# Lab 3 - Basic Encryption/Decryption using Caesar Additive Cipher

Khalid Asad
1147299
asadk
October 22nd, 2012
Lab 03
L06

## Problem:

The requirement of this application is to design and implement a program that will encrypt and decrypt messages using a Caesar additive cipher. These messages will be read from an input text file and the encrypted or decrypted result will be written to a text file which will both be stored in the same directory as the program. The program shall prompt the user with a choice of encryption or decryption. After either of these takes place, an analysis will be performed to calculate:

1. ASCII sum
2. Mean/average
3. Standard Deviation
4. Frequency and
5. Relative Frequency

## Introduction:

Encryption is the process of changing a message or piece of information (plain text) into an incomprehensible form using a complex algorithm (cipher). It can be read only in one way, with a key and another algorithm, this is called Decryption. The earliest time of using these processes (cryptography) was circa 1900BC where the Egyptians used hieroglyphs. As time progressed, Ibrahim Al-Kindi, an Arab mathematician, wrote a book on cryptography about deciphering or cracking cryptographic messages.

Cryptographic messages were very useful in passing information that only a certain person(s) would be able to read. This was convenient when sending radio signals during the world wars and during any modern war. This was crucial to the Allies during World War II, because it would allow them to send messages to each other without the German's finding out. Decryption of German messages was also very useful as it allowed the allies to understand what the enemy was up to.

In order to complete this program, a series of skills or knowledge in are required:

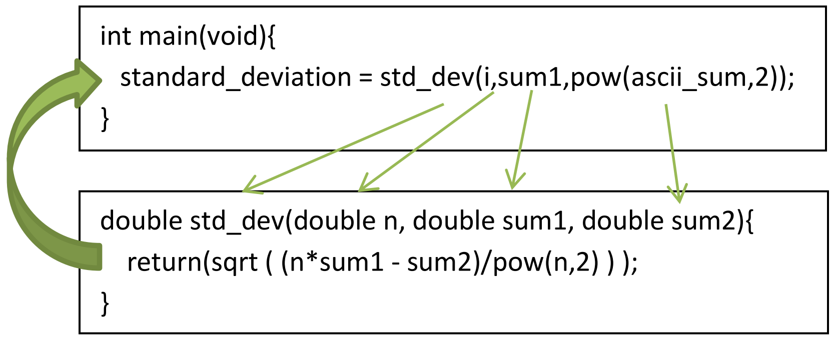
- Manipulate code to meet requirements
- Conditional statements (if, else if, else)
- Syntax
- Use of functions (creating, calling)
- Loops (specifically while)
- Structure use and manipulation
- Use of pointers and manipulation (FILE pointers)
- ASCII and character manipulation
- Globalization and localization of variables
- Array use and manipulation
- I/O operations
- Math.h library (power)

## Lab Questions:

**QUESTION 1**

The process of invoking the std_dev function:

1. Function assigns new memory for the variables to be used, in this case: number of characters, sum of characters, and sum of characters squared.
2. Based on the value of the argument values form of the function, they get copied to a formal parameter memory location.
3. The Function executes and then returns calculation.



```
int main(void){
  standard_deviation = std_dev(i,sum1,pow(ascii_sum,2));
}
```

```
double std_dev(double n, double sum1, double sum2){
  return(sqrt ( (n*sum1 - sum2)/pow(n,2) ) );
}
```

**QUESTION 2**

Through some thorough research, I discovered a type of encryption called ‘Solitaire Cipher’. This encoding essentially works with these steps:

1. Split message into groups of any number.
   EXA MPL E
2. Use solitaire to generate n key stream letters (same amount as the original message).
   AID XYT P
3. Convert both the message and the key stream letters into numbers. i.e. A=1
   EXA MPL E => 5,24,1  13,15,12  5
   AID XYT P => 1,9,4  24,25,20  15
4. Add the numbers for the message and key stream together and modulo 26.
   (5+1)%26 = 6, (24+9)%26 = 7, etc
5. Convert the added numbers back into letters.
   FGE KOG T

