Информация - от латинского слова **informatio** - разъяснение, сообщение, осведомленность.

в быту(житейский аспект) - сведения об окружающем мире и протекающих в нем процессах, воспринимаемые человеком или специальными устройствами.

в технике - сообщения, передаваемые в форме знаков или сигналов.

в теории информации - не любые сведения, а лишь те которые, снимают полностью или уменьшают существующую неопределенность. Информация – это снятая неопределенность.

в кибернетике - часть знаний, которая используется для ориентирования, активного действия, управления, т.е. в целях сохранения, совершенствования, развития системы.

в семантической теории (смысл сообщения) понимают сведения, обладающие новизной.

Информация - это отражение внешнего мира с помощью знаков и сигналов.

Свойства информации:

- Объективность. Информация объективна, если она не зависит от чьего – либо мнения.
- Достоверность. Информация достоверна, если она отражает истинное положение дел.
- Полнота. Информацию можно считать полной, если ее достаточно для понимания и принятия решения.
- Актуальность – важность, существенность для настоящего времени.
- Адекватность – определенный уровень соответствия создаваемого с помощью полученной информации образа реальному объекту, процессу, явлению.

Единицы измерения объема информации

В современные компьютеры мы можем вводить текстовую информацию, числовые значения, а также графическую и звуковую информацию.

Количество информации, хранящейся в ЭВМ, измеряется ее “длиной” (или “объемом”), которая выражается в битах.

Бит — минимальная единица измерения информации(от английского BInary digiT -- двоичная цифра). Каждый бит может принимать значение 0 или 1. Битом также называют разряд ячейки памяти ЭВМ. Для измерения объема хранимой информации используются следующие единицы:

1 байт = 8 бит;
1 Кбайт = 1024 байт (Кбайт читается как килобайт);
1 Мбайт = 1024 Кбайт (Мбайт читается как мегабайт);
1 Гбайт = 1024 Мбайт (Гбайт читается как гигабайт).

Хранение информации

Информация хранится в памяти компьютера в двоичном виде. Для этого каждому символу ставится в соответствие некоторое неотрицательное число, называемое кодом символа, и это число записывается в память ЭВМ в двоичном виде.

Кодирование данных двоичным кодом

Двоичное кодирование основано на представлении данных последовательностью всего двух знаков: 0 и 1. Эти знаки называют двоичными цифрами, по-английски – binary digit или сокращённо bit (бит). Одним битом могут быть выражены два понятия: 0 или 1 (да или нет, чёрное или белое, истина или ложь и т.п.). Если количество битов увеличить до двух, то уже можно выразить четыре различных понятия. Тремя битами можно закодировать восемь различных значений.

Конкретное соответствие между символами и их кодами называется системой кодировки.

Система кодировки ASCII

В персональных компьютерах обычно используется система кодировки ASCII (**American Standard Code for Information Interchange** — американский стандартный код для обмена информации).

Он введен в 1963 г. и ставит в соответствие каждому символу семиразрядный двоичный код. Легко определить, что в коде ASCII можно представить 128 символов.

Универсальная система кодирования текстовых данных

Если кодировать символы не восьмиразрядными двоичными числами, а числами с большим разрядом то и диапазон возможных значений кодов станет намного больше.

Такая система, основанная на 16-разрядном кодировании символов, получила название универсальной – **UNICODE.**

Шестнадцать разрядов позволяют обеспечить уникальные коды для 65 536 различных символов – этого поля вполне достаточно для размещения в одной таблице символов большинства языков планеты.

Язык как способ представления информации.

Знаковая форма восприятия, хранения и передачи информации означает использование какого-либо языка. Языки делятся на разговорные (естественные) и формальные.

Естественные языки носят национальный характер.

Формальные языки чаще всего относятся к специальной области человеческой деятельности (например, язык математики или язык флажков на флоте).

Основу языка составляют:

алфавит - конечный набор знаков (символов) любой природы, из которых конструируются сообщения, образует некоторого языка.

Простейшим алфавитом, достаточным для записи (представления) информации, является алфавит из двух символов, например, **0 и 1.**

слово - последовательность символов алфавита, кодирующая состояние источника и воспринимаемая адресатом как сообщение, как информация.

синтаксис - система правил, определяющих допустимые конструкции языка программирования.

семантика - система правил однозначного толкования отдельных языковых конструкций, позволяющих воспроизвести процесс обработки данных.

Понятие о языках программирования высокого уровня

Языки программирования, имитирующие естественные языки, обладающие укрупненными командами, ориентированные на решение прикладных содержательных задач, называются **языками "высокого уровня".**

Языки программирования высокого уровня

В настоящее время насчитывается несколько сотен таких языков, а с их диалектами - нескольких тысяч.

Особенности языков низкого уровня (машинно-ориентированных).

Команда на машинном языке содержит очень ограниченный объем информации, поэтому она обычно определяет простейший обмен содержимого ячеек памяти, элементарные арифметические и логические операции.

Команда содержит код и адреса ячеек, с содержимым которых выполняется закодированное действие.

Достоинства языков программирования высокого уровня.

