**Лабораторная работа № 0.**

**Знакомство с интегрированной средой разработки (IDE)  
Visual Studio.NET.   
Разработка консольных приложений.**

***Цель работы –*** знакомство с базовыми особенностями языка программирования C# и разработки простых программ с использованием среды Visual Studio.NET.

***Ключевые понятия:*** среда разработки (IDE), проект, решение, пространство имён, библиотека классов, типы данных, классы: Console, Convert, Math.

**Краткие сведения о платформе .NET**

Платформа .NET Framework – это технология, которая поддерживает создание и выполнение нового поколения приложений, состоит из *общеязыковой среды выполнения (CLR – Common Language Runtime) и библиотеки классов .NET Framework (Framework Class Library, FCL)*,которая содержит более 10000 различных типов: классов, структур, интерфейсов, перечислений и делегатов. Бесспорным достоинством FCL является то, что она полностью объектно-ориентированная и используется всеми языками, которые работают с платформой .NET.

Основой платформы .NET Framework является CLR. Среду выполнения можно считать агентом, который управляет кодом во время выполнения и предоставляет основные службы, такие как: управление памятью, управление потоками и удаленное взаимодействие. При этом накладываются условия строгой типизации и другие виды проверки точности кода, обеспечивающие безопасность и надежность. *Фактически основной задачей среды выполнения является управление кодом (в том числе CLR автоматически запускает вашу программу).* Код, который обращается к среде выполнения, называют управляемым кодом, а код, который не обращается к среде выполнения, называют неуправляемым кодом.

Библиотека классов является комплексной объектно-ориентированной коллекцией допускающих повторное использование типов, которые применяются для разработки приложений. Физически FCL представляет собой набор DLL файлов (файлов в формате динамических библиотек). Каждый файл DLL – это сборка, загружаемая CLR по запросу. Встроенные типы данных, такие как целые, вещественные, логические, реализованы в модуле Mscorlib.dll, другие типы разнесены по разным DLL файлам библиотеки FCL.

Для облегчения использования FCL, все её содержание хорошо структурировано в виде иерархически организованных групп типов. Каждая группа типов называется *пространством имен*. Всего в FCL более 80 таких пространств, в каждом из которых находятся тысячи классов, поскольку пространства имен используются для того, чтобы избежать путаницы с типами и методами, имеющими одинаковые имена. В каждом из них содержатся классы и другие типы, имеющие некоторое общее назначение.

*Пространства имен* можно сравнить с папкой в компьютерной файловой системе. Подобно папкам, пространства имен определяют для классов уникальные полные имена. Программа C# содержит одно или несколько пространств имен, каждое из которых либо определено программистом, либо определено как часть написанной ранее библиотеки классов.

Например, пространство имен [System](https://msdn.microsoft.com/ru-ru/library/system(v=vs.90).aspx) содержит класс [Console](https://msdn.microsoft.com/ru-ru/library/system.console(v=vs.90).aspx), который включает методы для чтения и записи в окне консоли. В пространство имен [System](https://msdn.microsoft.com/ru-ru/library/system(v=vs.90).aspx) входит ряд других пространств, например, [System.IO](https://msdn.microsoft.com/ru-ru/library/system.io(v=vs.90).aspx) и [System.Collections](https://msdn.microsoft.com/ru-ru/library/system.collections(v=vs.90).aspx).

**Структура программы на языке C#.**

Код всех программ на С# структурирован одинаково. Везде для простоты управления кодом применяются пространства имён, классы и методы (см. рис. 1). Каждое приложение в одном из классов должно содержать метод **Main** – с него начинается выполнение программы (*точка входа приложения)*.

Маленькая программа может состоять из одного класса и одного метода Main.

Рисунок 1. Пример структуры программы

При создании программ и комплексов программ в интегрированной системе разработки Microsoft Visual Studio.NET используются понятия решение (solution) и проект (project).

**Проект** представляет собой набор файлов исходных текстов, файлов графических изображений и других файлов, необходимых для создания программы.

Обычно сложные программные комплексы содержат в себе много программ и компонентов, причем, возможно, созданных с использованием разных языков программирования и даже рассчитанных для использования на различных компьютерных платформах. Для облегчения разработки Visual Studio.NET позволяет представить все создаваемые компоненты такого комплекса в виде набора проектов, объединенных общим **решением**.

**Первая программа на C#**

1. Запустите среду разработки Visual Studio.NET (из главного меню Windows или с рабочего стола).
2. После запуска откроется стартовая страница (Start Page). Если эта страница будет случайно закрыта или её не окажется на экране, то для повторного вызова стартовой страницы нужно выбрать команду Help – Show Start Page.
3. Для создания нового проекта выберите команду Создать проект … (New Project…) (см. рис. 2).

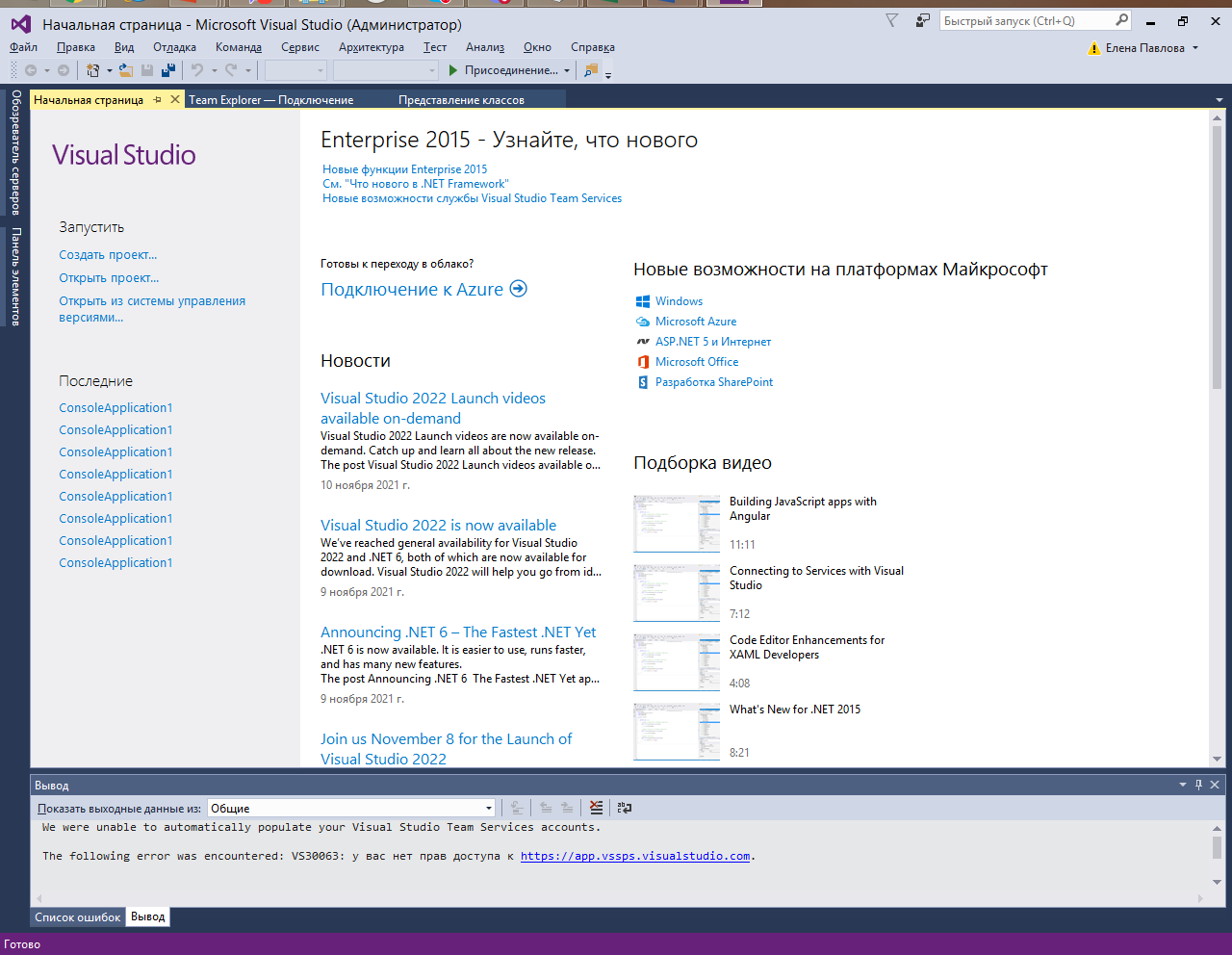


Рисунок 2. Фрагмент стартовой страницы

1. В результате появится диалоговое окно (см. рис. 3), в котором нужно указать:
   1. язык программирования (выбираем C#);
   2. тип приложения (выбираем консольное);
   3. в поле **Имя** наберите имя проекта (можно оставить то, которое задано по умолчанию);
   4. в поле **Расположение** (с помощью кнопки **Обзор** **…**) укажите путь, где будет храниться проект.

