**Write a C program for Caesar cipher involves replacing each letter of the alphabet with the letter standing k places further down the alphabet, for k in the range 1 through 25.**

#include <stdio.h>

#include <ctype.h>

void encrypt(char text[], int shift) {

for (int i = 0; text[i] != '\0'; i++) {

if (isalpha(text[i])) {

char base = isupper(text[i]) ? 'A' : 'a';

text[i] = (text[i] - base + shift) % 26 + base;

}

}

}

void decrypt(char text[], int shift) {

for (int i = 0; text[i] != '\0'; i++) {

if (isalpha(text[i])) {

char base = isupper(text[i]) ? 'A' : 'a';

text[i] = (text[i] - base - shift + 26) % 26 + base;

}

}

}

int main() {

char text[1000];

int shift;

printf("Enter the message: ");

fgets(text, sizeof(text), stdin);

do {

printf("Enter shift value (1-25): ");

scanf("%d", &shift);

} while (shift < 1 || shift > 25);

encrypt(text, shift);

printf("Encrypted message: %s\n", text);

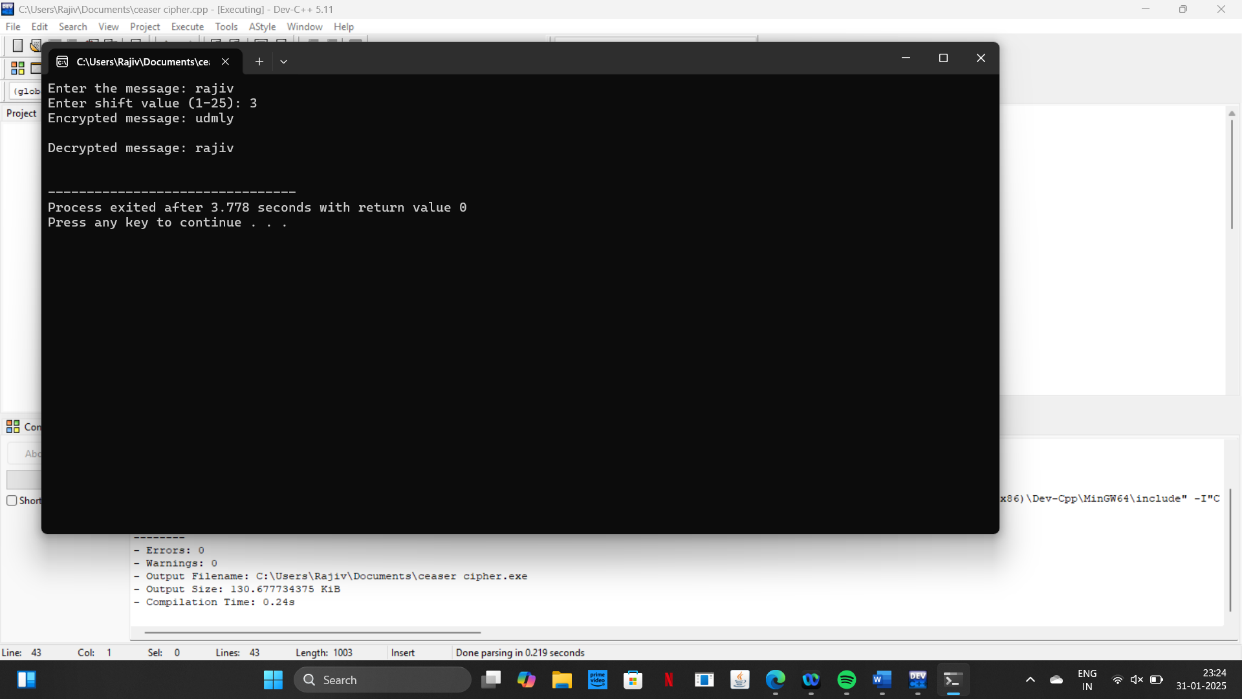
decrypt(text, shift);

printf("Decrypted message: %s\n", text);

return 0;

}

Output:



Write a C program for monoalphabetic substitution cipher maps a plaintext alphabet to a ciphertext alphabet, so that each letter of the plaintext alphabet maps to a single unique letter of the ciphertext alphabet.

