ЛАБОРАТОРНАЯ РАБОТА № 2

Разработка и отладка линейных алгоритмов и программ

Цель работы: обучиться приемам работы с инструментальной средой программирования Microsoft Visual Studio. Сформировать умения кодировать арифметические и логические выражения с использованием стандартных библиотечных функций ввода/вывода в языке программирования C#.

Оборудование: персональный компьютер, практикум, тетради для лабораторных работ.

Правила по технике безопасности: общие (приложение).

Литература:

1. Г. Шилдт, Полный справочник по C#.
2. Т.А. Павловская, Программирование на языке высокого уровня.

Время выполнения: 2 часа.

ВОПРОСЫ ВХОДНОГО КОНТРОЛЯ:

1. Перечислите основные типы данных.
2. Опишите структуру программы.
3. Перечислите основные операции в C#.

Краткие теоретические сведения

Язык С отражает возможности современных компьютеров.

Программы на С отличаются компактностью и быстротой исполнения. Структура языка С побуждает программиста использовать в своей работе нисходящее программирование, структурное программирование, пошаговую разработку модулей.

Данные различных типов хранятся и обрабатываются по-разному. Тип данных определяет:

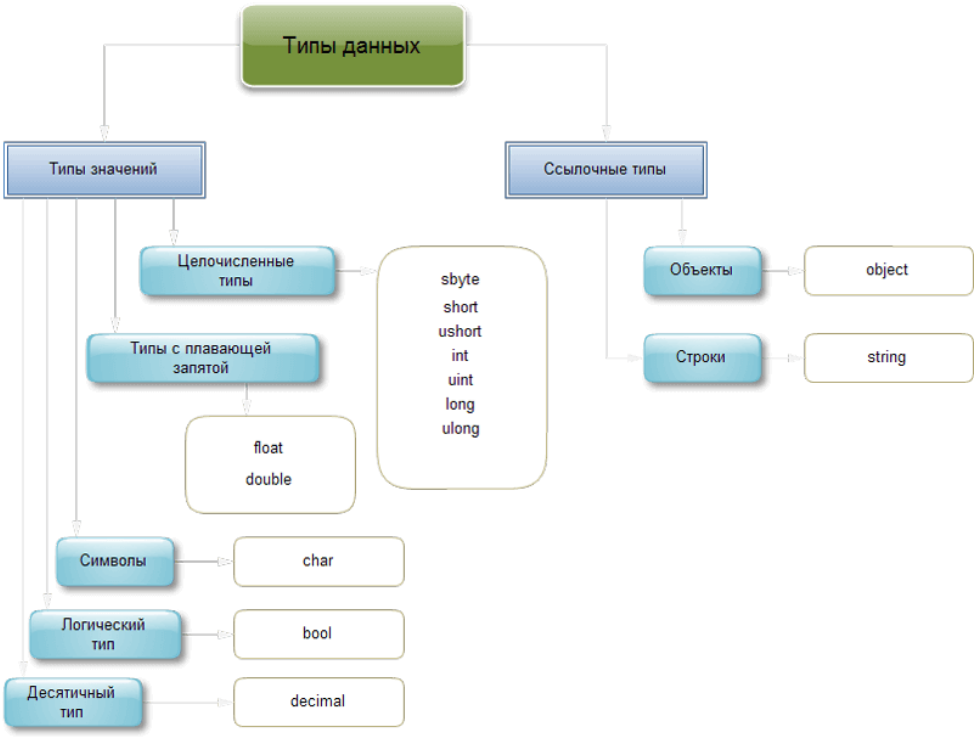
1) внутреннее представление данных в памяти компьютера;
2) множество значений, которые могут принимать величины данного типа;
3) операции и функции, которые можно применять к величинам данного типа.

Все типы данных языка С++ можно разделить на составные и основные. Определено 7 основных типов данных:

Типы данных имеют особенное значение в C#, поскольку это строго типизированный язык. Это означает, что все операции подвергаются строгому контролю со стороны компилятора на соответствие типов, причем недопустимые операции не компилируются. Следовательно, строгий контроль типов позволяет исключить ошибки и повысить надежность программ. Для обеспечения контроля типов все переменные, выражения и значения должны принадлежать к определенному типу. Такого понятия, как "бестиповая" переменная, в данном языке программирования вообще не существует. Более того, тип значения определяет те операции, которые разрешается выполнять над ним. Операция, разрешенная для одного типа данных, может оказаться недопустимой для другого.

В C# имеются две общие категории встроенных типов данных: типы значений и ссылочные типы. Они отличаются по содержимому переменной. Концептуально разница между ними состоит в том, что тип значения (value type) хранит данные непосредственно, в то время как ссылочный тип (reference type) хранит ссылку на значение.

