**Решение олимпиадных задач с использованием ветвления и циклов.**

1.Ветвления.

Простая форма ветвления – выбор из двух вариантов – полная или сокращенная формы: Если… То… Все; Если … То … Иначе … Все

Если в программе возникает необходимость выбора не из двух условий, а из большего количества, то в этом случае можно, конечно. Использовать несколько **if** подряд:

**if** условие1 **then** Команды;

**if**  условие2 **then** Команды;

**if** условие3  **then** Команды; и т.д.

В этом случае нужно следить, чтобы условия были несовместными, иначе выполнится несколько наборов команд. Например:

**if** a>4 **then** write (‘Best!’);

**if** a>3 **then** write (‘Good’);

на значение а=5 выведет и Best! И Good

не поможет и else в конце

**if** a>4 **then** write (‘Best!’);

**if** a>3 **then** write (‘Good’) **else**  write (‘BAD!’) ;

при а=5 выведет Best! и BAD! потому что else всегда относится только к последнему if. В таких многовариантных случаях на помощь приходит конструкция else if

**if** a>4 **then** write (‘Best!’)

**else if** a>3 **then** write (‘Good’) **else** write (‘BAD!’) ;

при таком подходе, даже если условия совместны, срабатывает только один вариант.

Если условий достаточно много (более трех) вложение **if** в **else** становится громоздким и удобнее воспользоваться оператором выбора **Case**

Оператор выбора позволяет выбрать одно из нескольких возможных продолжений программы. Параметром, по которому осуществляется выбор, служит ключ выбора – выражение любого порядкового типа.

Структура оператора выбора в Паскале:

**Case** <ключ\_выбора> **of**

<список\_выбора>

[**else** <оператор\_иначе>] **end**

Оператор выбора Паскаля работает следующим образом. Вначале вычисляется значение выражения <ключ\_выбора>, а затем в последовательности <список\_выбора> отыскивается константа, равная вычисленному значению. Выполняется оператор, который следует за найденной константой, после чего оператор выбора завершает работу. Если в списке выбора не будет найдена константа, соответствующая вычисленному значению ключа выбора, управление передается операторам, стоящим за словом else. Часть else <оператор\_иначе> можно опустить, тогда при отсутствии в списке выбора нужной константы не будет выполнено никаких действий, и оператор выбора просто завершит свою работу.

Например, составим программу, которая по номеру дня недели выводит на экран его название: Пример программы с использованием **Case of**

Program dni\_nedeli;

Var n: byte;

Begin

Readln(n);

Case n of

1: writeln(‘понедельник ’);

2: writeln(‘вторник ’);

3: writeln(‘среда ’);

4: writeln(‘четверг ’);

5: writeln(‘пятница ’);

6: writeln(‘суббота ’);

7: writeln(‘воскресенье’);

else writeln(‘дня недели с номером’, n,’нет’);

end;

end.

Следует помнить, что все константы из списка выбора должны быть различны.

Любому из операторов списка выбора может предшествовать не одна, а несколько констант выбора, разделенных запятыми. Например, следующая программа при вводе одного из символов ‘ y’ или ‘ Y’ выведет на экран «Да», а при вводе ‘ n’ или ‘ N’ – слово «Нет».

Пример программы с использованием **Case of с несколькими переменными**

Var ch: char;

Begin

Readln(ch);

Case ch of

N, n: writeln(‘Да ’);

Y, y: writeln(‘Нет ’);

End;

End.

Очевидно, что рассмотренные выше программы можно записать с помощью вложенных или последовательно расположенных условных операторов, но в подобных задачах использование оператора выбора является более простым. Основное отличие условного оператора от оператора выбора состоит в том, что в условном операторе условия проверяются одно за другим, а в операторе выбора значение ключа выбора непосредственно определяет одну из возможностей.

