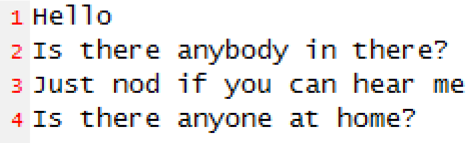
Working with this program is pretty simple. You should prepare input file contain text to encode/decode. Copy this file to application folder. Next create in your mind “secret word” used to encoding or decoding. “Secret word” should be confidential and known only by you and person authorized to decode file. Secret word can contain only letters, size of letters is not important (spaces are forbidden too).

Remember that you can cancel operation in every moment by entering “cancel” in any prompt.

Now you can run application. Input name of file to convert (with extension, like input.txt, file should exist in application folder) and tap enter. Next you will be asked for “secret word” (this should satisfying conditions from last paragraph). Next you will be asked for mode of processing: encrypting or decrypting. When you enter proper option and confirm that by enter, converting will start. After processing you will be asked for name of file where output will be saved. After this step you can find output of processing in folder of application.

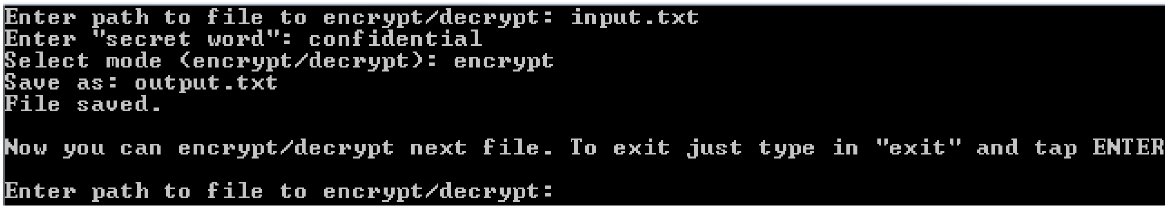
In this moment you can repeat all previous operation for other file or just type in “exit” to exit form application.

On the screenshoots you can see example of using of program.



```
Hello
Is there anybody in there?
Just nod if you can hear me
Is there anyone at home?
```

Input.txt content

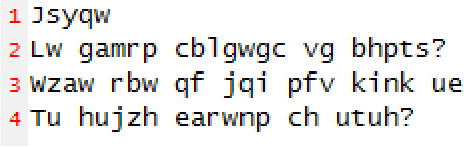


```
Enter path to file to encrypt/decrypt: input.txt
Enter "secret word": confidential
Select mode (encrypt/decrypt): encrypt
Save as: output.txt
File saved.

Now you can encrypt/decrypt next file. To exit just type in "exit" and tap ENTER

Enter path to file to encrypt/decrypt:
```

Application window with example of data



```
Jsyqw
Lw gamrp cblgwgc vg bhpts?
Wzaw rbw qf jqi pfv kink ue
Tu hujzh earwnp ch utuh?
```

Output.txt content

Functions used in application:

`int main()`

Description:

Main function

Input:

None

Output:

`OK` – all is right

`int isCharLetter(char c)`

Description:

Function to check whether the character is a letter

Input:

`char c` - char to check

Output:

`ERROR` - char is non letter

`OK` - char is letter

`int isUppercase(char c)`

Description:

Function to check whether char is uppercase

Input:

`char c` - char to check

Output:

`OK` - char is uppercase

`ERROR` - char is lowercase

`void toLowercase(char *string)`

Description:

Function converting string to lowercase

Input:

`char *string` - pointer to first element of string to check

Output:

None

`int clearTmp()`

Description:

Function to clearing tmp file

Input:

None

Output:

`OK` - tmp file cleared

`ERROR` - user have no rights to write in program folder or any other error with file handling occurred

---

`int dumpTmp(char *buffer)`

Description:

Function dumping content of buffer to tmp file

Input:

`char *buffer` - pointer to first element of buffer string

Output:

`OK` - dump complete

`ERROR` - problem with file handling occurred

```
void clearBuffer(char *buffer)
```

Description:

Function clearing content of buffer

Input:

`char *buffer` - pointer to first element of buffer string

Output:

None

```
int saveTmpToFile(char *filename)
```

Description:

Function function copying content of tmp file to user-defined path

Input:

`char *filename` - pointer to firest element to file path string

Output:

`OK` - file saved without any error

`ERROR` - error with file handling occured

```
int saveTmpToFileGUI()
```

Description:

Function creating GUI for saveTmpToFile function

Input:

None

Output:

`OPERATIONCANCEL` - operation canceled by user

`OK` - all is right

```
int encryptDecrypt(char actualChar, char *secretWord, char *buffer, int &i, char *mode)
```

Description:

Function encrypting/decrypting char with respect to secret word

Input:

`char actualChar` - char to encrypt/decrypt

`char *secretWord` - pointer to first element of secret word string

`char *buffer` - pointer to first element of global buffer string

`int &i` - reference to global counter value

`char *mode` - pointer to first element of mode string descrybing mode (encrypting/descrypting)

Output:

`ERROR` - any error occured

`OK` - all is right

```
int mainGUI()
```

Description:

Function creating GUI

Input:

None

Output:

`OPERATIONCANCEL` - operation canceled by user
`EXIT` - user enter "exit" command
`ERROR` - error occured
`OK` - all is right