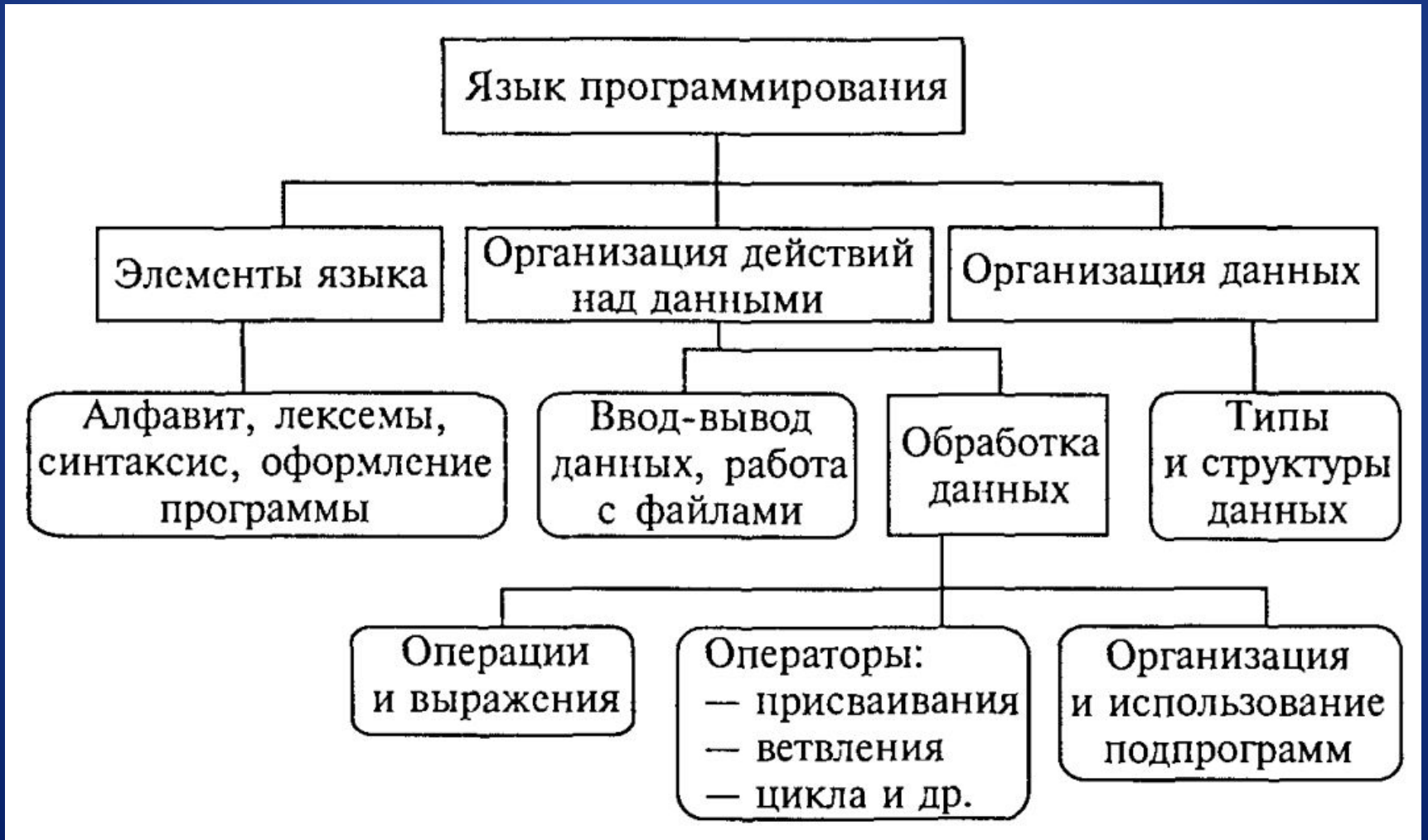
Основы программирования

PASCAL

Язык программирования Pascal

Разработан в 1971 г. швейцарским профессором Никлаусом Виртом для обучения структурному программированию.



Всякий язык программирования имеет три основные составляющие: *алфавит, синтаксис* и *семантику.*

Алфавит языка – это множество символов, которые можно использовать для записи правильных программ.

Синтаксис языка – это совокупность правил построения допустимых конструкций языка, форма их сочетаний при записи алгоритма, т.е. то что определяет правильность программ.