- **алфавит языка значительно шире машинного**, что делает его гораздо более выразительным и существенно повышает наглядность и понятность текста;
- **конструкции команд (операторов) отражают содержательные виды обработки данных** и задаются в удобном для человека виде;
- **используется аппарат переменных** и действий с ними;
- **поддерживается широкий набор типов данных.**

Языки программирования высокого уровня являются машинно-независимыми и требуют использования соответствующих программ-переводчиков **(трансляторов)** для представления программы на языке машины.

Примеры языков программирования высокого уровня

Fortran

Это первый компилируемый язык созданный Джимом Бэкусом в 50-е годы.Программисты,разрабатывавшие программы исключительно на ассемблере, выражали серьезное сомнение в возможности появления высокопроизводительного языка высокого уровня, поэтому основным критерием при разработке компиляторов Фортрана являлась эффективность исполняемого хода. Хотя в Фортране был впервые реализован ряд важнейших понятий программирования, удобство создания программ было принесено в жертву возможности получения эффективного машинного кода.

Cobol

Это компилируемый язык для применения в экономической области и решения бизнес-задач, разработанный в начале 60-х годов.Он отличается большой "многословностью"-его операторы выглядят как обычные английские фразы.В Коболе были реализованы очень мощные средства работы с большими объемами данных.

Algol

Компилируемый язык,созданный в 1960 году.Он был призван заменить Фортран,но из-за более сложной структуры не получил широкого распространения.

Pascal

Язык Паскаль,созданный в конце 70-х годов основоположником множества идей современного программирования Никлаусом Виртом, во многом напоминает Алгол,но в нем ужесточен ряд требований к структуре программы и имеются возможности, позволяющие успешно применять его при создании крупных проектов.

Basic

Для этого языка имеются и компиляторы,и интерпретаторы,а по популярности он занимает первое место в мире.Он создавался в конце 60-х годов в качестве учебного пособия и очень прост в изучении.

C

Данный язык был создан в лаборатории Bell и первоначально не рассматривался как массовый.Он планировался для замены ассемблера,чтобы иметь возможность создавать столь же эффективные и компактные программы,и в то же время не зависеть от конкретного вида процессора.

C++

C++-это объектно-ориентированное расширения языка Си,созданное Бьярном Страуструпом в 1980 году.Множество новых мощных возможностей,позволивших резко увеличить производительность программистов,наложилось на унаследованную от языка Си определенную низкоуровневость,в результате чего создание сложных и надежных программ потребовало от разработчиков высокого уровня прфессиональной подготовки.

Java

Этот язык был создан компанией Sun в начале 60-х годов на основе Си++.Он призван упростить разработку приложений на основе Си++ путем исключения из него всех низкоуровневых возможностей.Но главная особенность этого языка -компиляция не в машинный код,а в платформо-независимый байт-код.Этот байт-код может выполнятся с помощью интерпретатора-виртуальной машины Java-машины JVM(Java Virtyal Machine), версии которой созданы сегодня для любых платформ.

Алгоритм и алгоритмизация.

Слово **"алгоритм"** появилось в 9-м веке и связано с именем математика Аль-Хорезми.

Алгоритм представляет решение задачи в виде точно определенной последовательности действий (операций).

Процесс составления алгоритмов называют **алгоритмизацией.**

В процессе формального решения задачи, ее решение сначала описывается на языке математики в виде системы формул, а затем на языке алгоритмов в виде некоторого процесса. Таким образом, алгоритм – это связующее звено в цепочке "метод решения - реализующая программа".

Свойства алгоритма:

- Определенность - выполнив очередное действие, исполнитель должен точно знать, что ему делать дальше.
- Дискретность - прежде, чем выполнить определенное действие, надо выполнить предыдущее.
- Массовость - по одному и тому же алгоритму решаются однотипные задачи и неоднократно.
- Понятность - алгоритм строится для конкретного исполнителя человеком и должен быть ему понятен. Это облегчает его проверку и модификацию при необходимости.
- Результативность - алгоритм всегда должен приводить к результату.

Способы представления алгоритма.

- **Словесное** (с помощью обычных предложений русского или другого языка)
- **Графическое** или визуальное (с помощью блок-схемы),
- **Программное** (на языке программирования)

Все три способа представления алгоритмов взаимо дополняют друг друга.

Пример словесного описания алгоритма.

Начало

Ввести 2 числа;

Вычислить произведение чисел;

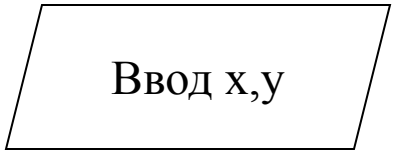
Вывести результат;

Конец.

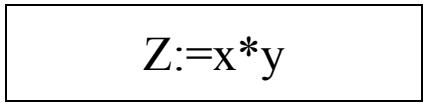
Графическое представление алгоритма(блок-схема алгоритма)

Блок-схема представляет алгоритм решения задачи в виде последовательности графических блоков.

