

Лабораторная работа №1

Выполнение программы простой структуры. Вычисление выражений с использованием стандартных функций

1. Цель задания:

- 1) Выполнение простой программы в системе программирования MS Visual Studio
- 2) Приобретение навыков в записи выражений на языке C# и использование стандартных функций.

2. Теоретические сведения

2.1. Общая характеристика платформы MSDN

Платформа MSDN (Microsoft.Net) предназначена для разработки и исполнения приложений различных типов:

- автономное консольное приложение с использованием текстового интерфейса пользователя;
- автономное Windows-приложение с использованием графического интерфейса пользователя;
- библиотека классов, которые предназначены для использования в других приложениях;
- Web-приложение, доступ к которому выполняется через браузер и которое по запросу формирует Web-страницу и отправляет ее клиенту по сети;
- Web-сервис – компонент, методы которого могут вызываться через Интернет.

Приложение может выполняться в режиме управляемого кода или небезопасного кода.

В первом случае исходный код должен быть переведен на специально разработанный для платформы промежуточный язык MSIL (MS Common Intermediate Language).

Язык CIL по синтаксису и мнемонике напоминает язык ассемблера. Его можно рассматривать как ассемблер виртуальной машины .NET. В то же время язык CIL содержит некоторые достаточно высокоуровневые конструкции, повышающие его уровень по сравнению с ассемблером для любой реально существующей машины, и писать код непосредственно на CIL легче, чем на ассемблере для реальных машин. Поэтому его можно рассматривать как своеобразный «высокоуровневый ассемблер».

Для исполнения кода на промежуточном языке приложения используется специальная программная компонента платформы – общезыковая среда исполнения CLR (Common Language Runtime) — «общезыковая исполняющая среда» — компонент пакета Microsoft .NET Framework, виртуальная машина, исполняющий программы, написанные на .NET-совместимых языках программирования.

CLR интерпретирует и исполняет код на языке MSIL, а также предоставляет MSIL-программам (а следовательно, и программам, написанным на языках высокого уровня, поддерживающих .NET Framework) доступ к библиотекам классов .NET Framework, или так называемой .NET FCL (Framework Class Library).

Одной из основных идей Microsoft .NET является совместимость программных частей, написанных на разных языках. Например, служба, написанная на C++ для Microsoft .NET, может обратиться к методу класса из библиотеки, написанной на Delphi; на C# можно написать класс, наследованный от класса, написанного на Visual Basic .NET, а исключение, созданное методом, написанным на C#, может быть перехвачено и

обработано в Delphi. Каждая библиотека (сборка) в .NET имеет сведения о своей версии, что позволяет устранить возможные конфликты между разными версиями сборок.

Во втором случае исходный код должен быть переведен на язык машинных команд. Машинный код выполняется непосредственно под управлением операционной системы.

Основные преимущества платформы проявляются в режиме управляемого кода. Этот режим принят по умолчанию. Все сказанное в дальнейшем относится к этому режиму.

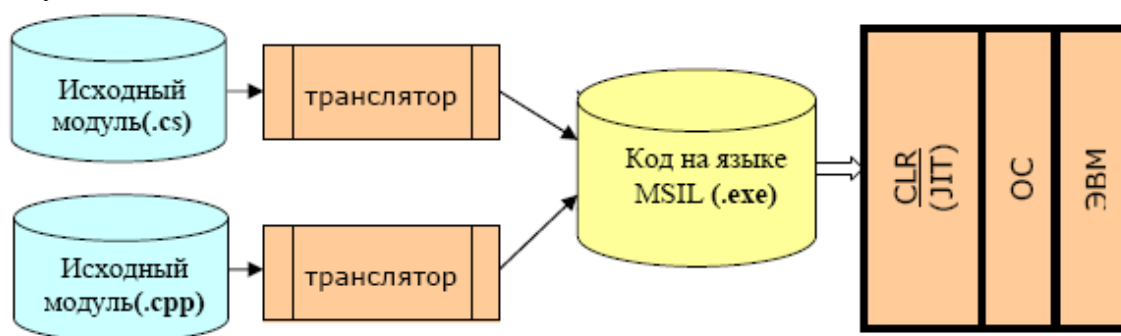


Рисунок 1. Трансляция исходных модулей на язык MSIL

Исходные модули транслируются на промежуточный язык MSIL, как показано на Рисунке 1. Код на промежуточном языке рассматривается средой исполнения CLR как исполняемый модуль. Среда исполнения представляет собой программу, настроенную над операционной системой Windows и выполняемую под ее управлением. С другой стороны, среда исполнения представляет собой функциональный аналог ЭВМ, - виртуальную машину, в которой выполняются программы на промежуточном языке.

При вызове метода среда исполнения активизирует транслятор CLR, который переводит код метода с промежуточного языка в машинный код и сохраняет его в памяти. При повторном вызове метода повторная трансляция не выполняется, используется машинный код, сохраненный в памяти. За счет такого подхода достигается эффективность, соизмеримая с эффективностью неуправляемого кода и экономия расхода основной памяти за счет перевода в машинный код не всей программы, а только тех методов, которые были реально вызваны в процессе выполнения программы.

Таким образом, каркас платформы образуют две компоненты, показанные на Рис.5:

- Статическая компонента – базовая библиотека классов, содержащая обширный набор готовых к использованию программных компонент на промежуточном языке. Базовая библиотека классов является общей для всех языков программирования, поддерживаемых в платформе.
- Динамическая компонента – общезыковая среда исполнения (CLR).

Указанные компоненты являются обязательными для исполнения программ на промежуточном языке MSIL в случае использования на ЭВМ операционных систем Windows. В перспективных операционных системах семейства Windows предполагается включение базовой библиотеки классов и средств исполнения в состав операционной системы.

Интегрированная среда разработки MS Visual Studio.NET представляет собой программную компоненту, поддерживающую процесс разработки программ. Возможности интегрированной среды для приложений на всех языках примерно равноценны, но в наибольшей степени возможности среды раскрываются при разработке программ на языке C#. С помощью средств MS Visual Studio.NET выполняется редактирование исходного кода, выполнение приложения в отладочном режиме, визуальное отображение логической структуры приложения, выдачи справочной информации по самой среде, платформе и языкам программирования, что является традиционным для большинства интегрированных сред.