Эти типы сохраняются в разных местах памяти: типы значений сохраняются в области, известной как стек, а ссылочные типы — в области, называемой управляемой кучей.



Целочисленные типы

В C# определены девять целочисленных типов: char, byte, sbyte, short, ushort, int, uint, long и ulong. Но тип char применяется, главным образом, для представления символов и поэтому рассматривается отдельно. Остальные восемь целочисленных типов предназначены для числовых расчетов. Ниже представлены их диапазон представления чисел и разрядность в битах:

Целочисленные типы C#

Тип	Тип CTS	Разрядность bit	Диапазон
byte	System.Byte	8	0:255
sbyte	System.SByte	8	-128:127
short	System.Int16	16	-32768 : 32767
ushort	System.UInt16	16	0 : 65535
int	System.Int32	32	-2147483648 : 2147483647
uint	System.UInt32	32	0 : 4294967295
long	System.Int64	64	-9223372036854775808 : 9223372036854775807
ulong	System.UInt64	64	0 : 18446744073709551615

Как следует из приведенной выше таблицы, в C# определены оба варианта различных целочисленных типов: со знаком и без знака. Целочисленные типы со знаком отличаются от аналогичных типов без знака способом интерпретации старшего разряда целого числа. Так, если в программе указано целочисленное значение со знаком, то компилятор C# сгенерирует код, в котором старший разряд целого числа используется в качестве флага знака. Число считается положительным, если флаг знака равен 0, и отрицательным, если он равен 1.

Отрицательные числа практически всегда представляются методом дополнения до двух, в соответствии с которым все двоичные разряды отрицательного числа сначала инвертируются, а затем к этому числу добавляется 1.

Вероятно, самым распространенным в программировании целочисленным типом является тип int. Переменные типа int нередко используются для управления циклами,

индексирования массивов и математических расчетов общего назначения. Когда же требуется целочисленное значение с большим диапазоном представления чисел, чем у типа int, то для этой цели имеется целый ряд других целочисленных типов.

Так, если значение нужно сохранить без знака, то для него можно выбрать тип uint, для больших значений со знаком — тип long, а для больших значений без знака — тип ulong. В качестве примера ниже приведена программа, вычисляющая расстояние от Земли до Солнца в сантиметрах. Для хранения столь большого значения в ней используется переменная типа long:

Пример программы:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            long result;
            const long km = 149800000; // расстояние в км.
            result = km * 1000 * 100;
            Console.WriteLine(result);
            Console.ReadLine();
        }
    }
}
```

Всем целочисленным переменным значения могут присваиваться в десятичной или шестнадцатеричной системе обозначений. В последнем случае требуется префикс 0x:

```
long x = 0x12ab;
```

Если возникает какая-то неопределенность относительно того, имеет ли целое значение тип int, uint, long или ulong, то по умолчанию принимается int. Чтобы явно специфицировать, какой другой целочисленный тип должно иметь значение, к числу можно добавлять следующие символы:

```
uint ui = 1234U;
long l = 1234L;
ulong ul = 1234UL;
```

U и L можно также указывать в нижнем регистре, хотя строчную L легко зрительно спутать с цифрой 1 (единица).

Типы с плавающей точкой

Типы с плавающей точкой позволяют представлять числа с дробной частью. В C# имеются две разновидности типов данных с плавающей точкой: float и double. Они представляют числовые значения с одинарной и двойной точностью соответственно. Так, разрядность типа float составляет 32 бита, что приближенно соответствует диапазону представления чисел от 5E-45 до 3,4E+38. А разрядность типа double составляет 64 бита, что приближенно соответствует диапазону представления чисел от 5E-324 до 1,7E+308.

Тип данных float предназначен для меньших значений с плавающей точкой, для которых требуется меньшая точность. Тип данных double больше, чем float, и предлагает более высокую степень точности (15 разрядов).

Если нецелочисленное значение жестко кодируется в исходном тексте (например, 12.3), то обычно компилятор предполагает, что подразумевается значение типа double. Если значение необходимо специфицировать как float, потребуется добавить к нему символ F (или f):