**пример:** Программа, спрашивающая у ученика его отметку по информатике и реагирующая на нее подходящим текстом

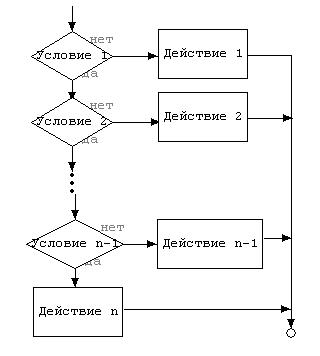
**Результат выполнения с использованием условного оператора if:**

|  |
| --- |
| **var** otmetka:**integer**;  **begin**  writeln('Отметка?');  readln(otmetka);  **if** (otmetka=1) **or** (otmetka=2) **then**  writeln ('кошмар!')  **else** **if** otmetka=3 **then**  writeln('плохо')  **else** **if** otmetka=4 **then**  writeln('неплохо')  **else** **if** otmetka=5 **then**  writeln('отлично!')  **else** writeln('таких отметок не бывает');  **end**. |

Данный пример демонстрирует нерациональную работу программиста и слишком запутанный код. Гораздо лаконичнее выглядит код при использовании оператора выбора.

**Результат выполнения с использованием оператора выбора case:**

|  |
| --- |
| **var** otmetka:**integer**;  **begin**  writeln('отметка?');  readln(otmetka);  **case** otmetka **of**  1,2 : writeln ('кошмар!');  3: writeln('плохо');  4: writeln('неплохо');  5: writeln('отлично!');  **else** writeln('таких отметок не бывает');  **end**  **end**. |

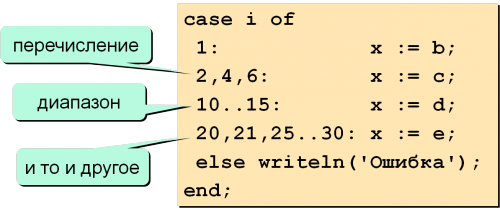
Блок-схема, соответствующая оператору выбора:  
[](https://labs-org.ru/wp-content/uploads/2016/09/1-1.jpg)

* Чтобы задать **диапазон значений**, используется один из двух вариантов:
  + Перечисление через запятую:

|  |
| --- |
| ...  **case** x **of**:  1,2,3,4,5: writeln('от одного до пяти');  ... |

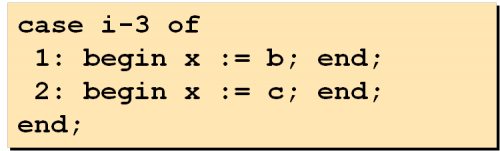
* + Использование ..:

|  |
| --- |
| ...  **case** x **of**:  1..5: writeln('от одного до пяти');  ... |

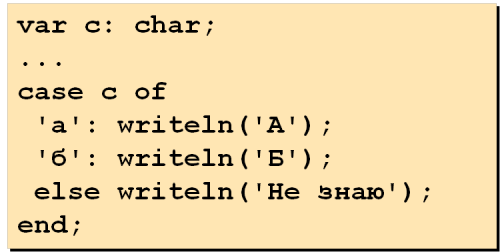
[](https://labs-org.ru/wp-content/uploads/2016/09/1-39.png)

Перечисление или диапазон

* После служебного слова case может стоять имя переменной или арифметическое выражение целого типа:

[](https://labs-org.ru/wp-content/uploads/2016/09/1_11-16.png)

* или символьного типа (char):

[](https://labs-org.ru/wp-content/uploads/2016/09/2-10.png)

**Задача 10. Case7:**  
Единицы массы пронумерованы следующим образом: 1 — килограмм, 2 — миллиграмм, 3 — грамм, 4 — тонна, 5 — центнер. Дан номер единицы массы (целое число в диапазоне 1–5) и масса тела в этих единицах (вещественное число). Найти массу тела в килограммах.

Хорошим примером задач на оператор выбора может быть задача: По номеру месяца определить количество дней в месяце данного года. (Вспомните условие для високосного года: год високосный если его номер делится без остатка на 4, кроме случаев, когда он делится на 100. Если номер года делится на 400, то он все равно високосный).

Задачи:

1. Бесконечная таблица.

|  |  |
| --- | --- |
| Натуральные числа записаны в таблицу (рис). Требуется по заданному числу вывести всех его соседей (числа, записанные в клетках сверху, справа, слева и снизу, если таковые имеются) | 1 |

ВХОДНЫЕ данные вводится одно натуральное число

Выходные данные: Программа должна вывести все числа, записанные в соседних клетках с данным, в порядке возрастания. Числа должны разделяться пробелом.

