## Практическая работа 5
## Преобразование типов. Явное и неявное преобразование

### Цель занятия

Получить практические навыки использования методов преобразования типов переменных.

### Перечень оборудования и программного обеспечения

Персональный компьютер
Microsoft Office (Word, Visio)
Microsoft Visual Studio 2010

### Краткие теоретические сведения

### Автоматическое преобразование типов

Когда данные одного типа присваиваются переменной другого типа, неявное преобразование типов происходит автоматически при следующих условиях:

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

Если оба эти условия удовлетворяются, то происходит расширяющее преобразование. Например, тип int достаточно крупный, чтобы вмещать в себя все действительные значения типа byte, а кроме того, оба типа, int и byte, являются совместимыми целочисленными типами, и поэтому для них вполне возможно неявное преобразование.

Числовые типы, как целочисленные, так и с плавающей точкой, вполне совместимы друг с другом для выполнения расширяющих преобразований. Рассмотрим пример:

```
{

shortnum1, num2;
num1 = 10;
num2 = 15;

Console.WriteLine("{0} + {1} = {2}",num1,num2,Sum(num1,num2));
Console.ReadLine();
}

staticint Sum(int x, int y)
```

```
{
return x + y;
    }
```

Метод Sum() ожидает поступления двух параметров типа int. В методе Main() ему передаются две переменных типа short. Программа будет компилироваться и выполняться без ошибок и возвращать в результате, как и ожидалось, значение 25.

Компилятор будет считать данный код синтаксически корректным, т.к. потеря данных здесь невозможна. Поскольку максимальное значение (32767), которое может содержать тип short, вполне вписывается в рамки диапазона типа int (максимальное значение которого составляет 2147483647), компилятор будет неявным образом расширять каждую переменную типа short до типа int. Формально термин "расширение" применяется для обозначения неявного восходящего приведения (upwardcast), которое не приводит к потере данных.

**Приведение несовместимых типов**

Неявные преобразования типов не способны удовлетворить все потребности в программировании, поскольку допускают лишь расширяющие преобразования совместимых типов. Во всех остальных случаях приходится обращаться к приведению типов. Приведение — это команда компилятору преобразовать результат вычисления выражения в указанный тип. А для этого требуется явное преобразование типов. Ниже приведена общая форма приведения типов:

(целевой_тип) выражение

Здесь целевой_тип обозначает тот тип, в который желательно преобразовать указанное выражение.

Если приведение типов приводит к сужающему преобразованию, то часть информации может быть потеряна. Например, в результате приведения типа long к типу int часть информации потеряется, если значение типа long окажется больше диапазона представления чисел для типа int, поскольку старшие разряды этого числового значения отбрасываются. Когда же значение с плавающей точкой приводится к целочисленному, то в результате усечения теряется дробная часть этого числового значения. Так, если присвоить значение 1,23 целочисленной переменной, то в результате в ней останется лишь целая часть исходного числа (1), а дробная его часть (0,23) будет потеряна.

**Пример:**

```
            int i1 = 455, i2 = 84500;
decimaldec = 7.98845m;
```

```
// Приводим два числа типа int
        // к типу short
Console.WriteLine((short)i1);
Console.WriteLine((short)i2);

// Приводим число типа decimal
        // к типу int
Console.WriteLine((int)dec);

Console.ReadLine();
```

**Объект Convert**

Для того чтобы преобразовать переменные к нужному типу данных используется объект Convert в определенным методом.

<переменная целого типа> = Convert.ToInt32(<строковая переменная>) - преобразует строковое представление числа с указанным основанием системы счисления в эквивалентное ему 32-битовое целое число со знаком.

<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;
 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, иначе возникает ошибка. Поэтому нужно внимательно следить за правильностью вводимых данных.

Метод ToString преобразует указанное значение в его эквивалентное строковое представление.