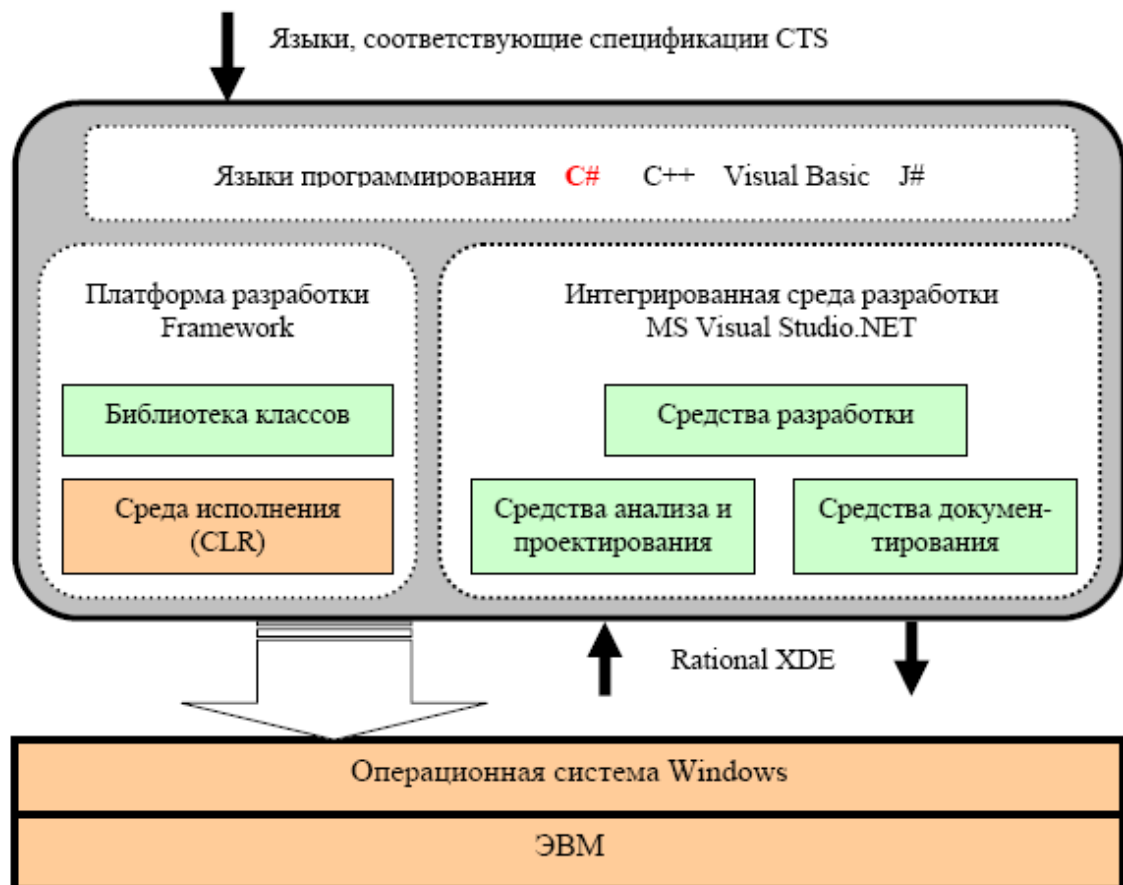


Рисунок 2. Интегрированная среда разработки MS Visual Studio.NET

Особенностью MS Visual Studio.NET является возможность автоматической генерации исходного кода по визуальному представлению диаграммы классов в нотации UML и наоборот, построение визуального представления диаграммы классов по исходному коду программы.

В MS Visual Studio.NET может быть интегрировано специализированное инструментальное средство Rational XDE, ориентированное на решение задач анализа и проектирования программ на языке C# на основе объектно-ориентированного подхода, в результате чего возможности среды по анализу, проектированию и документированию существенно расширяются.

Приложение в процессе разработки называют проектом. Проект логически объединяет все необходимые для создания приложения файлы, папки и прочие ресурсы. Типовая структура консольного приложения с некоторыми упрощениями приведена на Рисунке 3.

Описание структуры проекта хранится в специальном файле с расширением csproj. Несколько проектов логически могут быть объединены в одно решение. Структура решения хранится в файле с расширением sln. С помощью инспектора решения Solution Explorer, входящего в состав интегрированной среды можно просматривать логическую структуру всех проектов, включенных в решение и выполнять операции по изменению логической структуры решения и проектов.

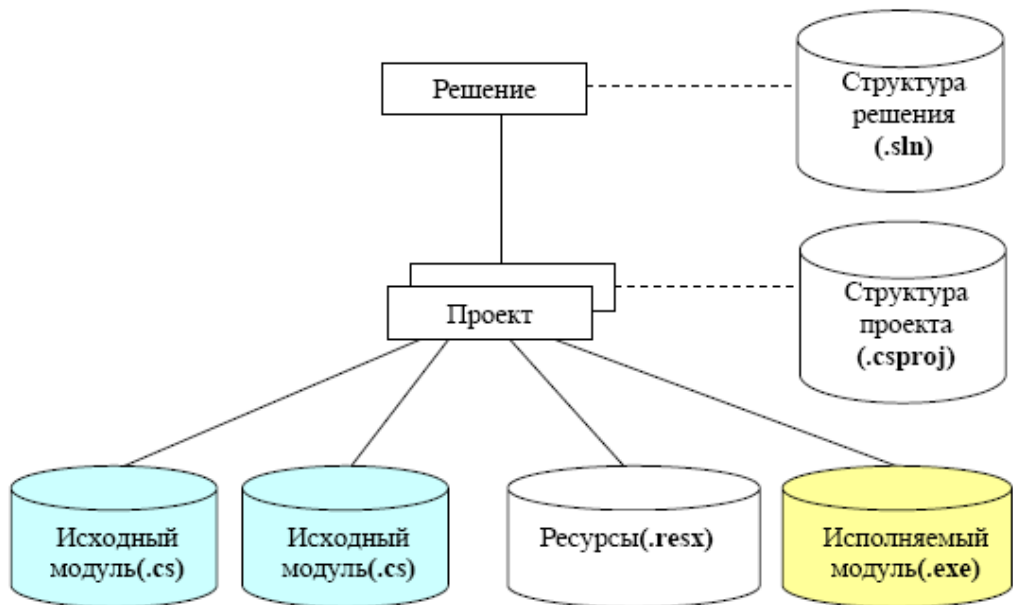


Рисунок 3 Типовая структура консольного приложения

2.2. Консольные приложения

Среда Visual Studio.NET работает на платформе Windows и ориентирована на создание Windows- и веб-приложений, однако разработчики предусмотрели работу с консольными приложениями. При запуске консольного приложения операционная система создает так называемое консольное окно, через которое идет весь ввод-вывод программы. Внешне это напоминает работу в операционной системе в режиме командной строки, когда ввод-вывод представляет собой поток символов.