```
float f = 12.3F;
```

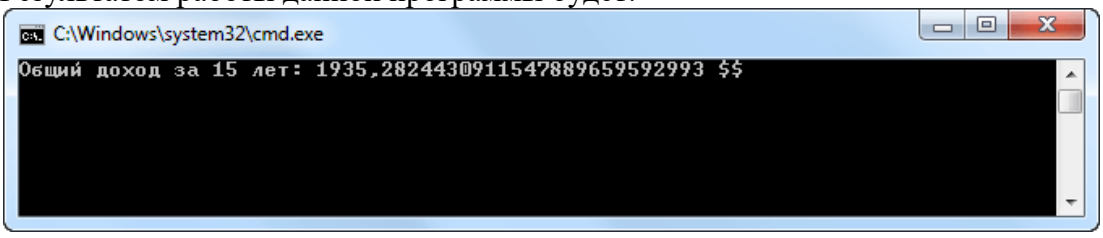
Десятичный тип данных

Для представления чисел с плавающей точкой высокой точности предусмотрен также десятичный тип decimal, который предназначен для применения в финансовых расчетах. Этот тип имеет разрядность 128 бит для представления числовых значений в пределах от 1E-28 до 7,9E+28. Вам, вероятно, известно, что для обычных арифметических вычислений с плавающей точкой характерны ошибки округления десятичных значений. Эти ошибки исключаются при использовании типа decimal, который позволяет представить числа с точностью до 28 (а иногда и 29) десятичных разрядов. Благодаря тому что этот тип данных способен представлять десятичные значения без ошибок округления, он особенно удобен для расчетов, связанных с финансами:

Пример программы:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            // *** Расчет стоимости капиталовложения с ***
            // *** фиксированной нормой прибыли***
         decimal money, percent;
         int i;
         const byte years = 15;

         money = 1000.0m;
         percent = 0.045m;

         for (i = 1; i <= years; i++)
         {
             money *= 1 + percent;
         }

         Console.WriteLine("Общий доход за {0} лет: {1} $$", years, money);
         Console.ReadLine();
        }
}
}
```

Результатом работы данной программы будет:



Символы

В C# символы представлены не 8-разрядным кодом, как во многих других языках программирования, например C++, а 16-разрядным кодом, который называется юникодом (Unicode). В юникоде набор символов представлен настолько широко, что он охватывает символы практически из всех естественных языков на свете. Если для многих естественных языков, в том числе английского, французского и немецкого, характерны относительно небольшие алфавиты, то в ряде других языков, например китайском, употребляются довольно обширные наборы символов, которые нельзя представить 8-разрядным кодом. Для преодоления этого ограничения в C# определен тип char,

представляющий 16-разрядные значения без знака в пределах от 0 до 65 535. При этом стандартный набор символов в 8-разрядном коде ASCII является подмножеством юникода в пределах от 0 до 127. Следовательно, символы в коде ASCII по-прежнему остаются действительными в C#.

Для того чтобы присвоить значение символьной переменной, достаточно заключить это значение (т.е. символ) в одинарные кавычки:

```
char ch;
ch = 'Z';
```

Несмотря на то что тип char определен в C# как целочисленный, его не следует путать со всеми остальными целочисленными типами. Дело в том, что в C# отсутствует автоматическое преобразование символьных значений в целочисленные и обратно. Например, следующий фрагмент кода содержит ошибку:

```
char ch;
ch = 8; // ошибка, не выйдет
```

Наравне с представлением char как символьных литералов, их можно представлять как 4-разрядные шестнадцатеричные значения Unicode (например, '\u0041'), целочисленные значения с приведением (например, (char) 65) или же шестнадцатеричные значения (например, '\x0041'). Кроме того, они могут быть представлены в виде управляющих последовательностей.

Логический тип данных

Тип bool представляет два логических значения: "истина" и "ложь". Эти логические значения обозначаются в C# зарезервированными словами true и false соответственно. Следовательно, переменная или выражение типа bool будет принимать одно из этих логических значений. Кроме того, в C# не определено взаимное преобразование логических и целых значений. Например, 1 не преобразуется в значение true, а 0 — в значение false.

Литералы

В C# литералами называются постоянные значения, представленные в удобной для восприятия форме. Например, число 100 является литералом. Сами литералы и их назначение настолько понятны, что они применялись во всех предыдущих примерах программ без всяких пояснений. Но теперь настало время дать им формальное объяснение.

В C# литералы могут быть любого простого типа. Представление каждого литерала зависит от конкретного типа. Как пояснялось ранее, символьные литералы заключаются в одинарные кавычки. Например, 'a' и '%' являются символьными литералами.

Целочисленные литералы указываются в виде чисел без дробной части. Например, 10 и -100 — это целочисленные литералы. Для обозначения литералов с плавающей точкой требуется указывать десятичную точку и дробную часть числа. Например, 11.123 — это литерал с плавающей точкой. Для вещественных чисел с плавающей точкой в C# допускается также использовать экспоненциальное представление.

У литералов должен быть также конкретный тип, поскольку C# является строго типизированным языком. В этой связи возникает естественный вопрос: к какому типу следует отнести числовой литерал, например 2, 12 3987 или 0.23? К счастью, для ответа на этот вопрос в C# установлен ряд простых для соблюдения правил:

- Во-первых, у целочисленных литералов должен быть самый мелкий целочисленный тип, которым они могут быть представлены, начиная с типа int. Таким образом, у целочисленных литералов может быть один из следующих типов: int, uint, long или ulong в зависимости от значения литерала.
- Литералы с плавающей точкой относятся к типу double.
- Если вас не устраивает используемый по умолчанию тип литерала, вы можете явно указать другой его тип с помощью суффикса.

