Переменные величины в ПАСКАЛЕ

Оператор присваивания состоит из знака присваивания := , слева от которого пишется переменная, а справа - число, переменная или выражение. При выполнении оператора присваивания компьютер "в уме" (то есть ничего не показывая на мониторе) вычисляет правую часть и присваивает вычисленное значение переменной, стоящей в левой части.

Например, после выполнения фрагмента программы:

. . . . $a:=2*3+4;$ $b:=a;$ $y:=a+b+1$. . .

компьютер будет знать, что a равно 10, b равно 10, y равно 21.

Описания переменных (VAR)

- **VAR** a,b : Integer;
  BEGIN
  a:=100;
  b:=20;
  WriteLn (a + b)
  END.

Первая строка называется **описанием** переменных величин *a* и *b*. Описание начинается со служебного слова VAR (от англ. variable - переменная). После него записываются имена всех переменных, встречающихся в программе с указанием через двоеточие типа значений, которые каждая переменная имеет право принимать.

Слово Integer (переводится ЦЕЛЫЙ) указывает, что *a* и *b* имеют право быть целыми числами, а не дробными.

Зачем нужно описание?

После того, как программист ввел программу в память, он приказывает компьютеру ее исполнить. Но компьютер при этом не сразу принимается выполнять программу, а сначала совершает **компиляцию**, то есть перевод программы с Паскаля на собственный машинный язык. Во время компиляции компьютер производит некоторые подготовительные действия, одним из которых является отведение в памяти места под переменные величины, упомянутые в программе. При этом компьютер "рассуждает" так: Раз в программе упомянута переменная величина, значит она в каждый момент времени будет иметь какое-то значение, которое, хочешь не хочешь, надо помнить. Лучше, чтобы не спутаться, заранее отвести в памяти определенное место для запоминания текущего значения каждой переменной величины и только потом уже выполнять программу. Будем называть место, отведенное в памяти под данную переменную, **ячейкой**.

- Значит, описание надо, ***чтобы перечислить компьютеру переменные, под которые он должен отвести ячейки в памяти.***

Integer и LongInt

- В Паскале принято правило, что если человек описал переменную, как Integer, то он разрешает ей принимать значения только целого числа. Число типа Integer занимает в памяти два байта. Значит, под переменные *a* и *b* компьютер отводит в памяти ячейки по два байта каждая. Два байта - это маленький объем памяти и уместиться в него может лишь небольшое целое число, а именно - число в диапазоне от -32768 до 32767. Если бы в предыдущем примере вам понадобилось взять a=40000, то Паскаль получил бы неправильную сумму.
- Для того, чтобы переменная имела право принимать значения больших целых чисел, она должна быть описана не как Integer, а как **LongInt** (сокращение от Long Integer - Длинное Целое). Под переменную типа LongInt компьютер отводит в памяти 4 байта и она может принимать значения в диапазоне от -2147483648 до 2147483647.

- **VAR** a,b,y : Integer;
  BEGIN
  a:= 10;
  b:= 6;
  y:= a+b+1;
  WriteLn (y+200)
  END.

ПОРЯДОК ИСПОЛНЕНИЯ ОПЕРАТОРОВ	ЧТО НАХОДИТСЯ В ЯЧЕЙКАХ ПАМЯТИ			ЧТО ВИДИМ НА ЭКРАНЕ
	a	b	y	
a:= 10	10	?	?	
b:= 6	10	6	?	
y:= a+b+1	10	6	17	
WriteLn (y+200)	10	6	17	217

- **Если в какое-нибудь место памяти или диска записывается новая информация, то старая информация, записанная там раньше, автоматически стирается, даже если она кому-то и нужна.**
- **VAR** f : Integer;
  BEGIN
  f:=30;
  f:=f+4;
  WriteLn (f)
  END.
- Оператор *f:=f+4* сначала вычислит правую часть *f+4* подставив туда вместо *f* его значение, взятое из ячейки, и получит 34. Затем число 34 будет записано в ячейку, отведенную под переменную, обозначенную в левой части, то есть опять в ячейку *f*. При этом старое значение 30 будет стерто.

Задания:

Определите без компьютера, что будет напечатано при выполнении следующих фрагментов программ:

- a:=100; a:=10*a+1; WriteLn (a)
- a:=100; a:=-a; WriteLn (a)
- a:=10; b:=25; a:=b-a; b:=a-b; WriteLn (a,' ',b)

Имена переменных

- Паскаль позволяет обозначать переменные не только буквами, но и целыми словами.

a:=3;
b:=4-a;
WriteLn (a,b+50)
	Summa:=3;
	ROBBY:=4-Summa;
	WriteLn (Summa,ROBBY+50)

Будем называть обозначение переменной **именем** или **идентификатором** этой переменной.

