# МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Національний університет “Львівська політехніка”**



**Інститут післядипломної освіти**

**Кафедра програмного забезпечення систем**

**ЗВІТ**

**Про виконання лабораторної роботи №2**

**«Реалізація алгоритмів з розгалуженням в мові С++»**

**з дисципліни «Основи програмування»**

Виконав:

слухач групи ПЗС-11

Гринчук Тарас

Прийняв:

доц. Макар В.М.

« »\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2013 р.

∑ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

ЛЬВІВ – 2013

**Тема роботи**: Реалізація алгоритмів з розгалуженням в мові С++.

**Мета роботи:** навчитися програмувати на мові С++ розгалужені обчислювальні процеси.

## 1. ТЕОРЕТИЧНІ ВІДОМОСТІ

1. Алгоритм – це зрозумілі й точні вказівки виконавцю (комп’ютеру)

здійснити певну послідовність дій (обчислювальних операцій) для розв’язання поставленої задачі. розрізняють

такі форми запису алгоритмів:

* словесний запис алгоритму (розмовна мова);
* псевдокод (структурно-стилізована мова);
* блок-схема (мова графічних символів);
* комп’ютерна програма (мова програмування високого рівня).

1. В теорії алгоритмів доведено, що будь-який алгоритм може бути побудований з використанням всього лише трьох базових конструкцій, а саме з ***конструкцій слідування, розгалуження та повторення***.
2. В блок-схемах використовують такі блоки: блок виконання дій, логічний блок, блок модифікації, блок виклику функції, блок з’єднання, блок початку/кінця.
3. Оператор умови **if/else**, який має таку синтаксичну структуру:

**if (<умова>) <оператор1>; else <оператор2>;**

Спочатку обчислюється **<умова>**, і якщо вона є

істинною (має ненульове значення), то виконується **<оператор1>**, а якщо умова є хибною (має нульове значення), то виконується **<оператор2>**.

Проексперементуємо з наступною програмою, задаючи додатнє, від’ємне та нульове значення n.

#include <iostream>

#include <conio.h>

void main() {

using namespace std;

int n;

cout<<"Enter n:";

cin>>n;

if (n) cout<<"TRUE"<<endl; // така умова в С++ є коректною!

if (n>0) cout<<"n is positive"<<endl;

if (n<0) cout<<"n is negative"<<endl;

if (n==0) cout<<"n is zero"<<endl;

cout<<"Value of operation > is "<<(n>0)<<endl;

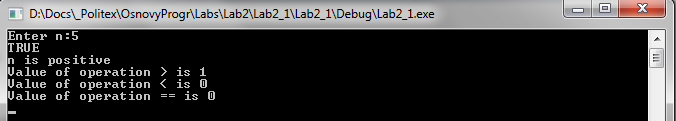
cout<<"Value of operation < is "<<(n<0)<<endl;

cout<<"Value of operation == is "<<(n==0)<<endl;

\_getch();

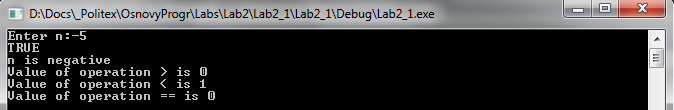
}

* n=5 (рис. 1.1). Умова «if (n)» виконується, оскільки n є не нулем, в іншому випадку цей оператор не виконається. Умова (n>0) рівна 1, (n<0) та (n==0) рівні 0, що ми й бачимо на екрані.



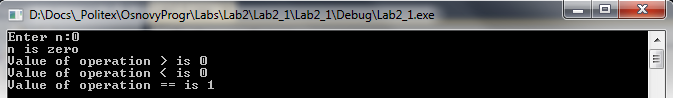
(Рис. 1.1)

* n=-5 (рис. 1.2). Як і в попередньому випадку умова «if (n)» виконується, оскільки n є не нулем. Також отримуємо підтвердження, що число є від’ємним, а умова (n<0) рівна 1.



