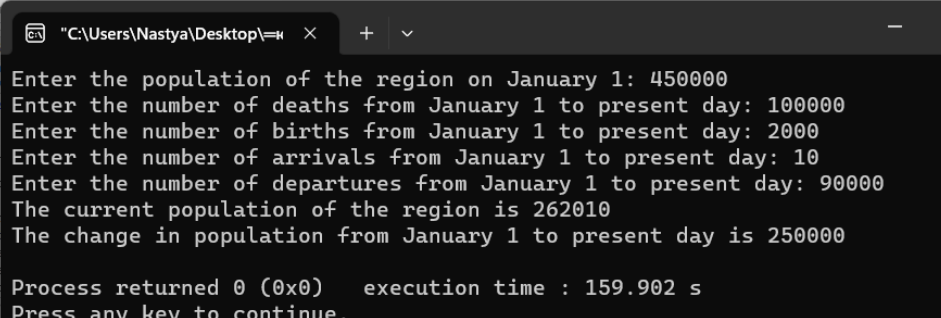
    population += births - deaths + arrivals - departures;

    cout << "The current population of the region is " << population;
    cout << "The change in population from January 1 to present day is " <<
(population - arrivals + departures - deaths - births);

    return 0;
```

}

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main() {
6      int population, deaths, births, arrivals, departures;
7
8      cout << "Enter the population of the region on January 1: ";
9      cin >> population;
10
11     cout << "Enter the number of deaths from January 1 to present day: ";
12     cin >> deaths;
13
14     cout << "Enter the number of births from January 1 to present day: ";
15     cin >> births;
16
17     cout << "Enter the number of arrivals from January 1 to present day: ";
18     cin >> arrivals;
19
20     cout << "Enter the number of departures from January 1 to present day: ";
21     cin >> departures;
22
23     population += births - deaths + arrivals - departures;
24
25     cout << "The current population of the region is " << population;
```



```
Enter the population of the region on January 1: 450000
Enter the number of deaths from January 1 to present day: 100000
Enter the number of births from January 1 to present day: 2000
Enter the number of arrivals from January 1 to present day: 10
Enter the number of departures from January 1 to present day: 90000
The current population of the region is 262010
The change in population from January 1 to present day is 250000

Process returned 0 (0x0)   execution time : 159.902 s
Press any key to continue.
```

Задача 9.2

```
#include <iostream>
```

```
using namespace std;
```

```
const int DAY_HOURS = 24;
```

```
const int BFT_SCALE = 12;
```

```
int main() {
```

```
    float wind_speed, min_speed = 1000.0;
```

```
    for (int hour = 0; hour < DAY_HOURS; hour++) {
```

```
        cout << "Enter wind speed for hour " << hour+1 << ": ";
```

```
        cin >> wind_speed;
```

```

int bft = 0;

while (wind_speed >= (bft + 1) * 0.3) {
    bft++;
}

if (bft < min_speed) {
    min_speed = bft;
}

}

cout << "The minimum wind speed in Beaufort scale is: " << min_speed <<
endl;

return 0;

}

```

The screenshot shows a C++ program in a code editor and its output in a terminal. The code defines a Beaufort scale calculation for 24 hours. It uses a while loop to increment the Beaufort scale (bft) until the wind speed is less than the current minimum. The output shows the wind speed for each hour and the final minimum Beaufort scale value of 13.

```

main.cpp x
1 #include <iostream>
2
3 using namespace std;
4
5 const int DAY_HOURS = 24;
6 const int BFT_SCALE = 12;
7
8 int main() {
9     float wind_speed, min_speed = 1000.0;
10    for (int hour = 0; hour < DAY_HOURS; hour++) {
11        cout << "Enter wind speed for hour " << hour+1 << ": ";
12        cin >> wind_speed;
13
14        int bft = 0;
15        while (wind_speed >= (bft + 1) * 0.3) {
16            bft++;
17        }
18
19        if (bft < min_speed) {
20            min_speed = bft;
21        }
22    }
23
24    cout << "The minimum wind speed in Beaufort scale is: " << min_speed
25
26    return 0;
27 }
28
29

