Лабораторно упражнение № 1

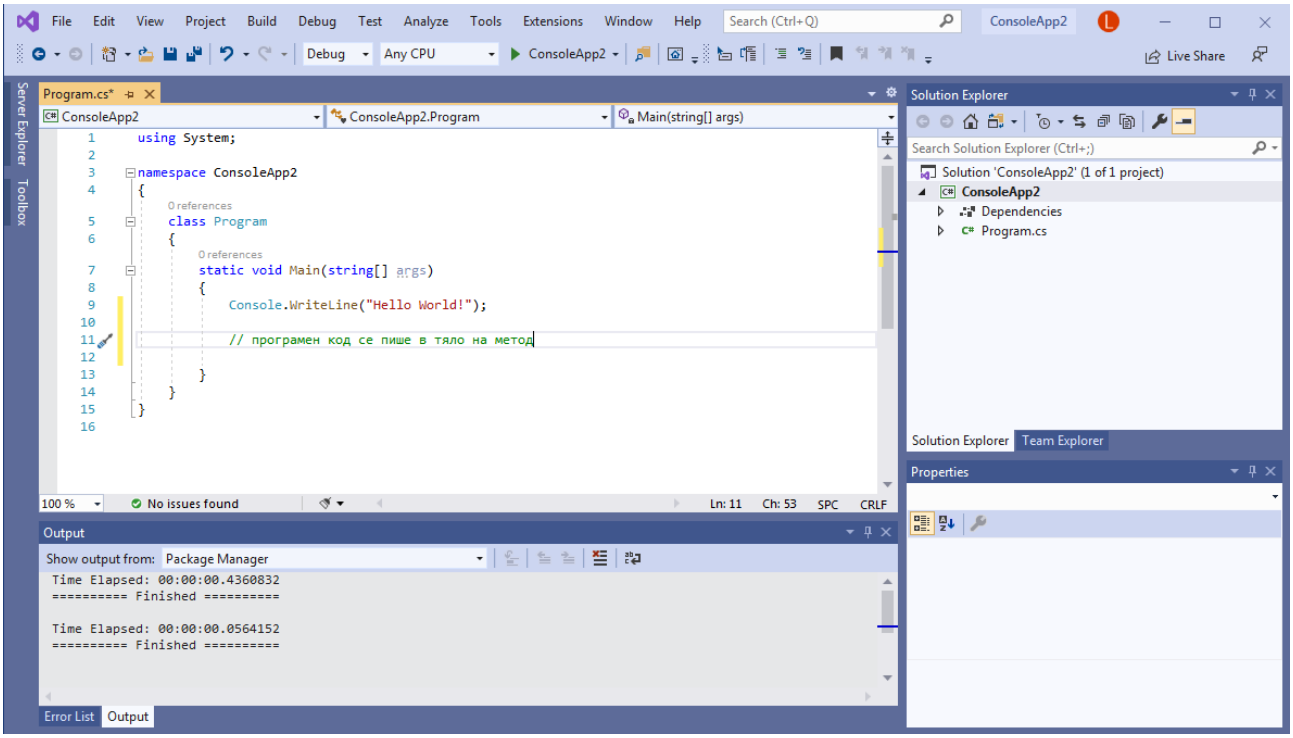
Структура на C# програма. Създаване на конзолни приложения чрез MS Visual Studio .NET. Въвеждане и извеждане на данни. Работа с масиви.

I. Теоретична част

1. Създаване на конзолни приложения чрез Visual Studio .NET

MS Visual Studio .NET е интегрирана развойна среда за разработка на програмно осигуряване.

За да се създаде проект се избира командна поредица File/New project, избира се програмен език (C#) и тип на приложението (в случая, конзолно приложение - Console Application). Освен това могат да се изберат и съответните настройки за самото приложение, име на приложението, както и къде да бъде записан проектът. Средата генерира скелета на приложението. Като пример за вече създадено конзолно приложение може да се види представеното на фиг. 1.



Фиг. 1. Пример за конзолно приложение

В посочения пример, приложението извежда на екрана низ "Hello World!".

2. Структура на C# програма. Метод Main().

Примерната програма, която извежда съобщението "Hello World!" в C# има следния вид:

```
class HelloWorld
{
```

```
    public static void Main()
      {    // тяло на метода
      System.Console.WriteLine("Hello World!");
      }
}
```

В синтаксиса на C#, отделните блокове с код са затворени в големи скоби {...}. Входна точка за приложението в статичния метод[1] Main(), който трябва да се съдържа в класа на приложението. Особеното в C# е, че главната функция всъщност е метод на клас. За да може да бъде извикана без да се създава обект (инстанция) на класа. Трябва да се обърне внимание и, че за разлика от C++, в C# ключова дума Main е с главна буква.

За да е общодостъпен методът Main() е необходимо да е деклариран като public. Отпечатването на израза на екрана е с помощта на метод WriteLine(), който е статичен метод на клас Console.

