Синтаксис C#

(арифметические и унарные операции, работа с типами данных, перечисления)

Артём Трофимушкин

Арифметические операции

```
var resultVariable = firstOperand operator secondOperand; // result
```

- `+` Сложение

```
float a = 7 + 0.5F;     // 7.5
```

- `−` Вычитание

```
int b = 7 - 2;          // 5
```

- `*` Умножение

```
float c = 10.1F * 3;    // 30.3
```

- `/` Деление

```
int d = 11 / 3;         // 3
double e = 11 / 3;      // 3
double f = 11.0 / 3;    // 3.66666666666667
```

- `%` Остаток (от деления)

```
int g = 11 % 3;         // 2
```

Унарные операции (с одним операндом)

Инкремент: увеличение значения на 1

- `++i` Префиксный инкремент

```
int a = 0;
int b = 0;
b = ++a; // first increment "a", then assign its value to "b"
Console.WriteLine(a);   // 1
Console.WriteLine(b);   // 1
```

- `i++` Постфиксный инкремент

```
int c = 0;
int d = 0;
d = c++; // first assign value of "c" to "d", then increment "c"
Console.WriteLine(c); // 1
Console.WriteLine(d); // 0
```

Унарные операции (с одним операндом)

Декремент: уменьшение значения на 1

- `--i` Префиксный декремент

```
int a = 0;
int b = 0;
b = --a;    // first decrement "a", then assign its value to "b"
Console.WriteLine(a); // -1
Console.WriteLine(b); // -1
```

- `i--` Постфиксный отложенный

```
int c = 0;
int d = 0;
d = c--; // first assign value of "c" to "d", then decrement "c"
Console.WriteLine(c); // -1
Console.WriteLine(d); // 0
```

Унарные операции (с одним операндом)

Отрицание: инвертирует значение булевой переменной

- ! Логическая инверсия

```
bool a = true;
Console.WriteLine(a);       // a is "true"

bool b = !a;
Console.WriteLine(b);       // a is "true", so !a is "false"

bool c = !b;
Console.WriteLine(b);       // b is "false", so !b is "true"
```

Операторы отношения (сравнения)

- `==` Равно

```
Console.WriteLine(10 == 15);         // False
Console.WriteLine(10.0 == 10);       // True
Console.WriteLine("abc" == "abc");   // True
Console.WriteLine(true == true);     // True
Console.WriteLine(true == false);    // False
```

- `!=` Не равно

```
Console.WriteLine(10 != 15);         // True
Console.WriteLine(10.0 != 10);       // False
Console.WriteLine("abc" != "abc");   // False
Console.WriteLine(true != true);     // False
Console.WriteLine(true != false);    // True
```

Операторы отношения (сравнения)

- `>` Больше

```
Console.WriteLine(10 > 15);         // False
Console.WriteLine(true > false);    // Compiler error!
Console.WriteLine("abc" > "def");   // Compiler error!
```

- `<` Меньше

```
Console.WriteLine(10 < 15);         // True
Console.WriteLine(15 < 10);         // False
```

- `>=` Больше или равно

```
Console.WriteLine(10 >= 15);        // False
Console.WriteLine(15 >= 15);        // True
```

- `<=` Меньше или равно

```
Console.WriteLine(10 <= 15);        // True
Console.WriteLine(15 <= 15);        // True
```

Приведение числовых типов

Неявное приведение (implicit casting) числовых типов данных

Можно неявно приводить целочисленные данные к дробным:

```
int a = 10;
double b = a;
Console.WriteLine(b);
```

Однако в обратную сторону это работать не будет:

```
double g = 9.8;
int c = g; // compiler gives an error for this line!
Console.WriteLine(g);`
```

- Ошибка CS0266: Cannot implicitly convert type 'double' to 'int'. An explicit conversion exists (are you missing a cast?)
- Невозможно неявно преобразовать тип 'double' в тип 'int'. Существует явное преобразование (вы забыли выполнить приведение типов?)