Пример:

Преобразует значение заданного 32-разрядного знакового целого числа в эквивалентное строковое представление.

```
int value = -16325;
// Display value using default ToString method.
Console.WriteLine(value.ToString("G"));  // Displays -16325
```

**Класс Math**

Предоставляет константы и статические методы для тригонометрических, логарифмических и иных общих математических функций:

**Тригонометрические функции**:

| | |
|---|---|
| **cos** | Вычислить cos (функция) |
| **sin** | Вычислить sin (функция) |
| **tan** | Вычислить tang(функция) |
| **acos** | Вычислить arccos (функция) |
| **asin** | Вычислить arcsin (функция) |
| **atan** | Вычислить arctang (функция) |

**Гиперболические функции**:

| | |
|---|---|
| **cosh** | Вычислить гиперболический cos (функция) |
| **sinh** | Вычислить гиперболический sin (функция) |
| **tanh** | Вычислить гиперболический tang (функция) |

**Экспоненциальные и логарифмические функции**:

| | |
|---|---|
| **exp** | Вычислить экспоненциальную функцию (функция) |
| **log** | Вычислить натуральный логарифм (функция) |
| **log10** | Вычислить десятичный логарифм (функция) |
| **modf** | Дробная и целая части (функция) |

**Степенные функции**

| | |
|---|---|
| **pow** | Возведение в степень (функция) |
| **sqrt** | Вычислить корень (функция) |

**Округление, абсолютное значение и остаток от деления функции**:

| | |
|---|---|
| **ceil** | Округленное значение (функция) |
| **abs** | Вычислить абсолютное значение (функция) |
| **floor** | Нижнее округленное (функция) |
| **fmod** | Вычислить остаток от деления (функция) |

| | |
|---|---|
| **max** | Возвращает большее из двух чисел (функция) |
| **min** | Возвращает меньшее из двух чисел (функция) |

**Поля**

| | |
|---|---|
| E | Представляет основание натурального логарифма, определяемое константой e |
| PI | Представляет отношение длины окружности к ее диаметру, определяемое константой π. |

**Пример:**

```
double radius, dlina;
string s;
    s = Console.ReadLine();
radius = Convert.ToDouble(s);
    dlina = Math.PI*Math.Pow(radius,2.0);
Console.WriteLine(dlina.ToString());
Console.ReadKey();
```

Класс **Math** является статическим, поэтому все его методы также являются статическими.

Рассмотрим методы класса Math:

`Abs(doublevalue)`: возвращает абсолютное значение для аргумента value

```
doubleresult = Math.Abs(-12.4); // 12.4
```

`Acos(double value)`: возвращаетарккосинус value. Параметр value должен иметь значение от -1 до 1

```
doubleresult = Math.Acos(1); // 0
```

`Asin(double value)`: возвращаетарксинус value. Параметр value должен иметь значение от -1 до 1

`Atan(double value)`: возвращаетарктангенс value

`BigMul(int x, int y)`: возвращает произведение x * y в виде объекта long

```
doubleresult = Math.BigMul(100, 9340); // 934000
```

`Ceiling(doublevalue)`: возвращает наименьшее целое число с плавающей точкой, которое не меньше value

```
doubleresult = Math.Ceiling(2.34); // 3
```

`Cos(double d)`: возвращает косинус угла d

`Cosh(double d)`: возвращает гиперболический косинус угла d

`DivRem(int a, int b, outintresult)`: возвращает результат от деления a/b, а остаток помещается в параметр result

```
intresult;
```

```
intdiv = Math.DivRem(14, 5, outresult);
//result = 4
// div = 2
```

`Exp(double d)`: возвращает основание натурального логарифма, возведенное в степень d

`Floor(decimal d)`: возвращает наибольшее целое число, которое не больше d

```
doubleresult = Math.Floor(2.56); // 2
```

`IEEERemainder(double a, double b)`: возвращает остаток от деления a на b

```
doubleresult = Math.IEEERemainder(26, 4); // 2 = 26-24
```

`Log(double d)`: возвращает натуральный логарифм числа d

`Log(double a, doublenewBase)`: возвращает логарифм числа a по основанию newBase

`Log10(double d)`: возвращает десятичный логарифм числа d

`Max(double a, double b)`: возвращает максимальное число из a и b

`Min(double a, double b)`: возвращает минимальное число из a и b

`Pow(double a, double b)`: возвращает число a, возведенное в степень b

`Round(double d)`: возвращает число d, округленное до ближайшего целого числа

```
doubleresult1 = Math.Round(20.56); // 21
doubleresult2 = Math.Round(20.46); //20
```

`Round(double a, round b)`: возвращает число a, округленное до определенного количества знаков после запятой, представленного параметром b

```
doubleresult1 = Math.Round(20.567, 2); // 20,57
doubleresult2 = Math.Round(20.463, 1); //20,5
```