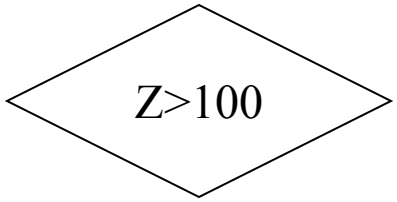
Блок начала или конца алгоритма



Блок ввода или вывода

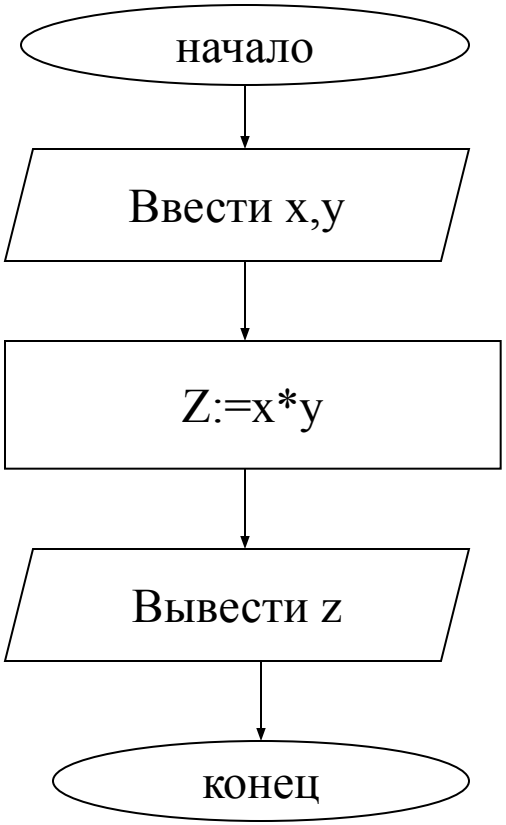


Блок действия



Блок условия, имеет 2 выхода

Пример блок-схемы алгоритма



Пример программного представления алгоритма (на Паскале)

```
Program myprog;
Var
X,Y,Z:integer;
Begin
WriteLn (‘Введите два числа’);
Read (x,y);
Z:=x*y;
WriteLn (‘Результат:’,z);
End.
```

Общие правила проектирования алгоритмов

- В начале алгоритма должны быть блоки ввода значений входных данных.
- После ввода значений входных данных могут следовать блоки обработки и блоки условия.
- В конце алгоритма должны располагаться блоки вывода значений выходных данных.
- В алгоритме должен быть только один блок начала и один блок окончания.
- Связи между блоками указываются направленными или ненаправленными линиями.

Классификация алгоритмов.

Алгоритмы разделяют на:

- **линейные,**
- **разветвленные**
- **циклические алгоритмы.**

Линейные алгоритмы.

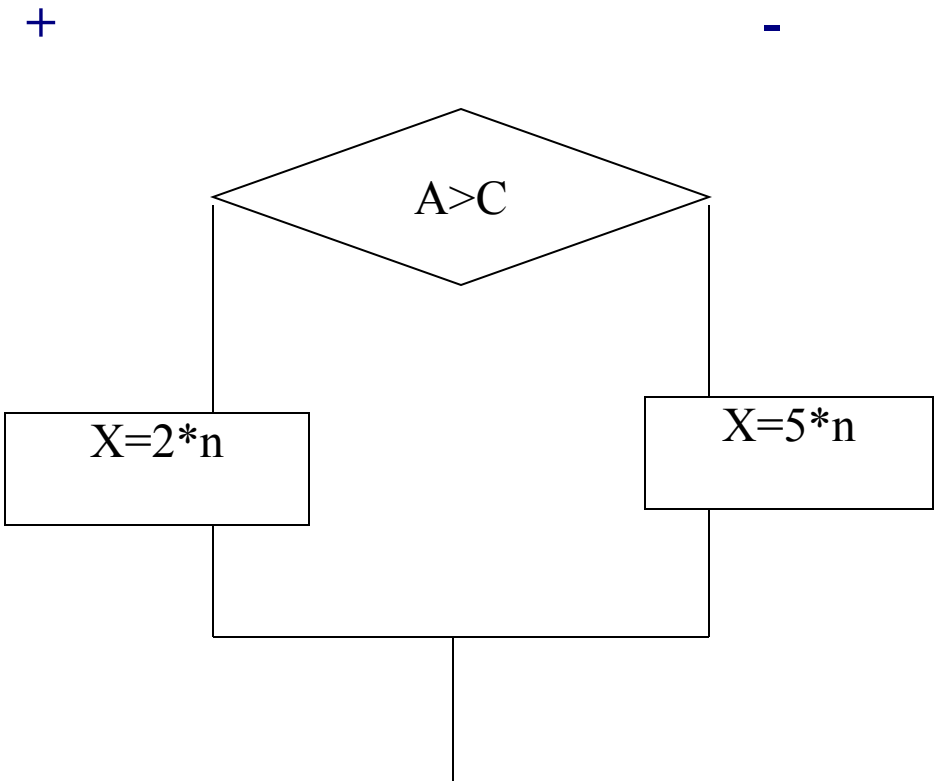
- Представляют решение задачи в виде последовательности действий.
- Не содержат блока условия.
- Предназначены для описания линейных процессов.

Пример линейного алгоритма.

