Теория и практика программирования

Шпилёв Пётр Валерьевич

Санкт-Петербургский государственный университет
Математико-механический факультет
Кафедра статистического моделирования

Лекция 2

Санкт-Петербург
2015 г.

e-mail: pitshp@hotmail.com

Пример 1

```
Console.WriteLine("Вы заказали " + 2 +
"предмета по цене" + 3 + "$ каждый.");
```

Пример 2

```
Console.WriteLine("Деление 10/3 дает:" + 10.0/3.0);
```

Пример 1

```
Console.WriteLine("Вы заказали " + 2 +
"предмета по цене" + 3 + "$ каждый.");
```

Результат: Вы заказали 2 предмета по цене 3$ каждый.

Пример 2

```
Console.WriteLine("Деление 10/3 дает:" + 10.0/3.0);
```

Пример 1

```
Console.WriteLine("Вы заказали " + 2 +
"предмета по цене" + 3 + "$ каждый.");
```

Пример 2

```
Console.WriteLine("Деление 10/3 дает:" + 10.0/3.0);
```

Пример 1

```
Console.WriteLine("Вы заказали " + 2 +
"предмета по цене" + 3 + "$ каждый.");
```

Пример 2

```
Console.WriteLine("Деление 10/3 дает:" + 10.0/3.0);
```

Результат: Деление 10/3 дает: 3.333333333333.

Общий вид

WriteLine("форматирующая строка", arg0, arg1, ... , argN);

форматирующая строка

"форматирующая строка" = Текст {спецификатор формата} текст {спецификатор формата} ...

Общий вид спецификатора формата

{argnum, width: fmt}, где argnum — номер выводимого аргумента, начиная с нуля; width — минимальная ширина поля; fmt — формат.

Общий вид

WriteLine("форматирующая строка", arg0, arg1, ... , argN);

форматирующая строка

"форматирующая строка" = Текст {спецификатор формата} текст {спецификатор формата} ...

Общий вид спецификатора формата

{argnum, width: fmt}, где argnum — номер выводимого аргумента, начиная с нуля; width — минимальная ширина поля; fmt — формат.

Общий вид

WriteLine("форматирующая строка", arg0, arg1, ... , argN);

форматирующая строка

"форматирующая строка" = Текст {спецификатор формата} текст {спецификатор формата} ...

Общий вид спецификатора формата

{argnum, width: fmt}, где argnum — номер выводимого аргумента, начиная с нуля; width — минимальная ширина поля; fmt — формат.

Общий вид

WriteLine("форматирующая строка", arg0, arg1, ... , argN);

форматирующая строка

"форматирующая строка" = Текст {спецификатор формата} текст {спецификатор формата} ...

Общий вид спецификатора формата

{argnum, width: fmt}, где argnum — номер выводимого аргумента, начиная с нуля; width — минимальная ширина поля; fmt — формат.

Пример 1: Console.WriteLine("В феврале {0} или {1} дней.", 28, 29);

Общий вид

WriteLine("форматирующая строка", arg0, arg1, ... , argN);

форматирующая строка

"форматирующая строка" = Текст {спецификатор формата} текст {спецификатор формата} ...

Общий вид спецификатора формата

{argnum, width: fmt}, где argnum — номер выводимого аргумента, начиная с нуля; width — минимальная ширина поля; fmt — формат.

Пример 1: Console.WriteLine("В феврале {0} или {1} дней.", 28, 29);

Результат: В феврале 28 или 29 дней

Общий вид

WriteLine("форматирующая строка", arg0, arg1, ... , argN);

форматирующая строка

"форматирующая строка" = Текст {спецификатор формата} текст {спецификатор формата} ...

Общий вид спецификатора формата

{argnum, width: fmt}, где argnum — номер выводимого аргумента, начиная с нуля; width — минимальная ширина поля; fmt — формат.