Консольные приложения наилучшим образом подходят для изучения языка, так как в них не используется множество стандартных объектов, необходимых для создания графического интерфейса.

2.3. Структура программы на C++

C# — объектно-ориентированный язык, поэтому написанная на нем программа представляет собой совокупность взаимодействующих между собой классов. Описание класса начинается с ключевого слова `class`, за которым следуют его имя и далее в фигурных скобках — список элементов класса (его данных и функций, называемых также методами).

Каждое приложение должно содержать метод `Main` — с него начинается выполнение программы. Все методы описываются по единым правилам.

Упрощенный синтаксис метода:

```

[ спецификаторы ] тип имя_метода ( [ параметры ] )
{
    тело метода: действия, выполняемые методом
}
  
```

Директива `using System` разрешает использовать имена стандартных классов из пространства имен `System` непосредственно (без указания имени пространства). Ключевое слово `namespace` создает для проекта собственное пространство имен.

```

//подключаем пространство имен System
using System;
//пространство имен
namespace App
{
  
```

```

    //класс
class Program
{
    //метод
    static void Main(string[] args)
    {
        // текст программы
    }
}
}

```

2.4. Элементы языка C#

1) Алфавит языка который включает

- прописные и строчные латинские буквы и знак подчеркивания;
- арабские цифры от 0 до 9;
- специальные знаки "{ } , | [] () + - / % * . \ ' : ; & ? < > = ! # ^
- пробельные символы (пробел, символ табуляции, символы перехода на новую строку).

2) Из символов формируются лексемы языка:

- *Идентификаторы* – имена объектов. В идентификаторе могут быть использованы латинские буквы, цифры и знак подчеркивания. Прописные и строчные буквы различаются, например, PROG1, prog1 и Progl – три различных идентификатора. Первым символом должна быть буква или знак подчеркивания (но не цифра). Пробелы в идентификаторах не допускаются. В идентификаторах C# разрешается использовать помимо латинских букв буквы национальных алфавитов, например, Массив – разрешенный идентификатор.
- *Ключевые (зарезервированные) слова* – это слова, которые имеют специальное значение для компилятора. Их нельзя использовать в качестве идентификаторов. Можно использовать ключевые слова в качестве идентификаторов, если поставить перед ними @, например, @if – разрешенный идентификатор.
- *Знаки операций* – это один или несколько символов, определяющих действие над операндами. Операции делятся на унарные, бинарные и тернарную по количеству участвующих в этой операции операндов.
- *Константы* – это неизменяемые величины. Существуют логические, целые, вещественные, символьные и строковые константы. Компилятор выделяет константу в качестве лексемы (элементарной конструкции) и относит ее к одному из типов по ее внешнему виду.
- *Разделители* – скобки, точка, запятая пробельные символы.

2.5. Константы в C#

Константа – это лексема, представляющая изображение фиксированного числового, строкового или символьного значения. Константы делятся на 5 групп:

- логические;
- целые;
- вещественные (с плавающей точкой);
- символьные;
- строковые.

Компилятор выделяет лексему и относит ее к той или другой группе, а затем внутри группы к определенному типу по ее форме записи в тексте программы и по числовому значению.

Целые константы могут быть десятичными и шестнадцатеричными.

Таблица 1. Целые константы

Название	Определение	Примеры
Логическая константа	Истина (true) или ложь (false)	true false
Десятичная константа	Последовательность десятичных цифр, за которой могут следовать символы U/u (unsigned) и/или L/l (long)	8, 0, 192345 8u, 10451, 34lu, 123UL
Шестнадцатеричная константа	Последовательность шестнадцатеричных цифр, которым предшествуют символы 0x или 0X, за цифрами могут следовать символы U/u (unsigned) и/или L/l (long)	0xA, 0X00F, 0x123 0x1AFLU, 0XFFu

Вещественные константы могут иметь две формы представления: с фиксированной точкой и с плавающей точкой.

Таблица 2. Вещественные константы

Название	Вид	Примеры
Константы с фиксированной точкой	[цифры].[цифры][суффикс] Суффикс – это символы F/f (float) или D/d (double) или M/m (decimal)	5.7, .0001, 41. 5.7d, .0001f, 41.M
Константа с плавающей точкой	[цифры][.][цифры]E e[+ -] [цифры] [суффикс]	0.5e5, .11e-5, 5E3 0.5e5d, .11e-5f, 5E3d

Символьная константа – представляет собой любой символ в кодировке Unicode. Символьные константы записываются в одной из четырех форм:

- «обычный» символ, имеющий графическое представление (кроме апострофа и символа перевода строки), — 'А', '5', '*', 'ю';
- управляющая последовательность — '\0', '\n';
- символ в виде шестнадцатеричного кода — '\xF', '\x74';
- символ в виде escape-последовательности Unicode — '\uA81B'.

Последовательности, начинающиеся со знака \, называются управляющими, они используются:

- для представления символов, не имеющих графического отображения, например:
 - \a – звуковой сигнал,
 - \b – возврат на один шаг,
 - \n – перевод строки,
 - \t – горизонтальная табуляция;
 - \0 – нуль-символ.
- для представления символов: \, ', ?, " (\\, \', \?, \");
- для представления символов с помощью шестнадцатеричных кодов (\x73, \0xF5).

Escape-последовательности Unicode служат для представления символа в кодировке Unicode с помощью его кода в шестнадцатеричном виде с префиксом \u или \U.

