Условная инструкция

Все ранее рассматриваемые программы имели линейную структуру: все инструкции выполнялись последовательно одна за одной, каждая записанная инструкция обязательно выполняется. По крайней мере, эти программы могли были сделаны именно так.

Допустим мы хотим по данному числу x определить его абсолютную величину (модуль). Для этого необходимо нарушить линейную логику программы. Программа должна напечатать значение переменной x, если x>0 или же величину -x в противном случае. Линейная структура программы нарушается: в зависимости от справедливости условия x>0 должна быть выведена одна или другая величина. Соответствующий фрагмент программы на C++ имеет вид:

```
int x;
cin >> x;
if (x > 0)
{
    cout << x;
}
else
{
    cout << -x;
}
return 0;
```

В этой программе используется условная инструкция `if` (если). После слова `if` в обязательных круглых скобках указывается проверяемое условие `(x > 0)`. После этого в фигурных скобках идет блок (последовательность) инструкций, который будет выполнен, если условие истинно, в нашем примере это вывод на экран величины `x`. Затем идет слово `else` (иначе) и после него блок инструкций, который будет выполнен, если проверяемое условие неверно, в данном случае будет выведено значение `-x`.

Итак, условная инструкция в C++ имеет следующий синтаксис:

```
if (Условие)
{
    Блок инструкций 1
}
else
{
    Блок инструкций 2
}
```

Блок инструкций 1 будет выполнен, если *Условие* истинно. Если *Условие* ложно, будет выполнен *Блок инструкций 2*.

В условной инструкции может отсутствовать слово `else` и последующий блок. Такая инструкция называется неполным ветвлением. Например, если дано число `x` и мы хотим заменить его на абсолютную величину `x`, то это можно сделать следующим образом:

```
if (x < 0)
{
    x = -x;
}
```

В этом примере переменной `x` будет присвоено значение `-x`, но только в том случае, когда `x<0`.

Вложенные условные инструкции

Внутри условных инструкций можно использовать любые инструкции языка C++, в том числе и условную инструкцию. Получаем вложенное ветвление – после одной развилки в ходе исполнения программы появляется другая развилка. Покажем это на примере программы, которая по данным ненулевым числам x и y определяет, в какой из четвертей координатной плоскости находится точка (x,y):

```
int x, y;
cin >> x >> y;
if (x > 0)
{
    if (y > 0)   // x>0, y>0
    {
        cout << "Первая четверть" << endl;
    }
    else            // x>0, y<0
    {
        cout << "Четвертая четверть" << endl;
    }
}
else
{
    if (y > 0)   // x<0, y>0
    {
        cout << "Вторая четверть" << endl;
    }
    else            // x<0, y<0
    {
        cout << "Третья четверть" << endl;
    }
}
```

В этом примере мы использовали *комментарии* – текст, который компилятор игнорирует. Комментариями в C++ является последовательность символов `//` и весь текст после этого символа до конца строки. Обратите также внимание на отступы в начале строк, используемые для облегчения понимания текста.

Операторы сравнения

В качестве проверяемого условия должно использоваться выражение логического типа `bool`. Переменные логического типа принимают два значения: `true` (истина) и `false` (ложь). Также любое целочисленное выражение можно трактовать, как логическое выражение, при этом нулевое целое число означает ложь, а ненулевое — истину. Таким образом, если вместо условия написать `false` или `0`, то оно будет всегда ложно, если же указать `true`, `1` или любое ненулевое число, то условие будет истинно.

Как правило, в качестве проверяемого условия используется результат вычисления одного из следующих операторов сравнения:

`<`
: Меньше — возвращает `true`, если первый операнд меньше второго.

`>`
: Больше — возвращает `true`, если первый операнд больше второго.

`<=`
: Меньше или равно.

`>=`
: Больше или равно.

`==`
: Равенство. Возвращает `true`, если два операнда равны.

`!=`
: Неравенство. Возвращает `true`, если два операнда неравны.

Например, условие `(x * x < 1000)` означает “значение `x * x` меньше 1000”, а условие `(2 * x != y)` означает “удвоенное значение переменной `x` не равно значению переменной `y`”.

Будьте аккуратны: оператор `==` (два знака равенства) — это проверка на равенство двух выражений, а оператор `=` (один знак равенства) — это присваивание одной переменной значения выражения и использование его в условии оператора ветвления в большинстве случаев является ошибкой.

