Lab 3

Substitution Cipher

Server Team

Karl Miller, Paul Shriner, Christian Messmer

# Compilation

# Screenshots

To compile execute the program simply run the make command in the directory that the zip file is extracted to.

A screen shot of a computer

Description automatically generated

The cipher module contains all functions pertaining the creation of a cipher as well as the encryption of a string.

Flex String is used to dynamically create a string size to be able to accept a string of any size.

Colors simply contains VTE commands to color the text of the program.

Main runs the program and uses the modules to run a basic encryption program using the randomized substitution cipher.

Screenshots of Program Running

# Source Code

## Cipher.h

/\*

    Class: ECET 4640-002

    Assignment: Lab Assignment 3

    Authors: Christian Messmer, Karl Miller, Paul Shriner

    Cipher.h: Function prototypes for Cipher.c

\*/

#ifndef Cipher\_h

#define Cipher\_h

/\*\*

    @brief Generates a random cipher.

    Modular, intended to be reusable.

    @details The array is first filled with the characters between start and end. Then the array is traversed. Each element is swapped with some random other element. Each element is swapped at least once.

    @param cipher The cipher to fill.

    @param start The character to start.

    @param end The character the cipher will end on (inclusive).

    @attention Cipher is at least end-start in length.

    @attention Mutates: Fills cipher randomly with characters between start and end

\*/

void GenerateCipher(char \*cipher, char start, char end);

/\*\*

    @brief Prints the cipher.

    Modular, intended to be reusable.

    @details Prints the cipher in a series of columns describing what each character will be transformed into.

    Uses colors.h.

    @param cipher The cipher to print.

    @param start The character started on cipher

    @param length The length of the cipher

\*/

void PrintCipher(char \*cipher, char start, char length);

/\*\*

    @brief Encrypts the given string

    Modular, intended to be reusable.

    @details Replaces the string in place, mutating it. Anything out of bounds of the cipher will not be encrypted and will stay as its original character.

    @attention mutatates: encypts the string in place, destroying the original characters

    @param string String to ecrypt.

    @param length Length of `string`

    @param cipher Cipher to use for encrypting the string. Must be (start-end)+1 in size.

    @param start The first character the cipher uses

    @param end The last character the ciper uses

\*/

void EncryptString(char \*string, int length, char \*cipher, char start, char end);

#endif

## Cipher.c

/\*

    Class: ECET 4640-002

    Assignment: Lab Assignment 3

    Authors: Christian Messmer, Karl Miller, Paul Shriner

    Cipher.c: Functions used for generating the cipher, printing it out, and encrypting a string.

\*/

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include "colors.h"

/\*\*

    @private

    @param array The array to fill.

    @param start The character to start at.

    @param end The character to end at.

    Fills a character array with sequential values

    Used by GenerateCipher

    @attention precondition: array is at least end-start length

    @attention precondition: start < end

\*/

void FillArraySequential(char \*array, char start, char end)

{

    char length = end - start + 1;

    char i;

    for (i = start; i <= end; i++)

    {

        array[i - start] = i;

    }

}

// See Cipher.h for header comments

void GenerateCipher(char \*cipher, char start, char end)

{

    time\_t t;

    srand((unsigned)time(&t));

    FillArraySequential(cipher, start, end);

    char length = end - start + 1;

    char hold, swap\_index, i;

    for (i = 0; i < length; i++)

    {

        swap\_index = rand() % length;

        hold = cipher[swap\_index];

        cipher[swap\_index] = cipher[i];

        cipher[i] = hold;

    }

}

// See Cipher.h for header comments

void PrintCipher(char \*cipher, char start, char length)

{

    printf("\nCipher:\n");

    char i;

    for (i = 0; i < length; i++)

    {

        printf("%s\'%s%c%s\'%s \u00BB %s\'%s%c%s\'%s      ", COLOR\_GRAY, COLOR\_RED, i + start, COLOR\_GRAY, COLOR\_RESET, COLOR\_GRAY, COLOR\_BLUE, cipher[i], COLOR\_GRAY, COLOR\_RESET);

        if ((i + 1) % 5 == 0)

        {

            printf("\n");

        }

    }

    printf("\n");

}

// See Cipher.h for header comments

void EncryptString(char \*string, int length, char \*cipher, char start, char end)

{

    char cipher\_l = end - start + 1;

    int i;

    for (i = 0; i < length; i++)

    {

        if (!(string[i] - start > end || string[i] < start))

        {

            string[i] = cipher[string[i] - start];

        }

    }

}

## FlexString.h

/\*

    Class: ECET 4640-002

    Assignment: Lab Assignment 3

    Authors: Christian Messmer, Karl Miller, Paul Shriner

    FlexString.h: Function prototypes for FlexString.c

\*/

#ifndef FlexString\_c

#define FlexString\_c

/\*

    FlexString is used to create a heap-allocated string that resizes as necessary for longer input strings.

\*/

#include <stdio.h>  // for reading input

#include <stdlib.h> // for memory allocation

/\*\*

    Fills a malloced string with input, reallocating if necessary.

    Called by FlexString

    @param string A char \*\*; it points to the string that will be populated.

