4. Логические операции и выражения. Условные операторы. Блок операторов

Логические величины типа bool (которые могут быть либо истиной, либо ложью) возникают, например, в результате применения к численным величинам **операций сравнения**, которые в математической записи имеют вид:

<, ≤, >, ≥, =, ≠

На языке C/C++ они пишутся, соответственно, так:

<, <=, >, >=, ==, !=

Над величинами логического типа определены **логические операции**: отрицание (унарная), конъюнкция и дизъюнкция (бинарные). Записываются они, соответственно, так:

!, &&, ||

и действуют по следующим правилам:

!A

|  |  |
| --- | --- |
| A | !A |
| true | false |
| false | true |

A && B

|  |  |  |
| --- | --- | --- |
| B  A | true | false |
| true | true | false |
| false | false | false |

A || B

|  |  |  |
| --- | --- | --- |
| B  A | true | false |
| true | true | true |
| false | true | false |

Как и прочие операции, операции сравнения и логические операции обладают приоритетом. В языке C/C++ для приоритетов операций имеются 14 градаций. Чтобы не перегружать читателя, мы сейчас перечислим не все, а только те из них, которые будут нами активно использоваться — как сейчас, так и в дальнейшем. Полный список можно найти, например, в [1] или [2]. Группы операций будут перечислены от высшего приоритета к низшим:

|  |  |  |
| --- | --- | --- |
| **Прио­ритет** | **Операция** | **Краткое описание** |
| 1 | *Унарные:* |  |
| ++ | увеличение на 1 |
| -- | уменьшение на 1 |
| ! | логическое отрицание |
| - | унарный минус |
| & | взятие адреса |
| \* | разадресация |
| new | выделение памяти |
| delete | освобождение памяти |
| (*тип*) | преобразование типа |
| 2 | \* | умножение |
| / | деление |
| % | остаток от деления |
| 3 | + | сложение |
| - | вычитание |
| 4 | … | ... |
| 5 | < | меньше |
| <= | меньше или равно |
| > | больше |
| >= | больше или равно |
| 6 | = | равно |
| != | не равно |
| 7 | … | ... |
| 8 | … | ... |
| 9 | … | ... |
| 10 | && | логическое «и» (конъюнкция) |
| 11 | || | логическое «или» (дизъюнкция) |
| 12 | ? : | тернарная условная операция |
| 13 | = | присваивание |
| += | сложение с присваиванием |
| -= | вычитание с присваиванием |
| \*= | умножение с присваиванием |
| /= | деление с присваиванием |
| %= | остаток от деления с присваиванием |
| 14 | , | операция «запятая» (последовательное вычисление) |

В ряде задач необходимо проверять какие-то условия и, в зависимости от этих условий, выполнять те или иные действия. Для программирования таких действий **применяют условные операторы**. Они имеют несколько форм записи.

1. Условный оператор if (полная форма):

if (*выражение*) *оператор\_1*; else *оператор\_2*;

Здесь *выражение* может быть типа арифметического, логического или указателя (см. далее, раздел «Массивы и указатели»). Оператор действует так: вначале вычисляется *выражение*. Если оно не равно 0 (имеет значение true), то выполняется *оператор\_1*, иначе — *оператор\_2*.

*Замечание.* Для большей иллюстративности точки с запятыми мы здесь специально выделили, хотя, формально говоря, они входят в состав соответствующих операторов.

Блок-схема оператора:

…

*оператор\_1*

*выражение*

true

false

*оператор\_2*

…

2. Условный оператор if (сокращённая форма):

if (*выражение*) *оператор\_1*;

Действует аналогично оператору в полной форме, только в случае, когда *выражение* равно 0 (имеет значение false), никакого действия не происходит.

Блок-схема:

…

*выражение*

*оператор\_1*

true

false

…

*Замечание 1*. По синтаксису языка C/C++ *оператор\_1* и *оператор\_2* — это одиночные операторы. Если же на их месте требуется присутствие сразу нескольких операторов, то они должны быть объединены в **блок операторов (составной оператор)**, посредством заключения их в фигурные скобки. Другой вариант — объединить эти действия в один оператор, записав их через запятую (использовать **операцию «запятая»**).

*Замечание 2.* Для операторов if возможна вложенная конструкция, когда один оператор if содержит другой оператор if, а тот может содержать третий if и т. д. В этом случае соответствие else и if определяется по принципу вложенных скобок. Отдельно следует оговорить ситуацию, когда в подобной вложенной цепочке у части операторов if присутствует else. а у другой части — нет (см. Пример 2 ).

*Замечание 3.* Сокращённая форма оператора if используется, когда в полной форме отсутствует *оператор\_2*. Если же отсутствует *оператор\_1*, то в качестве него можно использовать **пустой оператор**, который не производит ни одного действия (но присутствует в программе как оператор) и представляется одиночным символом «точка с запятой»:

if (*выражение*) ; else *оператор\_2*;

Пустой оператор используется чаще всего тогда, когда по синтаксису оператор требуется, а по смыслу — нет.