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#define ALPHABET "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

#define CIPHER "QWERTYUIOPASDFGHJKLZXCVBNM" // Substitution key

void encrypt(char text[]) {

for (int i = 0; text[i] != '\0'; i++) {

if (isalpha(text[i])) {

char is\_upper = isupper(text[i]);

char letter = toupper(text[i]);

for (int j = 0; j < 26; j++) {

if (ALPHABET[j] == letter) {

text[i] = is\_upper ? CIPHER[j] : tolower(CIPHER[j]);

break;

}

}

}

}

}

void decrypt(char text[]) {

for (int i = 0; text[i] != '\0'; i++) {

if (isalpha(text[i])) {

char is\_upper = isupper(text[i]);

char letter = toupper(text[i]);

for (int j = 0; j < 26; j++) {

if (CIPHER[j] == letter) {

text[i] = is\_upper ? ALPHABET[j] : tolower(ALPHABET[j]);

break;

}

}

}

}

}

int main() {

char text[1000];

printf("Enter the message: ");

fgets(text, sizeof(text), stdin);

encrypt(text);

printf("Encrypted: %s\n", text);

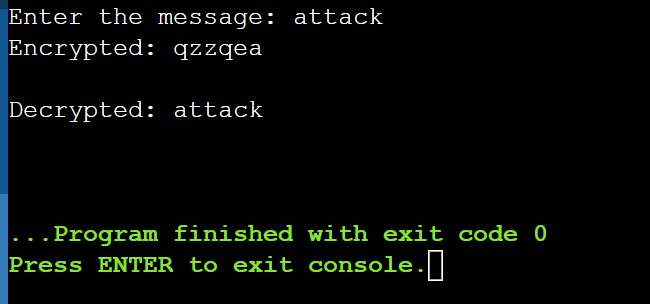
decrypt(text);

printf("Decrypted: %s\n", text);

return 0;

}

Output:



Write a C program for Playfair algorithm is based on the use of a 5 X 5 matrix of letters constructed using a keyword. Plaintext is encrypted two letters at a time using this matrix.

#include <stdio.h>

#include <ctype.h>

void encrypt(char text[], int shift) {

for (int i = 0; text[i] != '\0'; i++) {

if (isalpha(text[i])) {

char base = isupper(text[i]) ? 'A' : 'a';

text[i] = (text[i] - base + shift) % 26 + base;

}

}

}

void decrypt(char text[], int shift) {

for (int i = 0; text[i] != '\0'; i++) {

if (isalpha(text[i])) {

char base = isupper(text[i]) ? 'A' : 'a';

text[i] = (text[i] - base - shift + 26) % 26 + base;

}

}

}

int main() {

char text[1000];

int shift;

printf("Enter the message: ");

fgets(text, sizeof(text), stdin);

do {

printf("Enter shift value (1-25): ");

scanf("%d", &shift);

} while (shift < 1 || shift > 25);

encrypt(text, shift);

printf("Encrypted message: %s\n", text);

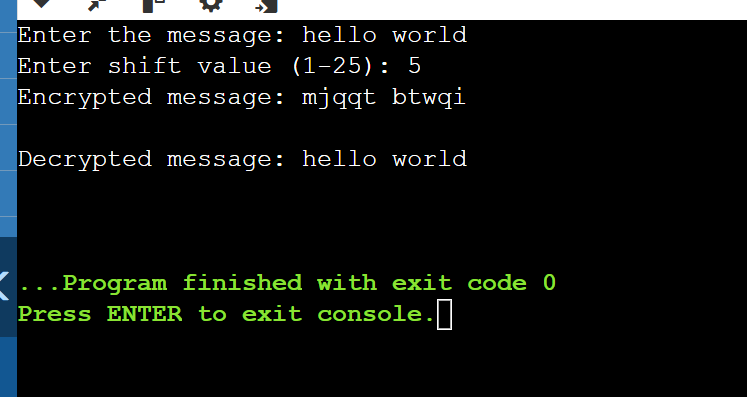
decrypt(text, shift);

printf("Decrypted message: %s\n", text);

return 0;

}

Output:



As you know, the most frequently occurring letter in English is e. Therefore, the first or second (or perhaps third?) most common character in the message is likely to stand for e. Also, e is often seen in pairs (e.g., meet, fleet, speed, seen, been, agree, etc.). Try to find a character in the ciphertext that decodes to e. 2. The most common word in English is “the.” Use this fact to guess the characters that stand for t and h. 3. Decipher the rest of the message by deducing additional words.

