Величины и выражения. Оператор присваивания

Понятие величины

Любая программа предназначена для обработки некоторых данных, которые хранятся в памяти компьютера в виде величин.

То есть, под величинами в языках программирования обычно понимаются именованные области памяти для хранения данных, с которыми работает программа

Величины делятся на:

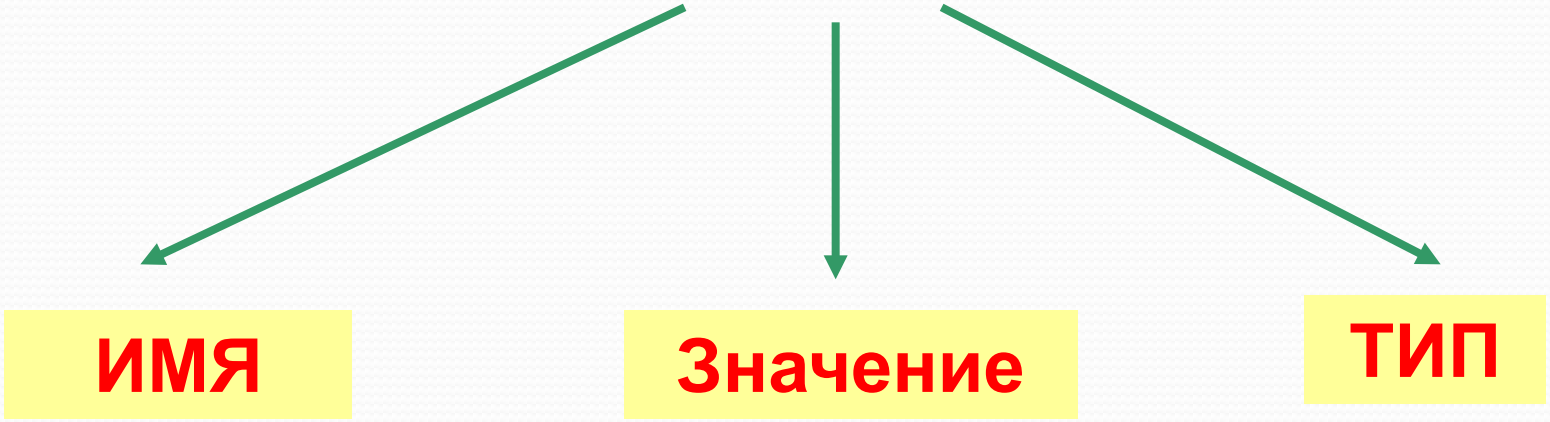
КОНСТАНТЫ и ПЕРЕМЕННЫЕ

Константа – постоянная величина, то есть ее значение не изменяется в процессе работы программы.

Переменная – это величина, значение которой можно изменять во время работы программы.

- Величина имеет три основных характеристики:

ВЕЛИЧИНА



Имя величины – это обозначение («метка», «бирка») той области памяти, которая отведена под данную величину.

Значение величины – это содержимое соответствующей величине области памяти

Тип величины – это характеристика, определяющая множество допустимых значений величины и множество допустимых операций над ней, а также форму внутреннего представления величины в памяти компьютера

Имена переменных

Могут включать

- **буквы**
- **знак подчеркивания _**
- **цифры 0-9**

Имя не может начинаться с цифры!