Так, для указания типа long к литералу присоединяется суффикс l или L. Например, 12 — это литерал типа int, а 12L — литерал типа long. Для указания целочисленного типа без знака к литералу присоединяется суффикс u или U. Следовательно, 100 — это литерал типа int, а 100U — литерал типа uint. А для указания длинного целочисленного типа без знака к литералу присоединяется суффикс ul или UL. Например, 984375UL — это литерал типа ulong.

Кроме того, для указания типа float к литералу присоединяется суффикс F или f. Например, 10.19F — это литерал типа float. Можете даже указать тип double, присоединив к литералу суффикс d или D, хотя это излишне. Ведь, как упоминалось выше, по умолчанию литералы с плавающей точкой относятся к типу double.

И наконец, для указания типа decimal к литералу присоединяется суффикс m или M. Например, 9.95M — это десятичный литерал типа decimal.

Несмотря на то что целочисленные литералы образуют по умолчанию значения типа int, uint, long или ulong, их можно присваивать переменным типа byte, sbyte, short или ushort, при условии, что присваиваемое значение может быть представлено целевым типом.

Шестнадцатеричные литералы

Вам, вероятно, известно, что в программировании иногда оказывается проще пользоваться системой счисления по основанию 16, чем по основанию 10. Система счисления по основанию 16 называется шестнадцатеричной. В ней используются числа от 0 до 9, а также буквы от A до F, которыми обозначаются десятичные числа 10,11,12,13, 14 и 15. Например, десятичному числу 16 соответствует шестнадцатеричное число 10. Вследствие того что шестнадцатеричные числа применяются в программировании довольно часто, в C# разрешается указывать целочисленные литералы в шестнадцатеричном формате. Шестнадцатеричные литералы должны начинаться с символов 0x, т.е. нуля и последующей латинской буквы "икс". Ниже приведены некоторые примеры шестнадцатеричных литералов:

```
count = 0xFF; // равно 255 в десятичной системе
incr = 0x1a; // равно 26 в десятичной системе
```

Управляющие последовательности символов

Большинство печатаемых символов достаточно заключить в одинарные кавычки, но набор в текстовом редакторе некоторых символов, например возврата каретки, вызывает особые трудности. Кроме того, ряд других символов, в том числе одинарные и двойные кавычки, имеют специальное назначение в C#, поэтому их нельзя использовать непосредственно. По этим причинам в C# предусмотрены специальные управляющие последовательности символов:

Управляющие последовательности C#

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

Строковые литералы

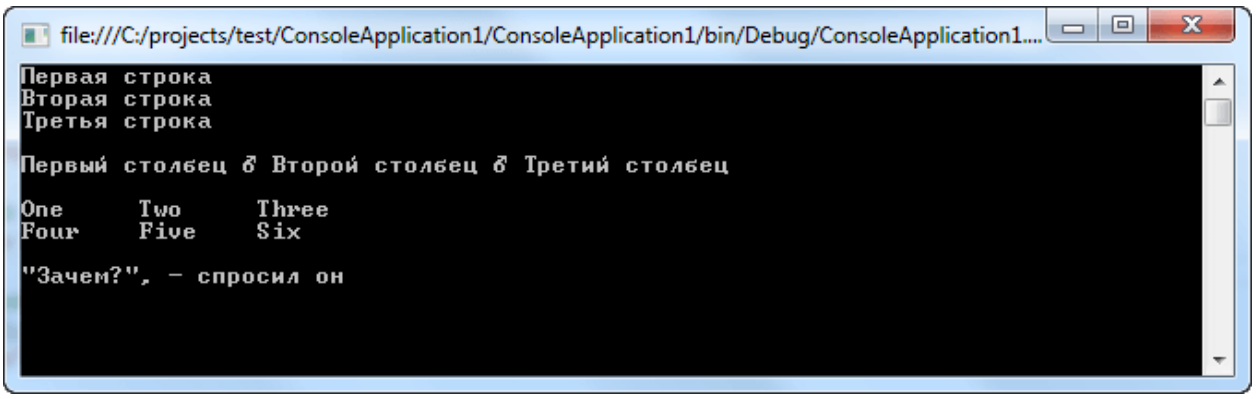
В C# поддерживается еще один тип литералов — строковый. Строковый литерал представляет собой набор символов, заключенных в двойные кавычки. Например следующий фрагмент кода:

```
"This is text"
```

