**Assignment 5 (Encryption Decryption using Playfair Cipher)**

|  |
| --- |
| #include<stdio.h> |
| #include <stdlib.h> |  | |
| #include <string.h> | |
|  | |
| #define SIZE 100 | |
|  | |
| // Function to encrypt using the Playfair Cipher | |
| void PlayfairCrypt(char str[], char keystr[]) | |
| { | |
| char keyT[5][5], ks, ps; | |
|  | |
| // Key | |
| ks = strlen(keystr); | |
| // Plaintext | |
| ps = strlen(str); | |
|  | |
| // Function to convert the string to uppercase | |
| // Can also use the library function toUpper here, but a function was written for better understanding of ascii values. | |
| void toUpperCase(char encrypt[], int ps) | |
| { | |
| int i; | |
| for (i = 0; i < ps; i++) { | |
| if (encrypt[i] > 96 && encrypt[i] < 123) | |
| encrypt[i] -= 32; | |
| } | |
| } | |
|  | |
| // Function to remove all spaces in a string | |
| int removeSpaces(char\* plain, int ps) | |
| { | |
| int i, count = 0; | |
| for (i = 0; i < ps; i++) | |
| if (plain[i] != ' ') | |
| plain[count++] = plain[i]; | |
| plain[count] = '\0'; | |
| return count; | |
| } | |
|  | |
| // Function to generate the 5x5 key square | |
| void generateKeyTable(char keystr[], int ks, char keyT[5][5]) | |
| { | |
| int i, j, k, flag = 0, \*dicty; | |
|  | |
| // a 26 character hashmap to store count of the alphabet | |
| dicty = (int\*)calloc(26, sizeof(int)); | |
| for (i = 0; i < ks; i++) { | |
| if (keystr[i] != 'j') | |
| dicty[keystr[i] - 97] = 2; | |
| } | |
|  | |
| dicty['j' - 97] = 1; | |
|  | |
| i = 0; | |
| j = 0; | |
|  | |
| for (k = 0; k < ks; k++) { | |
| if (dicty[keystr[k] - 97] == 2) { | |
| dicty[keystr[k] - 97] -= 1; | |
| keyT[i][j] = keystr[k]; | |
| j++; | |
| if (j == 5) { | |
| i++; | |
| j = 0; | |
| } | |
| } | |
| } | |
|  | |
| for (k = 0; k < 26; k++) { | |
| if (dicty[k] == 0) { | |
| keyT[i][j] = (char)(k + 97); | |
| j++; | |
| if (j == 5) { | |
| i++; | |
| j = 0; | |
| } | |
| } | |
| } | |
| } | |
|  | |
| // Function to make the plain text length even, and make pairs unidentical. | |
| int prepare(char str[], int ptrs) | |
| { | |
| int i, j, subs\_s = ptrs; | |
| for (i = 0; i < subs\_s; i += 2) { | |
| if(str[i]==str[i+1]){ | |
| for(j=subs\_s; j>i+1; j--){ | |
| str[j]=str[j-1]; | |
| } | |
| str[i+1]='x'; | |
| subs\_s+=1; | |
| } | |
| } | |
| str[subs\_s]='\0'; | |
|  | |
| if (subs\_s % 2 != 0) { | |
| str[subs\_s++] = 'z'; | |
| str[subs\_s] = '\0'; | |
| } | |
| return subs\_s; | |
| } | |
|  | |
| // Function to search for the characters of a digraph in the key square and return their position | |
| void search(char keyT[5][5], char a, char b, int arr[]) | |
| { | |
| int i, j; | |
|  | |
| if (a == 'j') | |
| a = 'i'; | |
| else if (b == 'j') | |
| b = 'i'; | |
|  | |
| for (i = 0; i < 5; i++) { | |
|  | |
| for (j = 0; j < 5; j++) { | |
|  | |
| if (keyT[i][j] == a) { | |
| arr[0] = i; | |
| arr[1] = j; | |
| } | |
| else if (keyT[i][j] == b) { | |
| arr[2] = i; | |
| arr[3] = j; | |
| } | |
| } | |
| } | |
| } | |
|  | |
| // Function for performing the encryption | |
| void encrypt(char str[], char keyT[5][5], int ps) | |
| { | |
| int i, a[4]; | |
|  | |
| for(i=0; i<ps; i+=2){ | |
|  | |
| search(keyT, str[i], str[i + 1], a); | |
|  | |
| if (a[0] == a[2]) { | |
| str[i] = keyT[a[0]][(a[1] + 1)%5]; | |
| str[i + 1] = keyT[a[0]][(a[3] + 1)%5]; | |
| } | |
| else if (a[1] == a[3]) { | |
| str[i] = keyT[(a[0] + 1)%5][a[1]]; | |
| str[i + 1] = keyT[(a[2] + 1)%5][a[1]]; | |
| } | |
| else { | |
| str[i] = keyT[a[0]][a[3]]; | |
| str[i + 1] = keyT[a[2]][a[1]]; | |
| } | |
| } | |
| } | |
|  | |
| ks = removeSpaces(keystr, ks); | |
| ps = removeSpaces(str, ps); | |
| ps = prepare(str, ps); | |
| generateKeyTable(keystr, ks, keyT); | |
| encrypt(str, keyT, ps); | |
| toUpperCase(str, ps); | |
| //cipher text - printed in upper case letters | |
| printf("Cipher text: %s\n", str); | |
| } | |
|  | |
| // Driver code | |
| int main() | |
| { | |
| char str[SIZE], keystr[SIZE]; | |
|  | |
| //Key used - to be entered in lower case letters | |
| printf("Enter the key: "); | |
| scanf("%[^\n]s", &keystr); | |
| printf("Key text: %s\n", keystr); | |
|  | |
| printf("Enter the plaintext: "); | |
| scanf("\n"); | |
| scanf("%[^\n]s", &str); | |
| printf("Plain text: %s\n", str); | |
|  | |
| //Calling the PlayfairCrypt function | |
| PlayfairCrypt(str, keystr); | |
|  | |
| return 0; | |
| } | |