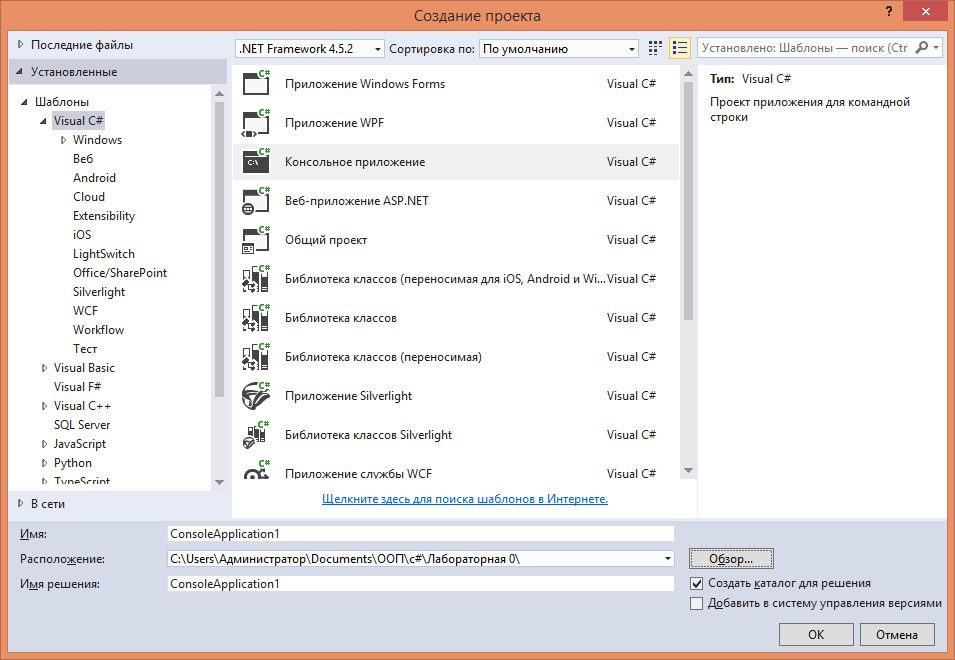


Рисунок 3. Окно диалога **Создать проект**…

1. Нажмите кнопу **ОК**, среда создаст решение и проект с указанным именем. Примерный вид окна приведён на рисунке 4.

Директива **using** **System** разрешает использовать имена стандартных классов непосредственно (без указания имени пространства).

Ключевое слово **namespace** создаёт для проекта собственное пространство имён, названное по умолчанию ConsoleApplication1. Это сделано для того, чтобы можно было давать программным объектам имена, не заботясь о совпадении с именами в других пространствах имён.

В нашей программе всего один класс с именем по умолчанию **Program**. В нём уже создан метод **Main**.

**static** – ключевое слово, обозначает, что при использовании метода не нужно создавать экземпляр класса.

**void** – ключевое слово, обозначает, что метод не возвращает значение (аналогия с процедурой из языка Pascal).

После имени метод всегда следуют круглые скобки, в которых могут указываться параметры метода.

***Важно****. Название класса мы задаём сами, а название пространства имен должно совпадать с именем проекта, и, соответственно, с именем исполняемой программы.*

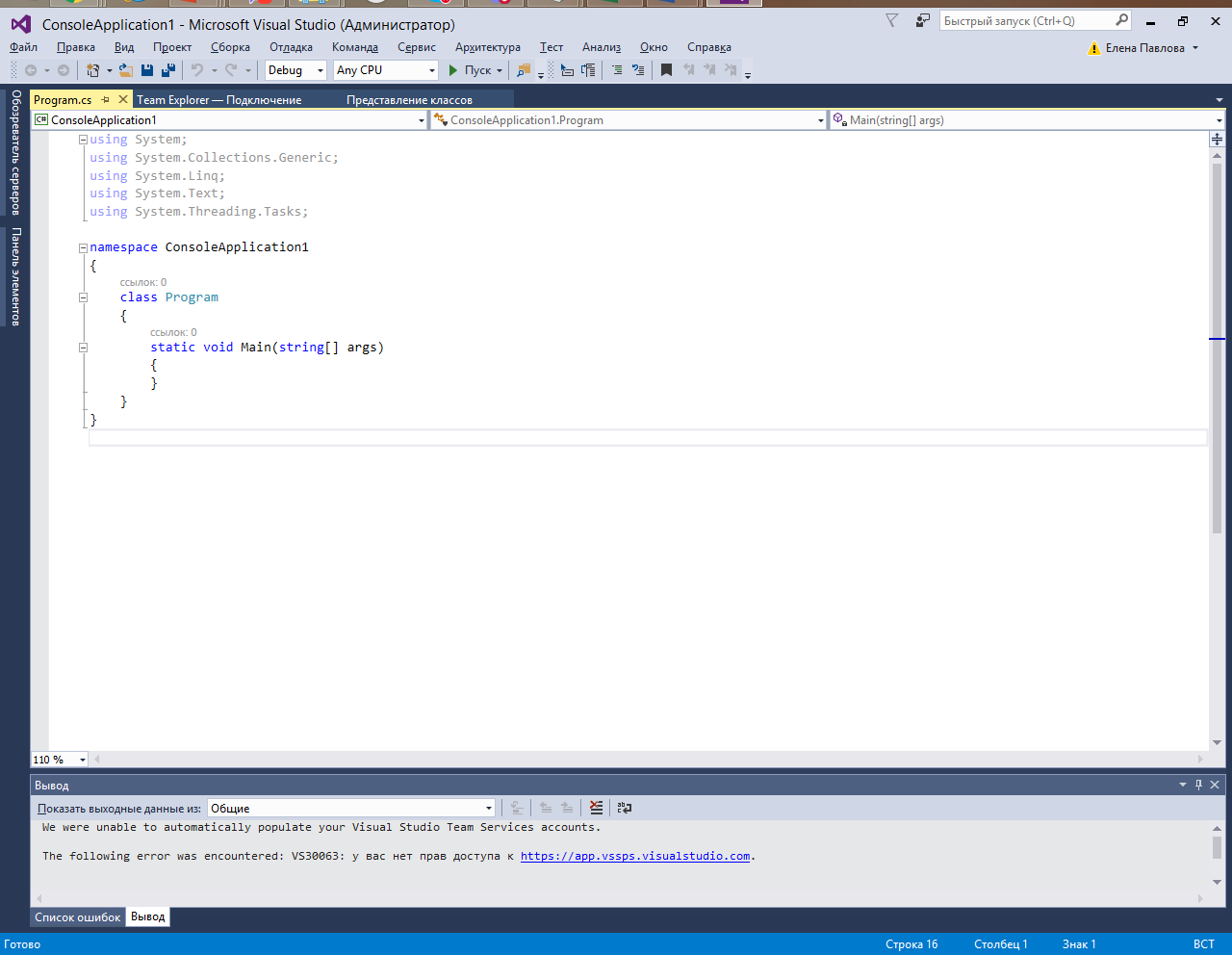


Рисунок 4. Примерный вид окна проекта консольного приложения

Фигурные скобки являются важным элементом синтаксиса. Каждой открывающей скобке должна соответствовать своя закрывающая, которая располагается ниже по тексту с тем же отступом. Фигурные скобки ограничивают блок, внутри которого могут располагаться вложенные блоки.