Пример 1: Console.WriteLine("В феврале {0} или {1} дней.", 28, 29);

Результат: В феврале 28 или 29 дней

Пример 2: Console.WriteLine("Деление 10/3 дает: { 0:#.##}", 10.0/3.0);

Общий вид

WriteLine("форматирующая строка", arg0, arg1, ... , argN);

форматирующая строка

"форматирующая строка" = Текст {спецификатор формата} текст {спецификатор формата} ...

Общий вид спецификатора формата

{argnum, width: fmt}, где argnum — номер выводимого аргумента, начиная с нуля; width — минимальная ширина поля; fmt — формат.

Пример 1: Console.WriteLine("В феврале {0} или {1} дней.", 28, 29);

Результат: В феврале 28 или 29 дней

Пример 2: Console.WriteLine("Деление 10/3 дает: { 0:#.##}", 10.0/3.0);

Результат: Деление 10/3 дает: 3,33

Таблица 1: Управляющие последовательности символов

Управляющая последовательность	Описание
\a	Звуковой сигнал (звонок)
\b	Возврат на одну позицию
\f	Перевод страницы (переход на новую страницу)
\n	Новая строка (перевод строки)
\r	Возврат каретки
\t	Горизонтальная табуляция
\v	Вертикальная табуляция
\0	Пустой символ
\'	Одинарная кавычка
\"	Двойная кавычка
\\	Обратная косая черта

Таблица 1: Управляющие последовательности символов

Управляющая последовательность	Описание
\a	Звуковой сигнал (звонок)
\b	Возврат на одну позицию
\f	Перевод страницы (переход на новую страницу)
\n	Новая строка (перевод строки)
\r	Возврат каретки
\t	Горизонтальная табуляция
\v	Вертикальная табуляция
\0	Пустой символ
\'	Одинарная кавычка
\"	Двойная кавычка
\\	Обратная косая черта

Пример: `Console.WriteLine("Первая строка \n Вторая строка");`

Таблица 1: Управляющие последовательности символов

Управляющая последовательность	Описание
\a	Звуковой сигнал (звонок)
\b	Возврат на одну позицию
\f	Перевод страницы (переход на новую страницу)
\n	Новая строка (перевод строки)
\r	Возврат каретки
\t	Горизонтальная табуляция
\v	Вертикальная табуляция
\0	Пустой символ
\'	Одинарная кавычка
\"	Двойная кавычка
\\	Обратная косая черта

Пример: Console.WriteLine("Первая строка \n Вторая строка");

Результат: Первая строка
Вторая строка

Литералы

Литералы это постоянные значения, представленные в удобной для восприятия форме.
Например, 12 — это литерал типа int, 12L — литерал типа long, 10.19F — это литерал типа float.

Шестнадцатеричные литералы

Шестнадцатеричные литералы должны начинаться с символов 0x, т.е. нуля и последующей латинской буквы "икс".
Например, number = 0xFF; // 255 в десятичной системе
age = 0x1a; // 26 в десятичной системе.

Строковые литералы

Строковый литерал представляет собой набор символов, заключенных в двойные кавычки.

Буквальный строковый литерал

Буквальный строковый литерал начинается с символа @, после которого следует строка в кавычках.

Литералы

Литералы это постоянные значения, представленные в удобной для восприятия форме.
Например, 12 — это литерал типа int, 12L — литерал типа long, 10.19F — это литерал типа float.

Шестнадцатеричные литералы

Шестнадцатеричные литералы должны начинаться с символов 0x, т.е. нуля и последующей латинской буквы "икс".
Например, number = 0xFF; // 255 в десятичной системе
age = 0x1a; // 26 в десятичной системе.

Строковые литералы

Строковый литерал представляет собой набор символов, заключенных в двойные кавычки.

Буквальный строковый литерал

Буквальный строковый литерал начинается с символа @, после которого следует строка в кавычках.

Литералы

Литералы это постоянные значения, представленные в удобной для восприятия форме.
Например, 12 — это литерал типа int, 12L — литерал типа long, 10.19F — это литерал типа float.

Шестнадцатеричные литералы

Шестнадцатеричные литералы должны начинаться с символов 0x, т.е. нуля и последующей латинской буквы "икс".
Например, number = 0xFF; // 255 в десятичной системе
age = 0x1a; // 26 в десятичной системе.

