Лабораторная работа 1: Разработка элементарных программ на языке программирования C#

Цель работы: Познакомиться с базовыми особенностями языка программирования C# и разработки программ с использованием среды Visual Studio.

Необходимые теоретические сведения

Краткие сведения о платформе .NET

Платформа .NET Framework определяет среду для поддержки создания и выполнения платформонезависимых гетерогенных приложений. Основными особенностями данной платформы являются не зависящая от языка среда исполнения (Common Language Runtime, CLR) и библиотека классов .NET

Основа языка C#

Обьявление и инициализация переменных:

Тип\_переменной имя\_переменной [=значение];

Примеры:

int x; //обьявление переменной x

x=100; //инициализация переменной x

long w,z=100; //обьявление переменных w и z и

//инициализация z

long q=100\*z; //обьявление переменной с динамической

//инициализацией

C# - язык со строгим контролем типов данных. Есть 2 основные категории встроенных типов данных в C# - простые типы и ссылочные типы.

Основные простые типы данных в C#:

тип Описание бит

bool Значение истина/ложь 1

byte 8-битовое беззнаковое целое 8

char Символ 16

decimal Числовой тип для финансовых вычислений 128

double Число двойной точности с плавающей точкой 64

float Число с плавающей точкой 32

int Знаковое целое 32

long Длинное знаковое целое 64

sbyte 8-битовое знаковое целое 8

short Короткое целое 16

uint Беззнаковое целое 32

ulong Беззнаковое длинное целое 64 8

ushort Беззнаковое короткое целое 16

Область видимости переменной в C# - блок кода (заключенный в фигурные скобки {}). Переменная создается при входе в область видимости и уничтожаются при выходе из нее.

Основные управляющие операторы:

Условный:

if (условие) оператор [else оператор];

if (условие1) оператор1;

else if (условие2) оператор2;

else if (условие3) оператор3;

…

Выбора:

switch (выражение){

case константа1:

оператор1;

…

break;

case константа2:

операторX1;

…

break;

…

default:

операторZ1;

…

break;

}

Цикла:

for(инициализация, условие\_выхода, итерация) оператор;

while (условие\_продолжения) оператор;

do оператор; while (условие продолжения);

Пространство имен:

Пространство имен определяет область объявления, что позволяет

хранить каждый набор имен отдельно от других наборов. В С# имена,

объявленные в одном пространстве имен, не конфликтуют с такими же 9

именами, объявленными в другом пространстве имен. Библиотекой .NET

Framework (библиотекой С#) используется пространство имен System.

Для того, чтобы чтобы сделать видимыми пространства имен без

указания полного имени (через ‘.’) используется директива using

Синтаксис:

using имя\_пространства\_имен;

также возможно использование псевдонимов для имен

using псевдоним = имя;

Пространство имен объявляется с помощью ключевого слова

namespace.

Синтаксис:

namespace имя {

члены\_пространства\_имен}

Пример программы на C#

Программа ‘Hello, World’ на языке C# выглядит следующим образом:

using System;

class HelloWorld01

{

public static void Main()

{

Console.Write("Hello, World!");

Console.ReadLine();

}

}

Контрольные вопросы

1. Что понимается под термином «.NET Framework»?

2. Зависят ли приложения, разрабатываемые в .NET, от платформы?

3. Возможно ли создание гетерогенных приложений в среде .NET?

4. Что означает аббревиатура «CLR»?

5. Является ли среда CLR многоязычной?

6. Приведите обобщенный синтаксис объявления переменной на языке

C#.

7. Приведите обобщенный синтаксис инициализации переменной на

языке C#. 10

8. Какая дисциплина (вариант контроля) типов принята в языке C#?

9. Каковы основные категории типов в языке C#?

10. Перечислите пять простых типов языка C#.

11. Что понимается под областью видимости переменной в языке C#?

12. Как обозначается область видимости переменной в языке C#?

13. Как соотносится время жизни переменной и область видимости?

14. Приведите синтаксис условного оператора в общем виде.

Проиллюстрируйте его фрагментом программы на языке C#.

15. Приведите синтаксис оператора выбора в общем виде.

Проиллюстрируйте его фрагментом программы на языке C#.

16. Что понимается под термином «пространство имен»?

17. В чем состоит назначение пространств имен в языке C#?

18. Благодаря какому механизму удается избежать коллизий имен в языке C#?

19. Какое пространство имен использует системная библиотека .NET Framework?

20. Какое пространство имен использует системная библиотека C#?

21. В чем состоит назначение директивы using?

22. Какой символ используется для указания полного имени объекта в языке C#?

23. Приведите синтаксис директивы using в общем виде. Проиллюстрируйте его фрагментом программы на языке C#.

24. Приведите синтаксис описания пространства имен в общем виде.

Проиллюстрируйте его фрагментом программы на языке C#.

Упражнение 1. Создание простой программы

В этом упражнении Вы напишите программу на языке C#, используя

среду разработки Visual Studio.NET. Программа будет спрашивать, как Вас

зовут и затем здороваться с Вами по имени.

