

ПЯВУ. Лекция 4.

Основы программирования.

А.М. Задорожный

Контрольные вопросы

1. Что такое Цикл в программировании?
2. Какой оператор C# позволяет организовать цикл?
3. Что указывается в круглых скобках оператора while?
4. Каким типом в C# представляются действительные числа?
5. Почему этот тип не представляет действительных чисел в математическом смысле?
6. Могут ли в одном выражении участвовать числовые данные разных типов?
7. Можно ли переменной типа double присвоить значение типа int? А наоборот?

Содержание

1. Текстовые данные. Типы:
 - a. Строки (string)
 - b. Символы (char)
2. Оператор цикла for
3. Сложение и сравнение строк
4. Ввод числовых данных с консоли.
Преобразование строк в числа.
5. Оператор do-while

Текстовая информация

- Важный вид общения между человеком и компьютером, а так же компьютером и компьютером (ASCII)
- Консоль работает ТОЛЬКО с текстовыми данными
- Даже если текст представляет число, для компьютера он остается текстом!
(Для текста $1 + 2 \Rightarrow 12$, для чисел $1 + 2 \Rightarrow 3$)

Тип String

- Для работы с текстовыми данными в C# существует тип String
- String – произвольная последовательность СИМВОЛОВ
- Строковые литералы задаются в кавычках - **""**

Специальные символы внутри строкового литерала

- \" – двойная кавычка
- \n – конец строки (переход на другую строку)
- \t – табуляция
- \b – беер
- \\ - обратный слеш

Примеры строк

```
string s = "Здравствуй, Мир!";
```

```
string t = " "; // Не пустая строка. Пробел – это символ  
string t = ""; // Пустая строка
```

```
Console.WriteLine("s = \"{0}\"", s );  
// s = "Здравствуй, Мир!"
```

```
Console.WriteLine("c:\\temp\\1.txt");  
// c:\temp\1.txt
```

Тип char

- Каждый отдельный символ строки имеет тип char (character - символ);
- Литералы для типа char задаются в одинарных кавычках - `'`.
- Символьный литерал может включать ТОЛЬКО 1 символ

Примеры символьных литералов

- `char c = 'a';`
- `char c = ' ';` // ~~`char c = " ";`~~ Пробел
- `char t = '\t';` // символ табуляции
- `char q = '\''` //Одинарная кавычка
- `char slash = '\\';` // Это 1 символ

Длина строки

- Свойство `Length` – определяет длину строки (количество символов в строке);

```
string s = "Здравствуй, Мир!";  
Console.WriteLine(s.Length); // 16
```

Отдельные символы строки

- Из строки можно ‘прочитать’ отдельный символ - операция []
- В квадратных скобках указываем номер символа
- **Символы строки нумеруются с 0**

```
string s = “Здравствуй, Мир!”;  
Console.WriteLine(s[0]); // З
```

Контрольные вопросы

1. Если строка – это последовательность символов. Может ли быть строка короче 1-го символа?
2. Строка состоит из двух слов, каждое из 5 букв. Сколько символов в строке?
3. Сколько символов в строке “\t”?
4. Как внутри строкового литерала задать символ двойной кавычки - “?”
5. Какой номер имеет последний символ в строке “12345”?

Оператор цикла for

- While – не единственный оператор цикла в C#.
- Более сложный оператор for

Задача: найти сумму N первых натуральных чисел.
Входные данные N

```
int sum = 0;
for(int i=1; i<=N; i = i + 1)
{
    sum = sum + i;
}
```

For и while

```
int sum = 0;
for(int i=1; i<=N; i = i + 1)
{
    sum = sum + i;
}
```

```
-----
int sum = 0, i = 1;
while(i<=N)
{
    sum = sum + i;
    i = i + 1;
}
```

Структура оператора for

```
for(<инициализатор>; <условие>; <итератор>)  
{  
    <тело цикла>  
}
```

Инициализатор выполняется 1 раз перед 1-ой итерацией

Итератор выполняется в конце каждой итерации

Условие – булевское выражение. Вычисляется перед каждой итерацией

Тело цикла выполняется на каждой итерации

Выполнение оператора for

```
for(<инициализатор>; <условие>; <итератор>)  
{  
    <тело цикла>  
}
```

1. Выполняются команды инициализатора
2. Вычисляется условие.
 - a) Если оно ложно, то выполнение оператора завершается и управление переходит к следующей инструкции.
 - b) Если условие истинно, то выполняется тело цикла.
 - i. Выполняется итератор.
 - ii. Переходим к п. 2.