Строковые литералы

Строковый литерал представляет собой набор символов, заключенных в двойные кавычки.

Буквальный строковый литерал

Буквальный строковый литерал начинается с символа @, после которого следует строка в кавычках.

Литералы

Литералы это постоянные значения, представленные в удобной для восприятия форме.
Например, 12 — это литерал типа int, 12L — литерал типа long, 10.19F — это литерал типа float.

Шестнадцатеричные литералы

Шестнадцатеричные литералы должны начинаться с символов 0x, т.е. нуля и последующей латинской буквы "икс".
Например, number = 0xFF; // 255 в десятичной системе
age = 0x1a; // 26 в десятичной системе.

Строковые литералы

Строковый литерал представляет собой набор символов, заключенных в двойные кавычки.

Буквальный строковый литерал

Буквальный строковый литерал начинается с символа @, после которого следует строка в кавычках.

```
// Продемонстрировать применение буквальных строковых литералов.
using System;
class Verbatim
{
    static void Main()
    {
        Console.WriteLine(@"Это буквальный
         строковый литерал,
         занимающий несколько строк.
         ");
        Console.WriteLine(@"А это вывод с табуляцией:
        1 2 3 4
        5 6 7 8
        ");
         Console.WriteLine(@"Отзыв программиста: ""Мне нравится
C#.""");
    }
}
```

Замечание

Все переменные в C# должны быть объявлены до их применения.

Общая форма задания переменной

тип имя_переменной;

Замечание

Все переменные в C# должны быть объявлены до их применения.

Общая форма задания переменной

тип имя_переменной;

Замечание

Все переменные в C# должны быть объявлены до их применения.

Общая форма задания переменной

тип имя_переменной;

Инициализация переменной:

тип имя_переменной = значение (или некоторое выражение);

Замечание

Все переменные в C# должны быть объявлены до их применения.

Общая форма задания переменной

тип имя_переменной;

Инициализация переменной:

тип имя_переменной = значение (или некоторое выражение);

Неявно типизированная переменная:

Например, var e = 2.7183; // Тип - double

var e = 2.7183F; // Тип - float

```
// Продемонстрировать область действия кодового блока.
using System;
class ScopeDemo
{
  static void Main()
  {
   int x; // Эта переменная доступна для всего кода внутри метода Main().
   x = 10;
   if(x == 10)
   {// начать новую область действия
     int y = 20;// Эта переменная доступна только в данном кодовом
блоке.
    // Здесь доступны обе переменные, x и y.
    Console.WriteLine("x и y:" + x + " " + y);
    x = y * 2;
   }
   // y = 100; // Ошибка! Переменная y здесь недоступна.
   // А переменная x здесь по-прежнему доступна.
   Console.WriteLine("x равно" + x);
  }
}
```

```
/**** Эта программа не может быть скомпилирована. ****/
using System;
class NestVar
{
    static void Main()
    {
        int i= 5;
        for (int count = 0; count < 10; count++)
        {
            Console.WriteLine("Это подсчет:" + count);
            for(int i = 0; i < 2; i++) // Недопустимо!!!
            Console.WriteLine("В этой программе есть ошибка!");
        }
    }
}
```

Автоматическое преобразование типов

Неявное преобразование типов происходит автоматически при следующих условиях:

- оба типа совместимы;
- диапазон представления чисел целевого типа шире, чем у исходного типа.

Автоматическое преобразование типов

Неявное преобразование типов происходит автоматически при следующих условиях:

- оба типа совместимы;
- диапазон представления чисел целевого типа шире, чем у исходного типа.

Пример:

```
int i;
double d;
i = 10;
d = i; // присвоить целое значение переменной типа double
```

Автоматическое преобразование типов

Неявное преобразование типов происходит автоматически при следующих условиях:

- оба типа совместимы;
- диапазон представления чисел целевого типа шире, чем у исходного типа.

Приведение типов

Приведение — это команда компилятору преобразовать результат вычисления выражения в указанный тип.