Помимо обычных символов, строковый литерал может содержать одну или несколько управляющих последовательностей символов, о которых речь шла выше. Также можно указать буквальный строковый литерал. Такой литерал начинается с символа @, после которого следует строка в кавычках. Содержимое строки в кавычках воспринимается без изменений и может быть расширено до двух и более строк. Это означает, что в буквальный строковый литерал можно включить символы новой строки, табуляции и прочие, не прибегая к управляющим последовательностям. Единственное исключение составляют двойные кавычки ("), для указания которых необходимо использовать двойные кавычки с обратным слэшем ("\"). Например:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            // Используем перенос строки
            Console.WriteLine("Первая строка\nВторая строка\nТретья
строка\n");

            // Используем вертикальную табуляцию
            Console.WriteLine("Первый столбец \v Второй столбец \v Третий
столбец \n");

            // Используем горизонтальную табуляцию
            Console.WriteLine("One\tTwo\tThree");
            Console.WriteLine("Four\tFive\tSix\n");

            //Вставляем кавычки
            Console.WriteLine("\"Зачем?\", - спросил он");

            Console.ReadLine();
        }
    }
}
```

Результат:



Стандартные математические функции

Декларации математических функций языка С содержатся в классе Math.

В математических функциях аргументы x и y имеют тип double, кроме abs(), параметр n имеет тип int.

Аргументы тригонометрических функций задаются в радианах (2π радиан = 360^o). Все приведенные математические функции возвращают значение (результат) типа double.

Таблица 2

Математическая функция	**Имя функции в языке С**
$\sqrt{x}$	Math.sqrt(x)
$\|x\|$	Math.Abs(x)
e^x	Math.Exp(x)
x^y	Math.Pow(x,y)
ln(x)	Math.Log(x)
lg10(x)	Math.Log10(x)
sin(x)	Math.Sin(x)
cos(x)	Math.Cos(x)
tg(x)	Math.Tan(x)
arcsin(x)	Math.Asin(x)
arctg(x)	Math.Atan(x)

Выражения и операции

Выражения – совокупность операндов (переменных, функций, констант), объединенных знаками операций.

Операции могут быть:

1. Унарными (с одним операндом)
2. Бинарными (с двумя операндами)
3. Тернарные (с тремя операндами).

Унарные:

++	Увеличение на 1 (инкремента)
--	Уменьшение на 1 (декремента)

Существует 2 формы записи: префиксная и постфиксная.