(Рис. 1.2)

* n=0 (рис. 1.3). Умова «if (n)» не виконується, тому що її вираз рівний нулю. Як слід було очікувати, умова (n==0) рівна 1, а на екрані з’явилось повідомлення, що n є нулем.



(Рис. 1.3)

1. У ролі **<оператор1>** та **<оператор2>** може бути будь-який допустимий оператор мови С++, в тому числі й сам оператор умови. У такому випадку отримуємо вкладені структури **if/else**, які дозволяють реалізувати розгалуження обчислювального процесу більше ніж у двох напрямках. Тут слід пам’ятати, що кожне службове слово **else** відноситься до першого перед ним слова **if**. Так, наприклад, при заданих початкових значеннях x=1; y=-1, після виконання оператора:

if (x>0) if (y>0) z=1; else z=2;

змінна z буде мати значення 2. Якщо ми хочемо, щоб гілка else z=2; відносилася не до умови if (y>0), а до if (x>0), то треба цей оператор записати так:

if (x>0) {if (y>0) z=1;} else z=2;

За правилами мови С++ **<оператор1>** та **<оператор2>** мають бути структурно одним єдиним оператором. Якщо користувачеві необхідно виконати в цих місцях кілька операторів, то їх треба взяти в операторні дужки { }, тобто зробити ці декілька операторів складеним оператором.

Приклади програм з розгалуженням.

Приклад 1. Задано три цілих числа a,b,c в діапазоні від 1000 до 9999. Знайти те з них сума цифр якого найбільша.

#include <iostream>

#include <conio.h>

void main() {

using namespace std;

int a,b,c,s,max,d; // оголошення змінних

cout<<"Enter a,b,c:";

cin>>a>>b>>c; // введення значення змінних

// перевірка на відповідність введених значень вказаному діапазону

if ((a<1000 || a>9999)||(b<1000 || b>9999)||(c<1000 || c>9999))

// хоча б одне число не попадає в діапазон

cout<<"All numbers must be between 1000 and 9999"<<endl;

else { // всі числа задані коректно, опрацьовуємо їх

s=a/1000+ a/100%10 + a/10%10 + a%10; // сума цифр числа a

max=s; // змінна max буде містити значення максимальної суми цифр

// змінна d буде містити саме число сума цифр якого максимальна

d=a;

s=b/1000+ b/100%10 + b/10%10 + b%10; // сума цифр числа b

// перевірка чи сума цифр числа b більша за поточне максимальне значення

if (s>max) {

max=s; // якщо так, то переприсвоюємо змінній max нове значення

d=b; // і запам'ятовуємо число b в змінній d

}

s=c/1000+ c/100%10 + c/10%10 + c%10; // сума цифр числа c

// перевірка чи сума цифр числа c більша за поточне макси мальне значення

if (s>max) {

max=s; // якщо так, то переприсвоюємо змінній max нове значення

d=c; // і запам'ятовуємо число c в змінній d

}

// тепер в змінній d маємо саме число, а в змінній max суму його цифр

// виводимо результат

cout<<"Number with max sum of digits:"<<d<<endl;

cout<<"Sum of digits is "<<max<<endl;

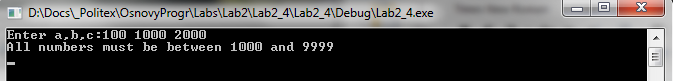
}

\_getch();

}

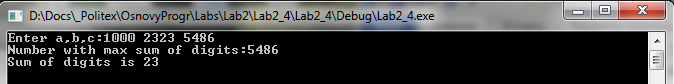
Запустимо програму на виконання зі значеннями змінних a,b,c: 100; 1000; 2000 (рис. 1.4). Як бачимо, виконалася умова: if ((a<1000 || a>9999)||(b<1000 || b>9999)||(c<1000 || c>9999))

і виконання перейшло на наступний оператор після блоку **if else**: \_getch();.