Строковая константа – это последовательность символов, заключенная в кавычки. Внутри строк также могут использоваться управляющие символы. Например:

`"\nНовая строка",`

`"\nНовый курс\"Алгоритмические языки программирования\""`.

В C# введен второй вид строковых констант – дословные литералы (verbatim strings). Эти литералы предваряются символом @, который отключает обработку управляющих последовательностей и позволяет получать строки в том виде, в котором они записаны.

`@"Новый курс "Алгоритмические языки программирования"`

2.6. Типы данных в C#

Тип данных однозначно определяет:

- внутреннее представление данных, а следовательно, и множество их возможных значений;
- допустимые действия над данными (операции и функции).

Типы данных можно *классифицировать* по различным признакам.

- По структуре типы C# можно разделить на простые и структурированные. К простым типам относят типы, которые характеризуются одним значением.
- По тому, кем создан тип данных, выделяют встроенные и пользовательские типы данных. Встроенные типы данных уже существуют в языке, пользовательские типы создаются программистом (классы).
- По тому, в какой момент выделяется память под переменные данного типа, выделяют статические и динамические данные. Под статические переменные память выделяется в процессе компиляции программы, под динамические данные память выделяется во время выполнения программы с помощью специальных операций (new).
- По способу хранения выделяют типы-значения и типы-ссылки. Переменные типов-значений представляют собой последовательность байтов в памяти, необходимый объем памяти выделяет компилятор. Переменная ссылочного типа хранит адрес, по которому расположены данные, сами данные хранятся в динамической памяти.

Таблица 3. Встроенные типы данных C#

Тип данных	Определение	Размер	Диапазон
bool	Логический тип данных, данные этого типа могут принимать значения true и false.	1 байт	true, false
sbyte	Знаковый целый тип	1 байт	-128 .. 127
byte	Беззнаковый целый тип	1 байт	0 .. 255
short	Знаковый целый тип	2 байта	-32768 .. 32767
ushort	Беззнаковый целый тип	2 байта	0 .. 65535
int	Знаковый целый тип	4 байта	$-2 \cdot 10^9 \dots 2 \cdot 10^9$
uint	Беззнаковый целый тип	4 байта	$0 \dots 4 \cdot 10^9$
long	Знаковый целый тип	8 байт	$-9 \cdot 10^{18} \dots 9 \cdot 10^{18}$
ulong	Беззнаковый целый тип	8 байт	$0 \dots 18 \cdot 10^{18}$
char	Символьный тип, Unicode-символ	1 байт	U+0000 .. U+ffff
float	Вещественный тип	4 байта	$1.5 \cdot 10^{-45} \dots 3.4 \cdot 10^{38}$
double	Вещественный тип	8 байт	$5.0 \cdot 10^{-324} \dots 1.7 \cdot 10^{308}$
decimal	Финансовый тип для денежных вычислений	16 байт	$1.0 \cdot 10^{-28} \dots 7.9 \cdot 10^{28}$

string	Строковый тип, строка Unicode символов	Длина ограничена объемом доступной памяти	
object	Всеобщий предок, можно хранить, что угодно		

2.7. Переменные

Переменная в C# – именованная область памяти, в которой хранятся данные определенного типа. У переменной есть имя и значение. Имя служит для обращения к области памяти, в которой хранится значение. Перед использованием любая переменная должна быть описана. Имя переменной должно соответствовать правилам, по которым формируются идентификаторы C#, отражать смысл хранимой величины и быть легко распознаваемым. Тип переменной выбирается исходя из диапазона и требуемой точности представления данных.

```
int a; float x;
```

2.8. Операции

В соответствии с количеством операндов, которые используются в операциях они делятся на унарные (один операнд), бинарные (два операнда) и тернарную (три операнда).

Операция	Описание
Унарные операции	
.	Доступ к элементу структуры
[]	Доступ к элементу массива
++	Увеличение на единицу: префиксная операция - увеличивает операнд до его использования, постфиксная операция увеличивает операнд после его использования.
--	Уменьшение на единицу: префиксная операция - уменьшает операнд до его использования, постфиксная операция уменьшает операнд после его использования.
typeof	Получение типа
-	Унарный минус
+	Унарный плюс
!	Логическое отрицание (НЕ). В качестве логических значений используется 0 (false) - ложь и не 0 (true) - истина, отрицанием 0 будет 1, отрицанием любого ненулевого числа будет 0.
new	Выделение памяти
(тип)	Преобразование типа
Бинарные операции	
Мультипликативные	
*	умножение операндов арифметического типа
/	деление операндов арифметического типа (если операнды целочисленные, то выполняется целочисленное деление)
%	получение остатка от деления целочисленных операндов
Аддитивные	
+	бинарный плюс (сложение арифметических операндов)

-	бинарный минус (вычитание арифметических операндов)
Операции отношения и проверки типа	
<	меньше, чем
<=	меньше или равно
>	больше
>=	больше или равно
is	проверка принадлежности типу
as	приведение типа
Операции сравнения	
==	равно
!=	не равно
Логические операции	
&&	конъюнкция (И) целочисленных операндов или отношений, целочисленный результат ложь(0) или истина(не 0)
	дизъюнкция (ИЛИ) целочисленных операндов или отношений, целочисленный результат ложь(0) или истина(не 0)
Тернарная	
?:	Условная операция в ней используется три операнда. Выражение1 ? Выражение2 : Выражение3; Первым вычисляется значение выражения1. Если оно истинно, то вычисляется значение выражения2, которое становится результатом. Если при вычислении выражения1 получится 0, то в качестве результата берется значение выражения3. Например: x<0 ? -x : x ; //вычисляется абсолютное значение x.
Присваивание	
=	присваивание
*=	умножение с присваиванием (мультипликативное присваивание)
/=	деление с присваиванием
%=	деление с остатком с присваиванием
+=	сложение с присваиванием
-=	вычитание с присваиванием

Приоритеты операций.