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#define MAX\_LEN 1000

void countFrequency(char \*ciphertext, int freq[26]) {

for (int i = 0; ciphertext[i] != '\0'; i++) {

if (isalpha(ciphertext[i])) {

freq[toupper(ciphertext[i]) - 'A']++;

}

}

}

void decryptMessage(char \*ciphertext, char mapping[26]) {

printf("\nDecrypted Message:\n");

for (int i = 0; ciphertext[i] != '\0'; i++) {

if (isalpha(ciphertext[i])) {

char original = toupper(ciphertext[i]);

char decrypted = mapping[original - 'A'];

if (islower(ciphertext[i])) {

decrypted = tolower(decrypted);

}

printf("%c", decrypted);

} else {

printf("%c", ciphertext[i]);

}

}

printf("\n");

}

int main() {

char ciphertext[MAX\_LEN];

int freq[26] = {0};

char mapping[26];

for (int i = 0; i < 26; i++) {

mapping[i] = 'A' + i;

}

printf("Enter the ciphertext: ");

fgets(ciphertext, MAX\_LEN, stdin);

countFrequency(ciphertext, freq);

printf("\nLetter Frequency:\n");

for (int i = 0; i < 26; i++) {

if (freq[i] > 0) {

printf("%c: %d\n", 'A' + i, freq[i]);

}

}

printf("\nEnter replacements (X Y to replace X with Y, type 0 0 to stop):\n");

char from, to;

while (1) {

printf("Replace: ");

scanf(" %c %c", &from, &to);

if (from == '0' && to == '0') break;

mapping[toupper(from) - 'A'] = toupper(to);

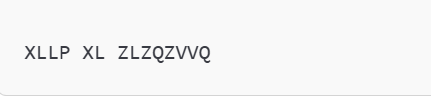
}

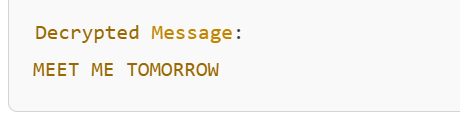
decryptMessage(ciphertext, mapping);

return 0;

}

Output:





Write a C program for monoalphabetic cipher is that both sender and receiver must commit the permuted cipher sequence to memory. A common technique for avoiding this is to use a keyword from which the cipher sequence can be generated. For example, using the keyword CIPHER, write out the keyword followed by unused letters in normal order and match this against the plaintext letters:

plain: a b c d e f g h i j k l m n o p q r s t u v w x y z

cipher: C I P H E R A B D F G J K L M N O Q S T U V W X Y Z

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#define ALPHABET\_SIZE 26

void generateCipherAlphabet(char \*keyword, char \*cipherAlphabet) {

char normalAlphabet[ALPHABET\_SIZE] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

int index = 0;

for (int i = 0; i < strlen(keyword); i++) {

if (!strchr(cipherAlphabet, toupper(keyword[i]))) {

cipherAlphabet[index++] = toupper(keyword[i]);

}

}

for (int i = 0; i < ALPHABET\_SIZE; i++) {

if (!strchr(cipherAlphabet, normalAlphabet[i])) {

cipherAlphabet[index++] = normalAlphabet[i];

}

}

cipherAlphabet[index] = '\0';

}

void encrypt(char \*plaintext, char \*cipherAlphabet, char \*normalAlphabet, char \*ciphertext) {

for (int i = 0; plaintext[i] != '\0'; i++) {

if (isalpha(plaintext[i])) {

char letter = toupper(plaintext[i]);

for (int j = 0; j < ALPHABET\_SIZE; j++) {

if (letter == normalAlphabet[j]) {

ciphertext[i] = cipherAlphabet[j];

break;

}

}

} else {

ciphertext[i] = plaintext[i];

}

}

ciphertext[strlen(plaintext)] = '\0';

}

int main() {

char keyword[] = "CIPHER";

char normalAlphabet[ALPHABET\_SIZE] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

char plaintext[100], ciphertext[100], cipherAlphabet[ALPHABET\_SIZE + 1];

generateCipherAlphabet(keyword, cipherAlphabet);

printf("Enter plaintext: ");

fgets(plaintext, sizeof(plaintext), stdin);

