Условные операторы в языке C++

АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ

Цели и задачи лекции

Цель лекции – изучить используемые в языке программирования C++ конструкции для принятия решений

Виды условных операторов

В языке программирования C++ используются несколько конструкций для принятия решений:

- оператор **if**;
- оператор **switch**;
- условный оператор **?** (тернарный оператор)

Оператор if

Общая форма записи оператора if:

if (expression)

program statement;

В операторе **if** используется результат вычисления условия, заключенного в круглые скобки, на основе которого принимается решение. Результат вычисления условия **expression** может быть арифметическим или логическим. Если результат выполнения условия **expression** будет истинным, то возможно выполнить несколько утверждений типа **program statement**

Оператор if

пример:

```
if (expression)
{
program1 statement1;
program2 statement2;
...
}
```

Конструкция if–else

Общая форма записи конструкции if–else:

```
if (expression)
program1 statement1;
else
program2 statement2;
```

Если выполняется условие expression, то будет выполняться фрагмент программы program1 statement1, в противном случае будет выполняться program2 statement2.

Конструкция if–else

Каждое из утверждений может быть множественным. В таком случае применяются фигурные скобки:

```
if (expression)
{
program1 statement1;
program2 statement2;
...
}
```

Конструкция if–else

```
else
{
program33 statement33;
program34 statement34;
...
}
```

Конструкция if–else if–else if–...–else

Форма записи конструкции if–else if–else if–...–else:

```
if (expression1)
program1 statement1;

else if (expression2)
program2 statement2;
```

Конструкция if–else if–else if–...–else

```
else if (expression3)
program3 statement3;
...
else
program statement;
```

Конструкция if–else if–else if–...–else

Приведенная конструкция используется для выбора возможных ситуаций, когда проверяются условия expression1, expression2, expression3,... . Соответственно будут выполняться действия program1 statement1, program2 statement2, program3 statement3 и т.д. В случае, когда ни одно из условий не выполняется, выполняются действия, прописанные после оператора else.

Конструкция if–else if–else if–...–else

В случае выполнения множественных действий применяются фигурные скобки для каждого из утверждений:

```
if (expression1)
{
program1 statement1;
...
}

else if (expression2)
```

Конструкция if–else if–else if–...–else

```
else if (expression2)
{
program2 statement2;
...
}

else if (expression3)
{
program3 statement3;
...
}
```

Конструкция if–else if–else if–…–else

```
…

else
{
program statement;
…
}
```

Оператор switch

Общая форма записи оператора **switch**:

```
switch (expression) {
case value1:
program statement;
...
break;

case value2:
program statement;
...
break;
```

Оператор switch

Общая форма записи оператора **switch**:

```
...
case valuen:
program statement;
...
break;
default:
program statement;
...
break;
}
```

Оператор switch

Выражение заключенного в круглые скобки оператора последовательно сравнивается со значениями value1, value2,..., valuen, которые должны быть простыми константами или константными выражениями. В том случае, когда одно из этих значений равно значению, выполняются утверждения, которые следуют за данным значением.

Оператор switch

Утверждение **break** сигнализирует об окончании выполнения утверждений и приводит к выходу из оператора **switch.** Утверждение **break** ставится в конце каждого варианта выбора. Если этого не сделать, то выполнение последовательности утверждений перейдет в следующий вариант выбора и будет выполняться до тех пор, пока не встретится утверждение **break**

Оператор switch

Специальный дополнительный вариант default будет выполнен в том случае, когда не будет найдено ни одного совпадения.

Операторы if и switch той или иной синтаксической конструкции существуют практически во всех языках программирования (в первую очередь языках высокого уровня), и их часто называют операторами ветвления.

Условный оператор ?

В отличие от других операторов языка С, которые могут быть унарными или бинарными, специфический оператор условия является тернарным оператором. Это означает, что у него может быть три операнда.

Общий формат записи оператора условия:

условие ? выражение_1 : выражение_2

Условный оператор ?

Если в результате вычисления условия будет получено значение TRUE (истина, не нуль), то выполняется выражение_1, и результатом выполнения оператора условия будет значение, полученное при вычислении этого выражения. Если в результате вычисления условия будет получено значение FALSE (ложь, т.е. нуль), то выполняется выражение_2, и результатом выполнения оператора условия будет значение, полученное при вычислении выражение_2.

Условный оператор ?

Оператор условия часто описывают как оператор **?**. Тернарный оператор условия **?** наиболее часто используется для присвоения переменной одного из двух значений в зависимости от некоторого условия.

