// **The encryption program in C++ and ASM** with a very simple encryption method - it simply adds 1 to the character.

// The encryption method is written in ASM. You will replace this with your allocated version for the assignment.

// In this version parameters are passed via registers (see 'encrypt' for details).

// Filename: "4473 FoCA Assignment 2 2017 - Encryption Original with ASM.cpp"

// Last revised Feb 2017 by A.Oram

**char EKey = 'x'; // Replace x with your Encryption key.**

**#define StudentName "Adrian Oram"**

**// \*\*\* PLEASE CHANGE THE NAME IN QUOTES ABOVE TO YOUR NAME \*\*\***

**// \*\*\* KEEP ALL COMMENTS UP-TO-DATE. COMMENT USEFULLY ALL CODE YOU PRODUCE. \*\*\***

#define MAXCHARS 6 // feel free to alter this, but 6 is the minimum

using namespace std;

<#includes omitted from listing>

#define dollarchar '$' // string terminator

char OChars[MAXCHARS],

EChars[MAXCHARS],

DChars[MAXCHARS] = "Soon!"; // Global Original, Encrypted, Decrypted character strings

//----------------------------- C++ Functions ----------------------------------------------------------

void get\_char (char& a\_character)

{

cin >> a\_character;

while (((a\_character < '0') | (a\_character > 'z')) && (a\_character != dollarchar))

{ cout << "Alphanumeric characters only, please try again > ";

cin >> a\_character;

}

}

//-------------------------------------------------------------------------------------------------------------

void get\_original\_chars (int& length)

{ char next\_char;

length = 0;

get\_char (next\_char);

while ((length < MAXCHARS) && (next\_char != dollarchar))

{

OChars [length++] = next\_char;

get\_char (next\_char);

}

}

//---------------------------------------------------------------------------------------------------------------

//----------------- ENCRYPTION ROUTINES -------------------------------------------------------------------------

void encrypt\_chars (int length, char EKey)

{ char temp\_char; // Character temporary store

for (int i = 0; i < length; i++) // Encrypt characters one at a time

{

temp\_char = OChars [i]; // Get the next char from Original Chars array

// Note the lamentable lack of comments below!

\_\_asm { //

push eax //

push ecx //

//

movzx ecx,temp\_char //

lea eax,EKey //

**call encrypt\_nn** //

mov temp\_char,al //

//

pop ecx //

pop eax //

}

EChars [i] = temp\_char; // Store encrypted char in the Encrypted Chars array

}

return;

// **Encrypt subroutine**. You should paste in the encryption routine you've been allocated from Bb and

// overwrite this initial, simple, version. Ensure you change the ‘call’ above to use the

// correct 'encrypt\_nn' label where nn is your encryption routine number.

// **Inputs: register EAX = 32-bit address of Ekey,**

**// ECX = the character to be encrypted (in the low 8-bit field, CL).**

**// Output: register EAX = the encrypted value of the source character (in the low 8-bit field, AL).**

// REMEMBER TO UPDATE THESE COMMENTS AS YOU DO THE ASSIGNMENT. DELETE OLD/STALE COMMENTS.

**\_\_asm {**

**encrypt\_nn:**

**mov eax,ecx // get character**

**inc eax // simply add 1 to character! EKey value not used in this simple version.**

**ret**

**}**

**//--- End of Assembly code**

}

// end of encrypt\_chars function

//---------------------------------------------------------------------------------------------------------------

//---------------------------------------------------------------------------------------------------------------

//----------------- DECRYPTION ROUTINES -------------------------------------------------------------------------

//

void decrypt\_chars (int length, char EKey)

{

/\*\*\* to be written by you \*\*\*/

return;

}

// end of decrypt\_chars function

//---------------------------------------------------------------------------------------------------------------

int main(void)

{

int char\_count (0); // The number of actual characters entered (upto MAXCHARS limit).

cout << "\nPlease enter upto " << MAXCHARS << " alphanumeric characters: ";

get\_original\_chars (char\_count);

ofstream EDump;

EDump.open("EncryptDump.txt", ios::app);

EDump << "\n\nFoCA Encryption program results (" << StudentName << ") Encryption key = '" << EKey << "'";

EDump << "\nDate: " << GetDate() << " Time: " << GetTime();

// Display and save initial string

cout << "\n\nOriginal string = " << OChars << "\tHex = ";

EDump<< "\n\nOriginal string = " << OChars << "\tHex = ";

for (int i = 0; i < char\_count; i++)

{

cout << hex << setw(2) << setfill('0') << ((int(OChars[i])) & 0xFF) << " ";

EDump<< hex << setw(2) << setfill('0') << ((int(OChars[i])) & 0xFF) << " ";

};

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Encrypt the string and display/save the result

encrypt\_chars (char\_count, EKey);

cout << "\n\nEncrypted string = " << EChars << "\tHex = ";

EDump<< "\n\nEncrypted string = " << EChars << "\tHex = ";

for (int i = 0; i < char\_count; i++)

{

cout << ((int(EChars[i])) & 0xFF) << " ";

EDump<< ((int(EChars[i])) & 0xFF) << " ";

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Decrypt the encrypted string and display/save the result

decrypt\_chars (char\_count, EKey);

cout << "\n\nDecrypted string = " << DChars << "\tHex = ";

EDump<< "\n\nDecrypted string = " << DChars << "\tHex = ";

for (int i = 0; i < char\_count; i++)

{

cout << ((int(DChars[i])) & 0xFF) << " ";

EDump<< ((int(DChars[i])) & 0xFF) << " ";

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

cout << "\n\n\n";

EDump << "\n\n-------------------------------------------------------------";

EDump.close();

system("PAUSE");

return (0);

} // end of whole encryption/decryption program --------------------------------------------------------------------