Source: *Available in references section.*

## Analysis:

Encrypting seems pretty forthright, but it really is not as easy as it looks. Depending on the type of encryption that is used, it can be difficult or easy, in our case (Caesar cipher), we have it easy. The Caesar/additive cipher using an encryption/decryption key was applied based on the last two digits of my student number, 99. Each character will pass through conditional statements to verify its ASCII value either as a capital letter or not, since we are only required to use capital letters in this lab. Characters are defined by their ASCII values, so when the ASCII value of the character is determined by its integer value stored in C, it will allow the program to perform one of the following computations: for any character between 'A' and 'Z'. After the character is valid, it will be sent off to be encrypted. It will be done by this pseudo-code piece, Encrypted Char = (ASCII value of character - 65 + key)%26 + 65 where the key is defined as 99. Each character will be outputted and written to the output text file.

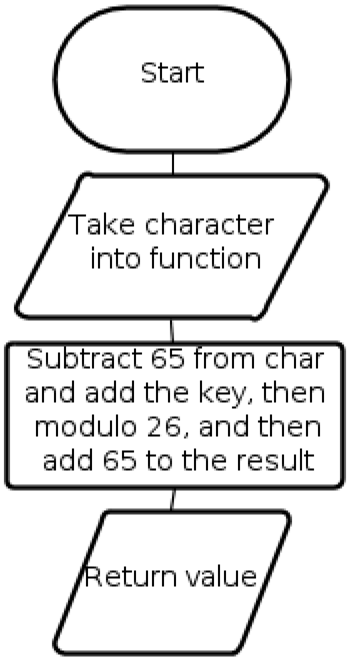
In the other case of Decrypting, it is essentially the same, except the pseudo-code piece is a bit different. In this case it will look a bit like this: Decrypted Char = ((26 - (key-(ASCII value of character - 65))%26)%26) + 65, where the key is 99. Again, each character will be outputted and written to the output text file.

The following analysis was taken into consideration when designing the program:

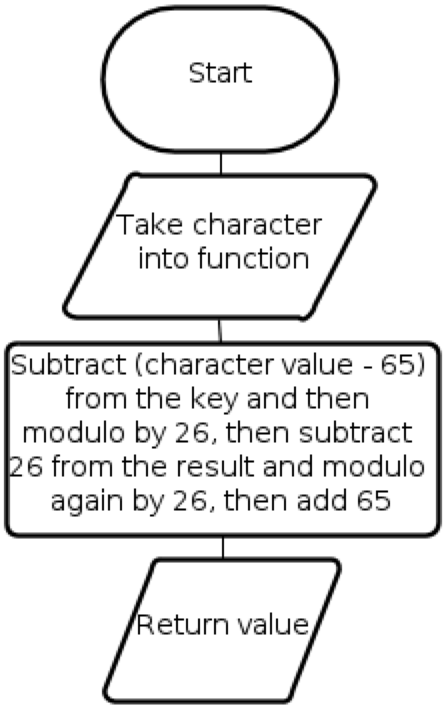
1. Sum the ASCII integer value of all characters
2. The mean of the data set (ASCII values)
3. The standard deviation of the entire data set (ASCII values)
4. Frequency of letters
5. Relative frequency of letters

## Design:

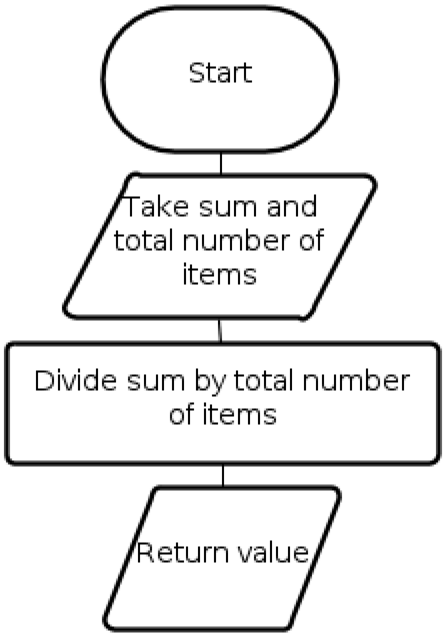
Encryption Function:


Start
Take character into function
Subtract 65 from char and add the key, then modulo 26, and then add 65 to the result
Return value