Ранг	Операции
1	() [] .
2	! - ++ -- (тип) sizeof
3	* / % (мультипликативные бинарные)
4	+ - (аддитивные бинарные)
5	< > <= >= (отношения)
6	== != (сравнения)
7	&& (конъюнкция «И»)
8	(дизъюнкция «ИЛИ»)
9	?: (условная операция)
10	= *= /= %= -= &= ^= = <<= >>= (операция присваивания)

2.9. Выражения

Из констант, переменных, разделителей и знаков операций можно конструировать выражения. Каждое выражение представляет собой правило вычисления нового значения. Каждое выражение состоит из одного или нескольких операндов, символов операций и ограничителей. Если выражение формирует целое или вещественное число, то оно называется арифметическим. Пара арифметических выражений, объединенная операцией сравнения, называется отношением. Если отношение имеет ненулевое значение, то оно – истинно, иначе – ложно.

2.10. Ввод и вывод данных

Любая программа при вводе исходных данных и выводе результатов взаимодействует с внешними устройствами. Совокупность стандартных устройств ввода и вывода, то есть клавиатуры и экрана, называется консолью. В языке C#, как и во многих других, нет операторов ввода и вывода. Вместо них для обмена с внешними устройствами применяются стандартные объекты. Для работы с консолью в C# применяется класс Console, определенный в пространстве имен System. В этом классе определены методы:

```
Console.Write(форматная строка, [параметры]);
```

```
Console.WriteLine(форматная строка, [параметры]);
```

где форматная строка – это строковая константа, в которой содержится текст для вывода, параметры – это переменные, значения которых будут при выводе подставлены в форматную строку.

Методы отличаются друг от друга тем, что метод WriteLine() выполняет переход на следующую строку после вывода форматной строки.

```
int x=10, y=15;  
Console.WriteLine("x={0}, \ny={1}", x, y);
```

```
int x=10, y=15;  
Console.WriteLine("x="+x);  
Console.WriteLine("y="+y);
```

```
int x=10, y=15;  
Console.Write("x="+x+"\n");  
Console.Write("y="+y+"\n");
```

При вводе данных с клавиатуры используются методы Console.Read() и Console.ReadLine().

В классе Console определены методы ввода строки и отдельного символа, но нет методов, которые позволяют непосредственно считывать с клавиатуры числа. Ввод числовых данных выполняется в два этапа:

- 1) Символы, представляющие собой число, вводятся с клавиатуры в строковую переменную.
- 2) Выполняется преобразование из строки в переменную соответствующего типа.

```
string buf;  
Console.WriteLine("Введите значение переменной");  
buf=Console.ReadLine();  
int x=int.Parse(buf); //преобразует строку в целое число
```

2.11. Использование математических функций

В выражениях часто используются математические функции, например синус или возведение в степень. Они реализованы в классе Math, определенном в пространстве имен System.

Таблица 4 Основные поля и методы класса Math

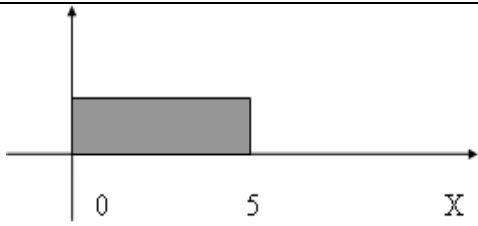
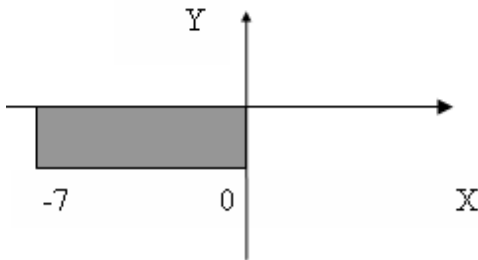
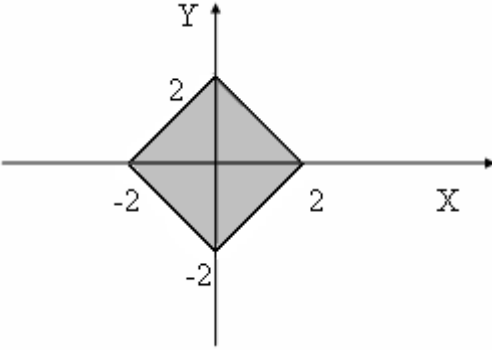
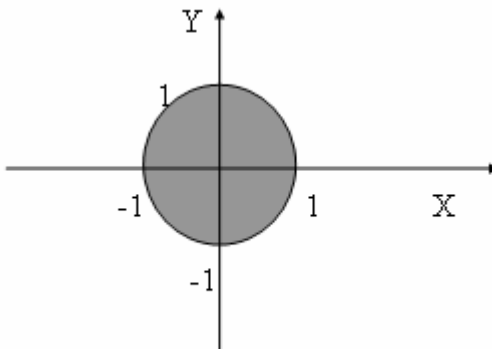
Функция	Описание	Пример вызова
Abs()	Абсолютное значение	int a=Math.Abs(-10);
Acos()	Арккосинус	double x; ... double y=Math.Acos(x);
Asin()	Арксинус	double x; ... double y=Math.Asin(x);
Atan()	Арктангенс	double x; ... double y=Math.Atan(x);
Atan2()	Арктангенс - угол, тангенс которого это результат деления второго аргумента на первый	double x,y; ... double z=Math.Atan(x,y);
BigMul()	Произведение	long x,y; ... long z=Math.BigMul(x,y);
Celling()	Округление до большего целого	double a=Math.Celling(3.4567);
Cos()	Косинус	double x; ... double y=Math.Cos(x);
Cosh()	Гиперболический косинус	double x; ... double y=Math.Cosh(x);
DivRem()	Деление и остаток	int b, c; c = Math.DivRem(15, 5, out b);
E	База натурального логарифма	2.71828
Exp()	Экспонента	double x; ... double y=Math.Exp(x);
Floor()	Округление до меньшего целого	double a=Math.Floor(3.4567);
IEEERemainder()	Остаток от деления	double a=Math. IEEERemainder (12, 5);
Log()	Натуральный логарифм	double x;..... double y=Math.Log(x);
Log10()	Десятичный логарифм	double x;..... double y=Math.Log10(x);
Max()	Максимум из двух чисел	int x= Math.Max(3,6);
Min()	Минимум из двух чисел	int x= Math.Min(3,6);
Pi	Значение числа пи	3.14159
Pow()	Возведение в степень	double x=Math.Pow(2,3);
Round()	Округление	double a=Math.Round (3.4567);
Sign()	Знак числа	int x=Math.Sign(-1);
Sin()	Синус	double x; ... double y=Math.Sin(x);
Sinh()	Гиперболический синус	double x; ... double y=Math.Sinh(x);
Sqrt()	Квадратный корень	double x; ... double y=Math.Sqrt(x);
Tan()	Тангенс	double x; ... double y=Math.Tan(x);
Tanh()	Гиперболический тангенс	double x; ... double y=Math.Tanh(x);