Семантика – это смысл конструкций языка, в том числе и программ, написанных на этом языке.

```
program <имя_программы>;
    [ uses <имена_подключаемых_модулей>;]
    [ label <список_меток>;]
    [ const <имя_константы> = <значение_константы>;]
    [ type <имя_типа> = <определение_типа>;]
    [ var <имя_переменной> : <тип_переменной>;]
    [ procedure <имя_процедуры> <описание_процедуры>;]
    [ function <имя_функции> <описание_функции>;]
    begin   {начало основного тела программы}
    <операторы>
    end.  (* конец основного тела программы *)
```

Основные символы языка Pascal

Это латинские буквы, цифры от 0 до 9 и специальные символы

```
+ — * / = , . : ; < > [ ] ( ) { } ^ @ $ #
```

Также есть служебные слова, которые не могут использоваться в качестве идентификаторов (т.е. имен переменных, подпрограмм, модулей). Например, слова **`var`**, **`type`**, **`if`**.

Идентификаторы

Это имена переменных, констант, подпрограмм, модулей. В программе **не может быть двух идентификаторов с одним именем**!

Правильный идентификатор **должен начинаться** с латинской буквы. В нем могут присутствовать цифры и знак подчеркивания.

Примеры:

x

X

summa

s1

m_

список наиболее часто встречающихся зарезервированных слов:

and	goto		set	
array		implementation		shl
begin		in	shr	
case		interface		string
const		label		then
div	mod		text	
do	nil	to		
downto		not	type	
else	of		unit	
end	or		until	
file	pointer		uses	
far	procedure		var	
for	program		while	
forward		record		with
Function		repeat		xor

Простейшие операторы

a:= b; - присваивание переменной a значения переменной b.

Операторные скобки, превращающие несколько операторов в один:

```
begin
  <несколько операторов>
end;
```

Комментарии

{ комментарий }
Другой вариант оформления комментария:
(* комментарий *)

Ввод и вывод

read(<список_ввода>) и readln(<список_ввода>);

write(<список_вывода>) и writeln(<список_вывода>).

Пример простейшей программы на языке Pascal

```
program start;
var s: string;
begin
 write('Пожалуйста, введите Ваше имя: ');
 readln(s);
 writeln('Мы рады Вас приветствовать, ',s,'!');
end.
```

Во время работы этой программы на экране появится следующее:
Пожалуйста, введите Ваше имя: Иван Иваныч
Мы рады Вас приветствовать, Иван Иваныч!

Типы данных

Объекты (константы, переменные, функции, выражения), которыми оперирует программа, относятся к определенному типу.

Тип — это множество значений, которые могут принимать объекты программы, и совокупность операций, допустимых над этими значениями.

Базовые типы данных	Дискретные типы данных		Дискретные типы данных		Адресные типы данных	Структурированные типы данных
	Логический	**Символьный (литерный)**	**Целые**	**Вещественные**	**Нетипизированный указатель**	
	boolean	char	shortint byte Integer Word longint	Real Single Double extended comp	pointer	
Конструируемые типы		**Перечисляемый**			**Типизированный указатель**	**Массив** array
		week = (su, mo, tu,we, th, fr,sa);			^<тип>	**Строка** string
		Интервал (диапазон)				**Запись** record

		budni = mo..fr;				**Файл**
						`text`
						`file`

						Процедурный
	Типы данных, конструируемые программистом					**Объектный**[1]

Перечисляемые типы данных задаются в разделе type

Например:

type week =(sun,mon,tue,wed,thu,fri,sat)

0 1 2 3 4 5 6

Интервальные типы данных

задаются только границами своего диапазона. Например:

type month = 1..12;

Комбинации

Например:

type valid_for_identifiers = 'a'..'z','A'..'Z','_','0'..'9';

Идентификатор	Длина (байт)	Диапазон значений	Операции
Целые типы			
integer	2	-32768..32767	+, -, /, *, Div, Mod, >=, <=, =, <>, <, >
byte	1	0..255	+, -, /, *, Div, Mod, >=, <=, =, <>, <, >
word	2	0..65535	+, -, /, *, Div, Mod, >=, <=, =, <>, <, >
shortint	1	-128..127	+, -, /, *, Div, Mod, >=, <=, =, <>, <, >
longint	4	-2147483648..2147483647	+, -, /, *, Div, Mod, >=, <=, =, <>, <, >
Вещественные типы			
real	6	$2,9x10^{-39}$ - $1,7x10^{38}$	+, -, /, *, >=, <=, =, <>, <, >
single	4	$1,5x10^{-45}$ - $3,4x10^{38}$	+, -, /, *, >=, <=, =, <>, <, >
double	8	$5x10^{-324}$ - $1,7x10^{308}$	+, -, /, *, >=, <=, =, <>, <, >
extended	10	$3,4x10^{-4932}$ - $1,1x10^{4932}$	+, -, /, *, >=, <=, =, <>, <, >
Логический тип			
boolean	1	true, false	Not, And, Or, Xor, >=, <=, =, <>, <, >
Символьный тип			
char	1	все символы кода ASCII	+, >=, <=, =, <>, <, >

Переменная

Данные хранятся в памяти компьютера, но для указания на конкретную информацию очень неудобно все время записывать физические адреса ячеек.