Пример программы

```
#include <stdio.h>
#include <conio.h>
#include <math.h>

int main(void) {
    float a, b, c;
    float D, x1, x2, x;
    printf("\n\t Equation a*x^2 + b*x + c = 0\n");
```

Пример программы

```
printf("\n\t Enter the coefficient a: ");
scanf_s("%f", &a);
printf("\t Enter the coefficient b: ");
scanf_s("%f", &b);
printf("\t Enter the coefficient c: ");
scanf_s("%f", &c);
D = b*b - 4*a*c;
if (D >= 0 && a != 0) {
```

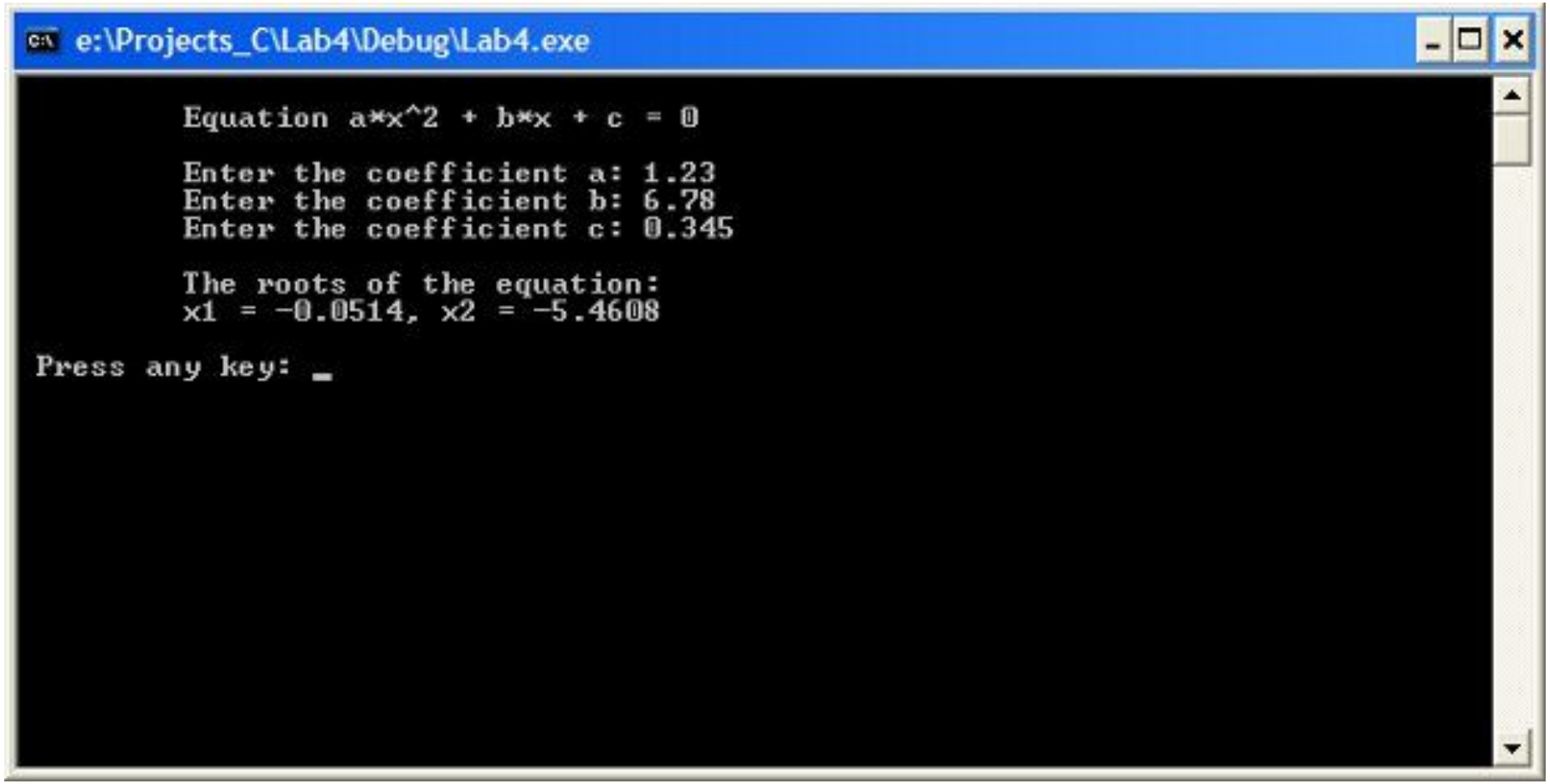
Пример программы

```
x1 = -b/(2*a) + (float)sqrt(D)/(2*a);
x2 = -b/(2*a) - (float)sqrt(D)/(2*a);
 printf("\n\t The roots of the equation:\n\t x1 = %1.4f, x2 = %1.4f\n",
x1, x2);
}
if (D < 0)
    printf("\n\t The roots of complex\n");
```

Пример программы

```
    if (a == 0 && b != 0) {
x = -c/b;
    printf("\n\t As a = %1.0f,\n\t the solution of the equation is: %1.4f\n", a, x); }
    printf("\n Press any key: ");
    _getch();
    return 0;
}
```

Результат работы программы



Спасибо за внимание