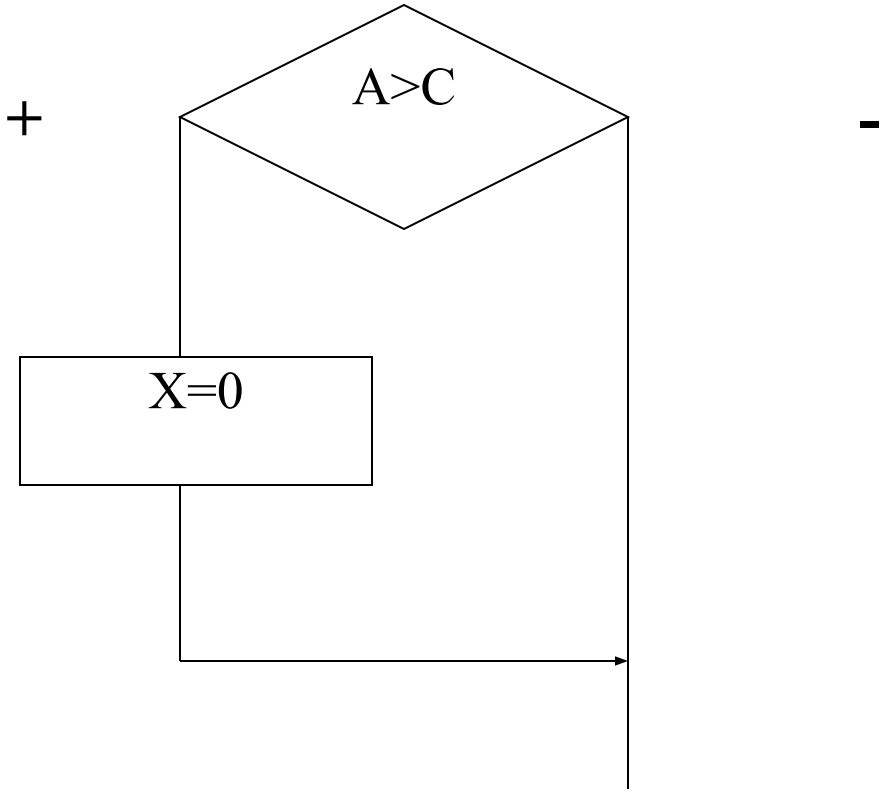
Задача. Вычислить площадь и периметр прямоугольника.

```
Program prog1;
Var
a,b,s,p:real;
Begin
        writeln(‘Введите длину сторон прямоугольника’);
        readln(a,b);
        s:=a*b;
        p:=2*(a+b);
        writeln(‘Площадь прямоугольника’,s:7:3);
        writeln(‘Периметр прямоугольника’,p:7:3);
End.
```

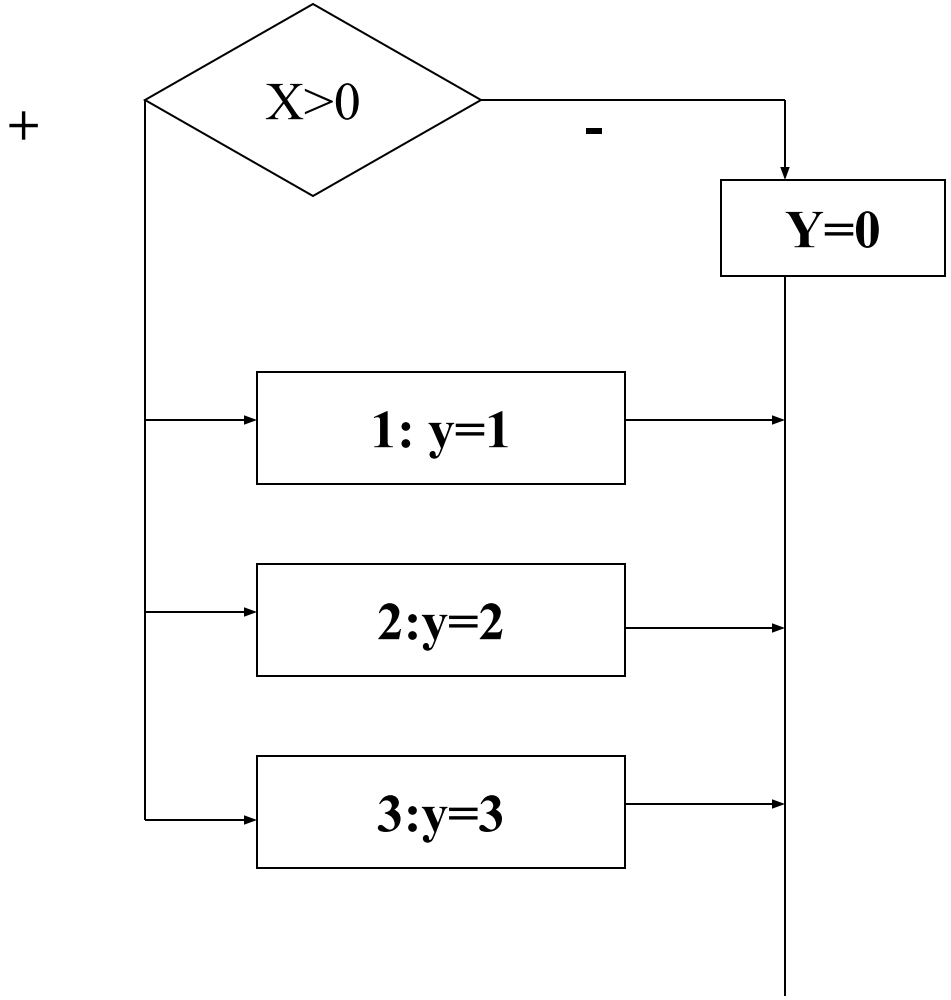
Разветвленные алгоритмы. Ветвление.