Приведение числовых типов

Явное приведение (explicit casting) числовых типов данных

Можно явно приводить дробные числа к целым отсекая (но не округляя!) дробную часть:

```
double c = 9.8;
int d = (int)c; // 'g' is 9 loosing its .8 part
Console.WriteLine(d);
```

Также явное приводятся числа больших типов данных к малым. Опасайтесь потери данных, так как любое значение, выходящее за рамки целевого типа данных будет приведено к -1:

```
long e = 10;
int f = (int)e;
Console.WriteLine($"e is {e} and f is {f}"); // e is 10 and f is 10
e = long.MaxValue;
f = (int)e;
Console.WriteLine($"e is {e} and f is {f}"); // e is 9223372036854775807
                                             // and f is -1
```

Округление чисел: Convert

Округление: берётся верхнее значение, если дробная часть больше 0.5, в противном случае дробная часть отсекается.

Однако (!), в .NET если целая часть чётная, дробная часть даже в случае 0.5 отсекается:

```
double i = 9.49;
double j = 9.5;
double k = 10.49;
double l = 10.5; // it will be just 10!
double m = 10.51;

Console.WriteLine(Convert.ToInt32(i)); // 9
Console.WriteLine(Convert.ToInt32(j)); // 10
Console.WriteLine(Convert.ToInt32(k)); // 10
Console.WriteLine(Convert.ToInt32(l)); // 10
Console.WriteLine(Convert.ToInt32(m)); // 11
```

Округление: Math.Floor и Math.Ceiling

Floor: Возвращает наибольшее целое число, которое меньше или равно указанному числу.

Ceiling: Возвращает наименьшее целое число, которое больше или равно заданному числу.

```
var d1 = 7.03m;
var d2 = 7.64m;
var d3 = 0.12m;
var d4 = -0.12m;
var d5 = -7.1m;
var d6 = -7.6m;

Console.WriteLine("{0}, {1}", Math.Ceiling(d1), Math.Floor(d1)); // 8, 7
Console.WriteLine("{0}, {1}", Math.Ceiling(d2), Math.Floor(d2)); // 8, 7
Console.WriteLine("{0}, {1}", Math.Ceiling(d3), Math.Floor(d3)); // 1, 0
Console.WriteLine("{0}, {1}", Math.Ceiling(d4), Math.Floor(d4)); // 0, -1
Console.WriteLine("{0}, {1}", Math.Ceiling(d5), Math.Floor(d5)); // -7, -8
Console.WriteLine("{0}, {1}", Math.Ceiling(d6), Math.Floor(d6)); // -7, -8
```

Другие математические операции

DivRem: Выполняет операцию деления с остатком и возвращает как результат деления, так и остаток

Abs: Возвращает абсолютное значение для аргумента

Sign: Возвращает число 1, если аргумент положительный, и -1, если отрицательный. Если он равен 0, то возвращает 0