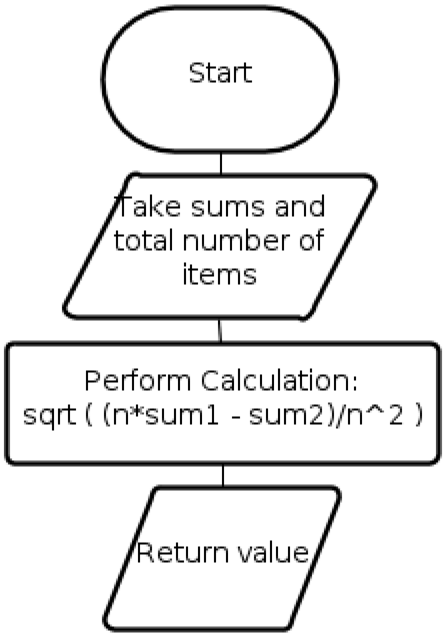
Decryption Function:


Start
Take character into function
Subtract (character value - 65) from the key and then modulo by 26, then subtract 26 from the result and modulo again by 26, then add 65
Return value

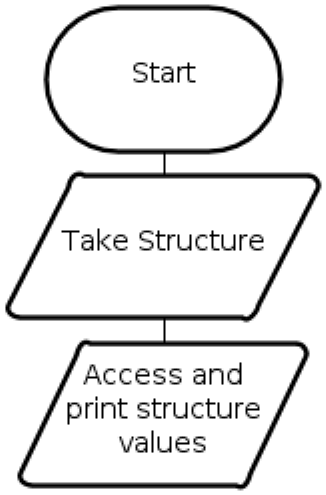
Average Function:


Start
Take sum and total number of items
Divide sum by total number of items
Return value

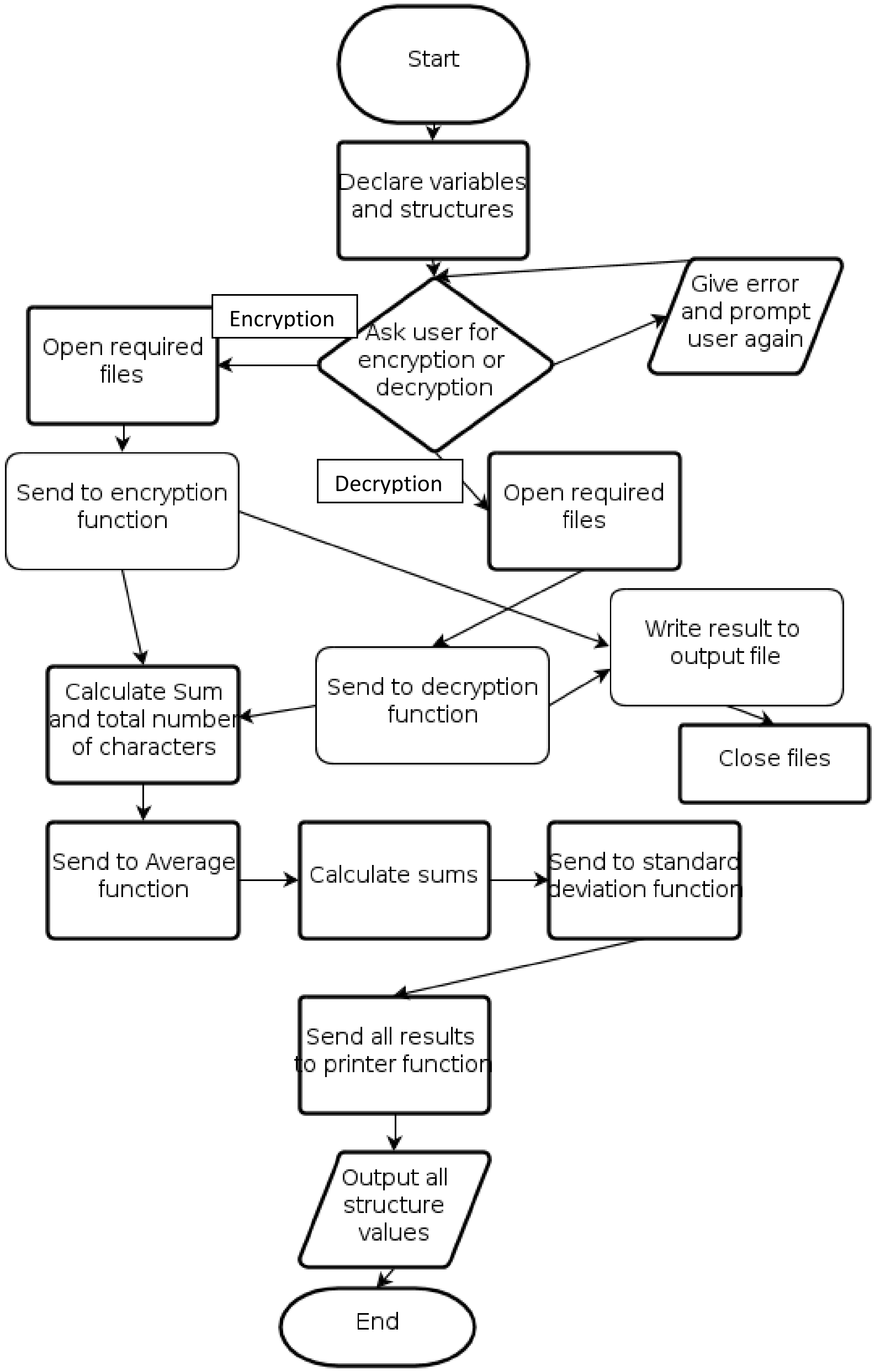
Standard Deviation Function:


Start
Take sums and total number of items
Perform Calculation: sqrt ( (n*sum1 - sum2)/n^2 )
Return value

Printer Function:

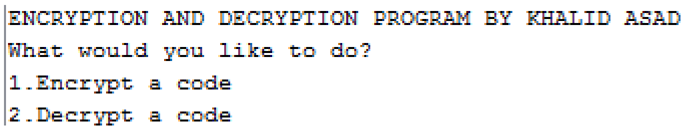


Main Function:


Start
Declare variables and structures
Ask user for encryption or decryption
Give error and prompt user again
Encryption
Open required files
Decryption
Open required files
Send to encryption function
Write result to output file
Send to decryption function
Calculate Sum and total number of characters
Close files
Send to Average function
Calculate sums
Send to standard deviation function
Send all results to printer function
Output all structure values
End

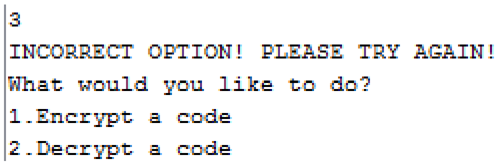
## Observations:

At the start of the program, the user is prompted to choose between encryption and decryption



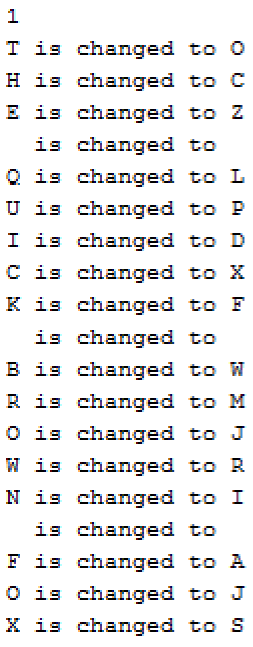
```
ENCRYPTION AND DECRYPTION PROGRAM BY KHALID ASAD
What would you like to do?
1.Encrypt a code
2.Decrypt a code
```

If the user picks a number outside of the range, the user will be given an error message and prompted again as to what they would like to do:

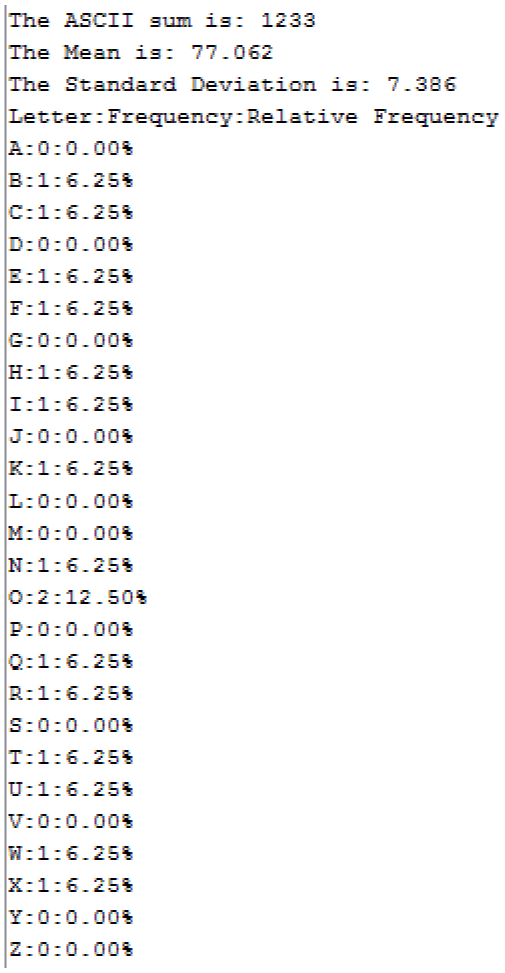


```
3
INCORRECT OPTION! PLEASE TRY AGAIN!
What would you like to do?
1.Encrypt a code
2.Decrypt a code
```

Let’s pick encryption this time by inputting ‘1’.The initial message that will be changed from the text file is “THE QUICK BROWN FOX”. The input file that is used is named: “plaintext.txt”



```
1
T is changed to O
H is changed to C
E is changed to Z
  is changed to  
Q is changed to L
U is changed to P
I is changed to D
C is changed to X
K is changed to F
  is changed to  
B is changed to W
R is changed to M
O is changed to J
W is changed to R
N is changed to I
  is changed to  
F is changed to A
O is changed to J
X is changed to S
```



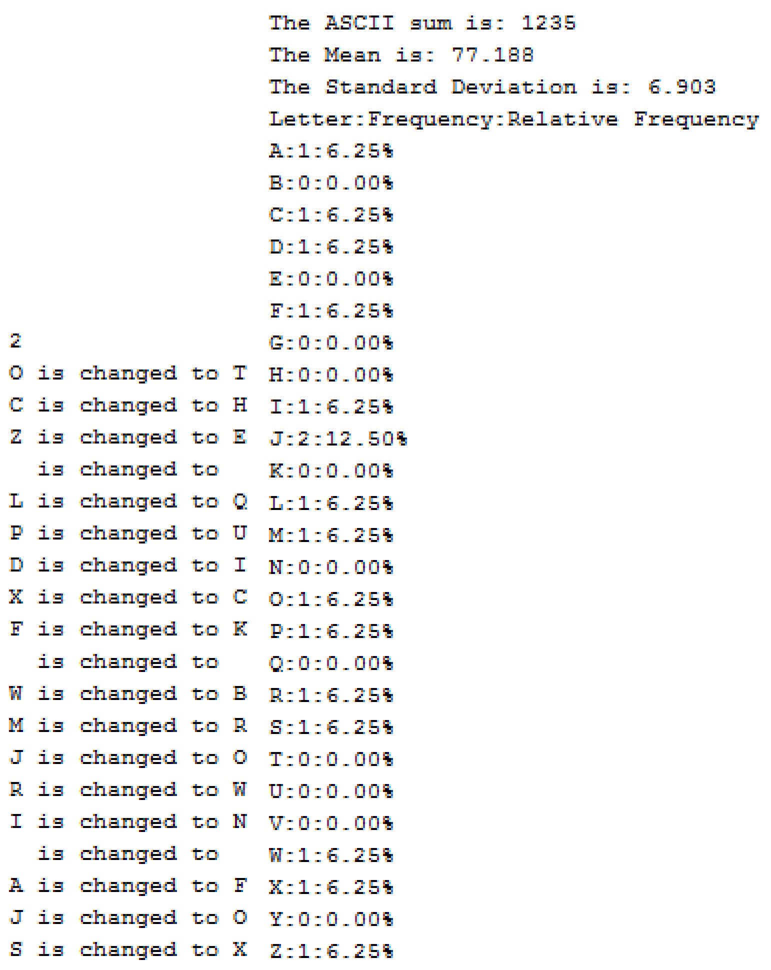
```
The ASCII sum is: 1233
The Mean is: 77.062
The Standard Deviation is: 7.386
Letter:Frequency:Relative Frequency
A:0:0.00%
B:1:6.25%
C:1:6.25%
D:0:0.00%
E:1:6.25%
F:1:6.25%
G:0:0.00%
H:1:6.25%
I:1:6.25%
J:0:0.00%
K:1:6.25%
L:0:0.00%
M:0:0.00%
N:1:6.25%
O:2:12.50%
P:0:0.00%
Q:1:6.25%
R:1:6.25%
S:0:0.00%
T:1:6.25%
U:1:6.25%
V:0:0.00%
W:1:6.25%
X:1:6.25%
Y:0:0.00%
Z:0:0.00%
```