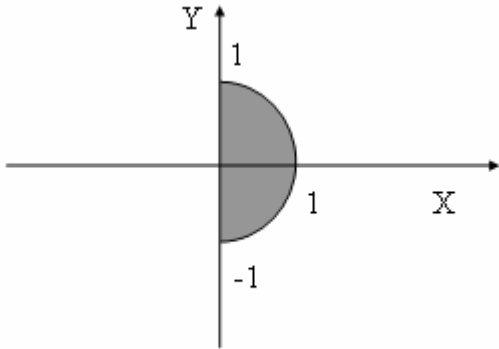
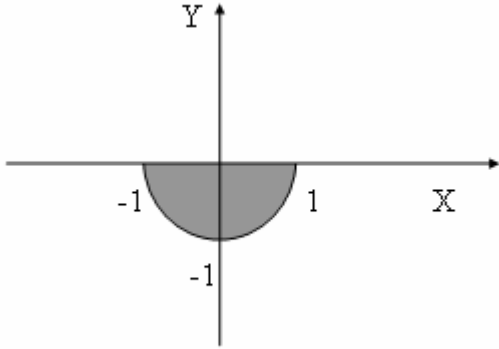
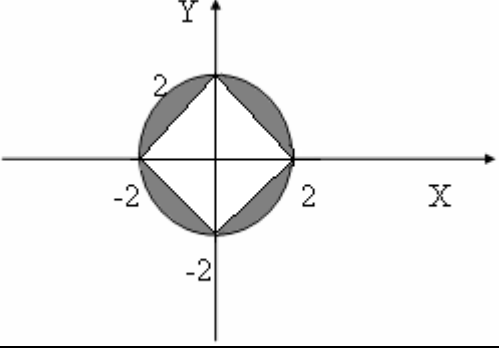
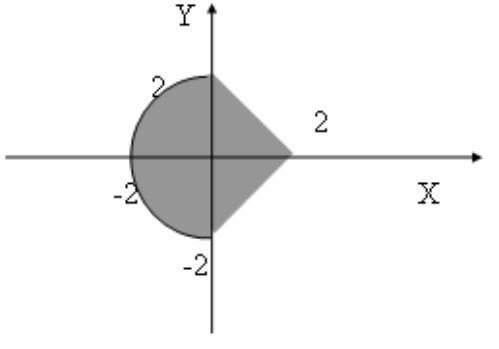
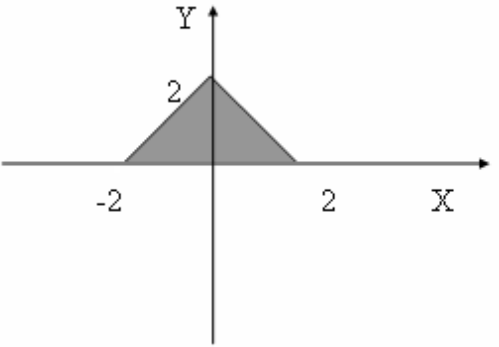
3. Постановка задачи

1. Для задачи 1 определить тип заданных выражений и найти их значения.
2. Составить систему тестов и вычислить полученное выражение для нескольких значений X, определить при каких X выражение не может быть вычислено.

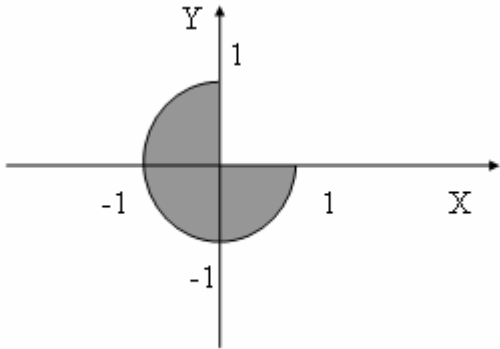
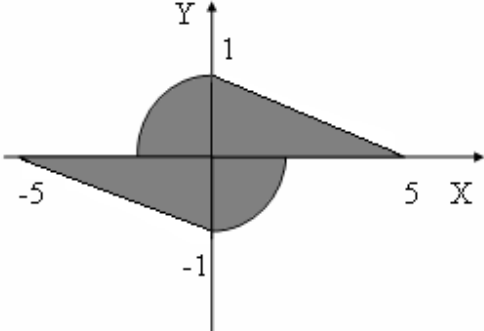
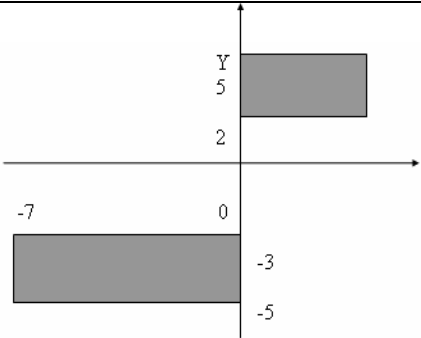
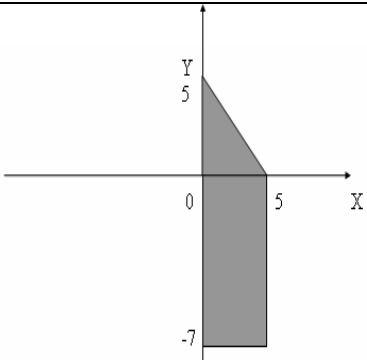
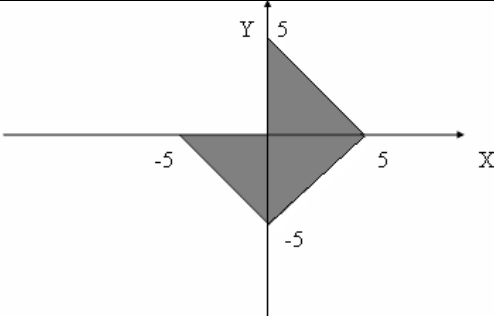
- Для задачи 2 записать выражение, зависящее от координат точки X1 и Y1 и принимающее значение TRUE, если точка принадлежит заштрихованной области, и FALSE, если не принадлежит.
- Составить систему тестов и вычислить полученное выражение для нескольких точек, принадлежащих и не принадлежащих заштрихованной области.
- Для задачи 3 вычислить значение выражения, используя различные вещественные типы данных (float и double).
- Результаты всех вычислений вывести на печать.
- Объяснить полученные результаты.