(Рис. 1.4)

Запустимо програму на виконання зі значеннями змінних a,b,c: 1000; 2323; 5486 (рис. 1.5). Оскільки, всі числа трьохзначні, виконання переходить в блок **else**. Обчислюється сума цифр числа a, яка буде рівною 1 (1+0+0+0), відповідно значення змінної max стає 1, а змінної d: 1000. Наступним оператором, виконується обчислення суми цифр числа b, яке стає рівним 10(2+3+2+3). Умова if (s>max) є істинною(10>1), відповідно після цього кроку: max=10; d=2323. Обчислюєм суму цифр числа c, яке буде рівна 23(5+4+8+6). Оскільки 23>10 (умова if (s>max) є істинною), отримуємо: max=23; d=5486. Наступним оператором, виводяться ці результати на екран.



(Рис. 1.5)

Приклад 2. Задано чотири цілих числа a,b,c,d. Відомо, що одне з них відрізняється від трьох інших, які однаковими. Знайти і вивести це число.

#include <iostream>

#include <conio.h>

void main() {

using namespace std;

int a,b,c,d; // оголошення змінних

cout<<"Enter a,b,c,d:";

cin>>a>>b>>c>>d; // введення значення змінних

if (b==c && c==d && a!=b)

cout<<"Number that differed is "<<a;

else if (a==c && c==d && b!=a)

cout<<"Number that differed is "<<b;

else if (a==b && a==d && a!=c)

cout<<"Number that differed is "<<c;

else if (a==b && a==c && a!=d)

cout<<"Number that differed is "<<d;

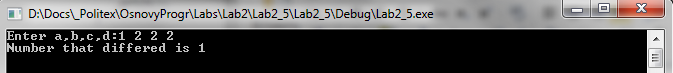
else

cout<<"Three of numbers must be equal!"<<endl;

\_getch();

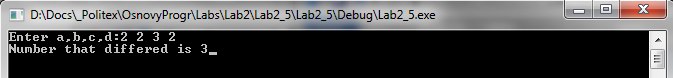
}

Запустимо програму на виконання зі значеннями змінних a,b,c,d: 1; 2; 2; 2 (рис. 1.6). Як бачимо виконується умова: if (b==c && c==d && a!=b) і виконання програми переходить на оператор \_getch().



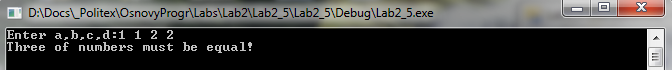
(Рис. 1.6)

Запустимо програму на виконання зі значеннями змінних a,b,c,d: 2; 2; 3; 2 (рис. 1.7). Умова: if (b==c && c==d && a!=b) не виконується. Наступною перевіряється умова if (a==c && c==d && b!=a), яка також є хибною. Виконання переходить на наступну гілку else, Оператором якої є умова: if (a==b && a==d && a!=c), яка є істинною. Відповідно виконується оператор виведення, який належить даній умові і виконання переходить на команду \_getch().



(Рис. 1.7)

Запустимо програму на виконання зі значеннями змінних a,b,c,d: 1; 1; 2; 2 (рис. 1.8), щоб проексперементувати з випадком, коли три числа не будуть рівними. Як бачимо всі умови if є хибними і виконання переходить на останній за вкладенням блок else, в якому виконується оператор виводу повідомлення: «Three of numbers must be equal!"».



(Рис. 1.8)

1. Особливітю виразу, яким задається умова в операторі **if**, є те, що результатом його виконання є або ненульове значення (означає, що умова є істинною або умова виконується), або значення 0 (означає, що умова є хибною або умова не виконується). У мові С++ відсутній спеціальний логічний тип, якому відповідають значення типу TRUE і FALSE (яку мові Паскаль). Аналогом значення TRUE у С++ є будь-яке ненульове значення, а значення FALSE – 0.
2. Вираз мовою С++, який перевіряє чи перша і остання цифри заданого

трьохзначного цілого числа k рівні:

if (int(k/100) == k%10)

int() – являється оператором приведення виразу k/100 до типу int, таким чином ми отримуємо першу цифру трьохзначного числа k. За допомогою операції k%10, отримуємо остачу від ділення k на 10 – це і буде остання цифра числа k.