**Примеры**

**входные данные**

1

**выходные данные**

3

**входные данные**

7

**выходные данные**

3 6 8 13

Это идейно достаточно сложная олимпиадная задача. Необходимо не только определить место введенного числа в таблице, но и понять, какие числа стоят справа и слева от него (верхнее и нижнее число всегда, если они есть отличаются на единицу) Заметим, что по нижнему краю идут квадраты чисел (1, 4, 9, 16…). Вычислим место в столбце, которое занимает введенное число. Для этого можно использовать функцию квадратного корня (в принципе, если использовать циклы, можно обойтись и без нее). int i = (int) Math.sqrt(n); Рассмотрим случаи:

n – внизу таблицы (квадрат) if (i \* i == n) { if (n != 1) { out.print (n – 1 + " "); } out.print ((i + 1)\* (i + 1) – 1 + " "); }

n - наверху таблицы else if (n == i \* i + 1) // n- наверху таблицы { out.print (n + 1 + " "); out.print ( (i + 1) \* (i + 1) - 2 + " "); }

Остается самый сложный случай n – в середине таблицы else { //… } Предлагаем вам разобрать его самостоятельно. В этой программе удобно использовать конструкцию else if. Она подробно разбирается в тексте занятия. Обратите внимание: при каждом выводе мы добавляем пробелы.

ЗАДАЧА Тип треугольника

Треугольник является • прямоугольным, если квадрат большей стороны равен сумме квадратов других сторон; • остроугольным, если меньше; • тупоугольным, если больше. Пожалуй, единственной сложностью этой задачи является то, что нам даны стороны в любом порядке. int a = in.nextInt(), b = in.nextInt(), c = in.nextInt(); Мы можем получить большую сторону (не меньшую двух других) в любой из трех переменных. И математических слов "без ограничения общности предположим, что наибольшая сторона с" на Java не существует. Задачу можно решить двумя принципиально разными способами. Во-первых, разобрать все три случая: if (a > b && a > b) { … } if (b > a && b > c) { … } if (c > a && c > b) { … } Алгоритмы. Олимпиадное программирование. Стартовый модуль Но даже уже в этом коде есть ошибка. Программа будет в некоторых случаях неправильно работать. Для равностороннего треугольника, например, она не выведет ничего… Замена условий на нестрогие не спасает ситуацию if (a >= b && a >= b) { … } if (b >= a && b >= c) { … } if (c >= a && c >= b) { … } В чем здесь будет ошибка, как ее исправить? Подсказка 4.1 Но даже, если написать условия правильно, все равно появится много ошибок. Внутри будет идейно одинаковый код, но с разным набором переменных. При этом очень легко ошибиться и потом трудно найти ошибку. Другой подход к решению состоит в том, чтобы поменять значения переменных местами, так чтобы, скажем, в переменной a лежала длина большей стороны. Это можно сделать так: if (a < b) //a пока не большая сторона { //меняем a и b местами int t = a; a = b; b = t; } if (a < с) //a пока не большая сторона //(она уже не меньше b, но меньше c) { //меняем a и c местами int t = a; a = c; c = t; } Теперь в переменной a точно лежит большая сторона. В каком порядке расположены переменные b и c нам безразлично. Необходимость в проверках на большую сторону отпадает. Программа становится фактически короче на две трети, при этом читаемость не ухудшается!

2.Циклы

Для того, чтобы организовать цикл в программе, можно использовать конструкцию whilе, которая очень похожа на if". Но не все так просто, конечно. В отличие от if при использовании while необходимо правильно организовать выход. Иначе цикл будет повторяться бесконечно, а бесконечно, он все равно повторяться не сможет: олимпиадную задачу тестирующая система остановит через секунду, максимум через несколько секунд с вердиктом "Превышено время работы" (Time Limit), что означает на обычном разговорном языке "Не тормози!". Вернее как раз наоборот: "Тормози!

Конечно, никто не забывает написать в программе условие цикла. Но вот начальную инициализацию и приращение счетчика очень легко забыть написать. И от этого цикл "зависает" или наоборот не выполняется ни разу.

В конструкции for все действия, кроме собственно команд собраны в одной строке.

Особенно повышается читаемость программы при использовании for вместо while при организации вложенных циклов ("один в другом"). Например, если надо вывести квадрат из N звездочек, можно написать так:

Попробуйте написать теперь текст программы, делающей то же самое, но при помощи while, а не for.