В зависимости от языка программирования прописные и строчные буквы в именах могут различаться или нет:

Различаются – в языке Си, например: **summa, Summa, SUMMA** - три разные переменные

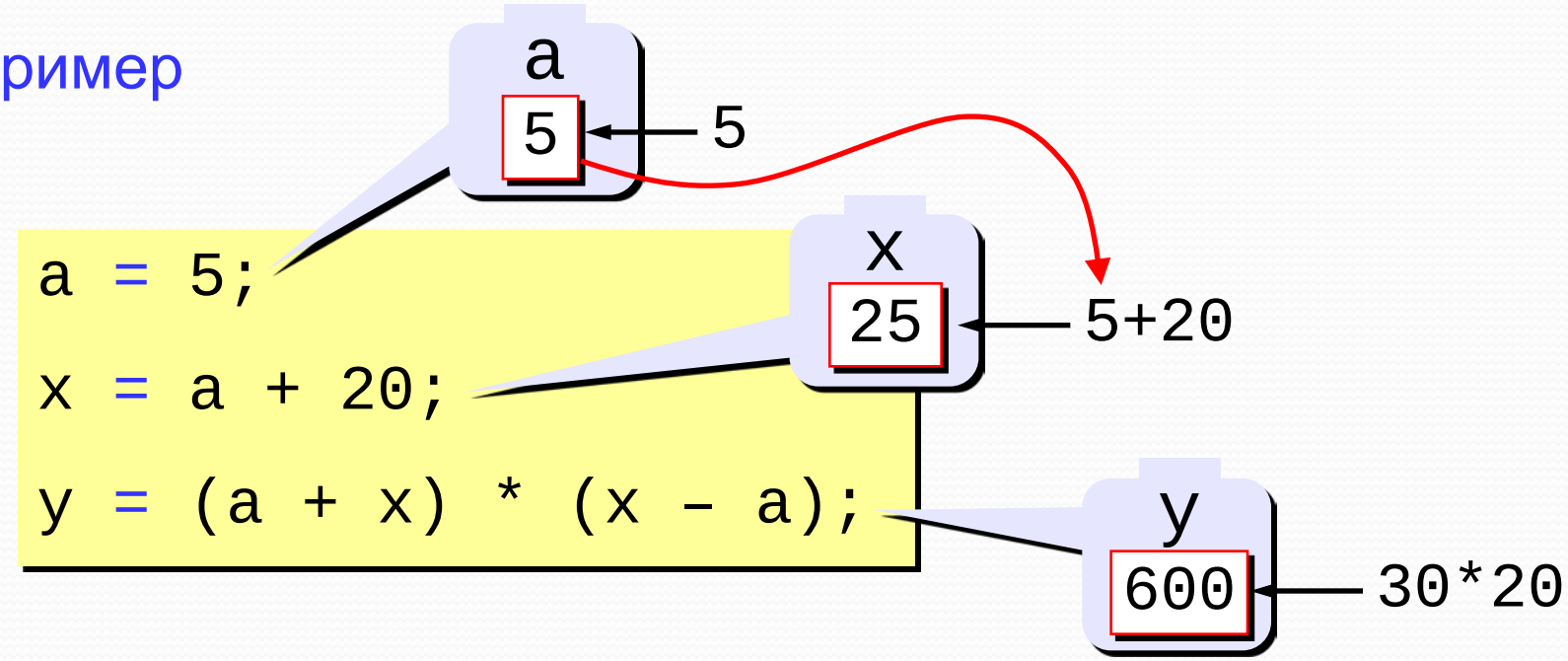
А в **языке Паскаль – не различаются**

Оператор присваивания

Оператор – это команда языка программирования высокого уровня.

Оператор присваивания (обозначается = или :=) служит для изменения значения переменной.

Пример



Оператор присваивания (язык Си)

Общая структура:

куда записать

что

имя переменной = *выражение*;

Арифметическое выражение может включать

- константы
- имена переменных
- знаки арифметических операций:

+ - * / %

умножение

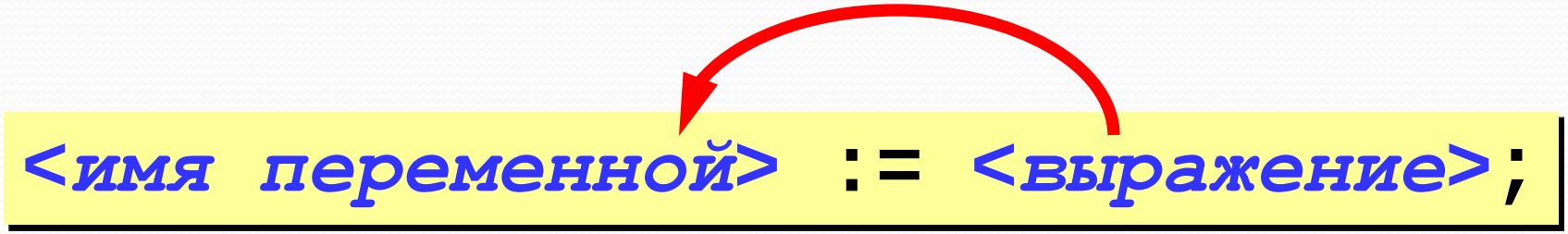
деление

остаток от деления

- вызовы функций
- круглые скобки ( )

? **Для чего служат круглые скобки?**

Оператор присваивания (язык Паскаль)



Арифметическое выражение может включать

- константы
- имена переменных
- знаки арифметических операций:

`+` `-` `*` `/` **`div`** **`mod`**

умножение — деление — деление нацело — остаток от деления

- вызовы функций
- круглые скобки ( )

Объявление переменных

Переменная должна быть объявлена до первого обращения к ней. По умолчанию не допускается в программном коде применять имена не объявленных переменных.

Объявить переменную = определить ее имя, тип, начальное значение, и выделить ей место в памяти.

Переменные в языке Паскаль

Простейшие типы:

- integer { целый }
- real { вещественный }
- char { один символ - символьный }
- boolean { логический }

Объявление переменных (пример):

```
var  a, b: integer;
     Q: real;
     s1, s2: char;
```

Объявление переменных в языке Си

```
main()
{
int a;
float b, c;
int Tu104, Il86=23, Yak42;
float x=4.56, y, z;
char c, c2='A', m;
}
```

Ввод и вывод данных. Линейные программы

Линейной называется программа, последовательность записи команд в которой совпадает с последовательностью их выполнения.

Простейшие линейные программы обычно сводятся к тому, что пользователь с клавиатуры вводит значения некоторых величин (числа или символы), затем эти значения обрабатываются, обычно с помощью команды присваивания, и полученный результат выводится на экран компьютера.

Таким образом, в любом языке программирования должны быть команды, позволяющие организовать обмен информацией между человеком и компьютером – ввод исходных данных и вывод результатов.

Организация ввода и вывода в программах на языке Паскаль.

Оператор вывода

Формат оператора:

Write[ln](<список вывода>);

Замечание: *Здесь и далее в квадратных скобках будем записывать необязательную часть каких-либо конструкций языка (она может быть, а может и отсутствовать).*

В данном случае сочетание **ln** означает перевод курсора на следующую строку экрана по окончании вывода всех элементов списка. При её отсутствии по окончании вывода курсор остается в текущей позиции строки вывода.

Список вывода **может содержать три вида элементов:**

1) **Имена переменных**
2) **Константы (числовые и строковые)**
3) **Выражения**

Разделителем в списке служит запятая.

При выполнении оператора вывода на экран соответственно выводятся:

1) **Значение переменной**
2) **Сама константа**
3) **Значение выражения**

Примеры

```
writeln ( a ); { вывод значения
                 переменной a и переход
                 на новую строчку}

 writeln ( 'Привет!' ); { вывод текста-
                          строковой константы}

 writeln ( 'Ответ: ', c );   { вывод
          текста и значения переменной c}

 writeln ( a, '+', b, '=', c );
```

Оператор ввода

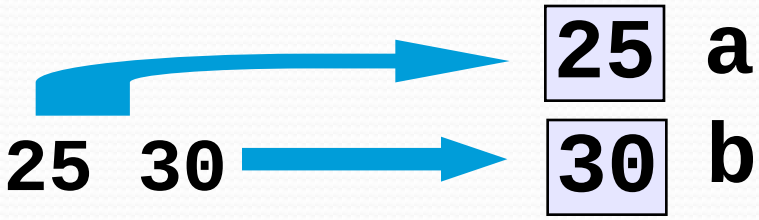
Формат оператора:

Read[ln](<список имен переменных>);

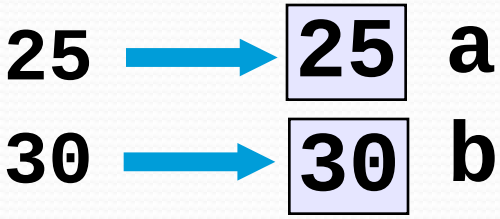
```
readln( a, b ); { ввод значений
                  переменных a и b}
```

Как вводить два числа?

через пробел:



через *Enter*:



Сложение двух чисел

Задача. Ввести два целых числа и вывести на экран их сумму.

Простейшее решение:

```
program sum2;
var a, b, c: integer;
begin
   readln( a, b );
   c := a + b;
   writeln( c );
end.
```

Форматы вывода

```
program qq;
var i: integer;
    x: real;
begin
  i := 15;
  writeln ( '>', i, '<' );
  writeln ( '>', i:5, '<' );
  x := 12.345678;
  writeln ( '>', x, '<' );
  writeln ( '>', x:10, '<' );
  writeln ( '>', x:7:2, '<' );
 end.
```

всего символов

```
>15<
>   15<

>1.234568E+001<
> 1.23E+001<
>  12.35<
```

всего символов

в дробной части

Полное решение

```
program sum2;
var a, b, c: integer;
begin
   write('Введите два целых числа: ');
   readln( a, b );
   c := a + b;
   writeln ( a, '+', b, '=', c );
end.
```

Протокол:

```
Введите два целых числа
25 30
25+30=55
```

компьютер

пользователь

Организация ввода и вывода в программах на языке Си-Шарп

В программировании существует специальное понятие консоль, которое обозначает клавиатуру при вводе и монитор при выводе

Ввод данных

Для того чтобы получить данные, вводимые пользователем вручную (то есть с консоли), применяются команды

```
<строковая переменная>=Console.ReadLine();
```

Для того чтобы преобразовать к нужному типу данных используем объект Convert

```
    <переменная целого типа> =
Convert.ToInt32(<строковая переменная>);
    <переменная действ типа> =
Convert.ToDouble(<строковая переменная>);
```

```
<переменная логического типа> =
Convert.ToBoolean(<строковая переменная>);

<переменная целого типа> =
Convert.ToByte(<строковая переменная>);

<переменная символьного типа> =
Convert.ToChar(<строковая переменная>);

<переменная целого типа> =
Convert.ToInt64(<строковая переменная>);
```

Пример

```
Int32 x; Double y; string s;
//с промежуточной строковой переменной
 s = Console.ReadLine();
 x = Convert.ToInt32(s);

//без промежуточной строковой переменной
y = Convert.ToDouble(Console.ReadLine());
```

Типы вводимых значений должны совпадать с типами указанных переменных,

Для того, чтобы обеспечить ожидание ввода любой клавиши в конце программы используют

```
Console.ReadKey();
```

Вывод данных

Для того чтобы вывести на экран какое-либо сообщение, применяются конструкции **Console.Write** или **Console.WriteLine**

Первая из них, напечатав на экране все, о чем ее просили, оставит курсор в конце выведенной строки, а вторая переведет его в начало следующей строчки.

Общий формат оператора:

```
Console.Write[Line](<список вывода> );
```

Примеры

```
Console.WriteLine(s); // переменная
Console.WriteLine(55.3); // константа
Console.WriteLine(y*3+7); // выражение
Console.Write(z); // переменная
Console.Write(-5.3); // константа
Console.Write(i*3+7/j); // выражение
```

```
Console.WriteLine("Это число A={0} далее
B={1} и, наконец их сумма {2}", a, b, a + b);
```

Для форматирования числовых результатов можно использовать метод **String.Format** или метод **Console.Write**, вызывающий метод String.Format.