```
int x=0;
++x; // увеличили на 1 (x=1)
--x; // уменьшили на 1 (x=0)
        int y=++x; /* префиксная запись. Сначала x увеличивается на 1, затем новое значение x
      присваивается y (x=1, y=1) */
        int z=x++; /* постфиксная запись. Сначала переменной z присваивается значение x,
```

затем x увеличивается на 1 (x=2, z=1) */	
sizeof	Получение размера типа данных в байтах sizeof(тип); sizeof выражение
~	Поразрядное отрицание (выполняется над одним операндом, вызывает побитовую инверсию двоичного числа) ~13 = -14 0000 0000 0000 1101 = 1111 1111 1111 0010
!	Логическое отрицание 1!1=0 1!0=0 0&1=1 0&0=1
-	Арифметическое отрицание (унарный минус)
+	Унарный плюс
new	Выделение памяти
delete	Освобождение памяти

Бинарные:

*, +, - ,		
/	Целочисленное деление (26/5=5)	
%	Остаток от деления (26/5=1)	

<<, >>	Сдвиг влево, вправо (двоичное представление числа сдвигается влево/вправо)
	
	25<<3; // 11001 << 3 = 11001000 25<<3=200
	25>>3; // 11001 >> 3 = 11 25>>3=3	

<, <=, >, >=	Меньше, меньше или равно, больше, больше или равно	
==	Сравнение на равенство	
!=	Не равно	
&	Поразрядная конъюнкция (И) 1&1=1 1&0=0 0&1=0 0&0=0	Операции битовой арифметики (произв. над двоичным представлением целых чисел)
^	Поразрядное исключающее ИЛИ 1^1=0 1^0=1 0^1=1 0^0=0	
|	Поразрядная дизъюнкция (ИЛИ) 1|1=1 1|0=1 0|1=1 0|0=0	
&&	Логическое И 1&&1=1 1&&0=0 0&&1=0 0&&0=1	Логические операции выполняются над логическими значениями истина/ложь. Результатами явл. значения истина/ложь.
||	Логическое ИЛИ 1||1=1 1||0=1 0||1=1 0||0=0	
=	Присваивание	
*=, +=, /=, %=, -=, <<=, >>=, &=, ^=, |=	Умножение с присваиванием y=y*3 можно записать y*=3; x=x+5 можно записать x+=5;	

Тернарная:

? :	Условная операция
	Выражение1 ? выражение2 : выражение3 (если значение выражения1 – истина, то вычисляется выражение2, иначе выражение3)
	int i=1, j=5, M;
	M= (i<j ? i : j); // M=1

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Создание проекта в Visual Studio 2012

При запуске Visual Studio 2012 открывается окно, отображенное на рисунке 1.1:

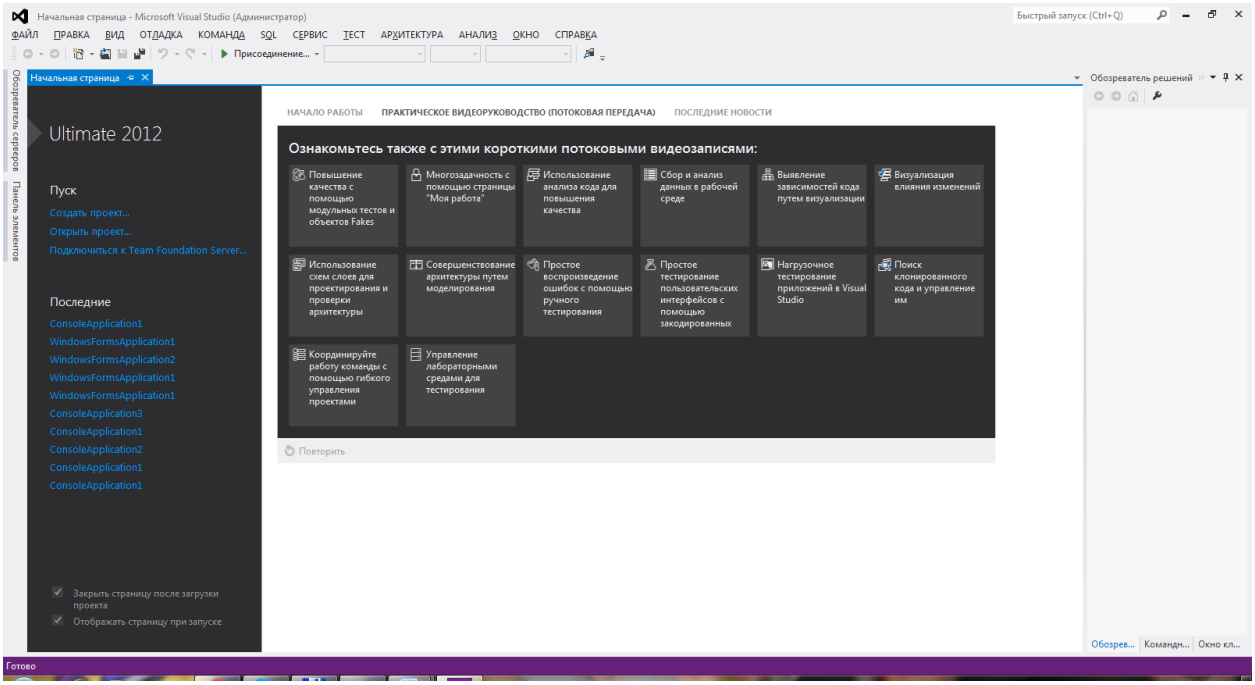


Рисунок 1.1 — Стартовое окно при запуске Visual Studio 2012

Для создания нового проекта надо нажать «Создать проект» (рисунок 1.2):

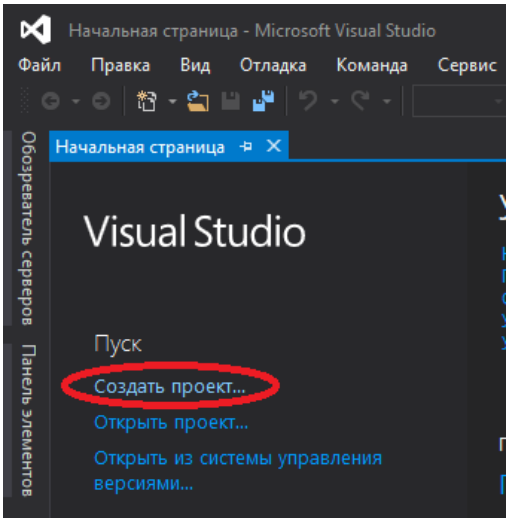


Рисунок 1.2 — Инструкция создания проекта

После этого откроется диалоговое окно «Создать проект» (рис. 1.3), где шужно выполнить 4 шага:

1. Нужно выбрать тип сосдаваемого проекта. В нашем случае создаем консольное приложение, для этого в типе проекта выбираем Win32**;**