Для решения задачи достаточно часто требуется использовать две и более циклические конструкции, одна из которых помещается в тело цикла другой. Такие конструкции называют вложенными циклами. Внутренний и внешний циклы могут быть любыми из трех рассмотренных ранее видов. Правила организации внешнего и внутреннего циклов такие же, как и соответствующих простых операторов. Однако при использовании вложенных циклов необходимо соблюдать следующее условие: внутренний цикл должен полностью укладываться в циклическую часть внешнего цикла.

**Пример 1. Даны натуральные числа n и k. Составить программу вычисления выражения 1k+2k+...+nk.**

***Ввод:***  
*1 строка - число N*  
*2 строка - число k*  
***Вывод:***  
*Сумма - S*  
  
Для вычисления указанной суммы целесообразно использовать  
1) Внешний цикл For с управляющей переменной i, изменяющейся от 1 до n.  
2) Внутренний цикл For с управляющей переменной j, изменяющейся от 1 до k, в котором будем вычислять p=i в степени k и сумму всех элементов.  
  
*Program z1 ;  
Var n, k, i, j: Integer;  
p,s:int64;  
Begin  
ReadLn (n,k) ;  
s:=0 ;  
For i:=1 To n Do  
Begin  
p:=1;  
For j:=1 To k Do  
p:=p\*i;*  
*s:=s+p;  
End;  
WriteLn(s);  
End.*

было небольшим.

**2. Измените программу так, чтобы вычислялась сумма 11+22+ + ...+nn .**

*Указание. Требуется изменить параметры у внутреннего оператора For (For j:=1 То i Do p:=p\*i;) и убрать переменную k.*

Задачи (acmp)

№63 ЗАГАДКА

Петя и Катя – брат и сестра. Петя – студент, а Катя – школьница. Петя помогает Кате по математике. Он задумывает два натуральных числа X и Y (X,Y≤1000), а Катя должна их отгадать. Для этого Петя делает две подсказки. Он называет сумму этих чисел S и их произведение P. Помогите Кате отгадать задуманные Петей числа.

## Входные данные

Входной файл INPUT.TXT содержит два натуральных числа S и P, разделенных пробелом.

## Выходные данные

В выходной файл OUTPUT.TXT выведите два числа Х и Y, загаданные Петей. Числа следует вывести в порядке неубывания своих значений, разделенные пробелом.

## Примеры

|  |  |  |
| --- | --- | --- |
| **№** | **INPUT.TXT** | **OUTPUT.TXT** |
| 1 | 4 4 | 2 2 |
| 2 | 5 6 | 2 3 |

Разбор решения

*(Время: 1 сек. Память: 16 Мб Сложность: 18%)*

Эту задачу можно решить, составив систему из двух уравнений и двух неизвестных:

X+Y=S

X\*Y=P

Выразив Х из первого уравнения и подставив во второе мы получим квадратное уравнение, из которого получим Х, а значит и Y. Но это решение не самое быстрое для реализации. Здесь можно использовать то, что компьютер способен выполнять миллионы простых операций в секунду. Поэтому можно реализовать полный перебор всех вариантов значений X и Y (очевидно, всевозможных значений получится не более миллиона). Причем, для удобства можно сразу положить что X<=Y. Таким образом, мы сократим перебор вариантов вдвое и сможем найти однозначное решение, которое без проблем возможно вывести в порядке неубывания. Следующий алгоритм реализует вышеописанные рассуждения:

read(s,p);

for x=1..1000{

for y=x..1000{

if(x+y==s и x\*y==p) write(x,' ',y);

}

}

Полезно уметь оценивать временную сложность задачи. Несмотря на то, что ЭВМ способна на многое, далеко не любая задача в случае неэффективного ее решения сможет уложиться в требуемые временные ограничения. Например, если бы дипазон чисел Х и Y был в диапазоне до 30000, то такое решение с полным перебором привело бы к провалу, т.к. в этом случае возможно потребовалось бы выполнение порядка миллиарда операций, что вполне сомнительно для ограничений в 1 секунду. Но и в этом случае перебор был бы возможен. Действительно, когда значение X определено, то не обязательно перебирать всевозможные значения Y, т.к. Y=S-X. Поэтому более короткий и быстрый алгоритм решения этой задачи может выглядеть следующим образом:

read(s,p);

for x=1..30000{

y=s-x;

if(x<=y и x\*y==p) write(x,' ',y);

}