Эта проблема в языке Pascal решена введением понятия переменной. Переменная – именованный участок памяти для хранения данных определенного типа. Значение переменной (информация в соответствующих ячейках памяти) в ходе выполнения программы может быть изменено. **Имя переменной должно быть правильным идентификатором!**

Объявляются переменные в специальном разделе `var` (см. структуру программы на языке Pascal). Например:

```
var
 x: integer;
 b, summa: integer;
 a: real;
```

Константа

Величина, значение которой в ходе выполнения программы не может быть изменено.

Константы бывают обычные (просто значения, например, 5, 6.7, ‘f’) и именованные.

Имя константы должно быть правильным идентификатором.

Именованные константы объявляются в разделе `const`.

Пример:

```
const
        pi=3.14;
        n=20;
```

Оператор присваивания

:=

<Переменная> := <Выражение> ;

В левой части может быть только 1 переменная, которой будет присвоено значение выражения из правой части. Тип переменной слева должен соответствовать типу выражения справа.

Выражение состоит из операндов, знаков операций и круглых скобок.

Операндами являются константы, переменные, обращения к функциям.

Примеры:

```
a:=6;
s:=a+5;
y:=sin(x);
z:=y+cos(x);
```

Ввод данных

```
read(<переменные>);
readln(<переменные>);
```

Пример 1:

```
readln(x);
```

Программа приостановит свое выполнение и будет ожидать ввод данных (в зависимости от типа переменной x). Завершается ввод нажатием клавиши Enter.

Пример 2:

```
readln(a,b,c);
```

Ввод нескольких переменных в одном выражении. При вводе с клавиатуры данные должны разделяться пробелом.

Вывод данных на экран

```
write(<выводимые данные>);
writeln(<выводимые данные>);
```

Во втором случае после вывода на экран будет произведен переход на следующую строку.

Примеры:

1.
```
write('Hello,');
write(' world!');
```

На экране появится:

```
Hello, world!
```

2.
```
writeln('Hello,');
writeln(' world!');
```

На экране появится:

```
Hello,
 world!
```

Особенности вывода

По умолчанию вывод происходит в поле вывода шириной в количество знаков выводимого числа. При этом вещественные числа выводятся с максимально возможным количеством знаков после вещественной точки.

Например:

```
x:=5/3;
writeln(x);
```

На экране появится

```
1.6666666666667
```

Но есть возможность управлять как количеством знаков в дробной части, так и шириной поля вывода числа.

В операторе write или writeln вещественное значение (а также целое или строковое) можно записать в виде:

переменная:ширина:точность

ширина - целое положительное число, определяющее, сколько экранных позиций отводится для вывода всего числа. Ширина определена для числовых значений любого типа и строк.

Точность - целое положительное число, определяющее, сколько цифр из ширины отводится на вывод дробной части числа. Значение точности определено только для вещественных чисел. Оно не учитывает позицию десятичной точки. Недопустимые значения ширины и точности не будут учтены при выводе.

```
write('Сумма введенных чисел равна',s:5:2);
```

Наименование Функции	Результат вычесления
МАТЕМАТИЧЕСКИЕ ФУНКЦИИ	
abs(x)	|x|
sin(x)	sin(x)
cos(x)	cos(x)
arctan(x)	arctg(x)
Sqrt(x)	корень квадратный
Sqr(x)	x^2
Exp(x)	e^x
Ln(x)	ln(x)
Power(a,b)	a^b

Наименование Функции	Результат вычесления
МАТЕМАТИЧЕСКИЕ ФУНКЦИИ	
Frac(x)	дробная часть "x"
Int(x)	целая часть "x"
Random	случайное число (0 < =y< 1)
Random(x)	случайное число (0 < =y< x)
Succ(c)	следующий за "c" символ
Pred(c)	предшествующий "c" символ
round(x)	округление вещественного числа x до ближайшего целого
trunc(x)	отбрасывает дробную часть числа
pi	π (π = 3,14265…)