The letters that are changed appear and are saved to a text file named “ciphertext.txt”. The requirements of the calculations are fulfilled and the frequency and relative frequency are formatted so that they are visible.

Sum = 79 + 67 + 90 + 76 + 80 + 68 + 88 + 70 + 87 + 77 + 74 + 82 + 73 + 65 + 74 + 83 = 1233
Average = Sum/# Characters = 1233/16 = 77.0625

Standard Deviation = $\sqrt{\frac{n\sum x^2 - (\sum x)^2}{n^2}}$. = root (((16*95891) – ($1233^2$))/($16^2$)) = 7.386378934

Now let’s pick Decryption and see what happens. This will use the input file “ciphertext.txt”



```
                  The ASCII sum is: 1235
                  The Mean is: 77.188
                  The Standard Deviation is: 6.903
                  Letter:Frequency:Relative Frequency
                  A:1:6.25%
                  B:0:0.00%
                  C:1:6.25%
                  D:1:6.25%
                  E:0:0.00%
                  F:1:6.25%
2                 G:0:0.00%
O is changed to T H:0:0.00%
C is changed to H I:1:6.25%
Z is changed to E J:2:12.50%
  is changed to   K:0:0.00%
L is changed to Q L:1:6.25%
P is changed to U M:1:6.25%
D is changed to I N:0:0.00%
X is changed to C O:1:6.25%
F is changed to K P:1:6.25%
  is changed to   Q:0:0.00%
W is changed to B R:1:6.25%
M is changed to R S:1:6.25%
J is changed to O T:0:0.00%
R is changed to W U:0:0.00%
I is changed to N V:0:0.00%
  is changed to   W:1:6.25%
A is changed to F X:1:6.25%
J is changed to O Y:0:0.00%
S is changed to X Z:1:6.25%
```

Evidently, with a shift of 99 (the last 2 digits of my student number), the decryption does indeed work, and changes the encrypted message back to what it originally was: “THE QUICK BROWN FOX”. This message is written to the file “messagetext.txt”. Also, all the calculations are fulfilled and the frequency and relative frequencies are shown.

## Conclusion:

The program that was implemented was somewhat easy yet also difficult, and the bonus option of cracking a cipher with an unknown key was not implemented. The program was coded with C language syntax, but could be easily converted to Java syntax, because most of the material is logic. This program incorporates most of the logical operations such as loops, conditional statements, single-dimension arrays, functions and structures for a well-rounded and efficient program. The program could be made more efficient by using multi-dimensional arrays, but that would be nearing the limitations of C. The use of multiple functions being called and structures being used takes into account the importance of object-oriented programming and how efficient it can make any logical system.

The program fulfilled all the requirements that were set out for it and used a minimal amount of RAM, therefore showing that it was an efficient and light-weight program that could be used by anyone. Cryptography is very important in the real world and it is used all over the place, from routers to signals to military use.