Sqrt: Возвращает квадратный корень аргумента

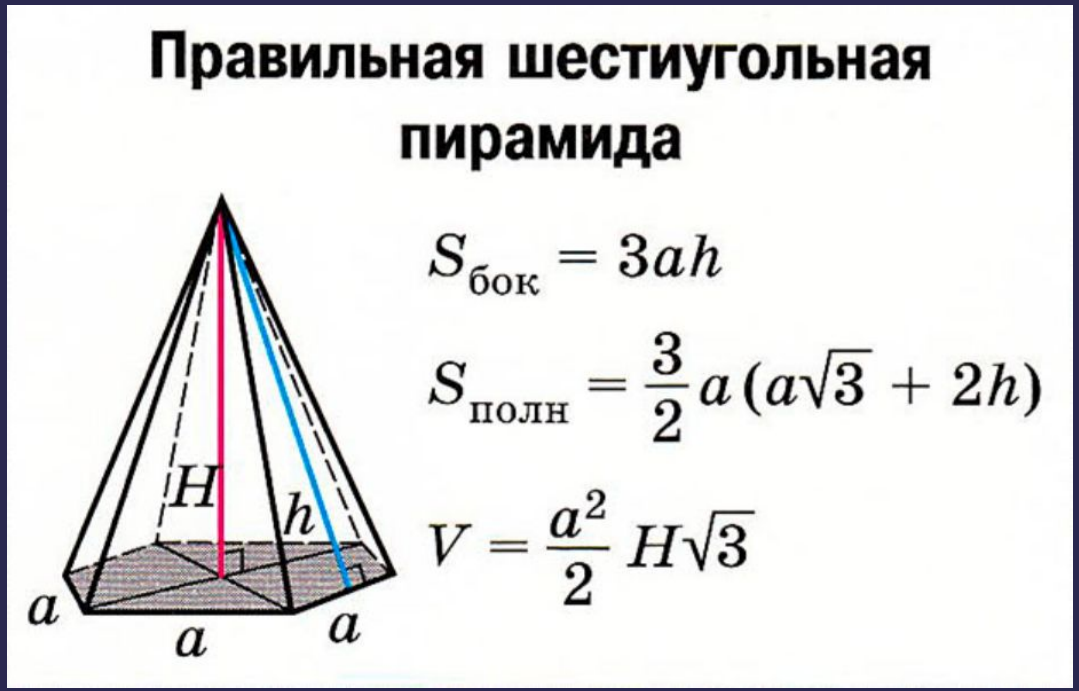
Cbrt: Возвращает кубический корень аргумента

Min: Возвращает минимальный из двух аргументов

Max: Возвращает максимальный из двух аргументов

Самостоятельная работа

Дана правильная шестиугольная пирамида.



Запросить a и h

Рассчитать

- S(бок)
- S(полн)
- V

Вывести их на экран

Приведение любого значения к строке

Переменные любого типа данных имеют метод ToString():

```
int number = 12;
Console.WriteLine(number.ToString());   // 12

bool boolean = true;
Console.WriteLine(boolean.ToString());  // True

DateTime now = DateTime.Now;
Console.WriteLine(now.ToString());      // 1/1/2019 2:15:00 PM

object me = new object();
Console.WriteLine(me.ToString());       // System.Object

string str = "abc";
Console.WriteLine(str.ToString());      // abc :)
```

Приведение строки к любому типу

Многие типы данных имеют метод Parse():

```
string strInt = "12";
int i = Int32.Parse(strInt);
Console.WriteLine(i * i);           // 144

string strFloat = "3.14159265";
float f = Single.Parse(strFloat);
Console.WriteLine(f * 2);           // 6.283185

string strBool = "true";
bool b = Boolean.Parse(strBool);
Console.WriteLine(b.ToString());    // True
```

Будьте внимательны! Если .NET не встретит ожидаемый формат, произойдёт ошибка преобразования типа!

Перечисления: enum

Тип перечисления предоставляет способ определения набора именованных целочисленных констант, который можно назначить переменной:

```
enum Day { Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday };
```

По умолчанию базовым типом каждого элемента перечисления является int. Можно задать другой целочисленный тип, используя двоеточие:

```
enum Month : byte { Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec };
```

Перечисления: enum

```
using System;
public class Program
{
    enum Day { Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday };
    enum Month : byte { Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec };

    public static void Main()
    {
        Day today = Day.Monday;
        int dayNumber = (int)today;
        Console.WriteLine("{0} is day number #{1}.", today, dayNumber);

        Month thisMonth = Month.Dec;
        byte monthNumber = (byte)thisMonth;
        Console.WriteLine("{0} is month number #{1}.", thisMonth, monthNumber);

        // Monday is day number #1.
        // Dec is month number #11.
    }
}
```