Создайте новое консольное C#-приложения

• Запустите Microsoft Visual Studio.NET. (Start → All Programs →

Visual Studio .NET → Microsoft Visual Studio.NET).

• Выберите пункт меню File→New→Project.

• На панели Project Types выберите Visual C# Projects.

• На панели Templates выберите Console Application.

• В текстовое поле Name введите имя проекта Greetings.

• В поле Location укажите каталог для проекта и нажмите OK.

• Измените имя класса на Greeter.

• Сохраните проект, выбрав пункт меню File→Save All.

Напишите код, запрашивающий имя пользователя и приветствующий его по имени.

• В методе Main вставьте следующую строку кода:

string myName;

• Напишите код, запрашивающий имя пользователя.

• Напишите код, считывающий введенное пользователем имя и

присваивающий полученное значение строковой переменной

myName.

• Добавьте код, который будет выводить на экран строку “Hello myName”, где myName – имя, введенное пользователем.

• Итоговый текст метода Main должен выглядеть следующим

образом:

static void Main(string[ ] args)

{

string myName;

Console.WriteLine("Please enter your name");

myName = Console.ReadLine( );

Console.WriteLine("Hello {0}", myName);

}

• Сохраните проект.

Откомпилируйте и запустите программу

• Выберите пункт меню Build→Build Solution (или Ctrl+Shift+B).

• При необходимости исправьте ошибки и откомпилируйте программу заново.

• Выберите пункт меню Debug→Start Without Debugging (или Ctrl+F5).

• В появившемся окне введите свое имя и нажмите ENTER.

• Закройте приложение.

Упражнение 2. Компиляция и запуск C#-программы из командной строки

В этом упражнении Вы откомпилируете и запустите Вашу программу

из командной строки.

Откомпилируйте и запустите Ваше приложение из командной строки

• Запустите Visual Studio .NET Command Prompt. (Start→All Programs→Visual Studio .NET→Visual Studio .NET Tools→Visual Studio .NET Command Prompt)

• Перейдите в каталог install folder\Labs\Lab02\Greetings

• Откомпилируйте Вашу программу, используя следующую команду: csc /out:Greet.exe Program1.cs

• Запустите программу, набрав в командной строке ее название: Greet

• Закройте окно командной строки.

Упражнение 3. Использование отладчика Visual Studio .NET

В этом задании Вы приобретете навыки работы с интегрированным

отладчиком Visual Studio .NET, изучите порядок отладки программы по шагам и окна для просмотра значения переменных.

Поставьте точки остановки и запустите пошаговое выполнение

• Запустите Visual Studio .NET, если она не запущена.

• Выберите пункт меню File→Open→Project.

• Откройте проект Greetings.sln .

• В редакторе кода класса Greeter щелкните по крайнему левому полю на уровне строки кода, где впервые встречается команда Console.WriteLine.

• Выберите пункт меню Debug→Start (или нажмите F5).

Программа запустится на выполнение, появится консольное окно и затем программа прервется в месте точки остановки.

Просмотрите значение переменной

• Выберите пункт меню Debug →Windows→Watch→Watch1.

• В окне Watch в список выражений для мониторинга добавьте

переменную myName.

• В окне Watch появится переменная myName с текущим значением

null.

Используйте команды пошагового выполнения

• Для выполнения первой команды Console.WriteLine выберите пункт

меню Debug→Step Over (или нажмите F10).

• Для выполнения следующей строчки кода, содержащей команду

Console.ReadLine, снова нажмите F10.

• Вернитесь в консольное окно, введите свое имя и нажмите ENTER.

Вернитесь в Visual Studio. Текущее значение переменной

myName в окне Watch будет содержать Ваше имя.

• Для выполнения следующей строчки кода, содержащей команду

Console. WriteLine, снова нажмите F10.

• Разверните консольное окно. Там появилось приветствие.

• Вернитесь в Visual Studio. Для завершения выполнения программы

выберите пункт меню Debug→Continue (или нажмите F5).

Упражнение 4. Добавление в C#-программу обработчика

исключительных ситуаций

В этом упражнении Вы напишите программу, в которой будет использоваться обработчик

исключительных ситуаций, который будет отлавливать ошибки времени выполнения. Программа будет

запрашивать у пользователя два целых числа, делить первое число на второе и выводить полученный

результат.

Создайте новую C#-программу

• Запустите Visual Studio .NET, если она не запущена.

• Выберите пункт меню File→New→Project

• На панели Project Types выберите Visual C# Projects.

• На панели Templates выберите Console Application.

• В текстовое поле Name введите имя проекта Divider.

• В поле Location укажите каталог для проекта install

folder\Labs\Lab02 и нажмите OK.

• Измените имя класса на DivideIt.

• Сохраните проект, выбрав пункт меню File→Save All.

Напишите код, запрашивающий у пользователя два целых числа.

• В методе Main() напишите код, запрашивающий у пользователя первое целое число.

• Напишите код, считывающий введенное пользователем число и присваивающий полученное значение переменной temp типа string.