    @param capacity A pointer to the capacity desired of the string. If 0 or less, it will be set to 11.

    @param file A pointer to a file. Can be stdin if reading from the console.

    @attention Precondition: String has already been malloced.

    @attention Mutates: May reallocate and change value pointed to by string.

    @attention Mutates: May change value pointed to by capacity.

    @return The number of characters read (not including new line, and null-terminator)

\*/

int FlexString\_Read(char \*\*string, size\_t \*capacity, FILE \*file);

/\*\*

    @brief Creates and fills a malloced string. Then calls FlexString\_Read to read from input, resizing as necessary.

    Called by main.

    @param string A pointer to a char\*, referencing the string that will be populated.

    @param capacity A pointer to the capacity desired of the string. If 0 or less, it will be set to 11.

    @param file A pointer to a file. Can be stdin if reading from the console.

    @attention Mutates: May reallocate and change value pointed to by string.

    @attention Mutates: May change value pointed to by capacity.

    @return The size of characters read, excluding the null-terminator and new-line (if present)

\*/

int FlexString(char \*\*string, size\_t \*capacity, FILE \*file);

#endif

## FlexString.c

/\*

    Class: ECET 4640-002

    Assignment: Lab Assignment 3

    Authors: Christian Messmer, Karl Miller, Paul Shriner

    FlexString.c: Functions for reading in a string.

\*/

#include "FlexString.h"

// See Cipher.h for header comments

int FlexString\_Read(char \*\*string, size\_t \*capacity, FILE \*file)

{

    int i = 0;

    char ch = '?';

    do

    {

        ch = getc(file);

        if (i >= \*capacity - 1)

        {

            \*capacity \*= 2;

            \*string = realloc(\*string, \*capacity \* sizeof(char));

        }

        (\*string)[i] = ch;

        i++;

    } while (ch != '\n' && ch != '\0');

    (\*string)[i - 1] = '\0';

    return i - 1;

}

// See Cipher.h for header comments

int FlexString(char \*\*string, size\_t \*capacity, FILE \*file)

{

    if (\*capacity <= 0)

    {

        \*capacity = 11;

    }

    \*string = malloc(\*capacity \* sizeof(char));

    return FlexString\_Read(string, capacity, file);

}

## color.h

/\*

    Class: ECET 4640-002

    Assignment: Lab Assignment 3

    Authors: Christian Messmer, Karl Miller, Paul Shriner

    colors.h: Define color macros for use with printing text to the console

\*/

#ifndef colors\_h

#define colors\_h

/\*

    Karl's magic color macros.

    These use Virtual Terminal escape sequences to trigger color changes on the console when printed.

    See the [man page](https://www.man7.org/linux/man-pages/man4/console\_codes.4.html) for more information.

\*/

#define COLOR\_RED "\e[38;2;255;75;75m"

#define COLOR\_BLUE "\e[38;2;0;240;240m"

#define COLOR\_GREEN "\e[38;2;0;240;0m"

#define COLOR\_YELLOW "\e[38;2;255;255;0m"

#define COLOR\_GRAY "\e[38;2;224;224;224m"

#define COLOR\_BOLD "\e[1m"

#define COLOR\_RESET "\e[0m"

#endif

## main.c

/\*

    Class: ECET 4640-002

    Assignment: Lab Assignment 3

    Authors: Christian Messmer, Karl Miller, Paul Shriner

    main.c: Entry point of the program.

\*/

#include "FlexString.h"

#include "Cipher.h"

#include "colors.h"

#define CIPHER\_START 32

#define CIPHER\_END 126

/\*\*

   Entry point of the program.

   Generates a randomized cipher, then prompts the user for a string to encrypt. Encrypts that string, then prints the encrpyted string. Finally, it prints the cipher.

   Calls GenerateCipher, FlexString, EncryptString, and PrintCipher.

   Uses macros from colors.h

\*/

int main()

{

    printf("%s\nSubstitution Cipher\n", COLOR\_YELLOW);

    printf("%sBy Karl, Christian, and Paul%s\n", COLOR\_GRAY, COLOR\_RESET);

    // reserve the cipher of the needed length

    char cipher[CIPHER\_END - CIPHER\_START + 1];

    // Generate the randomized cipher.

    GenerateCipher(cipher, CIPHER\_START, CIPHER\_END);

    // Get a flexible string from user input

    char \*string;

    size\_t capacity;

    printf("\nEnter a string: ");

    int n\_chars = FlexString(&string, &capacity, stdin);

    printf("\nEncrypting: %s%s%s\n", COLOR\_RED, string, COLOR\_RESET);

    // Encrypt the string

    EncryptString(string, n\_chars, cipher, CIPHER\_START, CIPHER\_END);

    // Print the result

    printf("    Result: %s%s%s%s\n", COLOR\_BLUE, COLOR\_BOLD, string, COLOR\_RESET);

    // Print the cipher.

    PrintCipher(cipher, CIPHER\_START, CIPHER\_END - CIPHER\_START + 1);

    printf("\n");

    return 0;

}

# Contributions

Karl re-used and modified the color codes and flex string from previous assignments. He contributed to writing this document.

All group members worked together and simultaneously using VSCode LiveShare to write the initial version of the cipher.

…