Рассмотрим эту типичную ошибку на следующем примере:

```
int a, b;
cin >> a >> b;
if (a = b)
{
    cout << "Числа равны" << endl;
}
else
{
    cout << "Числа не равны" << endl;
}
```

Здесь по ошибке вместо операции сравнения `==` использована операция присваивания `=`. Поэтому при любых значениях a и b переменной a будет присвоено значение переменной b, при проверке истинности выражения `a = b`. Но оператор присваивания еще и возвращает значение, поэтому если значение b было ненулевым (а это интерпретируется, как истина), то программа выведет строку "Числа равны", а если нулевым — то строку "Числа не равны". При этом значение переменной `a` может быть вообще любым.

3.3 Логические операторы

Иногда нужно проверить одновременно не одно, а несколько условий. Например, проверить, является ли данное число четным можно при помощи условия `(n % 2 == 0)` (остаток от деления `n` на `2` равен `0`), а если необходимо проверить, что два данных целых числа `n` и `m` являются четными, необходимо проверить справедливость обоих условий: `n % 2 == 0` и `m % 2 == 0`, для чего их необходимо объединить при помощи оператора && (логическое И): `n % 2 == 0 && m % 2 == 0`.

В C++ существуют стандартные логические операторы: логическое И, логическое ИЛИ, логическое отрицание.

Логическое И является бинарным оператором (то есть оператором с двумя операндами: левым и правым) и имеет вид && (два знака амперсанда). Оператор && возвращает `true` тогда и только тогда, когда оба его операнда имеют значение `true`.

Логическое ИЛИ является бинарным оператором и возвращает `true` тогда и только тогда, когда хотя бы один операнд равен `true`. Оператор “логическое ИЛИ” имеет вид `||` (два знака вертикальной черты).

Логическое НЕ (отрицание) является унарным (то есть с одним операндом) оператором и имеет вид `!` (восклицательный знак), за которым следует единственный операнд. Логическое НЕ возвращает `true`, если операнд равен `false` и наоборот.

Пример. Проверим, что хотя бы одно из чисел a или b оканчивается на 0:

```
if (a % 10 == 0 || b % 10 == 0)
```

Проверим, что число a — положительное, а b — неотрицательное:

```
if ( (a > 0) && !(b < 0) )
```

Или можно вместо `!(b < 0)` записать `(b >= 0)`.

Другая типичная ошибка новичков — неверное использование двойных неравенств. Пусть нужно проверить, является ли введенное число ball корректной школьной оценкой, то есть числом от 1 до 5. Правильное решение такое:

```
if (ball >= 1 && ball <= 5)
```

А вот такая запись будет синтаксически верной, но работать будет неправильно:

```
if (1 <= ball <= 5)
```

Почему? Потому что в записи `(1 <= ball <= 5)` два оператора сравнения. Они выполняются слева направо и скобки между ними расставляются так: `( (1 <= ball ) <= 5)`. Сначала выполняется первое сравнение: `(1 <= ball)`. Его результатом будет значение типа `bool`, то есть либо 0, либо 1. А на втором действии результат вычисления предыдущего выражения (то если либо 0, либо 1) сравнивается с числом 5. Поэтому итоговый результат всегда будет истинным, независимо от значения числа `ball`.

Упражнения

!!Задачи A-G считаются по ½ балла!!

A: Максимум двух чисел

Даны два целых числа. Выведите значение наибольшего из них.

B: Какое число больше?

Даны два целых числа. Программа должна вывести число 1, если первое число больше второго, число 2, если второе больше первого или число 0, если они равны.

C: Знак числа

В математике функция `sign(x)` (знак числа) определена так:
`sign(x) = 1`, если `x > 0`,
`sign(x) = -1`, если `x < 0`,
`sign(x) = 0`, если `x = 0`.

Для данного числа `x` выведите значение `sign(x)`.

D: Високосный год

Дано натуральное число. Требуется определить, является ли год с данным номером високосным. Если год является високосным, то выведите `YES`, иначе выведите `NO`.

Напомним, что год является високосным, если его номер кратен 4, но не кратен 100, а также если он кратен 400.

E: Максимум трех чисел

Даны три целых числа. Найдите наибольшее из них (программа должна вывести ровно одно целое число). Какое наименьшее число операторов сравнения (>, <, >=, <=) необходимо для решения этой задачи?

F: Существует ли треугольник?

Даны три натуральных числа a, b, c. Определите, существует ли треугольник с такими сторонами. Если треугольник существует, выведите строку `YES`, иначе выведите строку `NO`.

G: Сколько совпадает чисел

Даны три целых числа. Определите, сколько среди них совпадающих. Программа должна вывести одно из чисел: 3 (если все совпадают), 2 (если два совпадает) или 0 (если все числа различны).