1. Складений оператор має таку структуру:

{ <оператор1>; <оператор2>; ... <операторN> }

Складений оператор вживається у програмі завжди, якщо за синтаксисом мови С++ наступний оператор може бути тільки єдиним, а користувачеві потрібно виконати певну послідовність операторів.

У ролі прикладу використання складеного оператора в структурі **if/else**, розглянемо задачу обчислення виразу: з врахуванням обмеження на область визначення кореня.

#include <iostream>

#include <conio.h>

#include <math.h>

void main() {

using namespace std;

double x,y;

cout<<"Enter x:"<<endl;

cin>>x;

if (x<0)

cout<<"x<0. Negative numbers don't have square roots !!!"<<endl;

else {

y=0.75\*sqrt(x)-0.5\*pow(4,1.0/3);

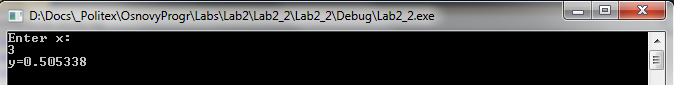
cout<<"y="<<y<<endl;

}

\_getch();

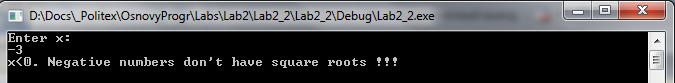
}

Спробуємо запустити програму на виконаня зі значенням х=3 (рис 1.9). Отримаємо результат виразу у=0,505338, що вказує на те, що виконались оператори блоку else.



(Рис 1.9)

Якщо значення х ввести від’ємним, отримаємо повідомлення, що не можливо отримати квадратний корінь з від’ємних чисел (рис. 1.10). Таким чином, бачимо, що виконався оператор, що відповідає умові if (x<0), натомість блоку else не виконався.



(Рис 1.10)

1. Мова С++ має умовну операцію ?:, яка є схожою на структуру if/else. Ця умовна операція є єдиною тернарною операцією, тобто такою, що має три операнда. Ці операнди разом з самою умовною операцією утворюють умовний вираз, який має такий вигляд:

<умова> ? <вираз1> : <вираз2> .

Тут, перший операнд є умовою, другий операнд містить значення умовного виразу в тому випадку, якщо умова є істинною, а третій операнд рівний значенню умовного виразу, якщо умова є хибною. Наприклад, оператор виведення

cout<< (grade>=50 ? “ЗДАВ” : “НЕ ЗДАВ”);

містить умовний вираз, значення якого рівне рядку “ЗДАВ”, якщо умова grade>=50 виконується, і рівне рядку “НЕ ЗДАВ”, якщо ця умова не виконується.

1. Операцій в мові С++, які завжди мають результат типу ‘істина’/’не істина’ є *логічні операції*. Цих операцій в С++ є три: операція логічного множення ***&&*** (логічне І), операція логічного додавання ***||*** (логічне АБО) та операція логічного заперечення ***!*** (логічне НЕ). Операції логіного множення і додавання працюють, як показано в табл. 1.1.

|  |  |  |
| --- | --- | --- |
| && | 0 | Не нуль |
| 0 | 0 | 0 |
| Не нуль | 0 | 1 |

|  |  |  |
| --- | --- | --- |
| || | 0 | Не нуль |
| 0 | 0 | 1 |
| Не нуль | 1 | 1 |

Таблиця 1.1

Логіка операції логічного заперечення така: !0=1 та !1=0.

1. *Оператор вибору* **switch**, має такий синтаксис:

**switch** (switch\_expression)

{ **case** constant1: statement1; [**break**;]

……………………

**case** constanti: statementi; [**break**;]

…………………......