За начинаещи програмисти е препоръчително първоначално програмният код да се пише в тялото на метод Main. Тялото на всеки един метод, както и тялото на функция, се формира от блока отваряща и затваряща фигурни скоби {…}.

2. Именувано обектно пространство

За разлика от други програмни езици, в C# не съществува библиотека с базови класове само за този език. Ако се направи сравнение между NET библиотеката и MFC, при втората има определен набор базови класове на C++ за създаване на диалогови прозорци, менюта, панели за управление и др. Вместо библиотеката с базови класове, разработчиците на C# използват библиотека с базови типове за всички среди NET. За по-добра организация на типовете, се използва концепцията за обектно пространство на имената.

Основната разлика с библиотеките за конкретен език, например MFC, е, че за всеки NET съвместим език се използват едни и същи типове и едни и същи именувани пространства като на C#. Например, за работа със системната конзола е необходим клас Console, който използва едно и също обектно пространство на имената System.

Обектно пространство на имената е начин за организация на типовете (класове, делегати, интерфейси, структури) в една група т.е. в едно множество. Дадено пространство може да съдържа в себе си също и друго обектно именувано пространство. Разбира се, в едно пространство могат да се обединяват взаимосвързани типове. Например, пространството System.Drawing обединява типовете, необходими за организиране на графичен изход на екрана.

За да може да се използва дадено обектно пространство, е необходимо да бъде обявено за използване чрез ключова дума **using**.

Пример:

```
using System;
```

[1] В обектно ориентираното програмиране методите са аналози на функциите в процедурно ориентираното програмиране.

Microsoft .NET Framework съдържа множество именувани пространства, най-важното от които е System. Това обектно пространство съдържа класове, които управляват входа и изхода с операционната система.

Когато бъде включено именувано пространство, не е необходимо то да се записва пред името на класа при извикване на метод. Така, програмата, която извежда низа "Hello World!" има следния вид:

```
using System;

class HelloWorld
{
    public static void Main()
      {
            Console.WriteLine("Hello World!");
      }
}
```

В случая не е необходимо да се посочва `System` пред метод `Console.WriteLine`.

Примери за обектни пространства на имена са:

System - множество класове за работа с прости типове на ниско ниво, изпълнение на математически операции, събиране на останки и други.

System.Collections - за работа с контейнерни обекти като: **ArrayList**, **Queue**, **SortedList** и др.

System.Data - за обръщение към база данни

System.Drawing - типове за GDI примитиви.

System.Web - класове, предназначени за използване на web-приложения.

System.Windows.Forms - класове за работа с елементи на интерфейса Windows, като прозорци, елементи на управление и др.

3. Елементарен вход-изход с използване на клас Console.

Клас Console осигурява на C# приложенията достъп до стандартния вход, стандартния изход и стандартните потоци за грешки. Тъй като стандартно входно устройство се явява клавиатурата, там се пренасочва и стандартния вход. Стандартните изход и потоци за грешки се пренасочват към екрана. Освен това, трите стандартни потока могат да бъдат пренасочени към файл или към друго устройство.

Трябва да се отбележи, че клас Console се използва при конзолни приложения, които се стартират в команден прозорец.

Най-често използваните методи на клас Console са:

WriteLine() - извежда низ на екрана с преминаване на нов ред

ReadLine() - чете низ от клавиатурата

Write() - извежда низ на екрана без преминаване на нов ред

Read() - чете единичен символ от клавиатурата.

3.1. Методи за извеждане на информация на екрана

За извеждане на данни на екрана се използват методи WriteLine() и Write() на клас Console. И двата метода извеждат низ на екрана, като разликата между тях е, че при първия маркерът преминава на нов ред т.е. добавя двойката управляващи символи CR/LF.

Подобно на функцията printf() в C/C++, методите Write и WriteLine изискват като параметри форматиращ низ и аргументи (евентуално). Форматиращия низ е заграден в кавички и това е текстът, който се извежда на екрана. Пример:

```
Console.WriteLine("Hello World!");
```

Мястото на извеждане на стойностите на аргументите се определя чрез фигурни скоби {}, като първият аргумент се определя със стойност {0}, следващите нарастват с единица. Пример:

```
Console.WriteLine("The sum of {0} and {1} is equal to {2}.", 10, 20, 10+20);
```

На екрана се извежда:

```
The sum of 10 and 20 is equal to 30.
```

Параметрите могат да бъдат отпечатани с ляво или дясно изравняване в определен брой позиции. Например:

```
Console.WriteLine("\"Left justified in a field with a 10: {0,-10}\"", 55);
Console.WriteLine("\"Right justified in a field with a 10: {0,10}\"", 55);
```

И в двата случая на екрана се отпечатва стойността 55 в 10 позиции, като в първия случай цифрите са ляво изравнени, а във втория – дясно изравнени. За да бъде отпечатан символ кавички (“) във форматиращият низ, пред символа се поставя обратна наклонена черта (\”).