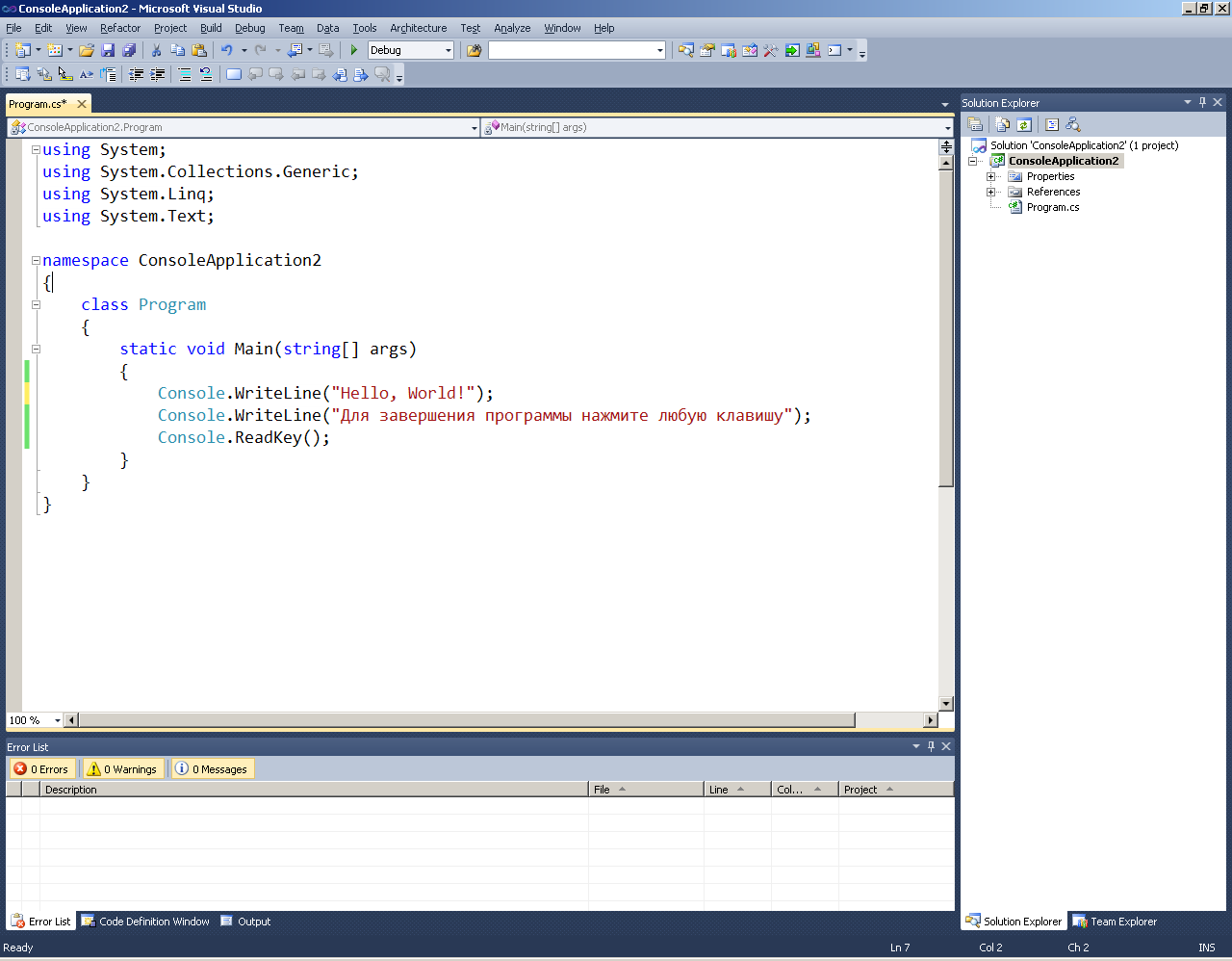
1. Наберите текст программы (см. рис. 5). Для перемещения и копирования фрагментов текста используйте команды Edit – Cut, Edit –Copy, Еdit – Paste.

Рисунок 5. Код простой программы

**Console** – это имя стандартного класса из пространства имён System. Его метод **Writeln** выводит на экран текст, который помещён между кавычками. Для обращения к методу класса используется конструкция:

**имя\_класса.имя\_метода**

Если слово Console (первое слово) было написано правильно, то после ввода точки среда выдаст подсказку, содержащую список всех доступных элементов класса **Console**. Эта функция среды называется *IntelliSense*. Она предлагает доступные способы завершения вводимой строки.

Метод **ReadKey** ожидаетнажатия любой клавиши клавиатуры, после того, как клавиша будет нажата, программа завершит работу, так как все операторы выполнены.

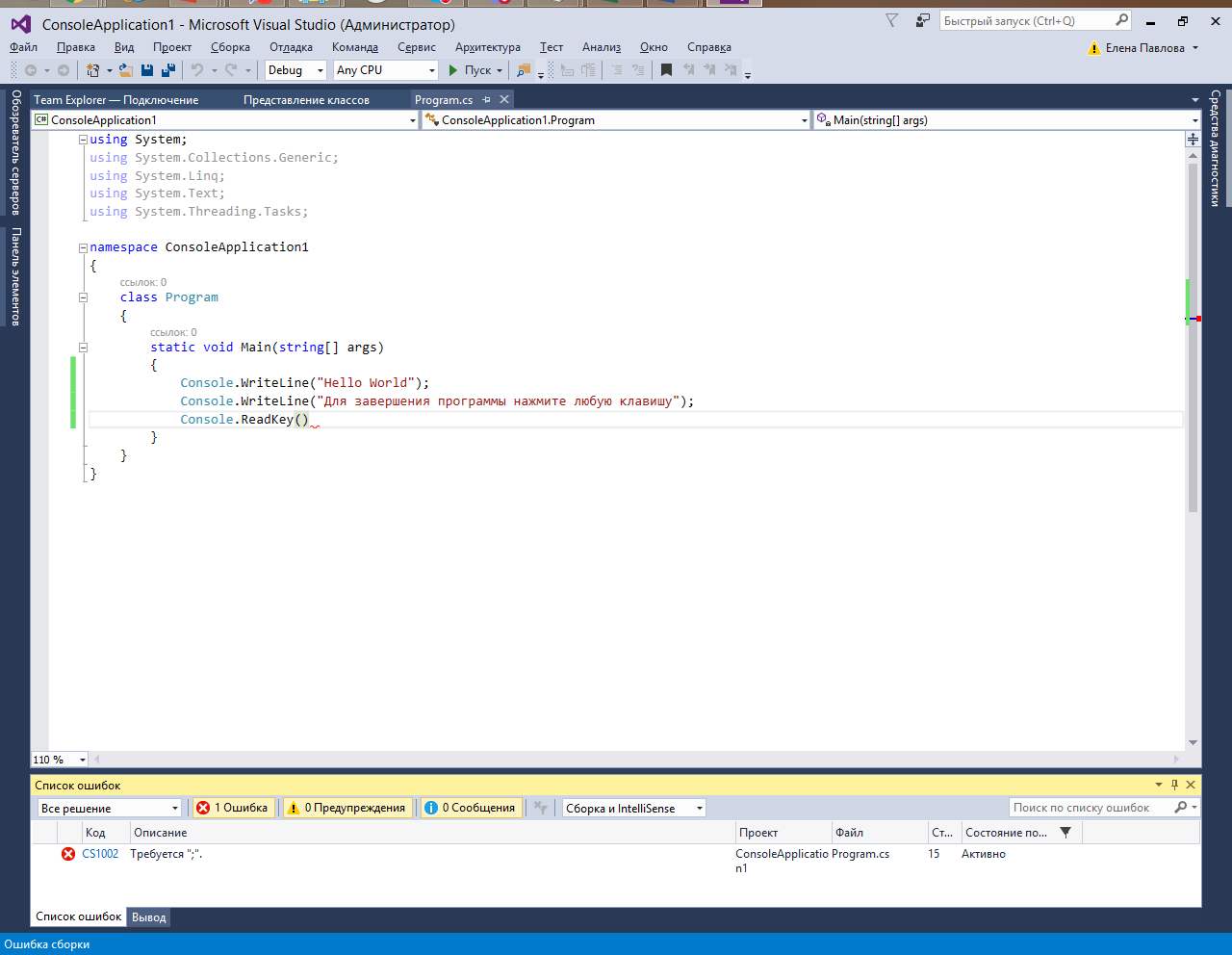








Рисунок 6. Допущена ошибка

**Важно!** Каждый оператор должен заканчиваться символом «;» (точка с запятой), в противном случае при компиляции программы появится ошибка (см. рис. 6), информация о которой появится в специальной области – **ErrorList**.