Неполное ветвление



Многоальтернативный выбор



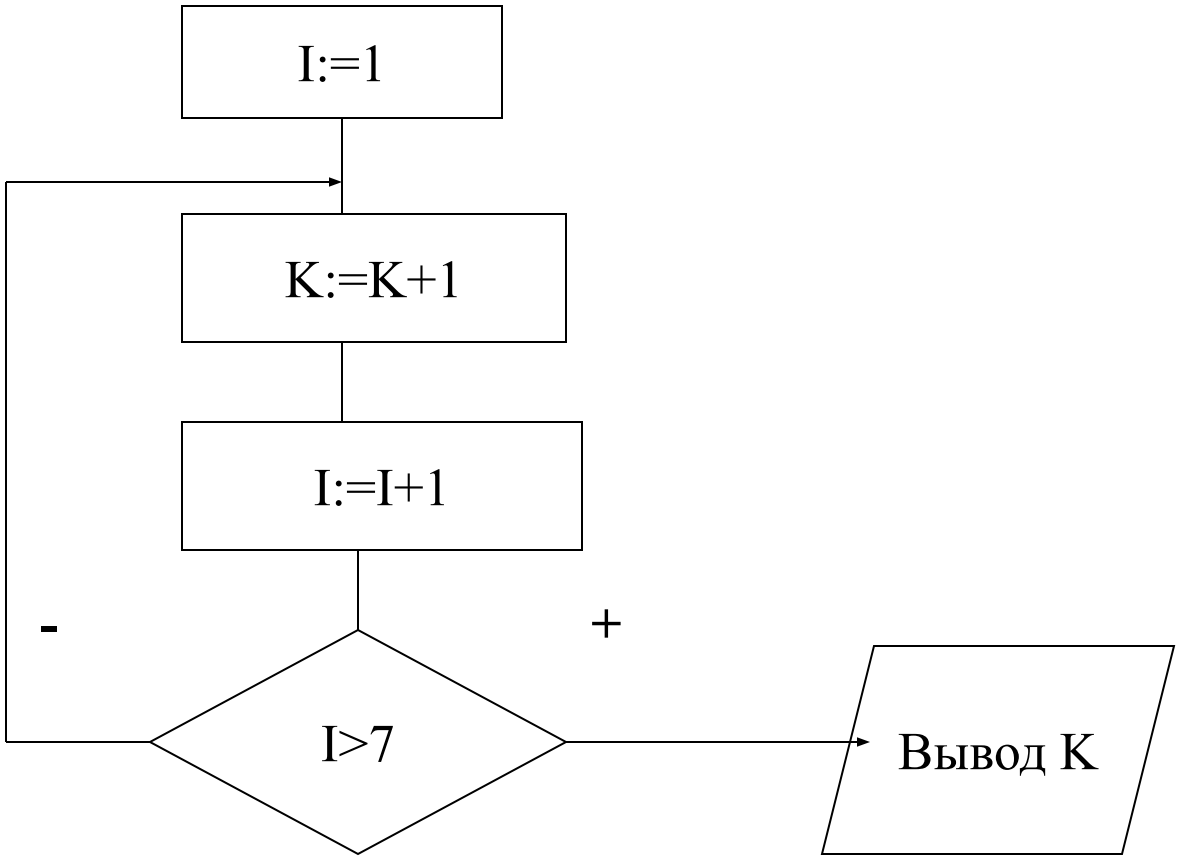
Циклические алгоритмы.

Цикл – повторяющаяся последовательность действий.