`Sign(doublevalue)`: возвращает число 1, если число value положительное, и -1, если значение value отрицательное. Если value равно 0, то возвращает 0

```
intresult1 = Math.Sign(15); // 1
intresult2 = Math.Sign(-5); //-1
```

`Sin(double value)`: возвращаетсинусугла value

`Sinh(double value)`: возвращаетгиперболическийсинусугла value

`Sqrt(doublevalue)`: возвращает квадратный корень числа value

```
doubleresult1 = Math.Sqrt(16); // 4
```

`Tan(doublevalue)`: возвращает тангенс угла value

`Tanh(doublevalue)`: возвращает гиперболический тангенс угла value

`Truncate(doublevalue)`: отбрасывает дробную часть числа value, возвращаяя лишь целое значние

```
doubleresult = Math.Truncate(16.89); // 16
```

Также класс Math определяет две константы: `Math.E` и `Math.PI`. Например, вычислим площадь круга:

```
Console.WriteLine("Введите радиус круга");
Double radius = Double.Parse(Console.ReadLine());
Double area = Math.PI * Math.Pow(radius, 2);
Console.WriteLine("Площадькругасрадиусом {0} равна {1}
```

Консольный вывод:

```
Введите радиус круга
20
Площадь круга с радиусом 20 равна 1256,63706143592
```

**Задания**

1 Изучить теоретические сведения и задание к работе

2 В соответствии с вариантом задания практической работы 4 разработать приложение WindowsForm.

3 На созданной форме расположить задание по варианту

**Порядок выполнения работы (Пример выполнения)**

**Исходные данные:**

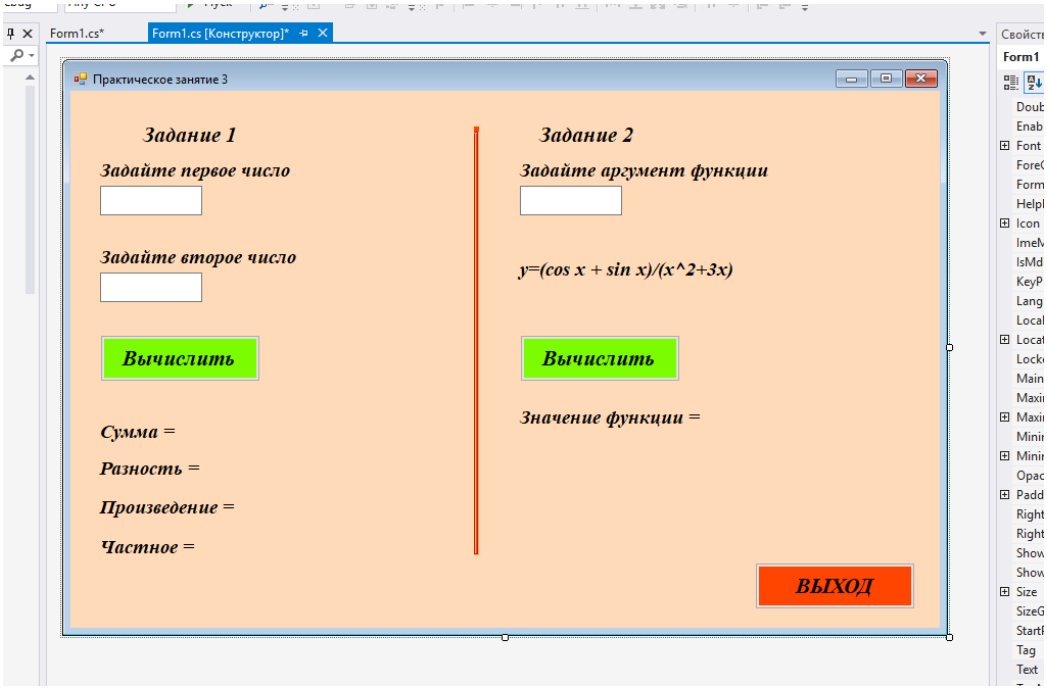
Даны два ненулевых числа. Найти сумму, разность, произведение и частное этих чисел