Именем переменной в Паскале может служить любая последовательность цифр, латинских букв и знака подчеркивания, не начинающаяся с цифры.

Примеры правильной записи имен:

- **a**
- **velichina**
- **zzz**
- **polnaja_summa**
- **tri_plus_dva**
- **s25**
- **k1**
- **_k1**
- **a1b88qq**

Примеры неправильной записи имен:

Математические возможности Паскаля

ДЕЙСТВИЕ	РЕЗУЛЬТАТ	СМЫСЛ
2 **+** 3	5	плюс
4 **-** 1	3	минус
2 ***** 3	6	умножить
10 **/** 5	2	разделить
17 **div** 5	3	целочисленное деление
17 **mod** 5	2	остаток от целочисленного деления

В Паскале в десятичных дробях принято вместо запятой ставить точку: *62.8*

Математические функции

ДЕЙСТВИЕ	РЕЗУЛЬТАТ	СМЫСЛ
Sqr (5)	25	возведение в квадрат
Sqrt (25)	5	корень квадратный
Pi	3.1415...	число пи
Frac (23.192)	0.192	дробная часть числа
Int (3.98)	3.0	целая часть числа
Round (5.8)	6	округление
Abs (-20)	20	абсолютная величина, модуль числа
Random	0.73088	случайное число из диапазона (0 - 1)
Random (200)	106	случайное целое число из диапазона (0 - 199)

Вещественные числа в Паскале

- ```
  VAR a,b,y : Integer;
  BEGIN
     a:=10;   b:=6;
     y:= a / b;
     WriteLn (y)
  END.
  ```
- Паскаль откажется выполнять эту программу, так как при делении целого на целое результат может получиться дробный, а это значит, что в ячейку y придется записывать дробное число. Но описание *VAR a,b,y : Integer* запрещает это делать.
- ```
  VAR      a,b:Integer;
           y:Real;
          BEGIN
            a:=10;   b:=6;
            y:=a / b;
               WriteLn (y)
          END.
  ```
- Под переменную типа Real Паскаль отводит в памяти ячейку размером в 6 байтов.

- В результате выполнения оператора *WriteLn (y)* должно получиться 1.666666
- Чтобы заставить Паскаль выводить вещественные числа в обычном, понятном виде, допишем кое-что в оператор вывода - *WriteLn (y :8:3).* Это значит, что мы хотим численное значение переменной *y* типа Real видеть на экране в привычном виде с 3 знаками после десятичной точки, а всё изображение числа не должно занимать больше 8 символов, включая целую часть, дробную часть, знак и десятичную точку. Этот оператор напечатает на экране **1.667**. Здесь напечатано действительно 8 символов (три пробела, предшествующие единице, видны вам, как пустое место). Вместо 8 и 3 в операторе программист может писать любые имеющие смысл числа.
- Поэкспериментируйте: *(y :38:3), (y :20:10), ('Результат равен', y :8:3)*.
-

Порядок составления программы:

- ***Задача:*** Известны размеры спичечной коробки: высота - 12.41 см., ширина - 8 см., толщина - 5 см. Вычислить площадь основания коробки и ее объем .
- **Программист сам должен знать решение задачи.** В нашем случае формулы для вычисления площади основания коробки и ее объема : площадь = ширина х толщина , объем = площадь х высота .
- **2. Нужно придумать имена переменным.** Имя переменной должно говорить о ее смысле. Если смыслом является ширина, называйте ее *shirina*.
- Удовлетворимся такими именами:
- shirina - ширина
  tol - толщина
  visota - высота
  pl - площадь
  V - объем

- **3. Нужно определить, какого типа будут переменные.** Поскольку ширина и толщина - целые, то и площадь будет целой. Высота и, следовательно, объем - вещественные. Первые две строки программы будут такими:
- *VAR shirina,tol,pl : Integer;*
- *visota,V : Real;*
- **4. Перед вычислениями нужно задать исходные данные решения задачи** *shirina:=8; tol:=5; visota:=12.41;*
- **5. Теперь нужно задать компьютеру действия, которые нужно проделать с исходными данными, чтобы получить результат.**
- *pl := shirina * tol;*
- *V := pl * visota;*
- **6. После получения результата его нужно напечатать**. Чтобы их узнать, человек в нашем

```
VAR   shirina,tol,pl :Integer;
      visota,V :Real;
BEGIN
      shirina:=8;   tol:=5;   visota:=12.41;
      pl := shirina * tol;
      V  := pl * visota;
      WriteLn (pl,'    ', V  :10:3)
END.
```

- **Программа напечатает два числа: 40 и 496.40**

Операторы ввода данных ReadLn и Read

- Задача: Сложить два числа - 20 и 16.