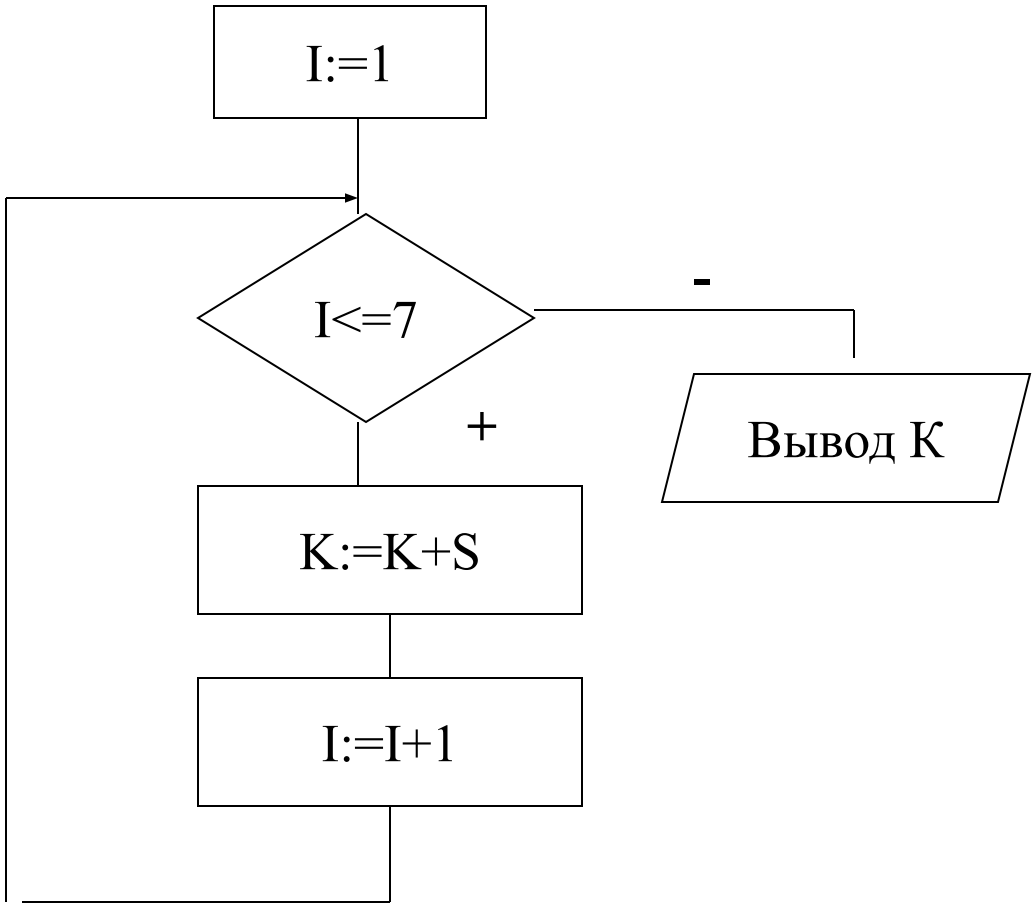
Цикл с предусловием. Сначала проверяется условие выхода из цикла. Условие - это логическое выражение, которое может принимать значение ИСТИНА или ЛОЖЬ. (Например, y>0). Если условие принимает значение ИСТИНА, то выполняются те действия, которые должны повторяться. В противном случае, если логическое выражение принимает значение ЛОЖЬ, цикл завершается.

Цикл с постусловием . Сначала один раз выполняются действия, которые подлежат повторению, затем проверяется логическое выражение , определяющее условие выхода из цикла, например, I>6 . Если условие выхода истинно, то цикл с постусловием прекращает свою работу, в противном случае - происходит повторение действий, указанных в цикле. Действия, повторяющиеся в цикле, называются "телом цикла".

Цикл с постусловием



Цикл с предусловием



Основные элементы языка Паскаль.

Основные Символы

Основные символы языка- буквы, цифры и специальные символы составляют его алфавит.

ТУРБО ПАСКАЛЬ включает следующий набор основных символов:

1. **26 латинских строчных и 26 латинских прописных букв:**

A B C D E F G H I J K L M N O P Q R S T U V W X Y Za b c d e f g h i j k l m n o p q r s t u v w x y z

2. **_ подчеркивание**
3. **10 цифр:**

0 1 2 3 4 5 6 7 8 9

4. знаки операций:

+ - * / = <> < > <= >= := @

5.ограничители:

. , ' ( ) [ ] (. .) { } (* *) .. : ;

6.спецификаторы:

^ # $

1. **служебные (зарезервированные) слова:**

ABSOLUTE EXPORTS LIBRARY SET ASSEMBLER EXTERNAL MOD SHL AND FAR NAME SHR ARRAY FILE NIL STRING ASM FOR NEAR THEN FORWARD NOT TO BEGIN FUNCTION OBJECT TYPE CASE GOTO OF UNIT CONST IF OR UNTIL CONSTRUCTOR IMPLEMENTATION PACKED USES DESTRUCTOR IN PRIVATE VAR DIV INDEX PROCEDURE VIRTUAL DO INHERITED PROGRAM WHILE DOWNTO INLINE PUBLIC WITH ELSE INTERFACE RECORD XOR END INTERRUPT REPEAT EXPORT LABEL RESIDENT

Кроме перечисленных, в набор основных символов входит пробел. Пробелы нельзя использовать внутри сдвоенных символов и зарезервированных слов.

Элементарные Конструкции

Элементарные конструкции языка ПАСКАЛЬ включают в себя **имена, числа и строки**.

Имя - это последовательность букв и цифр, начинающаяся с буквы. В именах может использоваться символ _ подчеркивание. Имя может содержать произвольное количество символов, но значащими являются 63 символа.

Имена (идентификаторы) называют элементы языка - константы, метки, типы, переменные, процедуры, функции, модули, объекты.

Не разрешается в языке ПАСКАЛЬ использовать в качестве имен служебные слова и стандартные имена, которыми названы стандартные константы, типы, процедуры, функции и файлы.

