

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Кафедра КІТАМ
Дисципліна «Програмування»

Звіт

з лабораторної роботи №4
«Умовні оператори мови C++»

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

Прийняла:
доц. Максимова С.С.,

Харків 2023

4. ОПЕРАТОРИ ЦИКЛІВ МОВИ C++

2.1 Мета роботи:

Вивчити особливості використання операторів циклу `while`, `for` і `do while`.

2.2 Теоретичні відомості:

Оператори циклів є невід'ємною частиною мови програмування C++, дозволяючи виконувати повторення певних дій. Вони не тільки спрощують код, але й надають потужні інструменти для ефективного управління ітераціями. У C++ існують три основних типи циклів: **for**, **while** та **do-while**.

1. [Цикл for](#): Це найбільш структурований цикл. Цей цикл часто використовується, коли точно відома кількість ітерацій. Його синтаксис складається з трьох частин: ініціалізація, умова продовження циклу та інкрементування/декрементування. Цикл **for** є особливо корисним для перебору елементів у масивах та контейнерах. Наприклад: `for(int i = 0; i < n; i++) { /* код */ }`. Цікавим є той факт, що усі три компоненти циклу **for** не є обов'язковими, що дозволяє створювати цикли з різними формами поведінки.
2. [Цикл while](#): Цей цикл використовується для виконання блоку коду, поки задана умова є істинною. Цикл **while** краще підходить для ситуацій, де кількість ітерацій не відома заздалегідь. Він перевіряє умову перед кожною ітерацією, що може впливати на продуктивність у випадку великої кількості ітерацій.
3. [Цикл do-while](#): Цей цикл схожий на цикл **while**, але з однією ключовою відмінністю: умова перевіряється після виконання ітерації, гарантуючи, що блок коду буде виконаний хоча б один раз. Це корисно в сценаріях, де необхідно спочатку виконати дію, а потім перевірити, чи слід продовжувати. Наприклад, це може бути використано для меню програми, яке має показатися принаймні один раз.

Нюанси та ефективність

- [Безкінечні цикли](#): Цикли можуть бути безкінечними, якщо умова виходу з них ніколи не виконується. Це може бути як помилкою, так і навмисним рішенням для створення постійно виконуваного коду, наприклад, в основному циклі гри або серверної програми.
- [Керування циклами](#): У C++ існують оператори **break** та **continue**, які дозволяють більш гнучко управляти виконанням циклів. **Break** негайно завершує цикл, тоді як **continue** завершує поточну ітерацію і переходить до наступної.
- [Вибір між for та while](#): Хоча обидва ці цикли можуть використовуватися замінно, важливо вибрати **for** для ситуацій, де кількість ітерацій заздалегідь відома або

має чітку межу. Цикл **while** ідеально підходить для ситуацій, де ітерації повинні продовжуватися до зміни певного стану або виконання умови.

2.3 Хід роботи:

Варіант 2

Написати програму на C++ :