**case** constantN: statementN; [**break**; ]

[**default**: statementN+1; ] }

Оператор **switch** виконується так. Спочатку обчислюється значення виразу **switch\_expression**. Тип значення повинен бути одним із цілих - char, int, unsigned int, long int і long unsigned. Обчислене значення зрівнюється зі значеннями *констант вибору* або *константних виразів* **constant1**, ... ,

**constantN**. Заборонено використовувати в якості константи вибору змінну. При співпадінні значення **switch\_expression** із constanti виконується оператор **statementi**. Потім керування передається на оператор відразу після switch, якщо в i-й гілці є присутнім оператор **break** (оператор break здійснює негайний вихід з оператора switch). У протилежному випадку

виконуються оператори в гілках i+1, i+2 і так далі доти, поки в них не зустрінеться оператор **break** або не буде виконаний оператор **statement N+1**.

Якщо значення switch\_expression не збіглося з жодною з констант **constanti,... , constantN,** виконується оператор у гілці, позначеній **default.** При її відсутності виконується наступний після **switch** оператор.

Для прикладу запишемо програмо, яка моделює роботу світлофора.

#include <iostream>

#include <conio.h>

void main() {

using namespace std;

char ch;

cout<<"Enter letter:";

cin>>ch;

switch (ch) {

case 'r': cout<<"Stop! RED"<<endl; break;

case 'y': cout<<"Attention! YELLOW"<<endl; break;

case 'g': cout<<"Go! GREEN"<<endl; break;

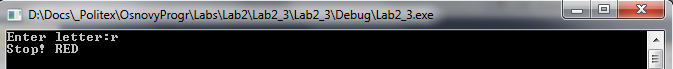
default: cout<<"Wrong letter!"<<endl;

}

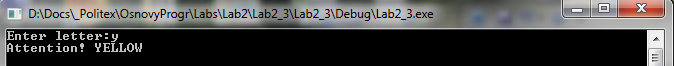
\_getch();

}

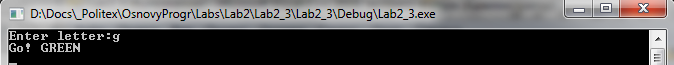
Запустимо програму на виконнання зі значеннями вхідного символа r (рис 1.11), y (рис 1.12), g (рис 1.13), j (рис 1.14). Як бачимо, виконався тільки блок операторів, що відповідає відповідній літері, в останньому випадку виконався блок default.



(Рис 1.11)



(Рис 1.12)



(Рис 1.13)



(Рис 1.14)

1. Для зміни послідовного виконання операторів використовується оператор безумовного переходу **goto**. Структура оператора goto:

goto <мітка>; M <мітка>: <оператор>;

Мітка задається за правилами запису ідентифікаторів. Якщо строго слідувати правилами структурного програмування, то нема ніякої необхідності в операторі безумовного переходу **goto**, який нещадно критикується в літературі по С++. Однак існують окремі випадки, коли його використання може принести певну користь, наприклад при роботі з вкладеними циклами, коли потрібно у внутрішньому циклі перервати його разом зі зовнішнім, чи перейти на наступну ітерацію зовнішнього циклу.

## ХІД РОБОТИ

**Індивідуальне завдання (варіант 5).** Задані дійсні числа а1, b1, с1, а2, b2, с2. Надрукувати координати точки перетину прямих, які описуються рівняннями: або повідомлення про те, що прямі не перетинаються.

Для знаходження точок перетину, можна скористатися методом Крамера. Відповідно до формули Крамера координати (x,y) точки перетину цих прямих обчислюються:

Якщо знаменник рівний нулю, то прямі паралельні або збігаються. Отже, перш за все, нам треба ввести параметри двох прямих, при цьому слід звернути увагу, що параметри a1 та b1 не можуть одночасно бути рівними нулю, тому що в цьому випадку, вираз перестане бути рівнянням прямої, відповідно, якщо користувач ввів значення: a1=0 та b1=0 – слід вивести відповідне повідомлення та припинити виконання програми. Аналогічні дії повторимо для a2 та b2.

Наступним кроком є обчислення знаменника та перевірка його на рівність нулю. Якщо ця умова справджується, то виводимо відповідне повідомлення, в протилежному випадку – обчислюємо x та y та виводимо їх на екран.

