

Sofia University
Department of Mathematics and Informatics

Course : **OO Programming with C#.NET**

Date: October 24, 2010

Student Name:

Lab No. 4

Submit the all C#.NET files developed to solve the problems listed below. Use comments and Modified-Hungarian notation.

Problem No. 1

Напишете C# конзолно приложение за шифроване на текстов низ от данни (*plaintext*) посредством алгоритъма на тъй наречения *Caesar cipher*. При този алгоритъм всяка буква в текста за **шифроване** (*plaintext*) се **замества с буква отстояща на дясно от нея на *SHIFT_LENGTH* позиции** и се получава шифрования текст (*ciphertext*)

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Например, в оригиналната версия на този алгоритъм *SHIFT_LENGTH=3* и думата **TOY** (*plaintext*) се шифрова като **WRB** (*ciphertext*)- забележете, **заместването с букви в края на азбуката води до циклично продължение от началото на азбуката**
Дешифрирането се извършва **по обратния път** (циклично заместване на всяка буква с тази отстояща от нея на *SHIFT_LENGTH* позиции) по зададени *ciphertext*.

Напишете конзолно приложение на Java, която **кодира** даден *plaintext* и **декодира** даден (*ciphertext*) в съответствие с по-горе описания алгоритъм.

Всички операции за обработка на низове да се сведат до работа с масиви като се **използват единствено** метода *ToCharArray()* на *class String* за преобразуване от *String* в масив от *char*. Обратното преобразуване от *char[]* в *String* става с използване на **конструктора** *String(char[])*. **Използване на други *String* методи не се допуска.**

При тестването на приложението използвайте меню в текстов режим за избиране на шифроване или дешифриране

Problem No. 2

A **simple transposition cipher** works as follows to encrypt a given string (**plaintext**). The user enters phrase and **the number of characters of the phrase** are considered as the **cipher key**. Then the user supplies the string to encrypt- the plaintext. The encryption procedure cuts the plaintext into series of characters, whose number is equal to the cipher key, where the last portion of characters is eventually padded with spaces to the size of the cipher key. These series are stored by rows of a matrix and the encrypted text comprises the sequence of columns of the thus obtained matrix. Example of encryption:

plaintext:	thisistheplaintext
key text	beauty → cipher key= 6

encryption

thisis
thepla
intext

cipher text: ttihhnietspeilxsat

Once the user provides the key text and the cipher text the decryption reconstructs the original plain text.

Write a **C# Windows** application to implement the encryption and decryption process using the `String.ToCharArray()` method and arrays of `char` datatype as in the following example

```
string delimStr = " ,.:";  
char [] delimiter = delimStr.ToCharArray();// to char array  
String text = new String(delimiter);      // to string
```

DO NOT USE ANY OTHER string METHODS!

Use **OO design** and in addition to the **GUI Window Windows** class define also a **class TransCipher** with respective data, constructors and methods, allowing you to implement the above encryption/ decryption functionality.

Test encryption / decryption of a string using different cipher keys.

Problem No. 3

For faster sorting of letters, the USA postal service encourages companies that send large volumes of mail to use a bar code denoting the ZIP code. The encoding scheme for a five-digit ZIP code is as follows:

Number/base code	7	4	2	1	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	1	0	0	0	1
8	1	0	0	1	0
9	1	0	1	0	0
0	1	1	0	0	0

Thus, each number is represented in terms of the coefficients (0's and 1's) of the base codes provided in the above table. For instance, the decimal number $1 = 0 \cdot 7 + 0 \cdot 4 + 0 \cdot 2 + 1 \cdot 1 + 1 \cdot 0$ and its barcode is 00011 (see the coefficients for the number 1 in the above table).