Автоматическое преобразование типов

Неявное преобразование типов происходит автоматически при следующих условиях:

- оба типа совместимы;
- диапазон представления чисел целевого типа шире, чем у исходного типа.

Приведение типов

Приведение — это команда компилятору преобразовать результат вычисления выражения в указанный тип.

Общая форма приведения типов:

(целевой_тип) выражение

Автоматическое преобразование типов

Неявное преобразование типов происходит автоматически при следующих условиях:

- оба типа совместимы;
- диапазон представления чисел целевого типа шире, чем у исходного типа.

Приведение типов

Приведение — это команда компилятору преобразовать результат вычисления выражения в указанный тип.

Общая форма приведения типов:

(целевой_тип) выражение

Пример:

```
double x=5, y=3;
int i;
i = (int) (x / y);
```

- ЕСЛИ один операнд имеет тип decimal, ТО и второй операнд продвигается к типу decimal (но если второй операнд имеет тип float или double, результат будет ошибочным).
- ЕСЛИ один операнд имеет тип double, ТО и второй операнд продвигается к типу double.
- ЕСЛИ один операнд имеет тип float, ТО и второй операнд продвигается к типу float.
- ЕСЛИ один операнд имеет тип ulong, ТО и второй операнд продвигается к типу ulong (но если второй операнд имеет тип sbyte, short, int или long,результат будет ошибочным).
- ЕСЛИ один операнд имеет тип long, ТО и второй операнд продвигается к типу long.
- ЕСЛИ один операнд имеет тип uint, а второй — тип sbyte, short или int, ТО оба операнда продвигаются к типу long.
- ЕСЛИ один операнд имеет тип uint, ТО и второй операнд продвигается к типу uint.
- ИНАЧЕ оба операнда продвигаются к типу int.

```
// Пример неожиданного результата продвижения типов!
using System;
class PromDemo
{
    static void Main()
    {
        byte b;
        b = 10;
        b = (byte) (b * b); // Необходимо приведение типов!!
        Console.WriteLine("b: "+ b);
    }
}
```

```
// Пример неожиданного результата продвижения типов!
using System;
class PromDemo
{
    static void Main()
    {
        char ch1 = 'a', ch2 = 'b';
        ch1 = (char) (ch1 + ch2);
        Console.WriteLine("ch1: " + ch1);
    }
}
```

```
// Пример неожиданного результата продвижения типов!
using System;
class PromDemo
{
    static void Main()
```

Упражнение 2.1

Написать программу для вычисления целой и дробной части заданного числа.

```
        Console.WriteLine("ch1: " + ch1);

    }
}
```

Таблица 2: Арифметические операторы в C#

Оператор	Действие
+	Сложение
−	Вычитание, унарный минус
*	Умножение
/	Деление
%	Деление по модулю
−−	Декремент
++	Инкремент

```
// Продемонстрировать применение оператора %.
using System;
class ModDemo
{
    static void Main()
    {
        int iresult, irem;
        double dresult, drem;
        iresult = 10 / 3;
        irem = 10 % 3;
        dresult = 10.0 / 3.0;
        drem = 10.0 % 3.0;
        Console.WriteLine("Результат и остаток от деления 10/3:" +
        iresult + " " + irem);
        Console.WriteLine("Результат и остаток от деления 10.0 / 3.0:" +
        dresult + " " + drem);
    }
}
```

Операторы инкремента(++) и декремента(−−)

Оба оператора инкремента и декремента можно указывать до операнда (в префиксной форме) или же после операнда (в постфиксной форме)

Операторы инкремента(++) и декремента(−−)

Оба оператора инкремента и декремента можно указывать до операнда (в префиксной форме) или же после операнда (в постфиксной форме)