```

```

Enter wind speed for hour 1: 4
Enter wind speed for hour 2: 8
Enter wind speed for hour 3: 55
Enter wind speed for hour 4: 555
Enter wind speed for hour 5: 5255
Enter wind speed for hour 6: 455
Enter wind speed for hour 7: 5
Enter wind speed for hour 8: 5
Enter wind speed for hour 9: 25
Enter wind speed for hour 10: 83
Enter wind speed for hour 11: 236
Enter wind speed for hour 12: 26
Enter wind speed for hour 13: 869
Enter wind speed for hour 14: 178
Enter wind speed for hour 15: 4441
Enter wind speed for hour 16: 466
Enter wind speed for hour 17: 235
Enter wind speed for hour 18: 588989
Enter wind speed for hour 19: 55
Enter wind speed for hour 20: 664
Enter wind speed for hour 21: 55
Enter wind speed for hour 22: 55
Enter wind speed for hour 23: 66
Enter wind speed for hour 24: 6666
The minimum wind speed in Beaufort scale is: 13

Process returned 0 (0x0)   execution time : 51.331 s
Press any key to continue.

```

Задача 9.3

```
#include <iostream>
```

```

int main() {
    unsigned int N;
    std::cin >> N;

```

```
bool D1 = (N >> 1) & 1;

int zero_count = 0;
if (D1 == 0) {
    for (int i = 0; i < 32; i++) {
        if ((N & (1 << i)) == 0) {
            zero_count++;
        }
    }
}
else {
    for (int i = 0; i < 32; i++) {
        if ((N & (1 << i)) != 0) {
            zero_count++;
        }
    }
}

std::cout << zero_count << std::endl;

return 0;
}
```

```
1 #include <iostream>
2
3 int main() {
4     unsigned int N;
5     std::cin >> N;
6
7     bool D1 = (N >> 1) & 1;
8
9     int zero_count = 0;
10    if (D1 == 0) {
11        for (int i = 0; i < 32; i++) {
12            if ((N & (1 << i)) == 0) {
13                zero_count++;
14            }
15        }
16    }
17    else {
18        for (int i = 0; i < 32; i++) {
19            if ((N & (1 << i)) != 0) {
20                zero_count++;
21            }
22        }
23    }
24
25    std::cout << zero_count << std::endl;
26 }
```

54884545554545
32
Process returned 0 (0x0) execution time : 3.395 s
Press any key to continue.

Задача 9.4

```
#include <iostream>
```

```
#include <cstring>
```

```
using namespace std;
```

```
void g_calculation();
```

```
void h_calculation();
```

```
void j_calculation();
```

```
void s_calculation()
```

```
{
```

```
}
```

```
int main()
{
    char input;

    do {
        cout << "Enter 'd', 'g', 'h', or 'j': ";
        cin >> input;

        switch (input) {
            case 'd':
                s_calculation();
                break;
            case 'g':
                g_calculation();
                break;
            case 'h':
                h_calculation();
                break;
            case 'j':
                j_calculation();
                break;
            default:
                cout << "\aError input! Enter 'd', 'g', 'h', or 'j'." << endl;
                continue;
        }

        cout << "Enter 'k' to exit, or anything else to continue: ";
        cin >> input;

    } while (input != 'k' && input != 'K' && input != '\k');
```



```
    return 0;
}
```

Відповіді на контрольні запитання

1. Яким оператором C/C++ можливо повноцінно замінити тернарний оператор? Відповідь обґрунтуйте й доведіть експериментально.

Тернарний оператор можливо замінити оператором if-else. Наприклад, замість "a ? b : c" можна написати "if (a) { b } else { c }". Це обґрунтовано тим, що обидва оператори дають однаковий результат, і у більш складних випадках if-else може бути більш зрозумілим для інших програмістів. Експериментально це можна перевірити наступним чином:

```
#include <iostream>

using namespace std;

int main() {
    int a = 1, b = 2, c = 3;
    int result1 = a ? b : c;
    int result2;
    if (a) {
        result2 = b;
    } else {
        result2 = c;
    }
    cout << "Result 1: " << result1 << endl;
    cout << "Result 2: " << result2 << endl;
    return 0;
}
```

2. Що в програмуванні розуміють під пріоритетом виконання операцій та асоціативністю?

Пріоритет виконання операцій визначає порядок виконання операцій виразу. Асоціативність визначає напрямок виконання операцій з однаковим пріоритетом. Наприклад, вираз "a + b * c" має пріоритет множення вище, тому спочатку буде виконано операцію "b * c", а потім додано до "a".