Формат задается с помощью строк формата. Спецификация формата:**{N,M:Axx}**, где

N указывает позицию элемента в списке выводимых переменных (нумерация начинается с 0);

M - задаёт ширину области, в которую будет помещено форматированное значение, если M отсутствует или отрицательно, значение будет выровнено влево, в противном случае - вправо;

Axx - является необязательной строкой форматирующих кодов, которые используются для управления форматированием чисел, даты и времени, денежных знаков и т.д.

Рассмотрим поддерживаемые строки стандартных форматов.

Строка формата принимает следующую форму:

Axx, где A — описатель формата, а xx — описатель точности.

Описатель формата управляет типом форматирования, применяемым к числовому значению, а описатель точности управляет количеством значащих цифр или десятичных знаков форматированного результата:

G или g Общий формат

```
Console.Write("{0:G}", 2.5);  // 2.5
```

- **N или n Числовой формат**

```
Console.Write("{0:N}", 2500000);   //2,500,000.00
```

- **X или x Шестнадцатеричный формат**

```
Console.Write("{0:X}", 250);
Console.Write("{0:X}", 0xffff);    // FA FFFF
```

- **D или d Десятичный формат**

```
Console.Write("{0:D5}", 25);   // 00025
```

- **E или e Инженерный формат**

```
Console.Write("{0:E}", 250000);   //2.500000E+005
```

- **F или f Формат с фиксированной запятой**

```
Console.Write("{0:F2}", 25);
Console.Write("{0:F0}", 25);  // 25.00 25
```

- **C или c Валюта**

```
Console.Write("{0:C}", 2.5);
Console.Write("{0:C}", -2.5); // 2.50р. (-2.50р.)
```

Пример

```
Int32 x; Double y; string s;

Console.Write("Введите X=");
    s = Console.ReadLine();
    x = Convert.ToInt32(s);
 Console.Write("Введите Y=");
    s = Console.ReadLine();
    y = Convert.ToDouble(s);
Console.WriteLine("Произведение X*Y= {0,5:g}", x * y);
            Console.ReadKey();
```

Арифметические и логические операции в Си-шарп

Все операции делятся на два типа: унарные и бинарные.

К унарным относятся операции, в которых участвует один операнд.

В бинарных операциях – два операнда.

Операнд – это данные, которые принимают участие в операции.

Бинарные операции

Операция	Запись
Сложение	a + b
Вычитание	a - b
Деление	a / b
Умножение	a * b
Нахождение остатка от деления	a % b

При делении двух целых чисел результатом также будет целое число. Например при делении 9/5 результатом будет число 1.

Чтобы получить точный результат с десятичной точкой, нужно чтобы делимое и/или делитель были типа float или double.

Чтобы явно присвоить тип, можно поместить десятичный разделитель после числа, как показано в следующем примере.

Пример 1:

```
Console.WriteLine(5 / 2);
Console.WriteLine(5D / 2);
Console.WriteLine(5 / 2.1);
Console.WriteLine(5.1 / 2);
Console.WriteLine(-5 / 2);
 /* напечатается
 2
 2.5
 2.38095238095238
 2.55
 -2 */
```

Оператор % - остаток от целочисленного деления, например A=Y % 6;

Пример 2:

```
Console.WriteLine(24 % 6);
Console.WriteLine(24 % 7);
Console.WriteLine(7 % 7);
Console.WriteLine(8 % 12);

/*напечатается
0
3
0
8  */
```

Унарные операции в Си-шарп

Унарных арифметических операторов в Си-шарп есть всего два:

инкрементация «++»
и декрементация «--»;

Инкрементация увеличивает операнд на единицу, а декрементация - уменьшает на единицу:

```
int a = 0, b = 5;
  a++;  // a=1;
  b--;  // b=4
```

Инкрементация и декрементация может быть префиксной и постфиксной. При префиксной форме оператор стоит перед операндом, а при постфиксной- после.