Наименование процедуры	Результат вычесления
МАТЕМАТИЧЕСКИЕ ПРОЦЕДУРЫ	
Inc(x)	Увеличивает "x" на 1 (x:=x+1;)
Dec(x)	Уменьшает "x" на 1 (x:=x-1;)
Inc(x, n)	Увеличивает "x" на n (x:=x+n;)
Dec(x, n)	Уменьшает "x" на n (x:=x-n;)
ФУНКЦИИ ПРЕОБРАЗОВАНИЯ ТИПОВ ПЕРЕМЕННЫХ	
Odd(x)	возвращает True если "x" - нечетное число
Chr(x)	возвращает символ по ASCII коду "x"
Ord(c)	ASCII код символа "c", порядковый номер символа "c"

Встроенные функции и операции Pascal

Арифметические операции

- * - умножение
- / - деление
- + - сложение
- - вычитание
- div - целая часть от деления
- mod - остаток от деления

Приоритет операций

*, /, MOD, DIV

+, -,

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

Условный оператор в языке Pascal

Условные операторы позволяют выбирать для выполнения те или иные части программы в зависимости от некоторых условий.

Синтаксис условного оператора:

```
if <условие> then
begin
…
end
else
begin
…
end;
```

Т.е. если условие верное, то выполняется блок операторов после `then`, иначе выполняется блок операторов после `else`.

Операторы сравнения

Используются в условиях.

Оператор	Операция	Тип результата
=	равно	boolean
<>	не равно	boolean
<	меньше	boolean
>	больше	boolean
<=	меньше или равно	boolean
>=	больше или равно	boolean

Например: `if a<=b then … else …`

Таблицы истинности логических операций

1 – true, 0 – false

0 and 0 = 0	0 or 0 = 0
0 and 1 = 0	0 or 1 = 1
1 and 0 = 0	1 or 0 = 1
1 and 1 = 1	1 or 1 = 1

Логические операции

NOT - логическое отрицание ("НЕ")
AND - логическое умножение ("И")
OR - логическое сложение ("ИЛИ")
XOR - логическое "Исключающее ИЛИ”

X	У	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Логические операции

Знак	Операция	Примеры
not	Логическое НЕ	`if not (x<y) then writeln('x >= y') else writeln('x<y');`
and	Логическое И	`if (x<y) and (y<z) then writeln('x < y < z');`
or	Логическое ИЛИ	`if (x=2) or (y=2) then writeln('Что-то равно 2');`

Оператор выбора *CASE*

Позволяет выбрать одно из нескольких возможных продолжений программы в зависимости от значения выражения:

```
case выражение of
  значение1 : оператор (группа операторов);
  значение2 : оператор (группа операторов);
  . . . . . .
  значениеN : оператор (группа операторов)
else оператор (группа операторов);
end;
```

Оператор выбора *CASE*

Пример 1:

```
Write('Введите  число: ');
  Readln( i );
  Case i of
      2, 4, 6, 8: Writeln('Четная цифра');
      1, 3, 5, 7, 9: Writeln('Нечетная цифра');
      10..100: Writeln('Число от 10 до 100');
    else
      Writeln ('Отрицательное число или больше 100')
  end;
```

Пример 2:

```
Write('Введите  номер месяца: ');
  Readln( MONTH );
  case MONTH of
       1, 2, 3 : writeln ('Первый квартал');
       4, 5, 6 : writeln ('Второй квартал');
       7, 8, 9 : writeln ('Третий квартал');
       10, 11, 12 : writeln ('Четвёртый квартал');
  end;
```

Операторы цикла

Цикл с предусловием while

Цикл – это последовательность операторов, которая может выполняться более одного раза.

Циклы с предусловием используются тогда, когда выполнение цикла связано с некоторым логическим условием. Оператор цикла с предусловием имеет две части: условие выполнения цикла и тело цикла.

При выполнении оператора while определенная группа операторов выполняется до тех пор, пока определенное в операторе while условие истинно. Если условие сразу ложно, то оператор не выполнится ни разу.

```
while <условие> do
begin
   группа операторов
end;
```

Цикл с предусловием while

При использовании цикла с предусловием надо помнить следующее:

- значение условия выполнения цикла должно быть определено до начала цикла;
- если значение условия истинно, то выполняется тело цикла, после чего повторяется проверка условия. Если условие ложно, то происходит выход из цикла;
- хотя бы один из операторов, входящих в тело цикла, должен влиять на значение условия выполнения цикла, иначе цикл будет повторяться бесконечное число раз.