Пример:

```
++x; // префиксная форма
x++; // постфиксная форма
```

```
// Продемонстрировать отличие между префиксной
// и постфиксной формами оператора инкремента (++).
using System;
class PrePostDemo
{
    static void Main()
    {
        int x, y;
        int i;
        x = 1;
        y = 0;
        Console.WriteLine("Ряд чисел, полученных" +
        "с помощью оператора y = y + x++;");
        for(i = 0; i < 10; i++)
        {
          y = y + x++; // постфиксная форма оператора ++
          Console.WriteLine(y + " ");
        }
        Console.WriteLine();
```

```
        x = 1;
        y = 0;
        Console.WriteLine( "Ряд чисел, полученных" +
        "с помощью оператора y = y + ++x;");
        for(i = 0; i < 10; i++)
        {
          y = y + ++x; // префиксная форма оператора ++
          Console.WriteLine(y + " ");
        }
        Console.WriteLine();
    }
}
```

Таблица 3: Операторы отношения в C#

Оператор	Значение
==	Равно
!=	Не равно
>	Больше
<	Меньше
>=	Больше или равно
<=	Меньше или равно

Таблица 4: Логические операторы в C#

Оператор	Значение
&	И
|	ИЛИ
^	Исключающее ИЛИ
&&	Укороченное И
||	Укороченное ИЛИ
!	НЕ

Таблица 5: Логические операторы в C#

p	q	p & q	p | q	p ^ q	!p
false	false	false	false	false	true
true	false	false	true	true	false
false	true	false	true	true	true
true	true	true	true	false	false

Таблица 5: Логические операторы в C#

p	q	p & q	p | q	p ^ q	!p
false	false	false	false	false	true
true	false	false	true	true	false
false	true	false	true	true	true
true	true	true	true	false	false

Упражнение 2.2

Написать программу для реализация операции импликации.

Таблица 5: Логические операторы в C#

p	q	p & q	p | q	p ^ q	!p
false	false	false	false	false	true
true	false	false	true	true	false
false	true	false	true	true	true
true	true	true	true	false	false

Упражнение 2.2

Написать программу для реализация операции импликации.

Упражнение 2.3

Написать программу укороченного варианта логического оператора "И"(&&) в которой обычный оператор(&) работать не будет.

Таблица 5: Логические операторы в C#

p	q	p & q	p | q	p ^ q	!p
false	false	false	false	false	true
true	false	false	true	true	false
false	true	false	true	true	true
true	true	true	true	false	false

Упражнение 2.2

Написать программу для реализация операции импликации.

Упражнение 2.3

Написать программу укороченного варианта логического оператора "И"(&&) в которой обычный оператор(&) работать не будет.

Упражнение 2.4

Написать программу с обратным примером.

Общий вид оператора присваивания:

имя_переменной = выражение

Общий вид оператора присваивания:

имя_переменной = выражение

Пример:

```
int x, y, z;
x = y = z = 100; // присвоить значение 100 переменным x, y и z
```

Общий вид оператора присваивания:

имя_переменной = выражение

Общий вид составного оператора присваивания:

имя_переменной оп= выражение,
где оп — арифметический или логический оператор, применяемый вместе с оператором присваивания

Общий вид оператора присваивания:

имя_переменной = выражение

Общий вид составного оператора присваивания:

имя_переменной оп= выражение,
где оп — арифметический или логический оператор, применяемый вместе с оператором присваивания

Таблица 6: Составные операторы присваивания

+=	−=	*=	/=
%=	&=	|=	^=

Общий вид оператора присваивания:

имя_переменной = выражение

Общий вид составного оператора присваивания:

имя_переменной оп= выражение,
где оп — арифметический или логический оператор, применяемый вместе с оператором присваивания

Таблица 6: Составные операторы присваивания

+=	−=	*=	/=
%=	&=	|=	^=

Пример:
Оператор x = x - 100;
эквивалентен оператору x −= 100;

Таблица 7: Поразрядные операторы

Оператор	Значение
&	Поразрядное И
|	Поразрядное ИЛИ
^	Поразрядное исключающее ИЛИ
>>	Сдвиг вправо
<<	Сдвиг влево
~	Дополнение до 1 (унарный оператор НЕ)