**Решение:**

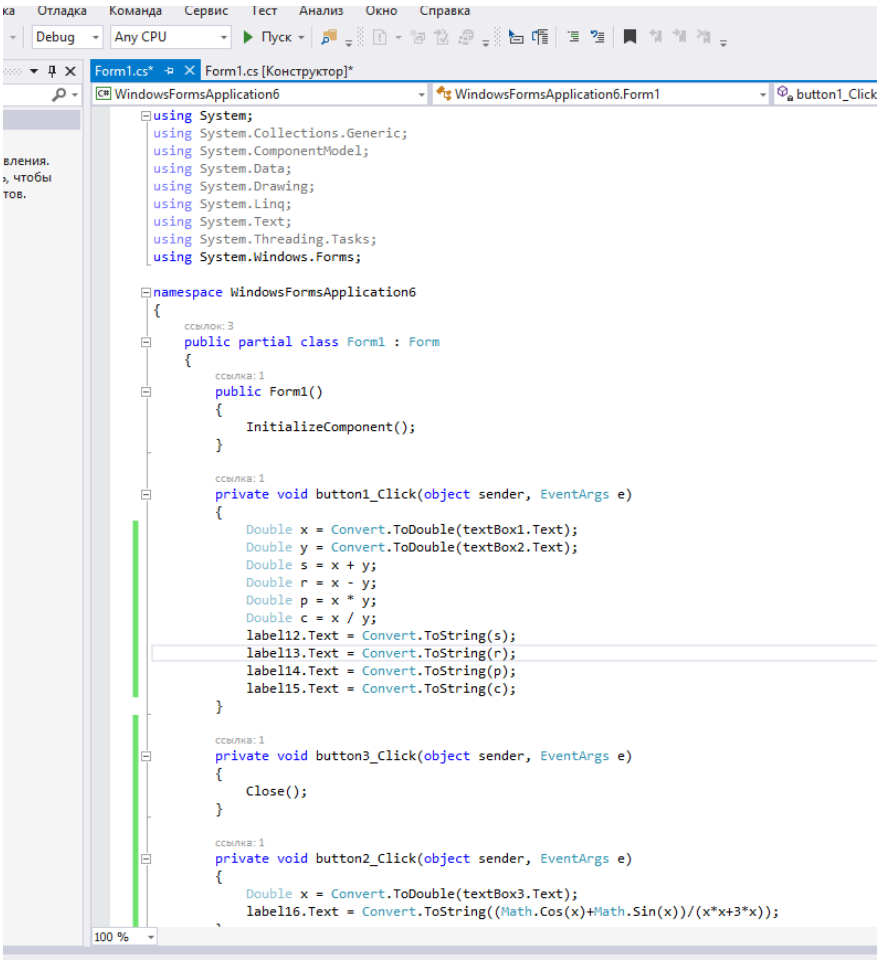
1 Текст программы практической работы 2

```
Double x = Convert.ToDouble(Console.ReadLine());
Double y = Convert.ToDouble(Console.ReadLine());
Double s = x + y;
Double r = x - y;
Double p = x * y;
Double c = x / y;
Console.WriteLine("  Сумма = "+s+"  Разность = "+ r+"  Произведение = "+ p+ "  Частное = "+c);
Console.ReadKey();
```

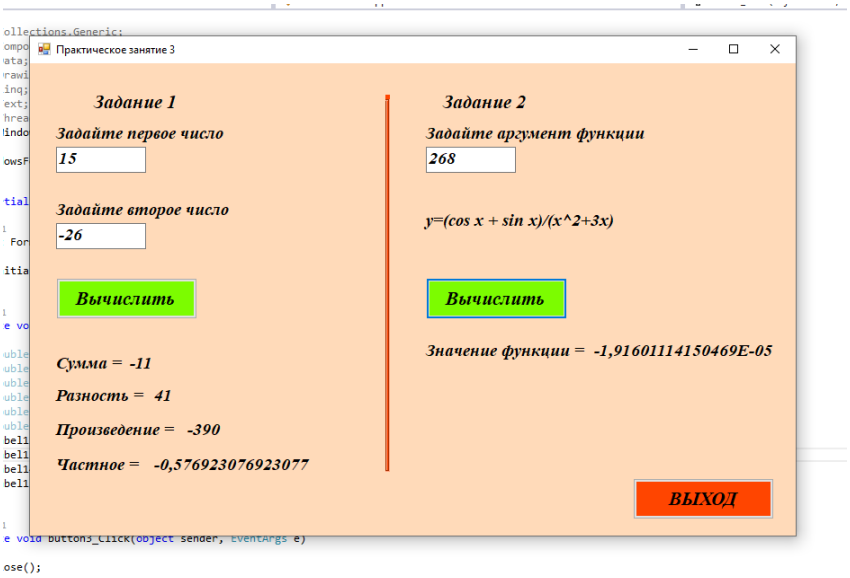
2 Интерфейс формы в VisualStudio




3 Код программы в VisualStudio




4 Результат работы программы:




## Содержание отчета

1 Название работы
2 Цель работы
3 Технические средства обучения
4 Задания (условия задач)
5 Порядок выполнения работы
6 Ответы на контрольные вопросы
**7** Вывод

## Варианты заданий

### Задание 3

| | | |
|---|---|---|
| Вариант 1<br>$\frac{\sqrt{x}}{x+1}$ | Вариант 2<br>$3\sin\sqrt{3}+0{,}35x-3{,}8$ | Вариант 3<br>$\cos\frac{2}{x}-2\sin\frac{1}{x}-\frac{1}{x}$ |
| Вариант 4<br>$\sqrt{1-x}-tgx$ | Вариант 5<br>$\sqrt{x^5-x+1{,}8}$ | Вариант 6<br>$\frac{x+2x^3+1{,}9}{\sqrt{x-1{,}5}}$ |
| Вариант 7<br>$2x\sin x-\cos x$ | Вариант 8<br>$x-2+\sin\frac{1}{x}$ | Вариант 9<br>$\frac{\sqrt{x+x^3+1{,}5}}{x+1}$ |
| Вариант 10<br>$1-x+\sin x-\cos(1+x)$ | Вариант 11<br>$3x-4\sin x^2$ | Вариант 12<br>$\sin x^2+\cos x^2-10x$ |

| | | |
|---|---|---|
| Вариант 13<br>$1-tgx-\cos(2+3x)$ | Вариант 14<br>$5x^2-4\cos x^3$ | Вариант 15<br>$\sin x^2-\cos x^2+6x$ |
| Вариант 16<br>$2x\sin x-\cos x\sin x$ | Вариант 17<br>$x-2+\sin\frac{1}{x}$ | Вариант 18<br>$\frac{\sqrt{x^3+2,5}}{x+2}$ |
| Вариант 19<br>$\sqrt{5-x}-tgx^2$ | Вариант 20<br>$x^5-x^2+2,6$ | Вариант 21<br>$\frac{x^2+2x^3+3x}{\sqrt{x-1}}$ |
| Вариант 22<br>$\frac{\sqrt{x+3}}{x-1}$ | Вариант 23<br>$\sin x\sqrt{3}+0,2x-3,5$ | Вариант 24<br>$\cos\frac{4}{x}+2\sin\frac{4}{x}+\frac{4}{x}$ |
| Вариант 25<br>$2-x+\sin x-\cos x$ | Вариант 26<br>$2x^3-3\sin x^2$ | Вариант 27<br>$\sin x^3+\cos x^3-3x^2$ |
| Вариант 28<br>$3+\sin x^2-\sqrt{5-x}$ | Вариант 29<br>$2\ln x^3-2\sin x+1$ | Вариант 30<br>$\sin x+\ln(3-x)$ |

**Контрольные вопросы:**

1 Когда в C# происходит неявное преобразование типов автоматически?

2 Почему приходится обращаться к явному приведению типов в языке C#?

3 Как преобразовать переменные к нужному типу данных?

4 Как преобразовать переменные к строковому представлению данных?

5 Как в C# записать математические вычисления с использованием основных функций?

**Используемая литература**

1. Гниденко, И. Г. Технология разработки программного обеспечения : учеб. пособие для СПО / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — М.: Издательство Юрайт, 2017.

2. Шарп Джон Ш26 Microsoft Visual C#. Подробное руководство. 8-е изд. — СПб.: Питер, 2017.

3. Васильев А.Н. Программирование на C# для начинающих. Основные сведения. – Москва: Эксмо, 2018.
4. Васильев А.Н. Программирование на C# для начинающих. Особенности языка. – Москва: Эксмо, 2019.
5. http://msdn.microsoft.com/ru-ru/library/67ef8sbd.aspx.