Лабораторная работа №7:
представление символов

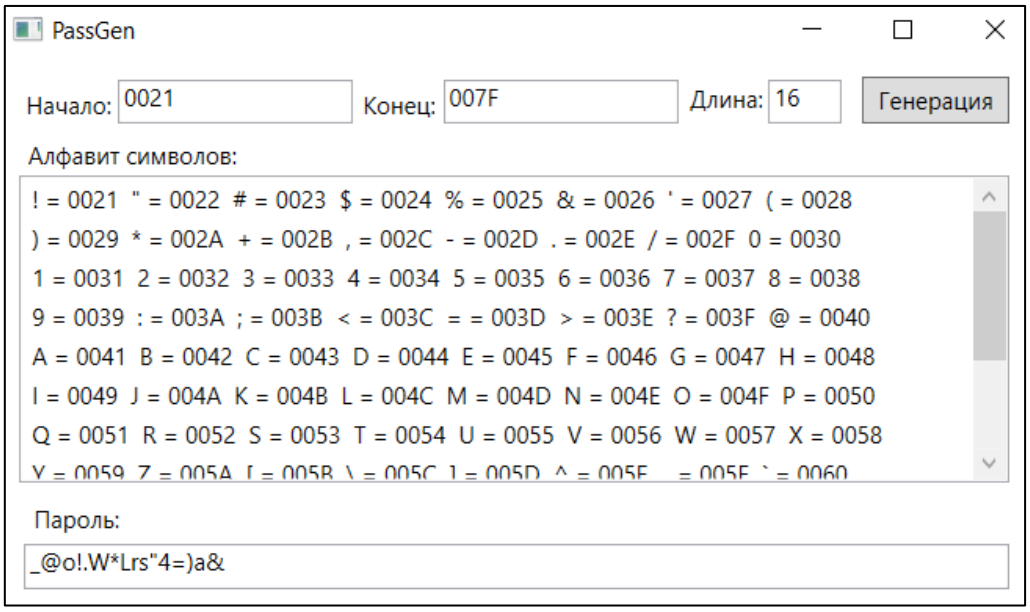
Цель:

Целью данной работы является получение базовых навыков работы с представлением символов в компьютерной технике

Задание:

Разработайте и реализуйте программное приложение, способное:

1. Получать на вход диапазон символов юникода и длину. Возвращать сгенерированный по указанному диапазону пароль, заданной длины.
   Пример:



2. Шифровать и дешифровать текстовые файлы, используя выбранный вами алгоритм шифрования.
   Пример результата шифрования файла при помощи функции XOR с ключом 27:

Краткая справка:

В вычислительной технике, как правило, для представления символов используют таблицы кодировок. Таблицы кодировок, это таблицы, которые устанавливают соответствие между числами и соответствующими им символами. Простейшим примером такой таблицы, является таблица ASCII, имеющая диапазон значений от 0 до 255 и, соответственно, позволяющая отобразить 255 символов.

Фрагмент ASCII таблицы:

DEC	OCT	HEX	BIN	Символ	HTML код	Мнемоника
32	040	0x20	00100000	Пробел	&#32;	
33	041	0x21	00100001	!	&#33;	
34	042	0x22	00100010	"	&#34;	&quot;
35	043	0x23	00100011	#	&#35;	
36	044	0x24	00100100	$	&#36;	
37	045	0x25	00100101	%	&#37;	
38	046	0x26	00100110	&	&#38;	&amp;

Как можно заметить, символу Пробела соответствует десятичный код 32 и двоичный код 0010 0000, а символу # соответствуют 35 и 0010 0011 соответственно.

На сегодняшний день, существуют таблицы кодировок, поддерживающие диапазон значений значительно больше 255 символов, например, Unicode.

Фрагмент таблицы Unicode 16 (UTF-16):

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
0010	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
0020		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0030	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0040	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O

Поскольку кодировка UTF-16 поддерживает диапазон до 1 112 064 символов, запись в десятичной, а тем более двоичной, системах была бы очень неудобной. Поэтому, для записи кодов символов, используется шестнадцатеричная система исчисления.

Шестнадцатеричная система исчисления – система, содержащая 16 цифр. Для записи первых 10 используются цифры десятичной системы, а оставшиеся 6 записываются в виде символов от A до F.

Таблица записи чисел в разных системах исчисления:

Десятичная запись:	Двоичная запись:	Шестнадцатеричная запись:
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Как видно из таблицы, используя 16-теричную систему исчисления, можно записывать символы в кодировке ASCII как 2х разрядные числа и символы UTF-16 как 4х разрядные. Например, числу 0031 – соответствует символ ‘1’, а числу 007E соответствует символ ‘~’.

Для считывания шестнадцатеричного числа из строки в целочисленную переменную, можно использовать код следующего вида:

```
int a = int.Parse(From.Text, System.Globalization.NumberStyles.HexNumber);
```

Для преобразования кода символа юникода в символ юникода, можно использовать код вида:

```
string str = char.ConvertFromUtf32(a).ToString();
```

Для того, чтобы вывести целое число в шестнадцатеричном виде, можно использовать параметр функции:

```
string str = i.ToString("X4");
```

В зависимости от того, каким образом вы решили шифровать содержимое файла, вы можете его считать, как последовательность строк, либо как массив байт.

Считать текст из файла, в виде последовательности байт, можно используя классы File и FileStream:

```
FileStream fs = File.OpenRead(dlg.FileName); //открытие файла на чтение

byte[] array = new byte[fs.Length]; //создание массива байт
fs.Read(array, 0, array.Length);    //запись содержимого файла в массив байт
string textFromFile = System.Text.Encoding.UTF8.GetString(array); //преобразование массива в строку
```

Одним из методов шифрования является применение операции “Исключающее ИЛИ” (XOR). Поскольку операция является обратимой, зная ключ шифрования, можно как зашифровать, так и расшифровать имеющиеся значения.

Пример:

```
byte symbol = 127; //символ
byte key = 33;     //код
symbol = (byte)(symbol ^ key); //шифрование символа
symbol = (byte)(symbol ^ key); //расшифровка символа
```

Записать зашифрованный текст в файл можно так же при помощи класса FileStream:

```
using (FileStream fstream = new FileStream(dlg.FileName, FileMode.OpenOrCreate))
{
    byte[] array = System.Text.Encoding.UTF8.GetBytes(text); //получение строки в виде массива байт
    fstream.Write(array, 0, array.Length); //запись массива байт в файл
}
```

Список литературы:

Таблица символов юникода: https://unicode-table.com/

Метод получения юникод символов: https://docs.microsoft.com/ru-ru/dotnet/api/system.char.convertfromutf32?view=net-5.0

Шифрование текста шифром Цезаря: https://planetcalc.ru/1434/

Шифр Вернама: https://thecode.media/vernam/

Криптографические алгоритмы: http://elcomdesign.ru/market/kriptograficheskie-algoritmy/