Нельзя использовать внутри имен и чисел пробелы.

Примеры имен языка ПАСКАЛЬ:

A b12 r1m SIGMA gamma I80_86

Числа в языке ПАСКАЛЬ обычно записываются в десятичной системе счисления. Они могут быть целыми и действительными. Положительный знак числа может быть опущен.

Целые числа записываются в форме без десятичной точки, например:

217 -45 8954 +483

Действительные числа записываются в форме с десятичной точкой или в форме с использованием десятичного порядка, который изображается буквой E:

28.6 0.65 -0.018 4.0 5E12 -1.72E9 73.1E-16

ПАСКАЛЬ допускает запись целых чисел и фрагментов действительных чисел в форме с порядком в шестнадцатиричной системе счисления:

$7F $40 $ABC0

Строки в языке ПАСКАЛЬ - это последовательность символов, записанная между апострофами. Если в строке в качестве содержательного символа необходимо употребить сам апостроф, то следует записать два апострофа. Примеры строк:

'СТРОКА' 'STRING' 'ПРОГРАММА' 'АД''ЮТАНТ'

КОНЦЕПЦИЯ ТИПА ДЛЯ ДАННЫХ

В любом алгоритмическом языке каждая константа, переменная, выражение или функция бывают **определенного типа.**

В языке ПАСКАЛЬ существует правило: **тип явно задается в описании переменной или функции, которое предшествует их использованию.**

В языке ПАСКАЛЬ существуют **скалярные и структурированные** типы данных.

К **скалярным типам** относятся **стандартные** типы и типы, **определяемые пользователем**.

Стандартные типы включают целые, действительные, символьный, логические и адресный типы.

Типы, определяемые пользователем, - перечисляемый и интервальный.

Структурированные типы имеют четыре разновидности: массивы, множества, записи и файлы.

Кроме перечисленных, TURBO PASCAL включает еще два типа - **процедурный и объектный.**

Из группы **скалярных типов** можно выделить порядковые типы, которые характеризуются следующими свойствами:

1. все возможные значения порядкового типа представляют собой ограниченное упорядоченное множество;
2. к любому порядковому типу может быть применена стандартная функция **Ord,** которая в качестве результата возвращает порядковый номер конкретного значения в данном типе;
3. к любому порядковому типу могут быть применены стандартные функции **Pred** и **Succ,** которые возвращают предыдущее и последующее значения соответственно;
4. к любому порядковому типу могут быть применены стандартные функции **Low** и **High**, которые возвращают наименьшее и наибольшее значения величин данного типа.

СТРУКТУРА ПРОГРАММЫ НА ПАСКАЛЕ.

```
PROGRAM  ИмяПрограммы;
    {Раздел описаний}
BEGIN
    {Раздел операторов}
END.
```

Раздел описаний.

Вся информация, с которой работает ЭВМ, называется **ДАННЫМИ**.

Каждый элемент данных является либо **константой**, либо **переменной** и указывается в программе своим именем-**идентификатором.**

Константы.

Константы – это данные, которые не изменяются в процессе выполнения программы.

Размещаются **в разделе описания констант**, например:

```
CONST
Max=200;
const rWeight: Real = 0.4;
```

Переменные.

Переменные – это данные, принимающие различные значения в процессе выполнения программы. Размещаются **в разделе описания переменных**

```
Var
имя переменной: тип;
```

например:

```
Var
a,b,c: Integer;
```

К переменным обращаются по именам (идентификаторам).

Правила задания имен переменных:

- строятся из букв, цифр и спецсимволов (например, подчерк), но начинаться должны с буквы.
- не могут быть зарезервированными словами
- не должны содержать пробелы.

Строчные и заглавные буквы в именах переменных не различаются.

В откомпилированной программе для всех переменных отведено место в памяти, и всем переменным присвоены нулевые значения.

Раздел операторов

Раздел операторов содержит последовательность операторов между служебными словами

begin.....end.

Операторы отделяются друг от друга символом точка с запятой - ;

Текст программы заканчивается символом точка.

Кроме описаний и операторов ПАСКАЛЬ - программа может содержать комментарии.

Комментарий записывается в фигурных скобках **{}**

ПРИМЕР ПРОГРАММЫ:

```
Program TRIANG;
var
A, B, C, S, P: Real;
begin
        Read(A,B,C);
        WriteLn(A,B,C);
         P:=(A+B+C)/2;
         S:=Sqrt(P*(P-A)*(P-B)*(P-C));
         WriteLn('S=',S:8:3);
 end.
```

Выражения

Выражение состоит из констант, переменных, указателей функций, знаков операций и скобок. Выражение задает правило вычисления некоторого значения. Порядок вычисления определяется старшинством (приоритетом) содержащихся в нем операций. В языке ПАСКАЛЬ принят следующий приоритет операций:

1. унарная операция not, унарный минус -, взятие адреса @
2. операции типа умножения * / div mod and shl shr
3. операции типа сложения + - or xor
4. операции отношения = <> < > <= >= in

Выражения входят в состав многих операторов языка ПАСКАЛЬ, а также могут быть аргументами встроенных функций.