Циклы и строки

Задача: Вывести каждый символ *строки* в отдельную строку на консоль

```
String s = "....."; // Входные данные
```

```
for(int i = 0; i < s.Length; i = i + 1)  
    Console.WriteLine(s[i]);
```

Циклы и строки 2

Задача: подсчитать количество пробелов в строке.

```
String s = "....."; // Входные данные
```

```
int n = 0; // аккумулятор  
for(int i = 0; i < s.Length; i = i + 1)  
{  
    if( s[i] == ' ')  
        n = n + 1;  
}
```

```
// Здесь в переменной n содержится количество  
пробелов.
```

Сложение строк

- Строки можно складывать:
“12345” + “67890” => “1234567890”
- Умножать, делить и вычитать строки
нельзя



Строки вообще нельзя изменять!

v

Если нужно получить строку,
отличающуюся от исходной, нужно
‘строить’ новую строку

Сравнение строк

- **Символы и строки** можно сравнивать операциями $==$ и $!=$
 $S == T$ или $S != T$.

Строки сравниваются 'посимвольно', т.е. сначала сравнивают первый символ S с первым символом T .

- Если совпали, то движемся дальше, если нет, то строки не равны.
- Если одна строка закончилась, а другая нет, то строки не равны.

✓

Строки равны только если у них одинаковое количество символов и все они (символы) попарно совпадают.

Циклы и строки 3

Задача: удалить все пробелы из исходной строки.

```
String s = ".....", t = "";
```

 // t - аккумулятор

```
for(int i = 0; i < s.Length; i = i + 1)
{
    if( s[i] != ' ')
        t = t + s[i];
}
```

Контрольные вопросы

1. Объясните, чем оператор `for` удобнее `while` для циклов с итератором (счетчиком).
2. Сколько секций в управляющей строке оператора `for`? За что они отвечают?
3. Какие секции оператора `for` выполняются обязательно?
4. Как узнать количество символов в строке?
5. Какой номер имеет последний символ строки?
6. Могут ли равные строки иметь различную длину?
7. Что можно сказать о строке `t`, если известно, что `s == s + t`?
8. Что можно сказать о строках `s` и `t`, если известно, что `s + t == t + s`?
9. Что предстоит сделать, если в строке `s` нужно удалить третий **СИМВОЛ**?

Ввод данных и строки

- С консоли читаются только текстовые данные – строки (ReadLine).
- Что бы превратить строку в число, у числовых типов существуют методы Parse (разбери) и TryParse (попытайся разобрать)

```
string s = Console.ReadLine();
```

Далее:

```
int x = int.Parse(s);
```

Или

```
int x;  
bool res = int.TryParse(s, out x);
```

Ввод чисел. Строгий, но дружелюбный

Пример кода для ввода целого числа 'a'.

```
int a;  
Console.Write("a=?");  
while(!int.TryParse(Console.ReadLine(), out a))  
    Console.Write("a=?");
```


Пояснение кода

Console.**Write**("a=?") выведет текст "a=?" и оставит курсор сразу за знаком вопроса.

Начнется цикл

```
while(!int.TryParse(Console.ReadLine(), out a))  
    Console.Write("a=?");
```

Оператор Console.ReadLine() прочтет строку с консоли и передаст ее первым параметром в TryParse.

TryParse попытается сформировать из строки целое число и поместить результат в 'a'

Если это удастся, то оператор вернет true, и цикл закончится.

Если нет, то оператор вернет false. Выполнится тело цикла (снова выведется строка "a=?") и т.д.

Оператор do-while

Рассмотренная задача – пример, в котором удобнее применить оператор цикла do-while.

```
int a;  
do  
{  
    Console.Write("a=?");  
}while(!int.TryParse(Console.ReadLine(), out a))
```

Он отличается от while тем, что сначала выполняется тело цикла (do), а потом проверяется условие продолжения цикла. Такие операторы называются операторами цикла с ПОСТУСЛОВИЕМ (в отличие от с ПРЕДУСЛОВИЕМ)

Тело оператора с постусловием обязательно выполнится хотя бы один раз. В нашем случае это позволило избежать повторения строки Console.**Write**("a=?");.

Контрольные вопросы

1. Какого типа данные можно ввести с консоли операцией `ReadLine`?
2. Как запрограммировать ввод числовых данных?
3. Как сделать ввод числовых данных устойчивым к ошибкам?
4. Опишите различия операторов `while` и `do-while`.
5. Когда удобно применять `do-while`?