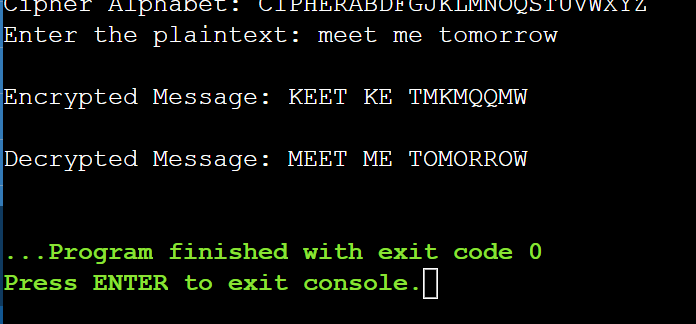
encrypt(plaintext, cipherAlphabet, normalAlphabet, ciphertext);

printf("Encrypted Message: %s\n", ciphertext);

return 0;

}

Output:



Write a C program for Playfair matrix:



Encrypt this message: Must see you over Cadogan West. Coming at once.

#include <stdio.h>

#include <string.h>

#include <ctype.h>

char matrix[5][5];

void prepareMatrix(const char \*key) {

int used[26] = {0}, x = 0, y = 0;

for (int i = 0; key[i] && x < 5; i++) {

char c = (key[i] == 'J') ? 'I' : toupper(key[i]);

if (isalpha(c) && !used[c - 'A']++) {

matrix[x][y++] = c;

if (y == 5) x++, y = 0;

}

}

for (char c = 'A'; x < 5 && c <= 'Z'; c++) {

if (c != 'J' && !used[c - 'A']++) {

matrix[x][y++] = c;

if (y == 5) x++, y = 0;

}

}

}

void findPosition(char c, int \*row, int \*col) {

for (int i = 0; i < 5; i++)

for (int j = 0; j < 5; j++)

if (matrix[i][j] == c) { \*row = i; \*col = j; return; }

}

void encrypt(char \*msg) {

for (int i = 0; msg[i]; i += 2) {

if (!msg[i + 1]) msg[i + 1] = 'X', msg[i + 2] = '\0';

int r1, c1, r2, c2;

findPosition(msg[i], &r1, &c1);

findPosition(msg[i + 1], &r2, &c2);

if (r1 == r2) msg[i] = matrix[r1][(c1 + 1) % 5], msg[i + 1] = matrix[r2][(c2 + 1) % 5];

else if (c1 == c2) msg[i] = matrix[(r1 + 1) % 5][c1], msg[i + 1] = matrix[(r2 + 1) % 5][c2];

else msg[i] = matrix[r1][c2], msg[i + 1] = matrix[r2][c1];

}

}

int main() {

char key[] = "ELARGDSTBCMFHIJKUNOPQVWXYZ", msg[] = "MUSTSEEYOUOVERCADOGANWESTX";

prepareMatrix(key);

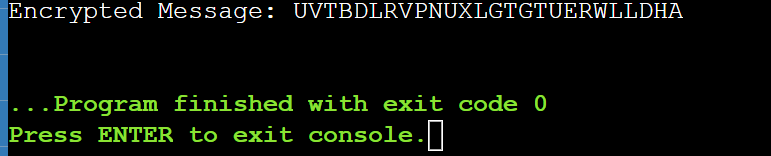
encrypt(msg);

printf("Encrypted Message: %s\n", msg);

return 0;

}

Output:



Write a C program to Encrypt the message “meet me at the usual place at ten rather than eight oclock” using the Hill cipher with the key. ( 9 4 ) (5 a. Show your calculations and the result. b. Show the calculations for the corresponding decryption of the ciphertext to recover the original plaintext

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#define MOD 26

void encrypt(const int key[2][2], char \*plaintext, char \*ciphertext) {

for (int i = 0; plaintext[i] && plaintext[i + 1]; i += 2) {

int x1 = toupper(plaintext[i]) - 'A';

int x2 = toupper(plaintext[i + 1]) - 'A';

ciphertext[i] = ((key[0][0] \* x1 + key[0][1] \* x2) % MOD) + 'A';

ciphertext[i + 1] = ((key[1][0] \* x1 + key[1][1] \* x2) % MOD) + 'A';

}

ciphertext[strlen(plaintext)] = '\0';

}

int main() {

int key[2][2] = {{9, 4}, {5, 7}};

char plaintext[] = "MEETMEATTHEUSUALPLACEATTENRATHERTHANEIGHTOCLOCK";

char ciphertext[100];

if (strlen(plaintext) % 2 != 0) strcat(plaintext, "X");

encrypt(key, plaintext, ciphertext);

printf("Plaintext: %s\n", plaintext);

printf("Ciphertext: %s\n", ciphertext);

return 0;

}