Цикл с предусловием while

Задача: найти сумму чисел, введенных пользователем.

```
Program Summa;
Var
    i, N : integer;
    x, S : real;
Begin
  write('Сколько чисел для сложения? ');
  readln (N);
  S:=0;
  i:=1;
  while i<=N do
  begin
    write('Введите ',i,'-е число ');
    readln (x);
    S:=S+x;
    i:=i+1;
  end;
  write('Сумма введенных чисел равна ',s:5:2);
End.
```

Цикл с предусловием while

Задача: Найти сумму цифр в записи данного натурального числа;

```
Program SUM;
Var a,b,s:Integer;
Begin
     write('Введите число: ');
     Readln(a);
     s:=0;
     While a<>0 do
     begin
          b:=a mod 10;
          s:=s+b;
          a := a div 10;
     end;
    Writeln(s);
End.
```

Цикл с постусловием repeat

Отличительной особенностью данного цикла является то, что тело цикла выполняется в любом случае как минимум 1 раз, т.к. условие выхода из цикла проверяется после тела цикла.

```
repeat
  группа операторов
until <условие>;  {до тех пор, пока условие не будет верным}
```

Для выполнения в цикле repeat нескольких операторов не следует помещать эти операторы в операторные скобки begin ... end. Зарезервированные слова repeat и until действуют как операторные скобки.

Примеры:

a)
```
repeat
      read (Number);
      Sum := Sum+Number;
 until Number=-1;
```

b)
```
repeat
        i := i+1;
        writeln (Sqr(i))
  until i=-1;
```

Цикл с постусловием *repeat*

Задача. Определить, является ли введенное число простым.

```
Program Prostoe;
Var
  i, {возможный делитель}
  Number : integer; {исследуемое число}
Begin
  writeln (‘Какое число должно быть проверено? ‘);
  read (Number);
  i := 1;
  repeat
    i := i+1;
  until Number mod i = 0;
    if Number=i
      then
        writeln (Number,’ является простым‘)
      else
        writeln (Number,’ делится на ‘,i);
  readln;
End.
```

Цикл со счетчиком for

Цикл со счетчиком представляет такую конструкцию, в которой выполнение тела цикла должно повторяться заранее определенное число раз.

```
for i := A to B do
begin
    операторы
end;
```

Или

```
for i := A downto B do
begin
    операторы
end;
```

Переменная i – управляющая переменная или переменная цикла (целый тип),

A – начальное значение переменной цикла,

B – конечное значение переменной цикла.

Управляющую переменную цикла нельзя менять в теле цикла!!!

Цикл со счетчиком for

При переходе к обработке оператора цикла for управляющей переменной присваивается заданное начальное значение.

Затем в цикле выполняется исполнительный оператор (или составной оператор).

Каждый раз при выполнении исполнительного оператора управляющая переменная увеличивается на 1 (для for…to) или уменьшается на 1 (для for…downto).

Цикл завершается при достижении управляющей переменной своего конечного значения.

Цикл со счетчиком for

При переходе к обработке оператора цикла for управляющей переменной присваивается заданное начальное значение.

Затем в цикле выполняется исполнительный оператор (или составной оператор).

Каждый раз при выполнении исполнительного оператора управляющая переменная увеличивается на 1 (для for…to) или уменьшается на 1 (для for…downto).

Цикл завершается при достижении управляющей переменной своего конечного значения.

Цикл со счетчиком for

Примеры:

```
for i := 1 to n do
begin
    readln (Number);
    S := S +Number;
end;
```

```
for Range := Number+1 to Multi*3
do
    writeln (Sqrt(Range));
```

```
for Dlina := 15 downto 1 do
    writeln (Sqr(Dlina));
```

```
for x := 1 to 10 do
  for y := 1 to 10 do
    writeln (x,'*',y,'=',x*y);
```

Вложенный цикл со счетчиком for

Часто исполнительная часть одного из циклов For является новым оператором цикла For.

Структуры такого рода называются **вложенными** циклами.

При завершении внутреннего цикла управляющая переменная внешнего цикла увеличивается, а внутренний цикл начинается заново.

Повторение этих действий будет продолжаться до завершения внешнего цикла.

Приведенный ниже вложенный цикл печатает пары чисел, начиная от (1,1), (1,2),... и кончая (10,10):

```
for x:= 1 to 10 do
  for y:= 1 to 10 do
    writeln ('(',x,',',y,'), ');
```

Пример цикла for

Вычислить N! (факториал):

```
Program Faktorial;
Var n, i, f: integer;
Begin
     f:=1;
     Write('Введите n: ');
     Readln(n);
     For i:=2 to n do
          f:=f*i;
     Writeln(n,'!=',f);
End.
```

Операторы языка Pascal

Безусловный переход:

Goto <метка>;

Безусловные операции:

Break - досрочное завершение цикла;

Continue- новая итерация цикла без завершения предыдущей;

Exit - завершение текущего блока программы;

Halt(n) – завершение работы программы с кодом завершения n.