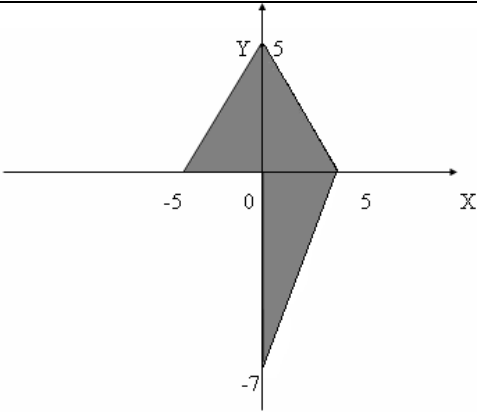
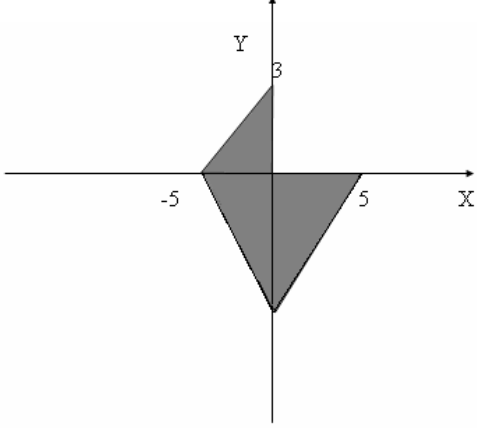
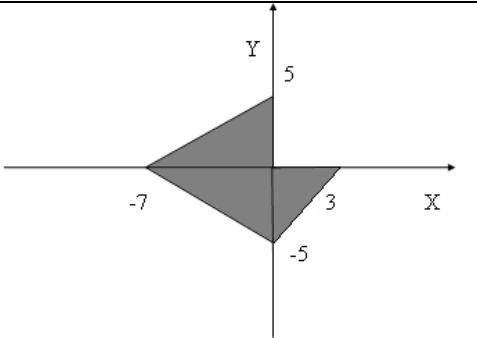
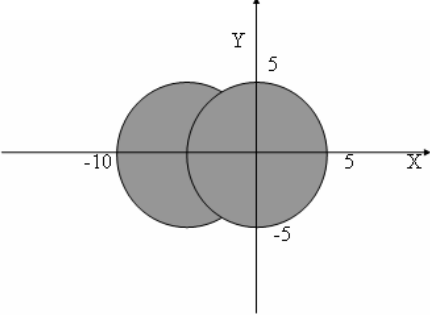
1. Варианты

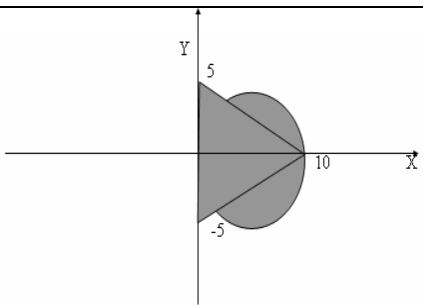
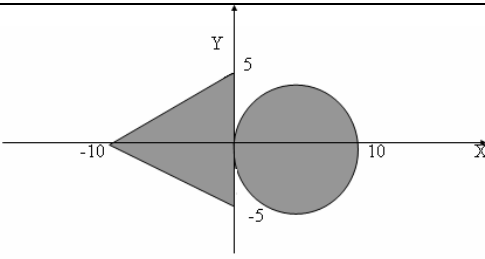
№	Задача 1	Задача 2	Задача 3
1	1) $n+++m$ 2) $m-->n$ 3) $n-->m$ 4) $\sin(x) + x^3 + \frac{1}{x^2 + 1}$		$\frac{(a+b)^2 - (a^2 + 2ab)}{b^2}$ $a=1000, b=0.0001$
2	1) $++n*++m$ 2) $m++<n$ 3) $n++>m$ 4) $x + \frac{1}{x^3 - x} - 2$		$\frac{(a-b)^2 - (a^2 - 2ab)}{b^2}$ $a=1000, b=0.0001$
3	1) $m--n$ 2) $m++<n$ 3) $n++>m$ 4) $x^4 - \cos(\arcsin(x))$		$\frac{(a+b)^3 - (a^3 + 3a^2b)}{3ab^2 + b^2}$ $a=100, b=0.001$
4	1) $n++*m$ 2) $n++<m$ 3) $--m>n$ 4) $\sqrt[3]{x - x^2 + x^5}$		$\frac{(a-b)^3 - (a^3)}{3ab^2 - b^3 - 3a^2b}$ $a=100, b=0.001$

5	1) $--m-n++$ 2) $m*m<n++$ 3) $n-->++m$ 4) $tg(x) - (5-x)^4$		$\frac{(a-b)^3 - (a^3 - 3a^2b)}{3ab^2 - b^3}$ $a=100, b=0.001$
6	1) $m-++n$ 2) $m++>--n$ 3) $m--<++n$ 4) $25x^5 - \sqrt{x^2 + x}$		$\frac{(a-b)^3 - (a^3 + 3ab^2)}{-3a^2b - b^3}$ $a=100, b=0.001$
7	1) $m+--n$ 2) $m++<--n$ 3) $--m>n---$ 4) $\sqrt[5]{x^3 + x^4} + ctg(arctg(x^2))$		$\frac{(a-b)^3 - (a^3)}{-b^3 + 3ab^2 - 3a^2b}$ $a=100, b=0.001$ a) $Y = \sqrt[5]{x^3 + x^4} + ctg(arctg(x^2))$
8	1) $n/m++$ 2) $m++<--n$ 3) $(m/n)++<n/m$ 4) $\sqrt{ x^3 - 1 } - 7\cos\sqrt[3]{x^4 + x}$		$\frac{(a+b)^3 - (a^3)}{b^3 + 3ab^2 + 3a^2b}$ $a=100, b=0.001$
9	1) $m++/n---$ 2) $++m<n--$ 3) $n-->m$ 4) $\sin x^3 + x^4 + \sqrt[5]{x^2 + x^3}$		$\frac{(a+b)^3 - (a^3 + 3ab^2)}{3a^2b + b^3}$ $a=100, b=0.001$