Асоціативність логічних та арифметичних операторів зазвичай лівоасоціативна, тобто вони виконуються зліва направо. Експериментально це можна перевірити таким чином:

```
#include <iostream>

using namespace std;

int main() {
    int a = 2, b = 3, c = 4;
    int result1 = a + b * c;
    int result2 = (a + b) * c;
    cout << "Result 1: " << result1 << endl;
    cout << "Result 2: " << result2 << endl;
    return 0;
}
```

3. Яку область видимості мають змінні, оголошені в тілі циклу або умови (вибору)? Відповідь обґрунтуйте та доведіть експериментально.

У мові програмування C++, змінні, оголошені в тілі циклу або умови (вибору), мають область видимості, яка обмежується циклом або умовою, і не можуть бути використані поза ними. Це означає, що змінні, оголошені в тілі циклу або умови, не будуть доступні за межами цього циклу або умови, іншими словами, вони будуть локальними змінними, які будуть існувати тільки в межах циклу або умови. Для демонстрації цього факту, можна написати наступний код на C++:

```
#include <iostream>

int main() {
    for (int i = 0; i < 5; i++) {
        int j = i * 2;
        std::cout << "i: " << i << ", j: " << j << std::endl;
    }

    std::cout << "i: " << i << std::endl;
```

```
std::cout << "j: " << j << std::endl;
```

```
return 0;
```

```
}
```

У цьому коді ми оголошуємо змінну *i* в тілі циклу `for`, а змінну *j* також в тілі циклу. Після завершення циклу ми намагаємося вивести значення цих змінних за межами циклу. Під час компіляції цього коду, отримуємо помилку компіляції з таким повідомленням:

```
error: 'i' was not declared in this scope
```

```
std::cout << "i: " << i << std::endl;
```

^

```
error: 'j' was not declared in this scope
```

```
std::cout << "j: " << j << std::endl;
```

^

Це підтверджує, що змінні *i* та *j* не доступні за межами циклу, оскільки вони існують тільки в межах тіла циклу. Отже, можна зробити висновок, що змінні, оголошені в тілі циклу або умови, мають область видимості, яка обмежується циклом або умовою, і не можуть бути використані поза ними.

4. Якою є асоціативність операцій арифметичних, логічних, логічних порозрядних, інкрементна, декрементна, тернарної та порівняння в мові програмування C/C++?

Асоціативність операцій визначає порядок виконання операцій з однаковим пріоритетом. У мові програмування C/C++ асоціативність операторів може бути лівоасоціативною, правоасоціативною або не асоціативною.

Арифметичні оператори:

Асоціативність операцій додавання (+) та віднімання (-) лівоасоціативна.

Асоціативність операцій множення (*) та ділення (/) лівоасоціативна.

Асоціативність операції залишку від ділення (%) правоасоціативна.

Логічні оператори:

Асоціативність операції логічного І (&&) та логічного АБО (||) лівоасоціативна.

Логічні порозрядні оператори:

Асоціативність операцій логічного І (&) та логічного АБО (|) лівоасоціативна.

Інкремент/декремент:

Асоціативність операцій інкремента (++) та декремента (--) правоасоціативна.

Тернарний оператор:

Не має асоціативності, бо складається з трьох операндів.

Оператор порівняння:

Не має асоціативності, бо порівнює два операнди.

Варто зауважити, що у випадку з операторами з однаковим пріоритетом, але з різною асоціативністю, порядок виконання операцій визначається асоціативністю.

5. Перелічіть випадки, за яких доцільно використовувати тернарний оператор C/C++, й наведіть приклад його запису.

Тернарний оператор в C/C++ зазвичай використовують для зменшення кількості коду, або коли треба швидко записати короткий умовний вираз. Ось декілька випадків, коли доцільно використовувати тернарний оператор:

Присвоєння залежно від умови:

```
int x = (a > b) ? a : b;
```

Додавання певної умови до рядка або повідомлення:

```
std::string message = "You have " + std::to_string(count) + " item" + ((count == 1) ? "" : "s") + " in your cart.";
```

Перевірка на наявність нульового вказівника:

```
int length = (strPtr != nullptr) ? strlen(strPtr) : 0;
```

Перевірка на парність чисел:

```
bool isEven = (num % 2 == 0) ? true : false;
```

Вивід на екран різних повідомлень залежно від умови:

```
std::cout << ((value > 0) ? "Positive" : "Negative") << std::endl;
```

У таких випадках тернарний оператор дозволяє зменшити кількість коду та зробити програму більш зрозумілою. Але потрібно пам'ятати, що якщо умовний вираз занадто складний, то може бути краще використати звичайний if-else

6. Яке значення міститиме змінна cnt після виконання наступної інструкції: cnt--; ?