## References:

**[1]** T. E. Doyle. Principles of Programming 2SH4 2012 Laboratory Manual, McMaster University 2012.

**[2]** B. Schneier. (May 26, 1999) The Solitaire Encryption Algorithm. [Online].
Available: http://www.schneier.com/solitaire.html

**[3]** (January 24, 2006) A Brief History of Cryptography. [Online].
Available: http://www.cypher.com.au/crypto_history.htm

## Appendices:

```
/*
 * File:   main.c
 * Author: Khalid
 * SN: 1147299
 * MACID: asadk
 * Created on October 15, 2012, 4:24 PM
 */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int shift = 99; // create global variable for key (last 2 digits of SN)
double ocurrence_sum = 0;
/*
 *
 */
typedef struct ciphertext_analysis_t{ // structure
   double ascii_sum;
   double mean;
   double standard_deviation;// define all variables as doubles
   double frequency[26];
   double relative_frequency[];
};// don't declare a structure here

double average(double sum, double count);// calculate average
double std_dev(double n, double sum1, double sum2);// standard deviation
char encrypt(char plaintext);// encryption function
char decrypt(char ciphertext);// decryption function
void printer(struct ciphertext_analysis_t cat); // function to print values

int main(int argc, char** argv) {
   FILE *input, *output; // declare pointers for file accessing
   int option = 4, check=0;
   double i=0, sum1=0;
   char cipher, final;

   struct ciphertext_analysis_t cat; // create a structure to be used

   int k;
   for (k=0; k<26; k++){// set all values inside the array to 0
      cat.frequency[k]=0;
   }
```

```
printf("ENCRYPTION AND DECRYPTION PROGRAM BY KHALID ASAD\n");

while (option == 4){ // goes on until user chooses to encrypt or decrypt
    printf("What would you like to do?\n1.Encrypt a code\n2.Decrypt a code\n");
    scanf("%d",&option);
    if (option == 1){// encryption
        check = 1;
        input = fopen("plaintext.txt","r");
        output = fopen("ciphertext.txt","w");
    }
    else if (option == 2){// decryption
        check = 2;
        input = fopen("ciphertext.txt","r");
        output = fopen("messagetext.txt","w");
    }
    else{
        option = 4; // error message
        printf("INCORRECT OPTION! PLEASE TRY AGAIN!\n");
    }

    if (check ==1 || check ==2){ // for encryption/decryption
        fscanf(input, "%c", &cipher);// read the next character
        while(!feof(input)){// until end of the file being read
            printf("%c is changed to ",cipher);
            if ((cipher < 'A') || (cipher > 'Z') ){// for non-capital-letter
                final = cipher; // leave character the same
            }
            else{ // if char is between A and Z
                cat.frequency[cipher-65]++;// add the frequency of this char
                ocurrence_sum++;// counter for occurrences
                if (check==1){// send to encryption function
                    final = encrypt(cipher);
                }
                else if (check==2){// send to decryption function
                    final = decrypt(cipher);
                }
                cat.ascii_sum+=final;// add character to sum
                i++;
                sum1 += pow(final,2);// add square character
            }
            fprintf(output, "%c", final);
            printf("%c\n", final); // print letters that are being changed
            fscanf(input, "%c", &cipher);
        }
        fclose(input);
        fclose(output);// close files
```

```
            cat.mean = average(cat.ascii_sum,i);// calculate average
            cat.standard_deviation = std_dev(i,sum1,pow(cat.ascii_sum,2));
            printer(cat);// send the structure to print out all variables
        }

    }
    return (EXIT_SUCCESS);
}

double average(double sum, double count){// calculates average
    return sum/count;
}

double std_dev(double n, double sum1, double sum2){// standard deviation
    return(sqrt ( (n*sum1 - sum2)/pow(n,2) ) );
}

char encrypt(char plaintext){// encryption
    return (plaintext - 65 + shift)%26 + 65;
}

char decrypt(char ciphertext){// decryption
    return ((26 - (shift-(ciphertext - 65))%26)%26) + 65;
}

void printer(struct ciphertext_analysis_t cat){// print all values
    printf("\nThe ASCII sum is: %.0f\n", cat.ascii_sum);
    printf("The Mean is: %.3f\n", cat.mean);
    printf("The Standard Deviation is: %.3f\n", cat.standard_deviation);
    printf("Letter:Frequency:Relative Frequency\n");
    int i;
    for (i =0; i<26; i++){// print both types of frequencies
        cat.relative_frequency[i]= (cat.frequency[i]/ocurrence_sum)*100;
        printf("%c:%.0f:%.2f%%\n",i+65,cat.frequency[i],cat.relative_frequency[i]);
    }
}
```