Таблица 8: Поразрядные операторы

p	q	p & q	p | q	p ^ q	!p
0	0	0	0	0	1
1	0	0	1	1	0
0	1	0	1	1	1
1	1	1	1	0	0

Таблица 7: Поразрядные операторы

Оператор	Значение
&	Поразрядное И
|	Поразрядное ИЛИ
^	Поразрядное исключающее ИЛИ
>>	Сдвиг вправо
<<	Сдвиг влево
~	Дополнение до 1 (унарный оператор НЕ)

Таблица 8: Поразрядные операторы

p	q	p & q	p | q	p ^ q	!p
0	0	0	0	0	1
1	0	0	1	1	0
0	1	0	1	1	1
1	1	1	1	0	0

Пример:

```
  1101 0011
& 1010 1010
-----------
  1000 0010
```

Упражнение 2.5

Применить поразрядный оператор "И",чтобы сделать число четным.

Упражнение 2.5

Применить поразрядный оператор "И",чтобы сделать число четным.

Упражнение 2.6

Применить поразрядный оператор "И",чтобы определить, является ли число нечетным.

Упражнение 2.5

Применить поразрядный оператор "И",чтобы сделать число четным.

Упражнение 2.6

Применить поразрядный оператор "И",чтобы определить, является ли число нечетным.

Упражнение 2.7

Применить поразрядный оператор "И",чтобы показать биты, составляющие байт.

Упражнение 2.5

Применить поразрядный оператор "И",чтобы сделать число четным.

Упражнение 2.6

Применить поразрядный оператор "И",чтобы определить, является ли число нечетным.

Упражнение 2.7

Применить поразрядный оператор "И",чтобы показать биты, составляющие байт.

Упражнение 2.8

Применить поразрядный оператор "ИЛИ",чтобы сделать число нечетным.

Свойство поразрядного оператора "исключающее ИЛИ"

$x = (x \hat{} y) \hat{} y;$

Свойство поразрядного оператора "исключающее ИЛИ"

```
x = (x ^ y) ^ y;
```

Пример:

```
int x= 33, y =71;
Console.WriteLine("Исходное значение:" + x);
x = (x ^ y) ^ y;
Console.WriteLine("Конечное значение:" + x);
```

Свойство поразрядного оператора "исключающее ИЛИ"

```
x = (x ^ y) ^ y;
```

Пример:

```
int x= 33, y =71;
Console.WriteLine("Исходное значение:" + x);
x = (x ^ y) ^ y;
Console.WriteLine("Конечное значение:" + x);
```

Результат:

Исходное значение: 33
Конечное значение: 33

Свойство поразрядного оператора "исключающее ИЛИ"

```
x = (x ^ y) ^ y;
```

Пример:

```
int x= 33, y =71;
Console.WriteLine("Исходное значение:" + x);
x = (x ^ y) ^ y;
Console.WriteLine("Конечное значение:" + x);
```

Результат:

Исходное значение: 33
Конечное значение: 33

Упражнение 2.9

Написать программу для шифрования текстовых сообщений. В качества ключа использовать число типа int.

Свойство поразрядного оператора "исключающее ИЛИ"

```
x = (x ^ y) ^ y;
```

Пример:

```
int x= 33, y =71;
Console.WriteLine("Исходное значение:" + x);
x = (x ^ y) ^ y;
Console.WriteLine("Конечное значение:" + x);
```

Результат:

Исходное значение: 33
Конечное значение: 33

Упражнение 2.9

Написать программу для шифрования текстовых сообщений. В качества ключа использовать число типа int.

Упражнение 2.10

Написать программу для расшифровки текстовых сообщений. Предполагается, что в качества ключа шифрования использовалось число типа int.

Общий вид оператора if:

```
if(условие)
{
    последовательность операторов
}
else
{
    последовательность операторов
}
```

Общий вид оператора if:

```
if(условие)
{
    последовательность операторов
}
else
{
    последовательность операторов
}
```

Компактная форма:

```
if(условие) оператор;
else оператор;
```

Общий вид оператора if:

```
if(условие)
{
    последовательность операторов
}
else
{
    последовательность операторов
}
```

Компактная форма:

```
if(условие) оператор;
else оператор;
```

Вложенный оператор if:

Вложенным называется такой оператор if, который является адресатом другого оператора if или же оператора else.