2. И выбираем консольное приложение Win32;
3. В поле «Имя» нужно указать имя проекта, а в поле «Расположение» выбрать папку, в которой будет храниться проект;
4. Нажать кнопку «ОК»

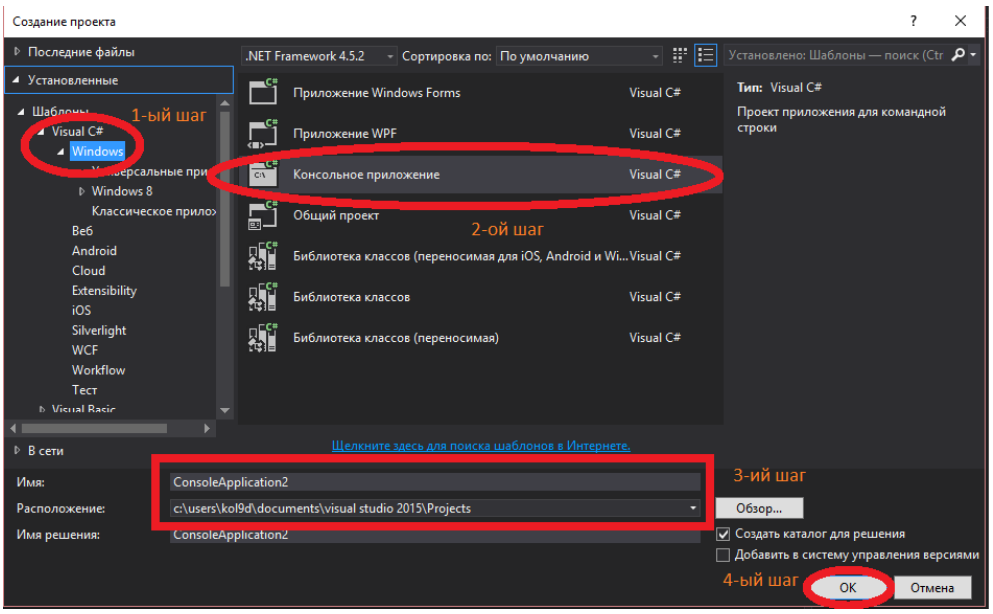


Рисунок 1.3 — Диалоговое окно «Создать проект»

После этого откроется диалоговое окно *Мастер приложений win32* (рисунок 1.4), в котором нажмем кнопку *Готово*:

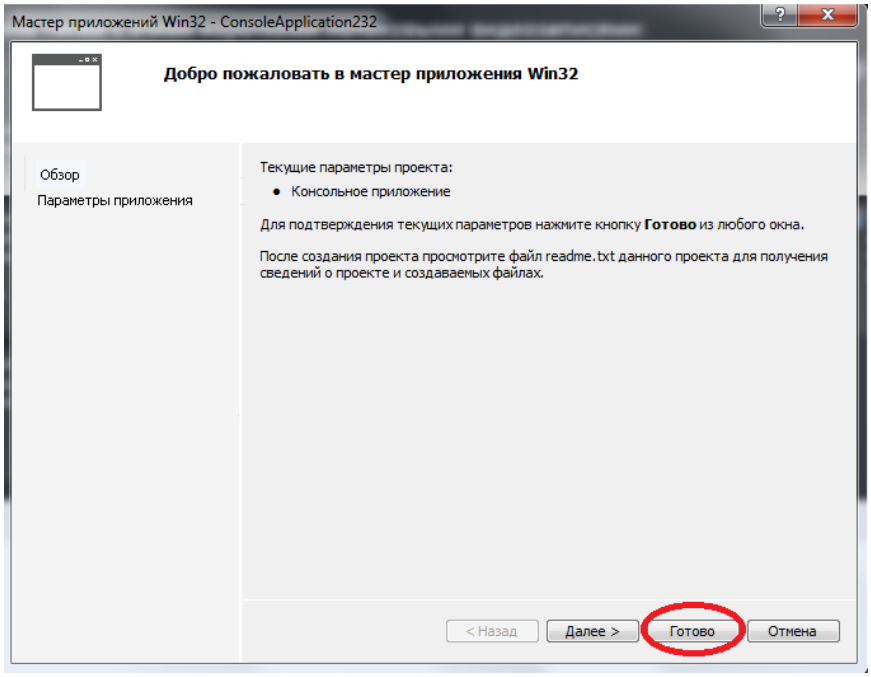


Рисунок 1.4 — Диалоговое окно «Мастер приложений win32»

Варианты индивидуальных заданий

Задание 1 (индивидуальное).

№	Вид формулы	№	Вид формулы
1	$3^{-x}-\cos x+\sin(2xy)$	2	$x-10^{\sin x}+\cos(x-y)$
3	$\frac{\sin x+\cos y}{\cos x-\sin y}\operatorname{tg} xy$	4	$\left(\frac{x+1}{x-1}\right)^{x}+18xy^{2}$
5	$\left(\frac{\ln\lvert\cos x\rvert}{\ln(1+x^{2})}\right)$	6	$\lvert x^{2}-x^{3}\rvert-\frac{7x}{x^{3}-15x}$
7	$\left(1+\frac{1}{x^{2}}\right)^{x}-12x^{2}y$	8	$x-10\sin x+\lvert x^{4}-x^{5}\rvert$
9	$\frac{\cos x}{\pi-2x}+16x\cos(xy)-2$	10	$2\operatorname{ctg}(3x)-\frac{1}{12x^{2}+7x-5}$
11	$3^{x}-4x+(y-\sqrt{\lvert x\rvert})$	12	$\sin\sqrt{x+1}-\sin\sqrt{x-1}$
13	$\frac{b+\sqrt{b^{2}+4ac}}{2a}-a^{3}c+b^{-2}$	14	$\frac{a}{c}\frac{b}{d}-\frac{ab-c}{cd}$
15	$\frac{x+y}{x+1}-\frac{xy-12}{34+x}$	16	$x-\frac{x^{3}}{3}+\frac{x^{5}}{5}$
17	$\frac{3+e^{y-1}}{1+x^{2}\lvert y-\operatorname{tg} x\rvert}$	18	$\frac{x^{2}-7x+10}{x^{2}-8x+12}$
19	$x\ln x+\frac{y}{\cos x-x}$	20	$\frac{1+\sin\sqrt{x+1}}{\cos(12y-4)}$

Задание 2 (индивидуальное).

1. Вычислить периметр и площадь прямоугольного треугольника по заданным длинам двух катетов a и b .

2. Заданы координаты трех вершин треугольника $(x1, y1)$, $(x2 , y2)$, $(x3, y3)$. Найти его периметр и площадь.

3. Вычислить длину окружности и площадь круга одного и того же заданного радиуса R .

4. Вычислить расстояние между двумя точками с данными координатами $(x1, y1)$ и $(x2 , y2)$.

5. Даны два действительных числа x и y . Вычислить их сумму, разность, произведение и частное.

6. Дана длина ребра куба. Найти площадь грани, площадь полной поверхности и объем этого куба.

7. Дана сторона равностороннего треугольника. Найти площадь этого треугольника, его высоты, радиусы вписанной и описанной окружностей.

8. Известна длина окружности. Найти площадь круга, ограниченного этой окружностью.

9. Найти площадь кольца, внутренний радиус которого равен r , а внешний – заданному числу R $(R > r)$.