1. Для запуска программы на выполнение используйте команды **Debug – Start Debug** (запуск с отладкой) или **Debug – Start Without Debugging** (запуск без отладки).
2. Окно работающей программы показано на рисунке 7.

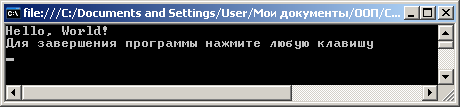


Рисунок 7. Окно работающей программы

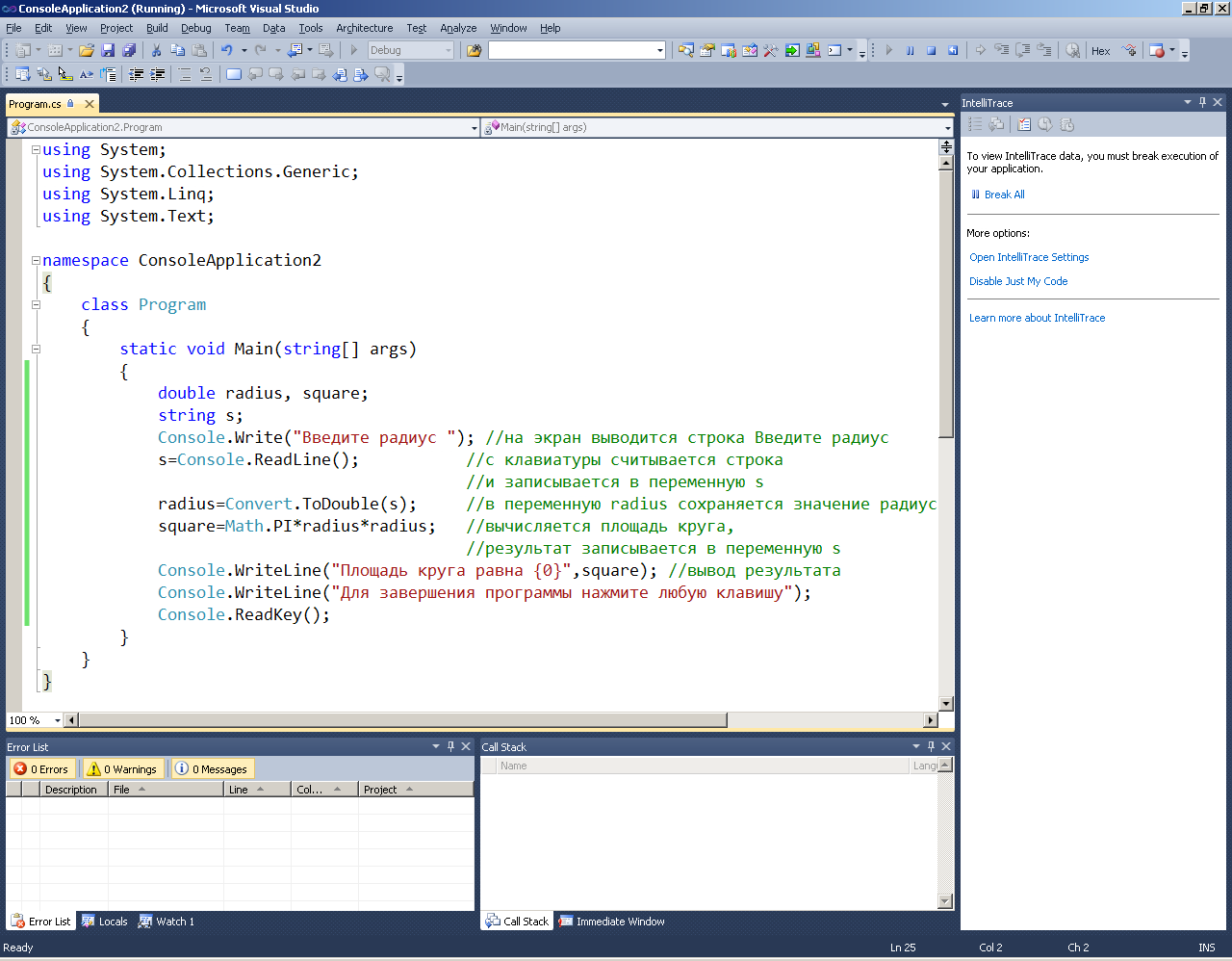
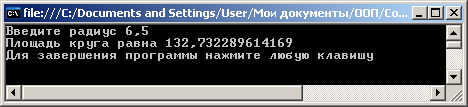
Напишем ещё одну программу – она будет вычислять и выводить на экран площадь круга заданного радиуса. Удалим код предыдущий программы и напишем новый (см. рис. 8) 

Рисунок 8. Код программы, вычисляющий площадь круга.



В программе использованы переменные radius, square вещественного типа (**double**) и строковая (**string**) переменная s, в которую считывается введённая строка, содержащая значение радиуса. Дело в том, что любой поток входных данных в C# воспринимается как строка. Если во входном потоке задано число, то строку нужно преобразовать к числовому типу данных. Преобразованиями из строки практически к любому типу занимается класс **Convert.** В данном случае использован его метод **ToDouble**, который преобразует строку в вещественной число.

Класс **Math** содержит два статических поля E (число е) и PI (число ПИ), а также 23 статических метода. Методы задают:

* модуль, корень, знак – Abs (), Sqrt (), Sign ();
* функции округления – Ceiling (), Floor (), Round ();
* минимум, максимум, степень – Min (), Max (), Pow ();
* тригонометрические функции – Sin (), Cos (), Tan ();
* экспоненту и логарифмические функции – Exp (), Log (), Log10 ().

**Синтаксис языка C#**

В C# используется кодировка символов Unicode. ***Кодировкой***, или***кодовой таблицей*** (character set), называется соответствие между символами и кодирующими их числами. Кодировка Unicode позволяет представить символы всех существующих алфавитов одновременно. Каждому символу соответствует свой уникальный код.

Алфавит C# включает:

* ***буквы*** (латинские и национальных алфавитов) и ***символ подчеркивания***  (*\_*), который употребляется наряду с буквами;
* ***цифры***;
* ***специальные символы***, например *+*, *\**, { и &;
* ***пробельные символы*** (пробел и символы табуляции);
* ***символы перевода строки***.

**Важно!** В С# различаются заглавные и прописные буквы.

Из символов составляются более крупные строительные блоки: лексемы, директивы препроцессора и комментарии.

***Лексема*** (token) — это минимальная единица языка, имеющая самостоятельный смысл. Существуют следующие виды лексем:

* ***имена*** (***идентификаторы***);
* ***ключевые слова***;
* ***знаки операций***;
* ***разделители***;
* ***литералы***(***константы***).

Лексемы языка программирования аналогичны словам естественного языка. Например, лексемами являются число 128 (но не его часть 12), имя *Vasia*, ключевое слово *goto* и знак операции сложения *+*. Далее мы рассмотрим лексемы подробнее.

***Директивы препроцессора***. Препроцессором называется предварительная стадия компиляции, на которой формируется окончательный вид исходного текста программы. Например, с помощью директив (инструкций, команд) препроцессора можно включить или выключить из процесса компиляции фрагменты кода. Директивы препроцессора не играют в C# такой важной роли, как в С++.

***Комментарии*** предназначены для записи пояснений к программе и формирования документации, текст комментария игнорируется компилятором. Внутри комментария можно использовать любые символы.

В C# есть два вида комментариев:

* **Однострочные**. Начинаются с двух символов прямой косой черты (*//*) и заканчиваются символом перехода на новую строку.
* **Многострочные**. Заключается между символами-скобками */\** и *\*/* и могут занимать часть строки, целую строку или несколько строк. Комментарии не вкладываются друг в друга.

Из лексем составляются выражения и операторы. ***Выражение*** задает правило вычисления некоторого значения. Например, выражение *a + b л*задает правило вычисления суммы двух величин.

***Оператор*** задает законченное описание некоторого действия, данных или элемента программы. Например:

int a; Это оператор описания целочисленной переменной *a*.