H: Високосный год

Дан номер года (положительное целое число). Определить количество дней в этом году, учитывая, что обычный год насчитывает 365 дней, а високосный — 366 дней. Високосным считается год, делящийся на 4, за исключением тех годов, которые делятся на 100 и не делятся на 400 (например, годы 300, 1300 и 1900 не являются високосными, а 1200 и 2000 — являются).

I: Ход ладьи

Шахматная ладья ходит по горизонтали или вертикали. Даны две различные клетки шахматной доски, определите, может ли ладья попасть с первой клетки на вторую одним ходом. Программа получает на вход четыре числа от 1 до 8 каждое, задающие номер столбца и номер строки сначала для первой клетки, потом для второй клетки. Программа должна вывести `YES`, если из первой клетки ходом ладьи можно попасть во вторую или `NO` в противном случае.

Пример

Ввод	Вывод
1 1 1 8	YES
3 4 4 3	NO

J: Ход короля

Решите аналогичную задачу для короля, который может ходить в клетку, которая является соседней с данной по стороне или углу.

K: Ход слона

Решите аналогичную задачу для слона, который ходит по диагонали.

L: Ход ферзя

Решите аналогичную задачу для ферзя, который ходит как ладья и слон (то есть по горизонтали, вертикали или диагонали).

M: Ход коня

Решите аналогичную задачу для коня, который ходит буквой “Г” — на две клетки по вертикали в любом направлении и на одну клетку по горизонтали, или наоборот.

N: Шахматная доска

Заданы две клетки шахматной доски. Если они покрашены в один цвет, то выведите слово `YES`, а если в разные цвета – то `NO`.

O: Шоколадка

Шоколадка имеет вид прямоугольника, разделенного на n×m долек. Шоколадку можно один раз разломить по прямой на две части. Определите, можно ли таким образом отломить от шоколадки ровно k долек.

Программа получает на вход три числа: n, m, k и должна вывести одно из двух слов: `YES` или `NO`.

Пример

Ввод	Вывод
3 2 4	YES
2 3 1	NO

P: Линейное уравнение

Даны числа a и b. Решите **в целых числах** уравнение $ax+b=0$. Выведите все решения этого уравнения, если их число конечно, выведите слово `NO`, если решений нет, выведите слово `INF`, если решений бесконечно много.

Пример

Ввод	Вывод
1 1	-1

2 1	NO

Q: Коровы

Для данного числа n<100 закончите фразу “На лугу пасется...” одним из возможных продолжений: “n коров”, “n корова”, “n коровы”, правильно склоняя слово “корова”. Программа должна вывести введенное число n и одно из слов (на латинице): `korov`, `korova` или `korovy`. Между числом и пробелом должен стоять ровно один пробел.

Пример

Ввод	Вывод
1	1 korova
2	2 korovy
5	5 korov

R: Яша плавает в бассейне

Яша плавал в бассейне размером N×M метров и устал. В этот момент он обнаружил, что находится на расстоянии x метров от одного из длинных бортиков (не обязательно от ближайшего) и y метров от одного из коротких бортиков. Какое минимальное расстояние должен проплыть Яша, чтобы выбраться из бассейна на бортик?

Программа получает на вход числа N, M, x, y. Программа должна вывести число метров, которое нужно проплыть Яше до бортика.

Пример

Ввод	Вывод
10 25 7 8	3

S: Тип треугольника

Даны три стороны треугольника a, b, c. Определите тип треугольника с заданными сторонами. Выведите одно из четырех слов: `rectangular` для прямоугольного треугольника, `acute` для остроугольного треугольника, `obtuse` для тупоугольного треугольника или `impossible`, если треугольника с такими сторонами не существует.

T: Билеты на метро — 1

Билет на одну поездку в метро стоит 15 рубля, билет на 10 поездок стоит 125 рублей, билет на 60 поездок стоит 440 рублей. Пассажир планирует совершить n поездок. Определите, сколько билетов каждого вида он должен приобрести, чтобы суммарное количество оплаченных поездок было не меньше n, а общая стоимость приобретенных билетов —

минимальна.

Программа получает на вход одно число n и должна вывести три целых числа, равных необходимому количеству билетов на 1, на 10, на 60 поездок.

U*: Билеты на метро — 2

Решите предыдущую задачу при наличии следующих билетов: 1 поездка — 15 рублей, 5 поездок — 70 рублей, 10 поездок — 125 рублей, 20 поездок — 230 рублей, 60 поездок — 440 рублей.