Оператор Присваивания :=

I:=I+1

Операторы Ввода и Вывода

Используются для организации ввода и вывода данных с терминального устройства (дисплей и клавиатура).

Для ввода и вывода данных используются стандартные процедуры ввода и вывода **Read** и **Write**, оперирующие стандартными последовательными файлами INPUT и OUTPUT.

Эти файлы разбиваются на строки переменной длины, отделяемые друг от друга признаком конца строки. Конец строки задается нажатием клавиши ENTER.

Для ввода исходных данных используются операторы процедур ввода:

Read(A1,A2,...AK);

ReadLn(A1,A2,...AK);

ReadLn;

При вводе исходных данных происходит преобразование из внешней формы представления во внутреннюю, определяемую типом переменных. Переменные, образующие список ввода, могут принадлежать либо к целому, либо к действительному, либо к символьному типам. Чтение исходных данных логического типа в языке ПАСКАЛЬ недопустимо.

Операторы ввода при чтении значений переменных целого и действительного типа пропускают пробелы, предшествующие числу. В то же время эти операторы не пропускают пробелов, предшествующих значениям символьных переменных, так как пробелы являются равноправными символами строк. Пример записи операторов ввода:

```
Var
rV, rS: Real;
iW, iJ: Integer;
chC, chD: Char;
Read(rV, rS, iW, iJ);
Read(chC, chD);
```

Для вывода результатов работы программы на экран используются операторы:

Write(A1,A2,...AK);

WriteLn(A1,A2,...AK);

WriteLn;

Первый из этих операторов реализует вывод значений переменных A1, A2,...,AK в строку экрана. Второй оператор реализует вывод значений переменных A1, A2, ..., AK и переход к началу следующей строки. Третий оператор реализует пропуск строки и переход к началу следующей строки.

Переменные, составляющие список вывода, могут относиться к целому, действительному, символьному или булевскому типам. В качестве элемента списка вывода кроме имен переменных могут использоваться выражения и строки.

Оператор вывода позволяет задать ширину поля вывода :

В виде **A:K**, где A - выражение или строка, K - выражение либо константа целого типа.

Для величин действительного типа элемент списка вывода может иметь вид **A:K:M**, где A - переменная или выражение действительного типа, K - ширина поля вывода, M - число цифр дробной части выводимого значения. K и M - выражения или константы целого типа.

В этом случае действительные значения выводятся в форме десятичного числа с фиксированной точкой.

Пример записи операторов вывода:

```
var
rA, rB: Real;
iP,iQ:Integer;
bR, bS: Boolean;
chT, chV, chU, chW: Char;
WriteLn(rA, rB:10:2);
WriteLn(iP, iQ:8);
WriteLn(bR, bS:8);
WriteLn(chT, chV, chU, chW);
```

Ветвящийся алгоритм. Условный оператор

If <условие> **then** <оператор> **else** <оператор>**;**

Пример

```
Program my_prog;
Var
t:real;
Begin
        writeln('введите средний балл');
        readln(t);
        if t>=4 then writeln ('Это успевающий студент')
        else
       writeln ('Вы плохо успеваете, надо быть прилежнее');
       readln;
End.
```

Составной оператор. Begin - end;

```
Program my_prog;
Var
t:real;
Begin
        writeln('введите средний балл');
        readln(t);
        if t>=4 then
        begin
                writeln ('Вы успевающий студент');
                writeln ('Вы заслуживаете поощрения')
        end;
        else
        begin
        writeln ('Вы плохо успеваете, надо быть прилежнее');
        writeln ('Придется побеседовать с Вашими родителями')
        end;
        readln;
End.
```

Оператор выбора

Case <переменная> of
<Значение>:<оператор>
<Значение>:<оператор>
<Значение>:<оператор>
……..
Else
<>
End;

Переменная может быть целочисленной или символьной.

```
Var
operation:Char;
x,y,z:real;
stop:Boolean;
…….
Case operatoin of
‘+’: z:=x+y;
‘-’: z:=x-y;
‘*’: z:=x*y;
‘/’: z:=x/y;
Else
Stop:=true;
End;
```

Цикл со счетчиком

For <идентификатор счетчика>=<начальное значение>

To <конечное значение> do <оператор, обычно составной>.

```
Var
Summa, I:Integer;
Begin
        Summa:=0;
        For I:=1 to 10 do
        Summa:=summa+I;
        Writeln('Сумма= ',summa);
        Readln;
End.
```

Цикл с предусловием

While <условие> do <оператор или составной оператор>

```
begin
I:=1;
While I<=10  do
Writeln('Значение счетчика=',I);
I:=I+1;
end.
```

Цикл с постусловием

Repeat <тело цикла> Until <условие>.

Не требует операторных скобок begin…end.

Задача: нахождение минимального числа элементов последовательности, сумма которых превышает значение предельной суммы.

```
Program my1;
Var
sp,s,i,a:Integer;
begin
i:=0;s:=0;
Writeln('Введите предельную сумму');
Readln(sp);
Repeat
Write('Введите элемент:');
Readln(a);
s:=s+a;
i:=i+1
Until s>sp;
Writeln('Число элементов равно ',I);
end.
```