Префиксная форма сначала увеличивает (уменьшает) значение, и после этого выполняются остальные действие,

а при постфиксной форме наоборот - сначала выполнится все действия, а после увеличится (уменьшится) значение:

```
int a = 2, b = 3, c, d = 3;
    c = a + ++b; // c = 6, сначала инкремент, потом сложение
    c = a + d++; // c = 5, сначала сложение, потом инкремент
```

Везде, где можно использовать инкрементацию/декрементацию стоит это делать, так как она работает быстрее оператора сложения/вычитания.

В Си-шарп также есть возможность использования краткой формы выражения:

```
int a = 2, b = 3;
    a += b; // равноценно выражению a = a + b;
    a -= b; // равноценно выражению a = a - b;
    a *= b; // равноценно выражению a = a * b;
    a /= b; // равноценно выражению a = a / b;
    a %= b; // равноценно выражению a = a % b;
```

```
 i += 7 * j; // Эквивалентно i = i + 7 * j;
m /= 3 + k; // Эквивалентно m = m / (3 + k);
```

Логические операции в Си-шарп

Логические операции в Си-шарп служат для работы с логическим типом данных (bool), который может принимать только два значения – true или false. Их можно разделить на две категории:

простые логические операции
и операции сравнения.

Логические операции (&&, ||, !, ^) применимы только к значениям типа bool. Их результатом также служат величины типа bool.

! – оператор «НЕ» является унарным и возвращает противоположное значение операнда:

```
bool a, b = true, c = false;
a = !b; // a = false
a = !c; // a = true
```

|| - оператор «ИЛИ» является бинарным и возвращает false только тогда, когда оба операнда равны false, в остальных случаях результат будет true:

```
bool a, bTrue = true, bFalse = false;
a = bFalse || bFalse;  // a = false
a = bFalse || bTrue;  // a = true
a = bTrue || bFalse;  // a = true
a = bTrue || bTrue;  // a = true
```

&& - оператор «И» является бинарным и возвращает true только тогда, когда оба операнда равны true, в остальных случаях результат будет false:

```
bool a, bTrue = true, bFalse = false;
a = bFalse && bFalse; // a = false
a = bFalse && bTrue; // a = false
a = bTrue && bFalse; // a = false
a = bTrue && bTrue; // a = true
```

^ исключающее или

F=A ^ B;

возвращает true только тогда, когда один операнд равен true, а другой – false. В остальных случаях результат будет false (оба true или оба false).

К операциям сравнения относятся:

Операция	Название
>	больше
<	меньше
>=	больше или равно
<=	меньше или равно
==	равно
!=	неравно

Пример:

```
bool a;
  int b = 2, c = 3, d = 2;
  a = b > c; // a = false
  a = b < c; // a = true
  a = b >= c; // a = false
  a = b >= d; // a = true
  a = b == c; // a = false
  a = b == d; // a = true
  a = b != c; // a = true
```

Приоритет (уровень значимости) операторов

Если выражения содержат операторы различных категорий, они вычисляются по следующим правилам:

Если в выражении расставлены скобки, то вычисления производятся в порядке, известном всем еще с начальной школы: чем меньше глубина вложенности скобок, тем позже вычисляется заключенная в них операция.

Если же скобок нет, то сначала вычисляются значения операций с более высоким приоритетом, затем - с менее высоким.

Несколько подряд идущих операций одного приоритета вычисляются в последовательности "слева направо".

1) x++, x--
2) унарные + и - , !, ++x, --x
3) /, *, %
4) +, -
5) < , > , <= , >=
6) ==, !=
7) ^
8) &&
9) ||
10) = , *= , /= , %= , += , -=