**Идентификаторы**

***Имена***, или ***идентификаторы***, служат для того чтобы обращаться к программным объектам и различать их, то есть идентифицировать. В идентификаторе могут использоваться буквы, цифры и символ подчеркивания. Прописные и строчные буквы различаются, например, *sysop*, *SySoP* и *SYSOP* — три разных имени.

Первым символом идентификатора может быть буква или знак подчеркивания, но не цифра. Длина идентификатора не ограничена. Пробелы внутри имен не допускаются.

В идентификаторах C# разрешается использовать, помимо латинских букв, буквы национальных алфавитов. Идентификатор создается на этапе объявления переменной (метода, типа и т.п.), после этого его можно использовать в последующих операторах программы. При выборе идентификатора необходимо следить, чтобы он не совпадал с ключевыми словами.

**Ключевые слова**

***Ключевые слова*** — это зарезервированные идентификаторы, которые имеют специальное значение для компилятора. Их можно использовать только в том смысле, в котором они определены. Ключевые слова обычно выделяются в коде программы синим цветом.

#### Знаки операций и разделители

***Знак операции*** — это один или более символов, определяющих действие над операндами. Внутри знака операции пробелы не допускаются. Например, в выражении *a += b* знак *+=* является знаком операции, *a* и *b* — операндами.

Операции делятся на ***унарные***, ***бинарные*** и ***тернарную*** по количеству участвующих в них операндов. Один и тот же знак может интерпретироваться по-разному в зависимости от контекста.

***Разделители*** используются для разделения или, наоборот, группирования элементов. Примеры разделителей: скобки, точка, запятая.

**Литералы**

***Литералами***, или ***константами***, называют неизменяемые величины. В C# есть логические, целые, вещественные, символьные и строковые константы, а также константа *null*. Компилятор, выделив константу в качестве лексемы, относит ее к одному из типов данных по ее внешнему виду. Программист может задать тип константы самостоятельно.

***Логических*** литералов два. Они широко используются в качестве признаков наличия или отсутствия чего-либо.

***Целые литералы*** могут быть представлены либо в десятичной, либо в шестнадцатеричной системе счисления, а ***вещественные*** — только в десятичной системе, но в двух формах: с фиксированной точкой и с порядком. Вещественная константа с порядком представляется в виде ***мантиссы*** и ***порядка***. Мантисса записывается слева от знака экспоненты (*E* или *e*), порядок — справа от знака. Значение константы определяется как произведение мантиссы и возведенного в указанную в порядке степень числа 10 (например, 1.3e2 = 1,3\*10*2* = 130). Пробелы внутри константы не допускаются.

Если требуется сформировать ***отрицательную*** целую или вещественную константу, то перед ней ставится знак унарной операции изменения знака (*–*), например, –218.

***Символьная константа*** представляет собой любой символ в кодировке Unicode. Например, 'A','ю', '\*'.

***Управляющей последовательностью***, или ***простой escape-последовательностью***, называют определенный символ, предваряемый обратной косой чертой. Управляющая последовательность интерпретируется как одиночный символ и используется для представления:

* кодов, не имеющих графического изображения (например,\*n* — переход в начало следующей строки);
* символов, имеющих специальное значение в строковых и символьных литералах, например, апострофа ( \*'* ).

**Строковые литералы**

В С# поддерживается еще один тип литералов — строковый. Строковый литерал представляет собой набор символов, заключенных в двойные кавычки. Помимо обычных символов, строковый литерал может содержать одну или несколько управляющих последовательностей символов.

**Типы данных**

Данные, с которыми работает программа, хранятся в оперативной памяти. Компилятору необходимо точно знать, сколько места они занимают, как именно закодированы и какие действия с ними можно выполнять. Все это задается при описании данных с помощью типа.

**Тип данных** однозначно определяет:

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

Например, целые и вещественные числа, даже если они занимают одинаковый объем памяти, имеют совершенно разные диапазоны возможных значений.

Каждое выражение в программе имеет определенный тип. Компилятор использует информацию о типе при проверке допустимости описанных в программе действий.

Память, в которой хранятся данные во время выполнения программы, делится на две области: стек (stack) и динамическая область, или куча (heap) (см. рис.).

Стек используется для хранения величин, память под которые выделяет компилятор, а в динамической области память резервируется и освобождается во время выполнения программы с помощью специальных команд. Основным местом для хранения данных в C# является куча.

Рисунок 9. Схематичная структура памяти

***Классификация типов***

Типы можно классифицировать по разным признакам. Если принять за основу строение элемента, все типы можно разделить на простые (не имеют внутренней структуры) и структурированные (состоят из элементов других типов). По своему «создателю» типы можно разделить на встроенные (стандартные) и определяемые программистом. Для данных статического типа память выделяется в момент объявления, при этом ее требуемый объем известен. Для данных динамического типа размер данных в момент объявления может быть неизвестен, и память под них выделяется по запросу в процессе выполнения программы (см. рис. 9).

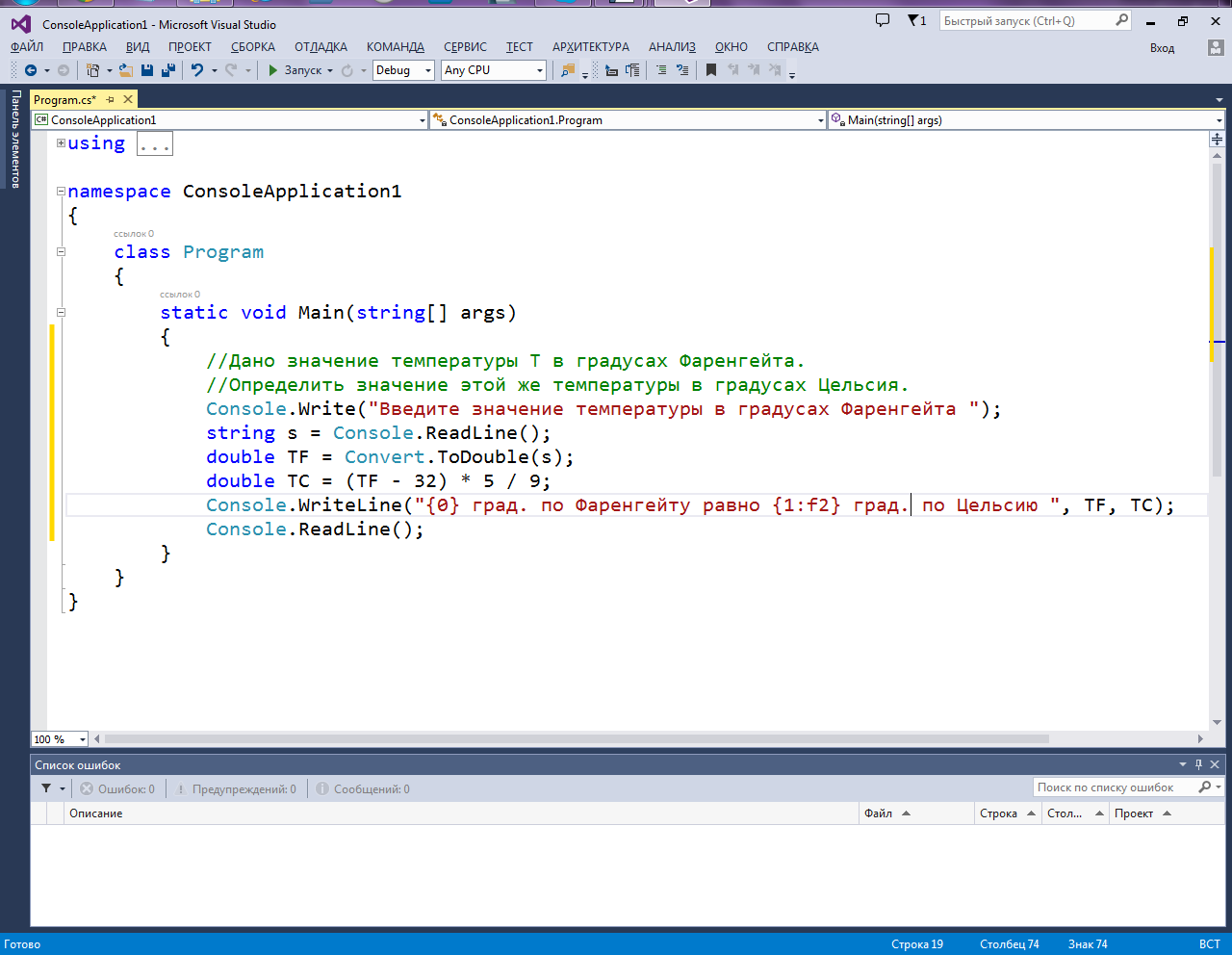
C# поддерживает следующие встроенные типы данных.