Побитовые или поразрядные операторы

Такие операторы проводят операции непосредственно на битах числа

- `|` ИЛИ (OR)
- `&` И (AND)
- `~` Инверсия / отрицание (NOT)
- `^` Исключающее ИЛИ (XOR)

  принимает значение “истина”, **если** всего один из аргументов имеет значение “истина”
- `<<` Сдвиг влево (left-shift)

  Сдвигает биты влево на определенное количество разрядов
  Биты, расположенные слева, удаляются, справа появляются нули
- `>>` Сдвиг вправо (right-shift)

  Сдвигает биты вправо на определенное количество разрядов
  Биты, расположенные справа, удаляются, слева появляются нули

Перечисления: enum как битовые флаги

Тип перечисления можно использовать для определения битовых флагов, благодаря чему экземпляр типа перечисления может хранить любую комбинацию значений, определенных в списке перечислителя.

```
[Flags]
enum Days
{
      None = 0x0,
      Sunday = 0x1,
      Monday = 0x2,
      Tuesday = 0x4,
      Wednesday = 0x8,
      Thursday = 0x10,
      Friday = 0x20,
      Saturday = 0x40
}

Days weekends = Days.Saturday | Days.Sunday;
```

Операторы для работы с флагами

При работе с флагами необходимо работать используя побитовые операторы

- AND & И
- OR | ИЛИ
- XOR ^ Исключающее ИЛИ

XOR – операция, которая принимает значение “истина” только если всего один из аргументов имеет значение “истина”.

Полезная статья про XOR: https://habr.com/ru/post/183462.

Установить / добавить бит можно через OR

```
nonWorkingDays = nonWorkingDays | Days.Friday;
```

Удалить бит можно через XOR

```
nonWorkingDays = nonWorkingDays ^ Days.Sunday;
```

Проверить, установлен ли бит можно AND

```
bool isThursdayWorking = (nonWorkingDays & Days.Thursday) != Days.Thursday;
```

Самостоятельная работа: enum [Flags]

Написать программу для добавления цветов заданной палитры в “избранное”.

Программа выводит список цветов с их порядковыми номерами и просит пользователя в цикле выбрать 4 цвета для добавления их в палитру “Избранное”.

Выбор производится путём введения порядковых номеров этих цветов.

После завершения ввода программа выводит список любимых цветов, а также отдельно список нелюбимых цветов.

Список допустимых цветов в палитре:

- Black
- Blue
- Cyan
- Grey
- Green
- Magenta
- Red
- White
- Yellow

Домашнее задание

Написать консольное приложение, которое будет спрашивать, “Какой объем сока (в литрах) требуется упаковать?”.

Затем оно будет рассчитывать и выдавать в качестве ответа необходимое количество контейнеров каждого типа.

В нашей модели будет 3 типа контейнеров: 1 литр, 5 литров, 20 литров.

Типы контейнеров должны быть определены в перечислении (представленным битовыми флагами).

Кроме количества контейнеров необходимо посчитать значение переменной типа Int32, в битах которой будет лежать признак наличия контейнера того или иного этого типа (0001 - 1л, 0010 - 5л, 0100 - 20л) .

При выводе, если бит, отвечающий за наличие хотя бы одного контейнера данного типа, равен 0, строку с данными по этому контейнеру не выводить.

Домашнее (примеры вывода)

Пример работы программы:

```
> Какой объем сока (в литрах) требуется упаковать?
> 76.4 /это ввод пользователя/
> Вам потребуются следующие контейнеры:
> 20 л: 3 шт.
>  5 л: 3 шт.
>  1 л: 2 шт.
```

Пример работы программы (где количество 5-ти литровых контейнеров равно 0):

```
> Какой объем сока (в литрах) требуется упаковать?
> 61.4 /это ввод пользователя/
> Вам потребуются следующие контейнеры:
> 20 л: 3 шт.
>  1 л: 2 шт.
```

Спасибо за внимание.