Блок-схема для даного алгоритму має вигляд:

a1,b1,c1

a1==0&&b1==0

1

-

+

a2,b2,c2

a2==0&&b2==0

+

2

z=a1\*b2-a2\*b1

-

z==0

-

+

x=-(c1\*b2-c2\*b1)/z

3

4

1

x, y

2

3

y=-(a1-a2\*c1)/z

Не перетинаються

4

Програмна реалізація даного алгоритму на мові С++ має вигляд:

#include <iostream>

#include <conio.h>

void main() {

using namespace std;

float a1,b1,c1,a2,b2,c2,x,y,z; // оголошення змінних

cout<<"Line 1. a1,b1,c1:";

cin>>a1>>b1>>c1; // введення значення змінних a1,b1,c1

if (a1==0 && b1==0) cout<<"It's not a line !"; //Це не пряма

else {

cout<<"Line 2. a2,b2,c2:";

cin>>a2>>b2>>c2; // введення значення змінних a2,b2,c2

if (a2==0 && b2==0) cout<<"It's not a line !"; //Це не пряма

else {

z=a1\*b2-a2\*b1; //обчислення знаменника

if (z==0) cout<<"Lines haven't point of intersection";

else {

x=-(c1\*b2-c2\*b1)/z; //обчислимо х

y=-(a1-a2\*c1)/z; //обчислимо у

cout<<"x="<<x<<"; y="<<y; //виведення результату

}

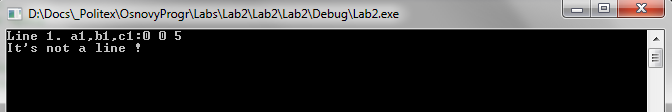
}

}

\_getch();

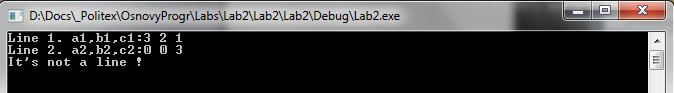
}

Запустимо програму на виконання з параметрами першої прямої a1=0; b1=0; c1=5 (рис. 2.1). В результаті того, що умова є if (a1==0 && b1==0) істинною, на екран з’являється відповідне повідомлення і виконнання програми переходить на наступний оператор після умови: \_getch().



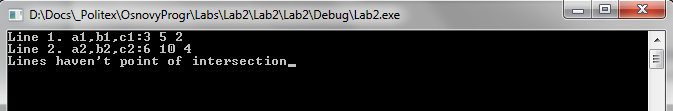
(Рис 2.1)

Запустимо програму на виконання з параметрами a1=3; b1=2; c1=1; a2=0; b2=0; c2=3 (рис. 2.2). Отримаємо аналогічне повідомлення, як і в попередньому випадку, тільки після введення параметрів другої прямої. Це свідчить, про те, що виконалась умова if (a2==0 && b2==0) і виконання перейшло на оператор \_getch().



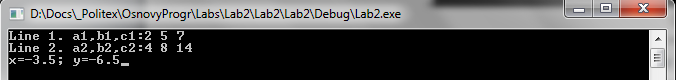
(Рис 2.2)

Спробуємо обчислити задачу з наступними параметрами: a1=3; b1=5; c1=2; a2=6; b2=10; c2=4 (рис. 2.3). В цей раз ми отримали повідомлення, що прямі не мають точки перетину. Виконалась умова if (z==0), адже 3\*10-6\*5=0.



(Рис 2.3)

Запустимо програму з параметрами: a1=2; b1=5; c1=7; a2=4; b2=8; c2=14 (рис. 2.4). Отримуємо ркзультат: х=-3,5; у=-6,5; Оскільки умова if (z==0) є хибною, виконується блок else, який відповідає цій умові і далі виконання програми передається на оператор \_getch().



(Рис 2.4)

## ВИСНОВКИ

На даній лабораторній роботі я навчився програмувати на мові С++ алгоритми з розгалуженнями. Зокрема, правильно реалізовувати, як і прості умови, так і вкладені блоки умов, а також ознайомився з роботою умовного оператора «?». Отримав практичні і теоретичні навики в представленні алгоритмів у вигляді блок-схем.