| **Тип данных** | | **Диапазон** | **Размер (байт)** |
| --- | --- | --- | --- |
| **Целые** | **byte** | 0 .. 255 | 1 |
| **sbyte** | -128 .. 127 | 1 |
| **short** | -32 768 .. 32 767 | 2 |
| **ushort** | 0 .. 65 535 | 2 |
| **int** | -2 147 483 648 .. 2 147 483 647 | 4 |
| **uint** | 0 .. 4 294 967 295 | 4 |
| **long** | -9 223 372 036 854 775 808 .. 9 223 372 036 854 775 807 | 8 |
| **ulong** | 0 .. 18 446 744 073 709 551 615 | 8 |
| **Вещественные** | **float** | -3,402823e38 .. -3,402823e38 .. | 4 |
| **double** | -1,79769313486232e308 .. 1,79769313486232e308 | 8 |
| **Финансовый** | **decimal** | -79228162514264337593543950335 .. 79228162514264337593543950335 | 32 |
| **Символьный** | **char** | Символ Юникода. | 2 |
| **Строковый** | **string** | Строка символов Юникода. |  |
| **Логический** | **bool** | true или false | 1 |
| **Тип объект** | **object** | Объект |  |

Если в объявлении переменной используется один из основных встроенных типов данных или определенная пользователем структура данных, значит мы имеем дело с типом значения. Исключение составляет тип данных **string и object**, которые являются ссылочными типами.

***Упражнения***

1. Определите виды лексем в программе, представленной на рисунке 10.

Рисунок 10. Код программы

1. Что будет выведено на экран в результате выполнения операторов:

Console.WriteLine(Math.Ceiling(value));

Console.WriteLine(Math.Floor(value));

Console.WriteLine(Math.Round(value));

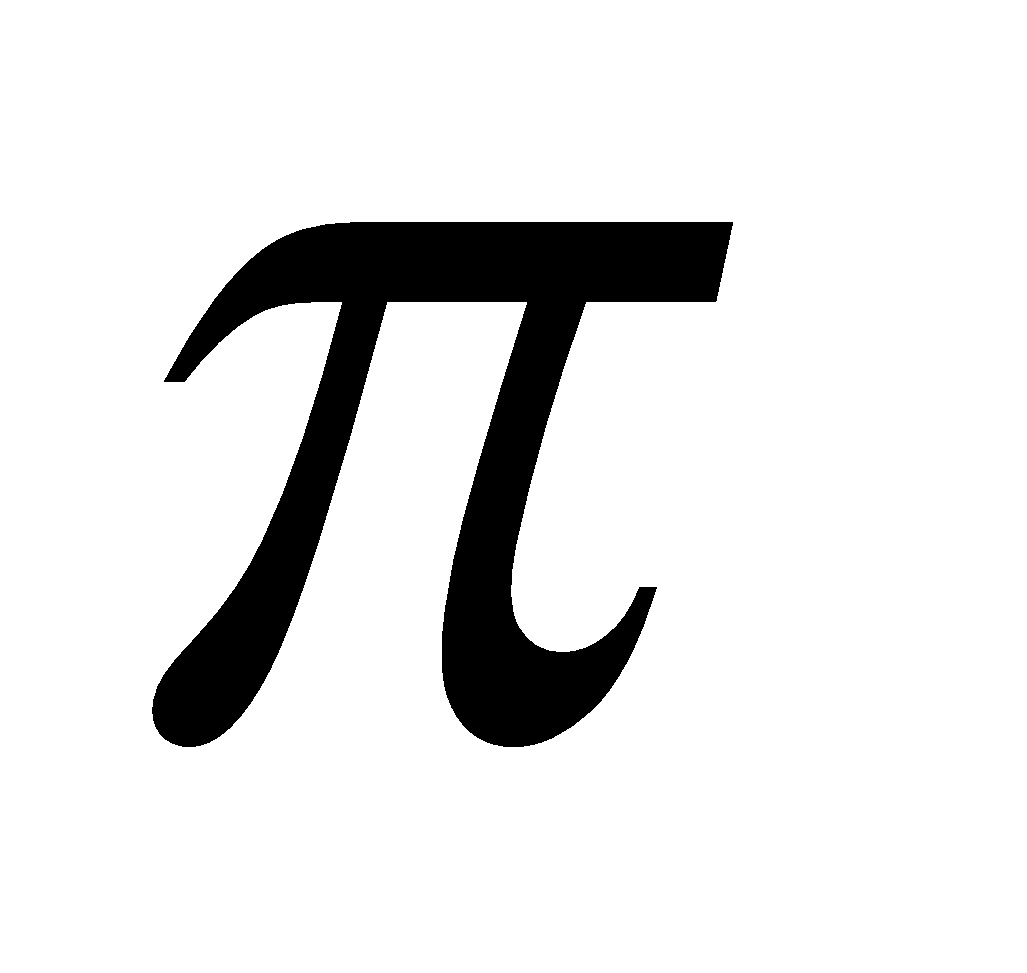
если value равно:

а). 7.03 б). 7.64 в). 0.12 г). -0.12 д). -7.1

е). -7.6 ж). 0.5 з). -0.5 и). -0.9 к). 0.1

1. Ниже приведены два столбца чисел. Подберите из правого столбца соответствующее представление на языке c# для чисел из правого столбца.

7,9985

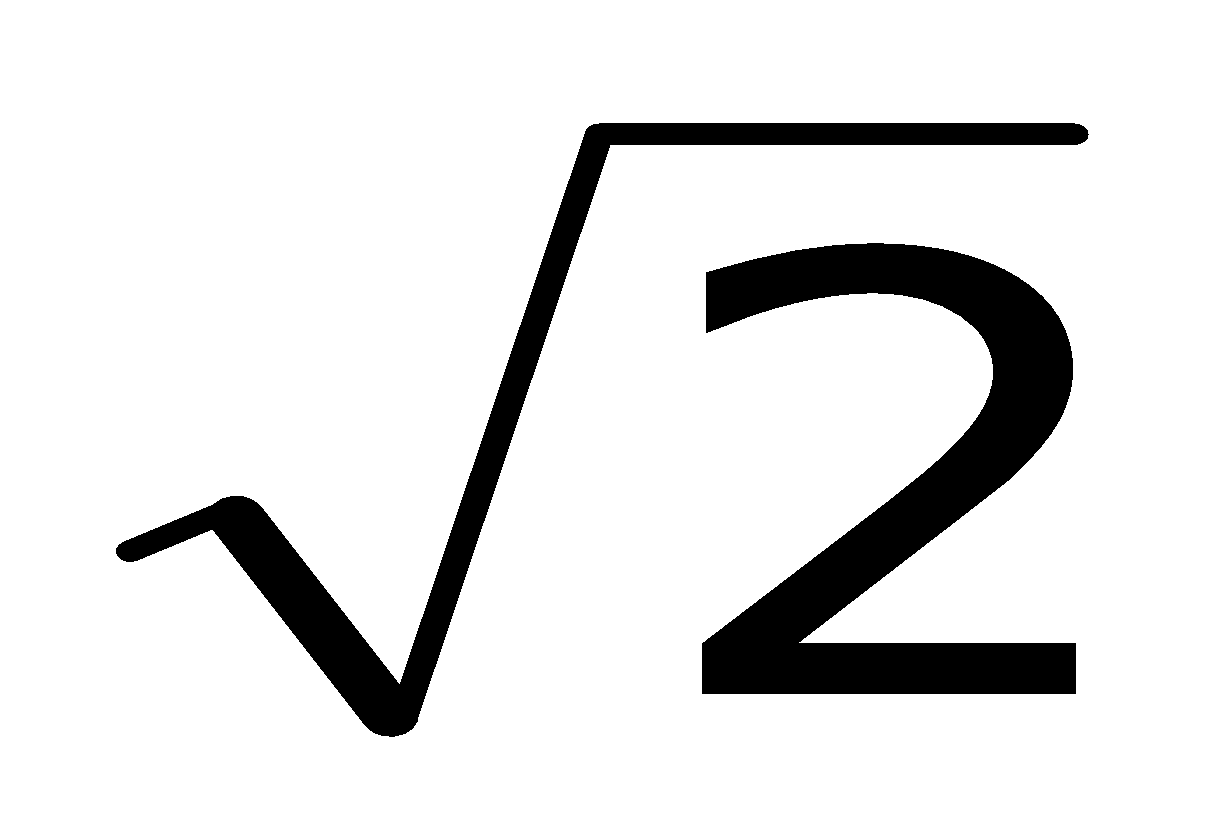


-1/3

106

-1/10000000

32782.82



6!