• Добавьте код, который переведет значение переменной temp из типа данных string в int и сохранит полученный результат в переменной i: int i = Int32.Parse(temp);

• Аналогичным образом создайте следующий код:

Запросите у пользователя второе целое число.

Считайте введенное пользователем число и присвойте полученное значение переменной temp.

Переведите значение переменной temp в тип данных int и сохраните полученный результат в переменной j.

Итоговый текст программы должен выглядеть следующим образом:

Console.WriteLine("Please enter the first integer");

string temp = Console.ReadLine( );

int i = Int32.Parse(temp);

Console.WriteLine("Please enter the second integer");

temp = Console.ReadLine( );

int j = Int32.Parse(temp);

• Сохраните проект.

Разделите первое число на второе и выведите результат на экран

• Напишите код, создающий новую переменную k типа int, в которую будет заноситься результат деления числа i на j, и поместите его после кода, созданного в предыдущем пункте.

int k = i / j;

• Добавьте код, выводящий значение k на экран.

• Сохраните проект.

Протестируйте программу

• Выберите пункт меню Debug→Start Without Debugging (или Ctrl+F5).

• Введите первое число 10 и нажмите ENTER.

• Введите второе число 5 и нажмите ENTER.

• Проверьте, что выводимое значение k будет равным 2.

• Снова запустите программу на выполнение, нажав Ctrl+F5.

• Введите первое число 10 и нажмите ENTER.

• Введите второе число 0 и нажмите ENTER.

• В программе возникнет исключительная ситуация (деление на ноль).

• Для очистки окна диалога Just-In-Time Debugging выберите No.

Добавьте в программу обработчик исключительных ситуаций

• Поместите код метода Main() внутрь блока try следующим образом: try

{

Console.WriteLine (...);

...

int k = i / j; 10

Console.WriteLine(...);

}

• В методе Main() после блока try добавьте блок catch, внутри которого должно выводиться краткое сообщение об ошибке:

catch(Exception e)

{

Console.WriteLine("An exception was thrown: {0}", e);

}

...

• Сохраните проект.

• Итоговый текст метода Main должен выглядеть следующим образом:

public static void Main(string[ ] args)

{

try {

Console.WriteLine ("Please enter the first integer");

string temp = Console.ReadLine( );

int i = Int32.Parse(temp);

Console.WriteLine ("Please enter the second integer");

temp = Console.ReadLine( );

int j = Int32.Parse(temp);

int k = i / j;

Console.WriteLine("The result of dividing {0} by

{1} is {2}", i, j, k);

}

catch(Exception e) {

Console.WriteLine("An exception was thrown: {0}",

e);

}

}

Протестируйте код обработчика исключительных ситуаций

• Снова запустите программу на выполнение, нажав Ctrl+F5.

• Введите первое число 10 и нажмите ENTER.

• Введите второе число 0 и нажмите ENTER.

В программе вновь возникнет исключительная ситуация (деление на ноль), но на этот раз ошибка перехватывается и на экран выводится сообщение.

**Варианты заданий**

Написать C# программу, реализующую функцию согласно варианту задания. Исходные данные вводятся с клавиатуры.

1. Реализовать функцию вычисления суммы двух целых чисел

2. Реализовать функцию вычисления разности двух целых чисел

3. Реализовать функцию вычисления произведения двух целых чисел

4. Реализовать функцию вычисления частного двух целых чисел

5. Реализовать функцию вычисления суммы двух вещественных чисел

6. Реализовать функцию вычисления разности двух вещественных

чисел

7. Реализовать функцию вычисления произведения двух вещественных

чисел

8. Реализовать функцию вычисления частного двух вещественных

чисел

9. Реализовать функцию возведения целого числа в квадрат

10. Реализовать функцию возведения в квадрат суммы двух целых

чисел 11

11. Реализовать функцию возведения в квадрат разности двух целых

чисел

12. Реализовать функцию возведения в квадрат произведения двух

целых чисел

13. Реализовать функцию возведения в квадрат частного двух целых

чисел

14. Реализовать функцию возведения в квадрат суммы двух

вещественных чисел

15. Реализовать функцию возведения в квадрат разности двух

вещественных чисел

16. Реализовать функцию возведения в квадрат произведения двух

вещественных чисел

17. Реализовать функцию возведения в квадрат частного двух

вещественных чисел

18. Реализовать функцию возведения в куб целого числа

28. Реализовать функцию возведения в куб суммы двух целых чисел

20. Реализовать функцию возведения в куб разности двух целых чисел

21. Реализовать функцию возведения в куб произведения двух целых

чисел

22. Реализовать функцию возведения в куб частного двух целых чисел

23. Реализовать функцию возведения в куб суммы двух вещественных

чисел

24. Реализовать функцию возведения в куб разности двух

вещественных чисел

25. Реализовать функцию возведения в куб произведения двух

вещественных чисел

26. Реализовать функцию возведения в куб частного двух

вещественных чисел

Порядок выполнения работы:

1) Реализовать программу на C# в соответствии с вариантом

исполнения.

3) Подготовить отчет в твердой копии и в электронном виде.