Після виконання інструкції cnt--, змінна cnt буде зменшена на 1. Це є скороченням запису для виразу cnt = cnt - 1.

7. Чим константна змінна, оголошена за допомогою кваліфікатора типів `const`, відрізняється від змінної? Сформулюйте правило, коли змінну варто оголошувати саме константною.

Константна змінна, оголошена з використанням ключового слова `const` в мові програмування, не може бути змінена після ініціалізації. Це означає, що коли змінна оголошується як константна, то її значення не може бути змінено за допомогою присвоєння. Константні змінні корисні, коли потрібно забезпечити сталість деяких даних у програмі і уникнути помилок, пов'язаних з їхньою зміною. Крім того, використання констант може полегшити читання і розуміння коду, оскільки вони вказують на те, що деякі значення залишаються незмінними на протязі виконання програми. Зазвичай варто оголошувати змінну як константну, якщо її значення не повинно змінюватися після ініціалізації, і коли зміна її значення може призвести до помилок в програмі. Наприклад, константні змінні можна використовувати для оголошення математичних констант, таких як числа π та експонента, які не змінюються в ході виконання програми. Константні змінні також можуть використовуватися для оголошення розмірів масивів, які відомі на момент компіляції програми.

8. Яких типів можуть бути операнди логічних операторів C/C++?

В мові C/C++ операндами логічних операторів можуть бути тільки вирази логічного типу (`bool`) або вирази, які можуть бути автоматично приведені до логічного типу. Такі вирази вважаються істинними, якщо їх значення відмінне від нуля, і хибними, якщо їх значення рівне нулю. Також можуть бути використані логічні оператори для виконання операцій з бітами. В такому випадку операндами можуть бути будь-які цілі числа (`char`, `short`, `int`, `long`), але результатом буде ціле число зі значенням 0 або 1. Наприклад, у такому виразі використовуються логічні оператори з бітами для перевірки, чи є біт на позиції 3 у змінній `x` рівний 1:

```
int x = 10; // бінарне представлення: 1010
```

```
bool is_bit_3_set = (x & (1 << 3)) != 0; // результат: false
```

У цьому випадку, якщо біт на позиції 3 в змінній `x` був би рівний 1, результатом би була істина (`true`).

9. Яке значення міститиме змінна `cnt` при: `bool cnt = !!0`; ?

У мові програмування C/C++, значення змінної `cnt` буде `false`. Оператор `!!` називається подвійним запереченням і застосовується для перетворення значення на тип `bool`. Якщо вихідне значення було рівним нулю, то результат буде `false`, в іншому випадку - `true`. Таким чином, `!!0` поверне значення `false`, яке буде присвоєне змінній `cnt` типу `bool`.

10. Сформулюйте правило запису виразу ініціалізації у циклах з параметром (`for`) C++.

Вираз ініціалізації у циклах з параметром (for) C++ складається з трьох частин, які розділяються крапкою з комою (;):

Оголошення та ініціалізація змінної (необов'язково).

Умова виконання циклу.

Вираз, який виконується після кожної ітерації циклу.

Наприклад, якщо потрібно пройти по елементах масиву `arr` типу `int`, то можна записати наступний цикл:

```
for (int i = 0; i < size; i++) {  
    // дії, що повторюються для кожного елемента масиву  
}
```

У цьому прикладі:

Оголошується змінна цілого типу `i` та ініціалізується значенням 0.

Встановлюється умова виконання циклу, що перевіряє, чи значення змінної `i` менше за розмір масиву `size`.

Вираз, який виконується після кожної ітерації циклу, збільшує значення змінної `i` на одиницю. Зверніть увагу, що оголошення змінної в циклі є необов'язковим і можна використовувати змінні, що були оголошені раніше.