LXXIV

10

1E6

0.327828E5

1\*2\*3\*4\*5\*6

74

7.9985

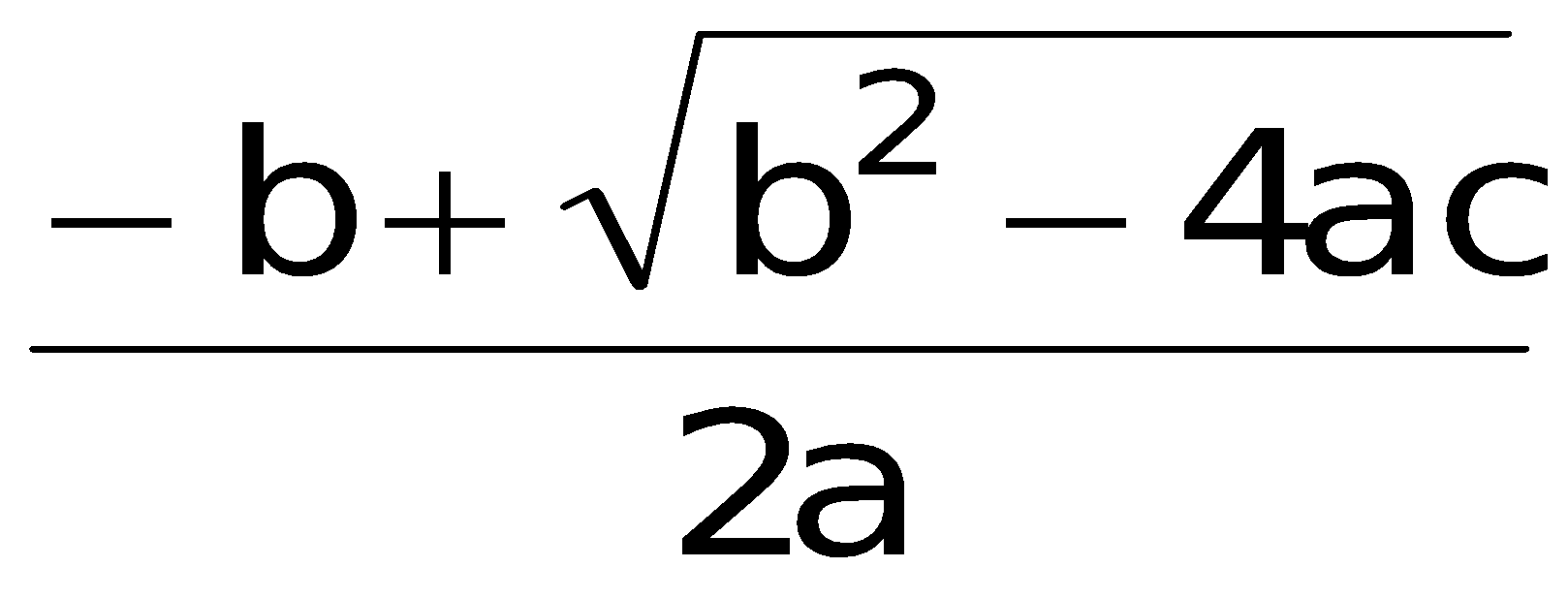
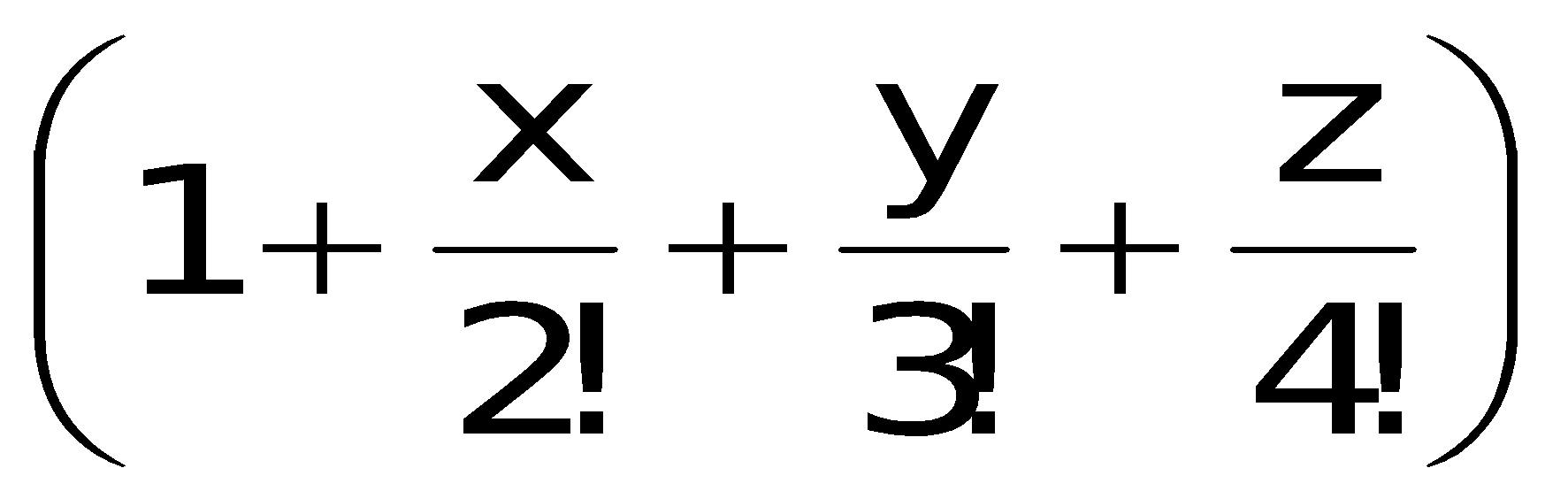
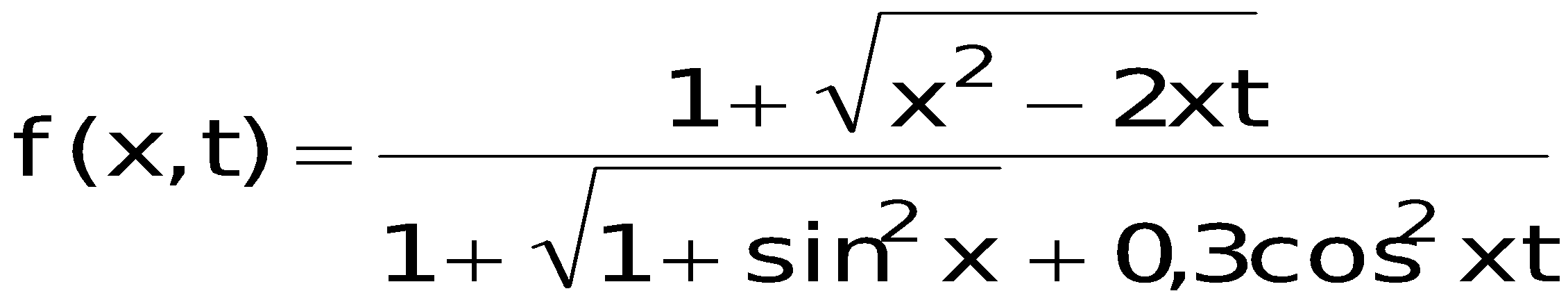
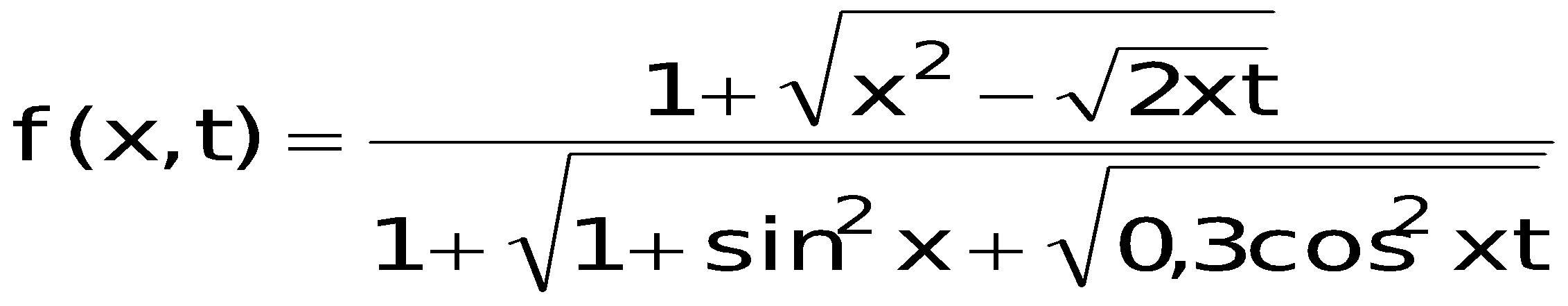
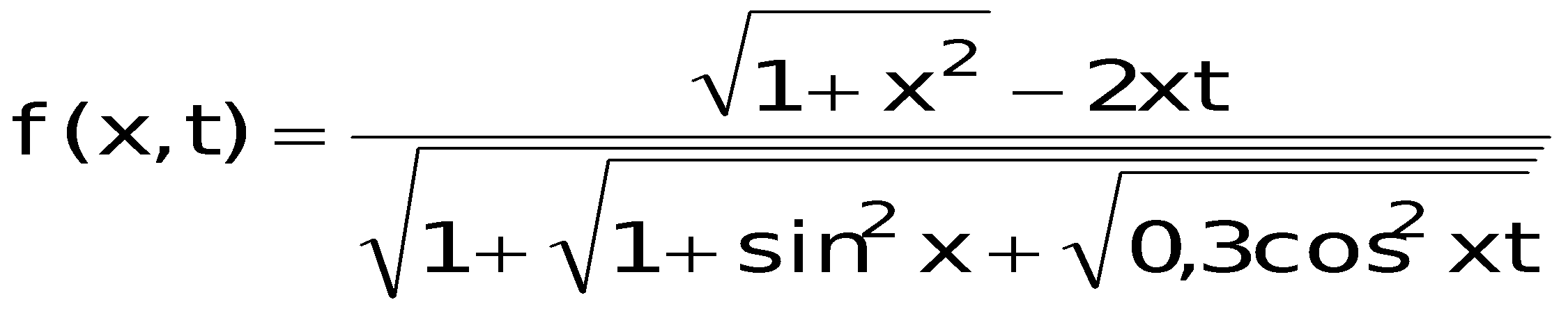
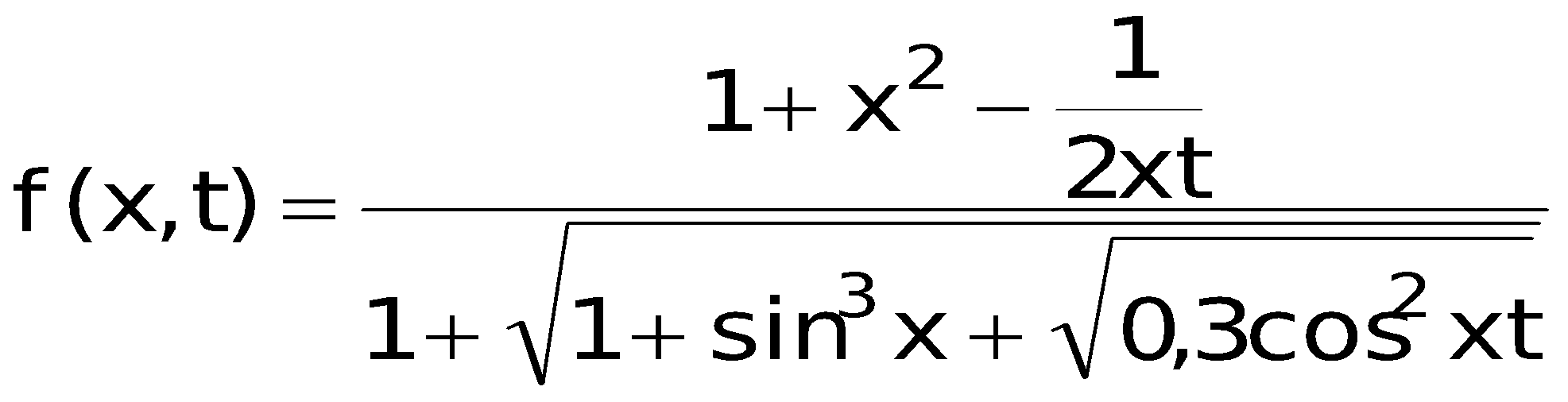
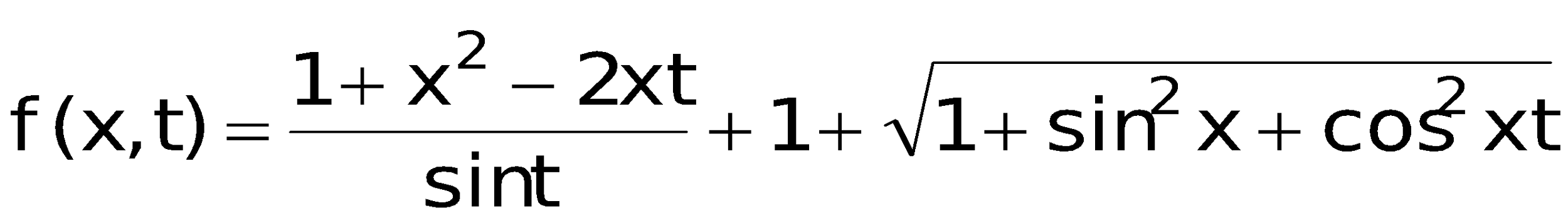
-0.3333