Сравним две программы решения:

- VAR a,b : Integer;
- BEGIN
- a:=20; b:=16;
- WriteLn (a+b)
- END.

Программы отличаются только одной строкой. Первая программа не требует пояснений - она печатает число 36. Во второй программе не сказано, чему равны a и b, а вместо этого включен оператор ReadLn ("читай строку«). Он приказывает компьютеру остановиться и ждать, когда человек введет с клавиатуры информацию, после чего продолжить работу. В частности,

- VAR a,b : Integer;
- BEGIN
- ReadLn (a,b);
- WriteLn (a+b)

- Человек должен на клавиатуре набрать число 20 (так как первым в списке оператора ReadLn стоит *a*), затем нажать клавишу пробела, затем набрать 16 и нажать клавишу Enter в знак того, что ввод чисел для ReadLn закончен.

Строковые переменные

- VAR a : String;

  BEGIN

  a:='Привет всем!';

  WriteLn (a)

  END.

- Описание *VAR a : String* говорит о том, что переменная a обязана иметь строковое (текстовое) значение, то есть ее значением будет не число, а произвольная цепочка символов, например, *Привет всем!*

Строковую переменную можно задавать не только оператором присваивания, но и оператором ReadLn:

```
VAR a : String;
BEGIN
 WriteLn ('Введите какое-нибудь слово');
 ReadLn (a);
 WriteLn ('Вы ввели слово  ' ,a)
END.
```

осуществляла бы такой диалог человека с компьютером:

- **КОМПЬЮТЕР ВЫВОДИТ НА ЭКРАН:**
  Здравствуй, я компьютер, а тебя как зовут?
- ***ЧЕЛОВЕК ВВОДИТ С КЛАВИАТУРЫ:***
  Коля
- **КОМПЬЮТЕР ВЫВОДИТ НА ЭКРАН:**
  Очень приятно, Коля. Сколько тебе лет?
- ***ЧЕЛОВЕК ВВОДИТ С КЛАВИАТУРЫ:***
  16
- **КОМПЬЮТЕР ВЫВОДИТ НА ЭКРАН:**

```
VAR   imya      :String;
      vozrast   :Integer;
BEGIN
 WriteLn ('Здравствуй, я компьютер, а тебя как зовут?');
 ReadLn (imya);
 WriteLn ('Очень приятно, ', imya, ' . Сколько тебе лет?');
 ReadLn (vozrast);
 WriteLn ('Ого! Целых ' ,vozrast, ' лет! Ты уже совсем взрослый!')
END.
```

Константы

- Кроме переменных величин в тексте программы мы встречаем ***константы.*** Это те значения, которые или присваиваются переменным, или встречаются в выражениях, или сравниваются с выражениями. Например:
- x:=25 **здесь числовая константа - 25**
- slovo:=’Волга’

  здесь строковая константа - Волга
- simvol:=’!’ **здесь символьная константа - !**
- y:=(x+1) / (200*x - 0,3)

Символьный тип данных Char

- Описание $VAR\ \ a,b:\ \ Char$ означает, что переменные a и b имеют право принимать значения любых символов, с которыми может работать компьютер.
- Мы можем записать $a:='Л';\ b:='+'$, что означает приказ присвоить переменной a значение символа $Л$, а переменной b - значение символа $+$.
- Мы можем записать $ReadLn\ \ (a)$, что означает приказ компьютеру ждать ввода с клавиатуры любого одного символа и присвоить его значение переменной a.
- Мы можем записать $WriteLn\ (b)$ и на экране появится плюсик.

- **Пример.** Программа, переворачивающая любое трехбуквенное слово, введенное человеком с клавиатуры:
- **VAR** c1,c2,c3: Char;
  BEGIN
  ReadLn (c1,c2,c3);
  WriteLn (c3,c2,c1)
  END.
- Если мы по оператору ReadLn введем символы $ТОК$, то оператор WriteLn напечатает $КОТ$. При вводе нескольких символов одним оператором ReadLn, **все символы набираются на клавиатуре подряд, без пробелов**, которыми мы привыкли разделять при вводе числовые данные. **После ввода последнего символа нажимаем клавишу Enter.** Таким образом, ввод трех символов одним оператором ReadLn не отличается от ввода одной трехсимвольной строки. Вообще, тип Char похож на тип String. Строка состоит из символов, да и сам символ - это как бы очень короткая строка длиной в один символ.