- 1.Знайти за допомогою while $f(x) = kx + b$, $x = 1, 2, \dots, 100$ (Оператор while та for)
- 2.Написати програму введення довільних символів до тих пір, поки не буде введений символ q (Оператор do while)

```
13 #include <iostream>
14 using namespace std;
15
16 int main() {
17     double k, b;
18     cout << "Введіть k: ";
19     cin >> k;
20     cout << "Введіть b: ";
21     cin >> b;
22
23     // 1. Знаходження  $f(x) = kx + b$ 
24     // Використання оператора while
25     cout << "1 Part. While Loop" << endl;
26     int x = 1;
27     while (x <= 100) {
28         double f_x = k * x + b;
29         // Доп перевірка яка робить перенос на нову строку після 10 ітерацій (для зручності зчитування інформ.
30         if (x % 10 == 0) {
31             cout << "f(" << x << ") = " << f_x << endl;
32             // Доп перевірка задля пропуску рядка при завершенні циклу (зручність зчитування в консолі
33             if (x == 100) {
34                 cout << "\n";
35             }
36         } else {
37             cout << "f(" << x << ") = " << f_x << "; ";
38         }
39         x++;
40     }
41
42     // Альтернативний варіант за допомогою оператора for
43     cout << "1 Part. For-Loop" << endl;
44     for (int x = 1; x <= 100; x++) {
45         double f_x = k * x + b;
46         if (x % 10 == 0) {
47             cout << "f(" << x << ") = " << f_x << endl;
48             if (x == 100) {
49                 cout << "\n";
50             }
51         } else {
52             cout << "f(" << x << ") = " << f_x << "; ";
53         }
54     }
55
56     // 2. Введення символів до 'q' (оператор do while)
57     cout << "2 Part. Do While Loop" << endl;
58     char ch;
59     do {
60         cout << "Введіть символ (введіть 'q' для виходу): ";
61         cin >> ch;
62     } while (ch != 'q');
63
64     cout << "Введено символ 'q', програма завершується." << endl;
65
66     return 0;
67 }
68
```

```
ПРОБЛЕМЫ    ВЫХОДНЫЕ ДАННЫЕ    КОНСОЛЬ ОТЛАДКИ    ТЕРМИНАЛ    ПОРТЫ    zsh - first_academic

● alonaratsebarska@MacBook-Pro-Alona nure_programming % cd first_academic_year
● alonaratsebarska@MacBook-Pro-Alona first_academic_year % g++ -o 4_laboratory 4_laboratory.cpp
● alonaratsebarska@MacBook-Pro-Alona first_academic_year % ./4_laboratory
Введіть k: 3
Введіть b: 5
1 Part. While Loop
f(1) = 8; f(2) = 11; f(3) = 14; f(4) = 17; f(5) = 20; f(6) = 23; f(7) = 26; f(8) = 29; f(9) = 32; f(10) = 35
f(11) = 38; f(12) = 41; f(13) = 44; f(14) = 47; f(15) = 50; f(16) = 53; f(17) = 56; f(18) = 59; f(19) = 62; f(20) = 65
f(21) = 68; f(22) = 71; f(23) = 74; f(24) = 77; f(25) = 80; f(26) = 83; f(27) = 86; f(28) = 89; f(29) = 92; f(30) = 95
f(31) = 98; f(32) = 101; f(33) = 104; f(34) = 107; f(35) = 110; f(36) = 113; f(37) = 116; f(38) = 119; f(39) = 122; f(40) = 125
f(41) = 128; f(42) = 131; f(43) = 134; f(44) = 137; f(45) = 140; f(46) = 143; f(47) = 146; f(48) = 149; f(49) = 152; f(50) = 155
f(51) = 158; f(52) = 161; f(53) = 164; f(54) = 167; f(55) = 170; f(56) = 173; f(57) = 176; f(58) = 179; f(59) = 182; f(60) = 185
f(61) = 188; f(62) = 191; f(63) = 194; f(64) = 197; f(65) = 200; f(66) = 203; f(67) = 206; f(68) = 209; f(69) = 212; f(70) = 215
f(71) = 218; f(72) = 221; f(73) = 224; f(74) = 227; f(75) = 230; f(76) = 233; f(77) = 236; f(78) = 239; f(79) = 242; f(80) = 245
f(81) = 248; f(82) = 251; f(83) = 254; f(84) = 257; f(85) = 260; f(86) = 263; f(87) = 266; f(88) = 269; f(89) = 272; f(90) = 275
f(91) = 278; f(92) = 281; f(93) = 284; f(94) = 287; f(95) = 290; f(96) = 293; f(97) = 296; f(98) = 299; f(99) = 302; f(100) = 305

1 Part. For-Loop
f(1) = 8; f(2) = 11; f(3) = 14; f(4) = 17; f(5) = 20; f(6) = 23; f(7) = 26; f(8) = 29; f(9) = 32; f(10) = 35
f(11) = 38; f(12) = 41; f(13) = 44; f(14) = 47; f(15) = 50; f(16) = 53; f(17) = 56; f(18) = 59; f(19) = 62; f(20) = 65
f(21) = 68; f(22) = 71; f(23) = 74; f(24) = 77; f(25) = 80; f(26) = 83; f(27) = 86; f(28) = 89; f(29) = 92; f(30) = 95
f(31) = 98; f(32) = 101; f(33) = 104; f(34) = 107; f(35) = 110; f(36) = 113; f(37) = 116; f(38) = 119; f(39) = 122; f(40) = 125
f(41) = 128; f(42) = 131; f(43) = 134; f(44) = 137; f(45) = 140; f(46) = 143; f(47) = 146; f(48) = 149; f(49) = 152; f(50) = 155
f(51) = 158; f(52) = 161; f(53) = 164; f(54) = 167; f(55) = 170; f(56) = 173; f(57) = 176; f(58) = 179; f(59) = 182; f(60) = 185
f(61) = 188; f(62) = 191; f(63) = 194; f(64) = 197; f(65) = 200; f(66) = 203; f(67) = 206; f(68) = 209; f(69) = 212; f(70) = 215
f(71) = 218; f(72) = 221; f(73) = 224; f(74) = 227; f(75) = 230; f(76) = 233; f(77) = 236; f(78) = 239; f(79) = 242; f(80) = 245
f(81) = 248; f(82) = 251; f(83) = 254; f(84) = 257; f(85) = 260; f(86) = 263; f(87) = 266; f(88) = 269; f(89) = 272; f(90) = 275
f(91) = 278; f(92) = 281; f(93) = 284; f(94) = 287; f(95) = 290; f(96) = 293; f(97) = 296; f(98) = 299; f(99) = 302; f(100) = 305

2 Part. Do While Loop
Введіть символ (введіть 'q' для виходу): f
Введіть символ (введіть 'q' для виходу): a
Введіть символ (введіть 'q' для виходу): y
Введіть символ (введіть 'q' для виходу): w
Введіть символ (введіть 'q' для виходу): q
Введено символ 'q', програма завершується.
● alonaratsebarska@MacBook-Pro-Alona first_academic_year %
```

ВИСНОВКИ

У процесі виконання лабораторної роботи оволоділа важливими навичками роботи з циклами **for**, **while** та **do-while**, що є важливими для розробки програм, здатних виконувати повторювані дії та обробляти послідовності даних. Зокрема, навчилася ефективно використовувати цикл **for** для випадків, коли кількість ітерацій заздалегідь відома, та цикл **while** для ситуацій, де ітерації залежать від виконання певних умов. Оволоділа застосуванням циклу **do-while** для гарантії виконання тіла циклу хоча б один раз, що є корисним у створенні інтерактивних консольних програм. У рамках лабораторної роботи було розроблено програму, що демонструє практичне використання цих циклів для розрахунку функції з заданими параметрами та введення даних користувачем до заданої умови.