Във форматиращия низ може да бъде определено как да бъдат отпечатани цифровите данни. Общият вид на низа е следния:

{N,M:FormatString}

където:

N е номер на параметъра;

M е позиции за изравняване (при положителна стойност изравняването е дясно, при отрицателна – ляво);

FormatString е параметър за форматиране.

Параметрите за форматиране в низа са следните:

C (**c**) - извеждане на стойност в паричен формат

D (d**)** - извеждане на десетична стойност

E (**e**) - извеждане на число в експоненциален формат

F (**f**) - извеждане на число във формат фиксирана запетая

G (**g**) - общ формат за извеждане в експоненциален формат или плаваща запетая

N (**n**) - стандартно числово форматиране с използване на разделители между хилядните

X (**x**) - извеждане на число във шестнадесетична формат.

В случая, няма значение дали се използва малка или главна буква. След форматиращия параметър може да се постави цифра. При паричния и форматите за реални числа, цифрата е броя на знаците след десетичната

точка, при целочислените формати – броя на разрядите, в които се извежда числото.

Примери за използване на метод **WriteLine** чрез използване на форматните спецификации:

```
    Console.WriteLine("Currensy format: {0,10:C}; {1,10:c4}",55.4,-55.4);
    Console.WriteLine("Integer format: {0,10:D}; {1,10:d4}",55,-55);
    Console.WriteLine("Exponencial format: {0,10:E}; {1,10:e4}",55.4,-
55.4);
    Console.WriteLine("Fix-point format: {0,10:F}; {1,10:f4}",55.4,-55.4);
    Console.WriteLine("General format: {0,10:G}; {1,10:g4}",55.4,-55.4);
    Console.WriteLine("Number format: {0,10:N}; {1,10:n4}",55.4,-55.4);
    Console.WriteLine("Hexadecimal format: {0,10:X}; {1,10:x4}",55,55);
```

3.2. Методи за въвеждане на информация от клавиатурата

За въвеждане на данни от клавиатурата се използват методите Read() и ReadLine(). Метод Read() чете символ от клавиатурата и го връща като резултат от тип int. При наличие на грешка, функцията връща стойност -1. Тъй като типа на функцията е int, при необходимост върнатата стойност се преобразува явно до тип char. Пример:

```
int i=Console.Read();
char c=(char)i;
Console.WriteLine("The symbol is {0}",c);
```

Метод ReadLine() чете низ от клавиатурата, като края на низа се фиксира с управляващите символи CR/LF. Пример:

```
Console.Write("name:");
string name=Console.ReadLine();
Console.WriteLine("name: {0}",name);
```

Тъй като методът въвежда низ от клавиатурата, ако трябва да бъдат въведени числа се използват функциите за преобразуване на низ в число. За целта могат да се използват методите на класове Int16, Int32, Int64, Double, Convert. Примери:

```
string temp = Console.ReadLine();
int i=Int32.Parse(temp);
Console.WriteLine("The value is {0:D}",i);

temp = Console.ReadLine();
double d = Double.Parse(temp);
Console.WriteLine(" The value  is {0:f}",d);
```

Класовете, които съответстват на типовете, например Int16, Int32, Int64, Double и др., съдържат метод Parse, който осъществява преобразуването до съответният тип. В случая, в програмен ред `int i=Int32.Parse(temp);` метод

`Parse` осигурява преобразуването на низа `temp` до число от тип int, за който отговаря клас `Int32`.

За преобразуване може да се използва и клас `Convert`. Той съдържа методите, които преобразуват стойността до съответния тип. Например, програмен ред `double d = Convert.ToDouble(temp);` осигурява конвертиране на низа `temp` до число от тип double.

II. Задачи за изпълнение

1. Да се създаде конзолно приложение, с което се прави демонстрация на възможностите на методите Write, WriteLine, Read и ReadLine за въвеждане и извеждане на данни.

2. Да се създаде конзолно приложение на C# за намиране лицата на следните фигури: квадрат, правоъгълник, кръг и триъгълни като изборът на фигура да е чрез реализация на потребителско меню.

Упътване: Решението на задачата може да е по два начина – с и без използване на функции (методи). При решението с използване на функции, намирането на лицето на всяка една от фигурите е с отделна функция. В случая, всяка една от функциите ще е оформена като статичен метод на клас Program.
За реализацията на потребителското меню се препоръчва да се използва конструкция switch.

3. Да се състави конзолно приложение на C# за преобразуване на температура от скалата на Фаренхайт в скала на Целзий и обратно.

Упътване: Формулите за преобразуване са: F=1.8C+32; $C = \frac{F - 32}{1.8}$,

където C и F са температура по Целзий и Фаренхайт, съответно.

4. Да се състави конзолно приложение на C# за преобразуване стойност на скорост от км/час в м/сек. и обратно.