Программа получает на вход одно число n и должна вывести пять целых чисел: количество билетов на 1, 5, 10, 20, 60 поездок, которое необходимо приобрести. Если для какого-то данного n существует несколько способов приобретения билетов одинаковой суммарной стоимости, необходимо вывести ту комбинацию билетов, которая дает большее число поездок.

V*: Узник замка Иф

За многие годы заточения узник замка Иф проделал в степе прямоугольное отверстие размером D×E. Замок Иф сложен из кирпичей, размером A×B×C. Определите, сможет ли узник выбрасывать кирпичи в море через это отверстие, если стороны кирпича должны быть параллельны сторонам отверстия.

Программа получает на вход числа A, B, C, D, E и должна вывести слово `YES` или `NO`.

Пример

Ввод	Вывод
2 1 3 2 2	YES
1 2 3 1 1	NO

W*: Коробки

Есть две коробки, первая размером $A_1 \times B_1 \times C_1$, вторая размером $A_2 \times B_2 \times C_2$. Определите, можно ли разместить одну из этих коробок внутри другой, при условии, что поворачивать коробки можно только на 90 градусов вокру ребер.

Программа получает на вход числа A_1, B_1, C_1, A_2, B_2, C_2. Программа должна вывести одну из следующих строчек:
`Boxes are equal`, если коробки одинаковые,
`The first box is smaller than the second one`, если первая коробка может быть положена во вторую,
`The first box is larger than the second one`, если вторая коробка может быть положена в первую,
`Boxes are incomparable`, во всех остальных случаях.

Пример

Ввод	Вывод

1 2 3 3 2 1	Boxes are equal
3 4 5 2 4 6	Boxes are incomparable

X*: Складирование ноутбуков

На склад, который имеет форму прямоугольного параллелепипеда, привезли ноутбуки, упакованные в коробки. Каждая коробка также имеет форму прямоугольного параллелепипеда. По правилам хранения коробки с ноутбуками должны быть размещены на складе с выполнением следующих двух условий:
1. Стороны коробок должны быть параллельны сторонам склада.
2. Коробку при помещении на склад разрешается расположить где угодно (с выполнением предыдущего условия), в том числе на другой коробке, но все коробки должны быть ориентированы одинаково (т.е. нельзя одну коробку расположить “стоя”, а другую —“лежа”)

Напишите программу, которая по размерам склада и размерам коробки с ноутбуком определит максимальное количество ноутбуков, которое может быть размещено на складе. Программа получает на вход шесть натуральных чисел. Первые три задают длину, высоту и ширину склада. Следующие три задают соответственно длину, высоту и ширину коробки с ноутбуком. Программа должна вывести одно число — максимальное количество ноутбуков, которое может быть размещено на складе.

Пример

Ввод	Вывод
100 200 300 1 2 3	1000000
100 200 300 3 2 1	1000000
100 100 1 2 2 2	0
7 7 7 3 3 3	8

Y*: Аншлаг, аншлаг!

В одном из популярных кинотеатров все сеансы проходят с аншлагом, поэтому все места в зале всегда заняты. К сожалению, расстояние между рядами в кинозале маленькое, и зрители, пробираясь перед началом фильма к своим местам, вынуждены спотыкаться о ноги уже сидящих. Заходя в зал, зритель думает, с какой стороны ряда он будет пробираться к своему месту (с левой или с правой), и выбирает сторону так, чтобы споткнуться о меньшее число людей. В случае равенства зритель выбирает ту сторону, к которой его место ближе.

Вася, ярый любитель кино и столь же ярый ненавистник математики, первым купил билет на очередную премьеру. Когда Вася вошёл в зал и сел на своё место, он увидел, что все остальные кресла в его ряду ещё пустуют. Вася точно знал, что к началу сеанса зал

заполнится до отказа, а это значило, что с минуты на минуту о его ноги начнут спотыкаться другие кинолюбители, пробирающиеся к своим местам. Несмотря на всю свою нелюбовь к математике, Вася мгновенно оценил, какое максимальное количество человек может спотнуться о его ноги, прежде чем все зрители займут свои места. А вы сможете?

Программа получает на вход два целые числа n и k — количество мест в том ряду, где сидит Вася, и номер его места соответственно ($1 \le k \le n \le 50$, n — чётно). Места в ряду нумеруются с единицы.

Программа должна Вывести максимальное количество человек, которое может споткнуться о ноги Васи.

Пример

Ввод	Вывод
4 1	1