-1E-7

1.4142

1E1

3.1416

1. Запишите на языке c# следующие формулы:
   1. 
   2. {[(ax – b)x + c]x + d}x – e
   3. 
2. Напишите программу, которая вычисляет длину окружности по известному радиусу.
3. Напишите программу, которая по известным длинам сторон вычисляет площадь и периметр прямоугольника.
4. Найдите расстояние между двумя точками в двухмерном пространстве.
5. Перепишите программу из нахождения расстояния между двумя точками для трехмерного случая.
6. Город А находится в N милях от Лондона. Составьте программу вычисления расстояния между этими городами в километрах, если 5 сухопутных миль равны 8,045 км.
7. Посчитайте, сколько очков набрала команда «Динамо» в первом круге чемпионата России по хоккею, если известно, что m встреч она выиграла, n встреч проиграла, k встреч закончились ничьими, полагая, что за выигрыш команда получает 2 очка, за ничью – 1 очко, за проигрыш – 0 очков.
8. Напишите программу, которая выводит на экран последнюю цифру заданного числа n.
9. Напишите программу, которая целой переменной x присваивает сумму цифр заданного трёхзначного числа.
10. Напишите программу, которая меняет местами значения целых переменных m и n, не используя дополнительные переменные.
11. Пусть n – целое число от 1 до 365. присвойте целой переменной m значение 1, 2, …, 6 или 7 в зависимости от того, на какой день недели (понедельник, вторник, …, субботу или воскресенье) приходится n-ный день невисокосного года, в котором 1 января – среда.
12. Найти значение y = f(x,t), где 
13. Найти значение y = f(x,t), где 
14. Найти значение y = f(x,t), где 
15. Найти значение y = f(x,t), где 
16. Найти значение y = f(x,t), где

***Контрольные вопросы***

1. Что такое «пространство имён»?
2. Что такое «проект»?
3. Что такое «решение»?
4. Что такое FCL? Каково её предназначение?
5. Что такое CLR? Каковы её функции?
6. Зависит ли значение переменной от ее имени?
7. Зависит ли имя переменной от ее типа?
8. Какие формы записи вещественных чисел используются в языке c#?
9. Назовите функцию, которая вычисляет модуль вещественного числа х.
10. Опишите классификацию типов данных в языке с#.