10	1) $m/-n++$ 2) $m/n < n$ — 3) $m+n++ > n+m$ 4) $x^5 \sqrt{ x-1 } + 25-x^5 $		$\frac{(a+b)^3 - (a^3 + 3a^2b)}{3ab^2 + b^3}$ $a=100, b=0.001$
11	1) $n+++m--$ 2) $n*m < n++$ 3) $n-->++m$ 4) $2^x x \cos(x) + 1$		$\frac{(a+b)^4 - (a^4 + 4a^3b)}{6a^2b^2 + 4ab^3 + b^4}$ $a=10, b=0.01$
12	1) $n+++m$ 2) $m--<n$ 3) $++m>n$ 4) $\sqrt{x + \sqrt[4]{ x }} + x $		$\frac{(a+b)^4 - (a^4)}{6a^2b^2 + 4ab^3 + b^4 + 4a^3b}$ $a=10, b=0.01$
13	1) $(n++/-m)++$ 2) $++m < n$ — 3) $--m > ++n$ 4) $\sqrt[3]{e^x + \operatorname{tg} x} + \frac{1}{x}$		$\frac{(a+b)^4 - (a^4 + 6a^2b^2 + 4ab^3)}{b^4 + 4a^3b}$ $a=10, b=0.01$
14	1) $n+++--m$ 2) $n--<m++$ 3) $--n > --m$ 4) $\sqrt[4]{ x+1 } + \frac{1}{x^2}$		$\frac{(a+b)^4 - (a^4 + 6a^2b^2 + b^4)}{4ab^3 + 4a^3b}$ $a=10, b=0.01$

15	1) $n++/--m$ 2) $n-->n/m++$ 3) $m<n++$ 4) $1+x\cos^2(x)+\sin^3(x)$		$\frac{(a-b)^4 - (a^4 - 4a^3b)}{6a^2b^2 - 4ab^3 + b^4}$ $a=10, b=0.01$
16	1) $m/--n++$ 2) $m/n<n--$ 3) $m+n++>n+m$ 4) $\sqrt{\sin(x)+ x^2+x }$		$\frac{(a-b)^4 - (a^4)}{6a^2b^2 - 4ab^3 + b^4 - 4a^3b}$ $a=10, b=0.01$
17	1) $n+++m--$ 2) $n*m<n++$ 3) $n-->++m$ 4) $\arcsin(x+x^2)$		$\frac{(a-b)^4 - (a^4 + 6a^2b^2 - 4ab^3)}{b^4 - 4a^3b}$ $a=10, b=0.01$
18	1) $n+++*m$ 2) $m--<n$ 3) $++m>n$ 4) $\cos(\arctg(x))$		$\frac{(a-b)^4 - (a^4 + 6a^2b^2 + b^4) - 4ab^3 - 4a^3b}{b^2}$ $a=10, b=0.01$
19	1) $(n++/--m)++$ 2) $++m<n--$ 3) $--m>++n$ 4) $7\arctg(x^2)$		$\frac{(a+b)^2 - (a^2 + 2ab)}{b^2}$ $a=1000, b=0.0001$

20	1) $n++*--m$ 2) $n--<m++$ 3) $--n>--m$ 4) $5x^3\sqrt[5]{\frac{1}{x^2} + \frac{1}{x^3}}$		$\frac{(a-b)^2 - (a^2 - 2ab)}{b^2}$ $a=1000, b=0.0001$
21	1) $n++/--m$ 2) $n-->n/m++$ 3) $m<n++$ 4) $\sqrt[3]{e^x - \sin x}$		$\frac{(a+b)^3 - (a^3 + 3a^2b)}{3ab^2 + b^2}$ $a=100, b=0.001$
22	1) $n++*m$ 2) $n++<m$ 3) $--m>n$ 4) $2^{-x}\sqrt{x + \sqrt[4]{ x }}$		$\frac{(a-b)^3 - (a^3)}{3ab^2 - b^3 - 3a^2b}$ $a=100, b=0.001$
23	1) $--m-n++$ 2) $m*m<n++$ 3) $n-->++m$ 4) $1 + \frac{1}{x} + \frac{1}{x^2}$		$\frac{(a-b)^3 - (a^3 - 3a^2b)}{3ab^2 - b^3}$ $a=100, b=0.001$

24	1) m-++n 2) m++>--n 3) m--<++n 4) $\arcsin(x+1)$		$\frac{(a-b)^3 - (a^3 + 3ab^2) - 3a^2b - b^3}{a=100, b=0.001}$
25	1) m+--n 2) m++<--n 3) --m>n— 4) $\arccos(x+x^2)$		$\frac{(a-b)^3 - (a^3) - b^3 + 3ab^2 - 3a^2b}{a=100, b=0.001}$

5. Методические указания

- Ввод данных для заданий А и Б организовать с клавиатуры.
- Вывод результатов для задания А организовать в виде:

```
n?1
m?2

m=3 n=1 m++n=3
Press any key to continue
```

- Для проверки возможности вычислений использовать условный оператор if:

```
if(n==1) Console.WriteLine("Нельзя вычислить");
else {
    k=m/--n;
    Console.WriteLine("m++ +n={0}, m={1},n={2}", k, m, n);
}
```
- При выполнении задачи 2 использовать переменную логического типа, а не условный оператор.
- При выполнении задачи 3 использовать вспомогательные переменные для хранения промежуточных значений.
- При работе с данными типа float использовать операцию приведения типа:

```
float c = (float)Math.Pow(a + b, 2);
```

6. Содержание отчета

- Постановка задачи (общая и конкретного варианта).
- Формулы, используемые при решении задачи (математическая модель).
- Программы для решения задач на языке C#.
- Описание используемых в программе методов класса Math.
- Система тестов для проверки правильности работы программы и результаты выполнения тестов.
- Объяснение результатов работы программы.