3. Тернарная условная операция. В ней используются символы ? и : и три операнда-выражения:

*выражение\_1* ? *выражение\_2* : *выражение\_3*;

Действует так. Сначала вычисляется *выражение\_1*. Если оно не равно нулю (истинно), то вычисляется *выражение\_2*, значение которого становится результатом. Если *выражение\_1* равно нулю (ложно), то в качестве результата берётся значение *выражения\_3*.

**Пример 1.** Даны вещественные *x*, *y*. Вычислить величины *s* и *t*:





Для упрощения математических формул в программе будут использованы промежуточные переменные a и b. Для вычисления t реализовано вложение одного оператора if в другой.

#include<stdio.h>  
#include<math.h>  
int main() {

float x,y,s,t,a,b;

printf(”Введите x,y\n”); //запрос на ввод  
 scanf(”%f%f”,&x,&y);

a=fabs(x)+fabs(y);

b=x\*x+y\*y;

if(x<2\*y) s=a; else s=log(b+1); //вычисл. s

if(b<1) t=0.5\*a; else //вычисление b

if(b<=4) t=coz(a); else t=sqrt(b-1);

printf(”s=%f t=%f\n”,s,t);

return 0;

}

Вариант вычисления s с помощью тернарного условного оператора:

x<2\*y ? s=a : s=log(b+1);

Вариант вычисления t с использованием операторов if в краткой форме:

if(b<1) t=0.5\*a;

if(b>=1&&b<=4) t=coz(a);

if(b>4) t=sqrt(b-1);

Следующий «вариант» вычисления t является ошибочным:

if(b<1) t=0.5\*a;

if(b>=1&&b<=4) t=coz(a); else t=sqrt(b-1);

***Упражнение.*** Объяснить, почему.

**Пример 2** (вложенные операторы if, из которых один полный, другой сокращённый). Дана точка на плоскости (*x*, *y*). Если точка принадлежит 1‑му квадранту или нижней полуплоскости, вывести об этом сообщение.

Приведём сначала *ошибочный вариант* решения:

#include <iostream>

using namespace std;

int main() {

double x,y;

cout<<”input x,y\n”; // запрос на ввод

cin>>x>>y;

if(x>0) // вложенные усл. операторы

if(y>0) cout<<”1-й квадрант\n”;

else cout<<”нижняя полуплоскость\n”;

return 0;

}

По логике задачи, else в условном операторе должно относиться к первому (внешнему) if, а второй if должен быть сокращённым. Однако согласно правилам языка, компилятор отнесёт это else ко второму if, и в случае, когда точка принадлежит 3‑му квадранту (относящемуся к нижней полуплоскости), программа не даст никакого ответа. Один из вариантов правильного решения — организовать внутренний if в блок из одного оператора, с помощью фигурных скобок:

if(x>0)

{if(y>0) cout<<”1-й квадрант\n”;}

else cout<<”нижняя полуплоскость\n”;

Другой вариант — сделать внутренний if полным, с помощью пустого оператора:

if(x>0)

if(y>0) cout<<”1-й квадрант\n”; else;

else cout<<”нижняя полуплоскость\n”;

**Пример 3.** Даны три числа. Найти из них наибольшее и наименьшее. В программе используется конструкция блока.

#include<stdio.h>  
int main() {

float x,y,z,max,min;

printf(”Введите x,y,z\n”);

scanf(”%f%f”,&x,&y,&z);

if(x<y){max=y; min=x;} else {max=x; min=y;}

if(z>max) max=z;

if(z<min) min=z;

printf(”max=%f min=%f\n”,max,min);

return 0;}

**Задачи**

1. Даны четыре вещественных числа. Найти произведение двух меньших из них.
2. Даны вещественные *a*, *b*. Найти решения уравнения *ax*=*b* (во всех вариантах: *a*≠0, *a*=0).
3. Даны *a*, *b*. Найти и вывести решения неравенства *ax*<*b* (во всех вариантах).
4. Даны длины трех отрезков *a*, *b*, *c*. Выяснить, существует ли треугольник с такими сторонами, если существует, то вычислить его площадь по формуле Герона: , где . Если не существует, то вывести сообщение «треугольник не существует».
5. Найти вещественные корни уравнения . Задачу сделать в двух вариантах: с *a* ненулевым и с *a* произвольным.
6. Найти вещественные корни биквадратного уравнения *ax*4*+bx*2*+c=*0*.* Если корней нет, то вывести сообщение об этом.
7. Известны площади круга и квадрата. Определить, уместится ли круг в квадрате?
8. Известны площади круга и квадрата. Определить, уместится ли квадрат в круге?
9. Определить, существует ли треугольник со сторонами *,,* и является ли он равнобедренным?
10. Даны *x*, *y* — координаты точки на плоскости. Выяснить, принадлежит ли эта точка второй четверти кольца с центром в начале координат, малым радиусом *r* и большим радиусом *R* (*r* и *R* даны).