

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Кафедра КІТАМ
Дисципліна «Програмування»

Звіт

з лабораторної роботи №3
«Умовні оператори мови C++»

Виконала:
ст.гр. АКТСІз-23-1
Рацебарська А.Д.

Прийняла:
доц. Максимова С.С.,

Харків 2023

3. УМОВНІ ОПЕРАТОРИ МОВИ C++

2.1 Мета роботи:

Вивчити особливості використання умовних операторів if і switch.

2.2 Теоретичні відомості:

Умовні оператори мови C++

Умовні оператори використовуються для контролю потоку виконання програми на основі певних умов. Основні умовні оператори в C++:

if та else: Як було зазначено, ці оператори використовуються для виконання різних дій залежно від того, чи є певна умова істинною чи ні.

else if: Це використовується для перевірки декількох умов одна за одною. Якщо умова в if не виконується, то програма може перевірити додаткову умову в else if.

switch: Дозволяє виконувати різні дії на основі значення змінної. Це корисно, коли потрібно порівняти одну змінну з багатьма можливими значеннями.

Інші важливі оператори в C++:

Циклічні оператори: Це включає **for, while, do-while**, які використовуються для виконання блоку коду багаторазово до тих пір, поки виконується певна умова.

Оператор goto: Хоча використання goto у більшості випадків не рекомендується, оскільки воно може ускладнити читання та розуміння коду, він дозволяє перейти до певного рядка в програмі.

Оператори зворотного зв'язку (break та continue): break використовується для негайного виходу з циклу або switch-блоку. Continue перериває поточну ітерацію циклу та переходить до наступної ітерації.

Тернарний оператор (?:): Це укорочена форма if-else, яка дозволяє виконувати присвоєння або повертати значення на основі умови в одному рядку.

2.3 Хід роботи:

Варіант 2

Написати програму на C++ для:

1. перевірки попадання введеного числа в діапазон від -2 до 2 (Оператор if);
2. перекладу введеного символу від А до F в нижній регістр (Оператор case).

```
first_academic_year > C++ 3_laboratory.cpp > main()

13  #include <iostream>
14  #include <cctype> // Для використання функції tolower()
15  using namespace std;
16
17  int main() {
18      // Частина 1: Перевірка числа
19      double num;
20      cout << "Введіть число для перевірки: ";
21      cin >> num;
22
23      if (num >= -2 && num <= 2) {
24          cout << "Число " << num << " знаходиться в діапазоні від -2 до 2.\n";
25      } else {
26          cout << "Число " << num << " не знаходиться в діапазоні від -2 до 2.\n";
27      }
28
29      // Частина 2: Переклад символу в нижній регістр
30      char ch;
31      cout << "Введіть символ від A до F: ";
32      cin >> ch;
33
34      switch (ch) {
35          case 'A' :
36          case 'B' :
37          case 'C' :
38          case 'D' :
39          case 'E' :
40          case 'F' :
41              cout << "Символ у нижньому регістрі: " << char(tolower(ch)) << "\n";
42              break;
43          default:
44              cout << "Введено неправильний символ.\n";
45      }
46
47      return 0;
48  }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
● alonaratsebarska@MacBook-Pro-Alona first_academic_year % g++ -o 3_laboratory 3_laboratory.cpp
● alonaratsebarska@MacBook-Pro-Alona first_academic_year % ./3_laboratory
Введіть число для перевірки: 1
Число 1 знаходиться в діапазоні від -2 до 2.
Введіть символ від A до F: C
Символ у нижньому регістрі: c
● alonaratsebarska@MacBook-Pro-Alona first_academic_year % ./3_laboratory
Введіть число для перевірки: 5
Число 5 не знаходиться в діапазоні від -2 до 2.
Введіть символ від A до F: G
Введено неправильний символ.
○ alonaratsebarska@MacBook-Pro-Alona first_academic_year %
```

***Пояснення виразу `char(tolower(ch))`

Вираз `char(tolower(ch))` в кодї C++ використовується для перетворення символу `ch` у нижній регістр, використовуючи функцію `tolower()` з бібліотеки `cctype`. Вказівка `char` перед `tolower(ch)` є **явним приведенням типу**, яке забезпечує, що результат, повернутий функцією `tolower()`, буде оброблено як символ типу `char`.

Тобто це важливо бо функція `tolower()` працює з цілочисельними значеннями (`int`), а не безпосередньо з символами (`char`). Коли символ подається на вхід `tolower()`, він автоматично перетворюється в цілочисельне значення, що відповідає його ASCII-коду.

Після того, як `tolower()` обробляє символ, вона повертає цілочисельний результат, який відповідає ASCII-коду символу у нижньому регістрі. Явне приведення типу за допомогою `char` перетворює це цілочисельне значення назад у символ типу `char`. Це важливо для коректного виведення символу, а не його числового ASCII-коду. Отже, використання `char(tolower(ch))` гарантує, що результатом буде символ у нижньому регістрі, який можна вивести або використати далі як символьне значення.

ВИСНОВКИ

У процесі виконання цієї лабораторної роботи було засвоїла важливі аспекти використання умовних операторів в мові програмування C++. Отримала практичний досвід роботи з операторами **if**, **else**, та **switch**, що є ключовими для контролю потоку виконання програм. Навчилася використовувати оператор **if** для перевірки попадання числа в певний діапазон та оператор **switch** для обробки вводу символів і переведення їх у нижній регістр. Також засвоїла як використовувати функцію **tolower()** для перетворення символів і правильно застосовувати явне приведення типів у C++. Умовні оператори є фундаментальними у мові C++ і дозволяють писати ефективні програми, здатні приймати рішення на основі даних, введених користувачем або отриманих під час виконання програми.