Пример:

```
if (i == 10)
{
    if(j < 20) a = b;
    if(k > 100) c = d;
    else a = c; // этот оператор else связан с оператором if(k > 100)
}
else a = d; // этот оператор else связан с оператором if(i == 10)
```

Вложенный оператор if:

Вложенным называется такой оператор if, который является адресатом другого оператора if или же оператора else.

Многоступенчатая конструкция if-else-if:

```
if(условие) оператор;
else if (условие) оператор;
else if (условие) оператор;
else оператор;
```

Общий вид оператора switch:

```
switch(выражение)
{
    case константа1:
    последовательность операторов
    break;
    case константа2:
    последовательность операторов
    break;
    case константа3:
    последовательность операторов
    break;
    default:
    последовательность операторов
    break;
}
```

Общий вид оператора switch:

```
switch(выражение)
{
    case константа1:
    последовательность операторов
    break;
    case константа2:
    последовательность операторов
    break;
    case константа3:
    последовательность операторов
    break;
    default:
    последовательность операторов
    break;
}
```

Замечание

Для управления оператором switch может быть использовано выражение любого целочисленного типа, включая и char

Пример 1:

```
for(int i=1; i < 5; i++)
switch(i)
{
    case 1:
    case 2:
    case 3: Console.WriteLine("i равно 1, 2 или 3");
    break;
    case 4: Console.WriteLine("i равно 4");
    break;
}
```

Пример 2:

```
switch(ch1)
{
    case 'A': Console.WriteLine("Эта ветвь A — часть" +
                        "внешнего оператора switch.");
    switch(ch2)
    {
        case 'A':
        Console.WriteLine("Эта ветвь A — часть" +
                    "внутреннего оператора switch");
        break;
        case 'B': // ...
    } // конец внутреннего оператора switch
    break;
    case 'B': // ...
}
```

Общий вид оператора for:

```
for(инициализация; условие; итерация)
{
    последовательность операторов;
}
```

Общий вид оператора for:

```
for(инициализация; условие; итерация)
{
    последовательность операторов;
}
```

Компактная форма:

```
for(инициализация; условие; итерация) оператор;
```

Общий вид оператора for:

```
for(инициализация; условие; итерация)
{
    последовательность операторов;
}
```

Компактная форма:

```
for(инициализация; условие; итерация) оператор;
```

Упражнение 2.11

Использовать операторы цикла for для выявления простых чисел в пределах от 2 до 20. Если число оказывается непростым, вывести его наибольший множитель.

Замечание

В операторе цикла for разрешается использовать две или более переменных для управления циклом. В этом случае операторы инициализации и инкремента каждой переменной разделяются запятой.

Замечание

В операторе цикла for разрешается использовать две или более переменных для управления циклом. В этом случае операторы инициализации и инкремента каждой переменной разделяются запятой.

Пример: // Использовать запятые в операторе цикла for.

```
using System;
class Comma
{
    static void Main()
    {
        int i, j;
        for(i=0, j=10; i < j; i++, j--)
        Console.WriteLine("i и j:" + i + " " + j);
    }
}
```

Замечание

В операторе цикла for разрешается использовать две или более переменных для управления циклом. В этом случае операторы инициализации и инкремента каждой переменной разделяются запятой.

Пример: // Использовать запятые в операторе цикла for.

```
using System;
class Comma
{
    static void Main()
    {
        int i, j;
        for(i=0, j=10; i < j; i++, j--)
        Console.WriteLine("i и j:" + i + " " + j);
    }
}
```

Упражнение 2.12

Использовать две переменные управления одним циклом for для выявления наибольшего и наименьшего множителя заданного целого числа.

Замечание

Условным выражением, управляющим циклом for, может быть любое действительное выражение, дающее результат типа bool.

Замечание

Условным выражением, управляющим циклом for, может быть любое действительное выражение, дающее результат типа bool.