Write a Console application that asks the user for a ZIP code (a three digit decimal number) and prints the respective barcode for this number. Use the character ':' to display halfbars (the zeroes in the barcode) and the character '|' for full bars (the 1's in the barcode). Thus, the ZIP code 111 should be displayed as

: : : | | : : : | | : : : | |

Problem No. 4

Consider the **two- dimensional** and **one- dimensional array** structures defined below:

```
table
    3, 4, 5, 6, 3
    2, 1, 2, 1, 1,
    3, 2, 1, 4, 9
    1, 0, 0, 1, 1
    9, 2, 6, 4, 9

1st
2, 1, 2, 3, 4
```

Write a C# Console application with data members *table* and *1st*. Use **implicit type** to define initialize *table* and *1st* as above given in the class *Main()* method .

Write a method *Subtract* with nested *for* loop structure to subtract the array *1st* from each row of the two-dimensional *table*, where method *Subtract* takes as arguments two arguments- a two dimensional and a single dimensional *integer* arrays. It should return the obtained two dimensional array after the subtraction. For example, after completion, the first row of table that have to be returned will be

1,3,3,3, -1 since $3-2=1$, $4-1=3$, $5-2=3$, $6-3=3$, $3-4=-1$

Additionally, write a method *Multiply* that takes as arguments two arguments- a two dimensional and a single dimensional *integer* arrays and **returns by reference** a single dimensional array that is the product of the two arrays.

Display in a dialog *MessageBox* the arrays obtained after executing *Subtract* and *Multiply* by rows and columns as well as the given initial values of *table* and *1st*.

Problem No. 5

Write an application to simulate the rolling of two dice. The application should use an object of **class Random** once to roll the first die and again to roll the second die. The sum of the two values should then be calculated. Each die can show an integer value from 1 to 6, so the sum of the values will vary from 2 to 12, with 7 being the most frequent sum and 2 and 12 being the least frequent sums. The figure below shows the 36 possible combinations of the two dice. Your application should roll the dice 36,000 times. Use a one-dimensional array to tally the number of times each possible sum appears. Display the results in tabular format.

Determine whether the totals are reasonable (e.g., there are six ways to roll a 7, so approximately one-sixth of the rolls should be 7).

	1	2	3	4	5	6
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12

Write methods to compute the averages by rows and columns. Display these values appropriately formatted with two digits after the decimal point.

Problem No. 6.

Морз кодовете на английската азбука е показано на таблицата по- долу, където **точките** и **тиретата** представят късите и дългите сигнали по Морз.

A	. -	B	- . . .	C	- . - .	D	- . .	E	.	F	. . - .
G	- - .	H	I	. .	J	. - - -	K	- . -	L	. - . .
M	- -	N	- .	O	- - -	P	. - - .	Q	- - . - -	R	. - .
S	. . .	T	-	U	. . . -	V	. . . -	W	. - -	X	- . . -
Y	- . - -	Z	- - . .								

Напишете *class MorseCode* , което има клас данна *code* , която е **едномерен *String* масив константа** и е **инициализиран** с **точките** и **тиретата** за всяка буква **A- Z** **в съответствие с горната таблица**. (елементът с индекс 0 е низа “. -“, елементът с индекс 1 е низа “- - .“ и т. н.)

Напишете метод *encode()* на *class MorseCode* , който взима за аргумент *String* **променлива (English текст в главни букви за кодиране)** , а връща *String* със **съответния Morse код**. Разделяйте, съответните Морз кодове със един празен символ.

Напишете метод *decode()* на *class MorseCode* , който взима за аргумент *String* **променлива (Morse код текст)** , а връща *String* със **съответния English текст в главни букви** . Използвайте това, че празен символ е разделител между отделните групи Морз кодове.

Напишете *class MorseCode Test* за тестване на всеки от тези методи . **Изведете на печат резултатите от изпълнение на тези методи, подходящо форматирани.** Всички операции за **обработка на низове да се сведат до работа с масиви** като се **използват единствено** метода *ToCharArray()* на *class String* за преобразуване от *String* в масив от *char* . Обратното преобразуване от *char[]* в *String* става с използване на **конструктора** *String(char[])* .

Упътване : Ако текущо прочетения символ от английския текст е *x* , то *x* - ' A ' е номер на позицията в масива *code* , съдържащ **Морз кода** за символа *x* .