10. Треугольник задан величинами своих углов и радиусом описанной окружности. Найти стороны треугольника.

11. Найти площадь равнобедренной трапеции с основаниями a и b и углом α при большем основании a .

12. Вычислить корни квадратного уравнения $ax2 + bx + c = 0$, заданного коэффициентами a, b и c (предполагается, что $a \neq 0$ и что дискриминант уравнения неотрицателен).

13. Дано действительное число x . Не пользуясь никакими другими арифметическими операциями, кроме умножения, сложения и вычитания, вычислить за минимальное число операций $2x^4 - 3x^3 + 4x^2 - 5x + 6$.

14. Найти площадь треугольника, две стороны которого равны a и b, а угол между этими сторонами q .

15. Найти сумму членов арифметической прогрессии, если известны ее первый член, знаменатель и число членов прогрессии.

16. Найти все углы треугольника со сторонами a , b , c . Предусмотреть в программе перевод радианной меры угла в градусы, минуты и секунды.

17. Три сопротивления $R1$, $R2$, $R3$ соединены параллельно. Найдите сопротивление соединения.

18. Текущее показание электронных часов: m часов $(0 \le m \le 23)$, n минут $(0 \le n \le 59)$, k секунд $(0 \le k \le 59)$. Какое время будут показывать u1095 часы через p ч q мин r с ?

19. Составить программу вычисления объема цилиндра и конуса, которые имеют одинаковую высоту H и одинаковый радиус основания R .

20. Ввести любой символ и определить его порядковый номер, а также указать предыдущий и последующий символы.

21. Дана величина A , выражающая объем информации в байтах. Перевести A в более крупные единицы измерения информации.

22. Составить программу для вычисления пути, пройденного лодкой, если ее скорость в стоячей воде v км/ч, скорость течения реки w км/ч, время движения по озеру $t1$ ч, а против течения реки – $t2$ ч.

23. Дано x . Получить значения $-2x + 3x^2 - 4x^3$ и $1+ 2x + 3x^2 + 4x^3$ с наименьшим числом произведенных операций.

Содержание отчета

1. Титульник;
3. Цель работы;
4. Текст задания (постановка задачи) шрифтом Times New Roman;
5. Блок-схема алгоритма решения поставленной задачи;
6. Текст программы шрифтом Courier New;
7. Результаты выполнения программы;
8. Выводы.

ВОПРОСЫ ВЫХОДНОГО КОНТРОЛЯ:

1. Перечислите типы данных языка C#.
2. Перечислите спецификаторы языка C#.
3. Перечислите ввода/вывода информации.
4. Перечислите основные математические операции.
5. Дайте понятие выражение.
6. Перечислите основные операции языка C#.

Домашнее задание: закрепить умения разработки линейных программ на ЯП C#.