Пример: // Условием выполнения цикла может служить любое выражение типа bool.

```
using System;
class forDemo {
    static void Main()
    {
        int i, j;
        bool done = false;
        for(i=0, j=100; !done; i++, j--)
        {
            if(i*i >= j) done = true;
            Console.WriteLine("i, j:" + i + " " + j);
        }
    }
}
```

Пример 1: // Отдельные части цикла for могут оставаться пустыми.

```
int i;
for(i = 0; i < 10; )
{
    Console.WriteLine("Проход №" + i);
    i++; // инкрементировать переменную управления циклом
}
```

Пример 1: // Отдельные части цикла for могут оставаться пустыми.

```
int i;
for(i = 0; i < 10; )
{
    Console.WriteLine("Проход №" + i);
    i++; // инкрементировать переменную управления циклом
}
```

Пример 2:

```
int i= 0; // исключить инициализацию из определения цикла
for(; i < 10; )
{
    Console.WriteLine("Проход №" + i);
    i++; // инкрементировать переменную управления циклом
}
```

Пример 1: // Отдельные части цикла for могут оставаться пустыми.

```
int i;
for(i = 0; i < 10; )
{
    Console.WriteLine("Проход №" + i);
    i++; // инкрементировать переменную управления циклом
}
```

Пример 2:

```
int i= 0; // исключить инициализацию из определения цикла
for(; i < 10; )
{
    Console.WriteLine("Проход №" + i);
    i++; // инкрементировать переменную управления циклом
}
```

Пример 3: // Отдельные части цикла for могут оставаться пустыми.

```
int i, sum = 0;
// получить сумму чисел от 1 до 5
for(i = 1; i <= 5; sum += i++);
Console.WriteLine("Сумма равна" + sum);
```

Общий вид оператора while

```
while (условие)
{
    последовательность операторов;
}
```

Общий вид оператора while

```
while (условие)
{
    последовательность операторов;
}
```

Общий вид оператора do-while

```
do
{
    последовательность операторов;
}
while (условие);
```

Общий вид оператора while

```
while (условие)
{
    последовательность операторов;
}
```

Общий вид оператора do-while

```
do
{
    последовательность операторов;
}
while (условие);
```

Упражнение 2.13

С помощью оператора do-while отобразить цифры целого числа в обратном порядке

Замечание

С помощью оператора break можно специально организовать немедленный выход из цикла в обход любого кода, оставшегося в теле цикла, а также минуя проверку условия цикла.

Замечание

С помощью оператора break можно специально организовать немедленный выход из цикла в обход любого кода, оставшегося в теле цикла, а также минуя проверку условия цикла.

Замечание

С помощью оператора continue можно организовать преждевременное завершение шага итерации цикла в обход обычной структуры управления циклом.

Замечание

С помощью оператора break можно специально организовать немедленный выход из цикла в обход любого кода, оставшегося в теле цикла, а также минуя проверку условия цикла.

Замечание

С помощью оператора continue можно организовать преждевременное завершение шага итерации цикла в обход обычной структуры управления циклом.

Пример:

```
// вывести четные числа от 0 до 100.
for (int i = 0; i <= 100; i++)
{
    if((i%2) != 0) continue; // перейти к следующему шагу итерации
    Console.WriteLine(i);
}
```

Оператор goto

Оператор goto представляет собой оператор безусловного перехода. Когда в программе встречается данный оператор, ее выполнение переходит непосредственно к тому месту, на которое указывает этот оператор.

Оператор goto

Оператор goto представляет собой оператор безусловного перехода. Когда в программе встречается данный оператор, ее выполнение переходит непосредственно к тому месту, на которое указывает этот оператор.

Пример:

```
// Продемонстрировать практическое применение оператора goto.
int i=0, j=0, k=0;
for(i=0; i < 10; i++)
{
    for(j=0; j < 10; j++ )
    {
        for(k=0; k < 10; k++)
        {
            Console.WriteLine("i, j, k:" + i + " " + j + " " + k);
            if(k == 3) goto stop;
        }
    }
}
stop:
Console.WriteLine("Остановлено! i, j, k:" + i + "," + j + " " + k);
```