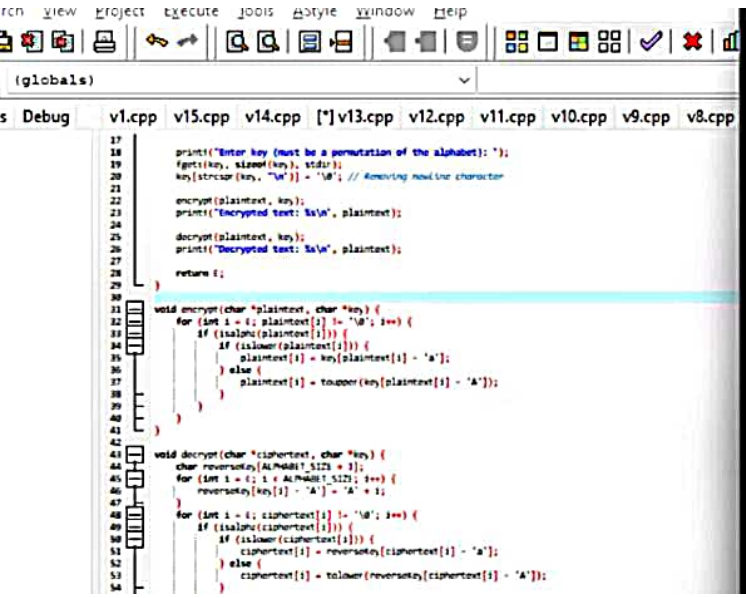
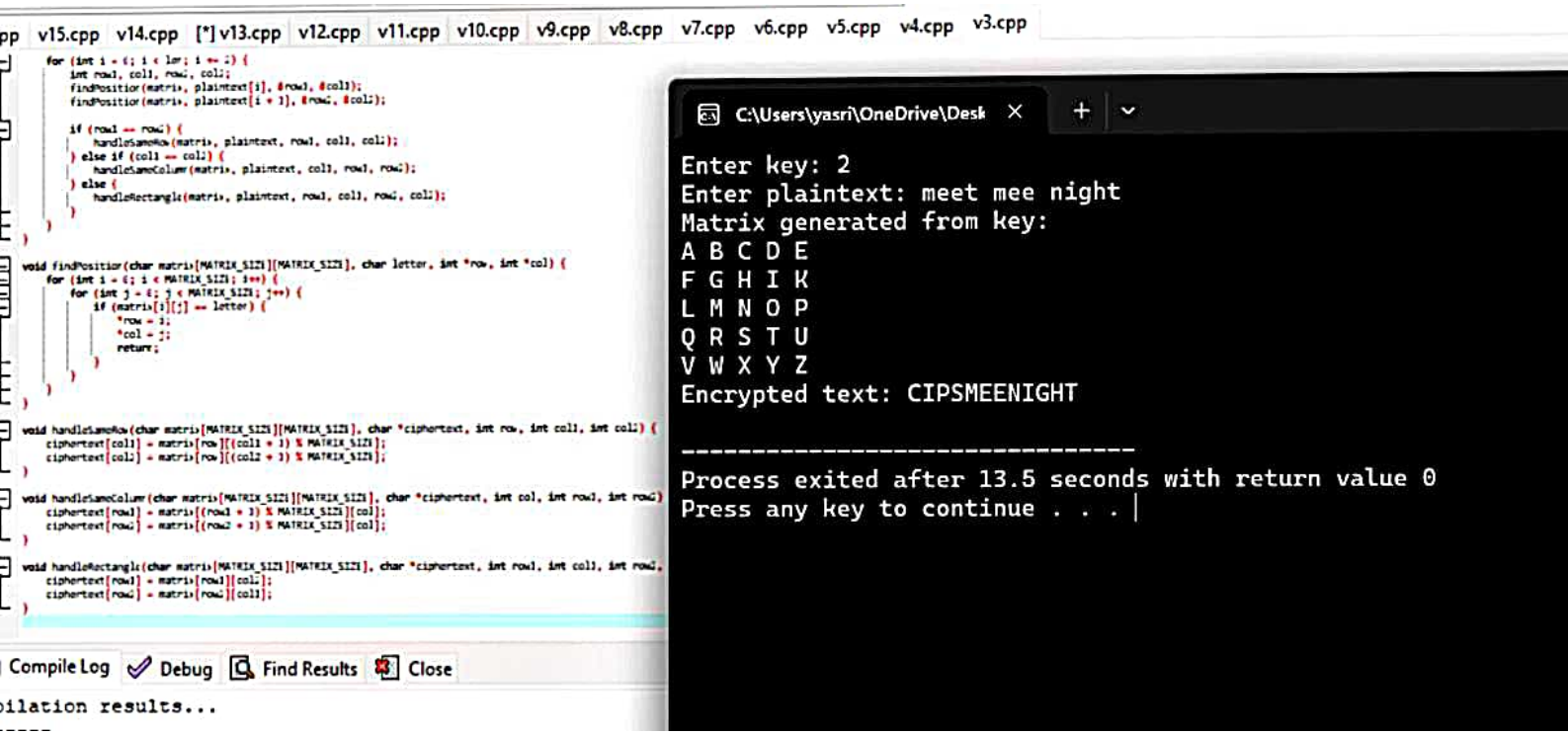
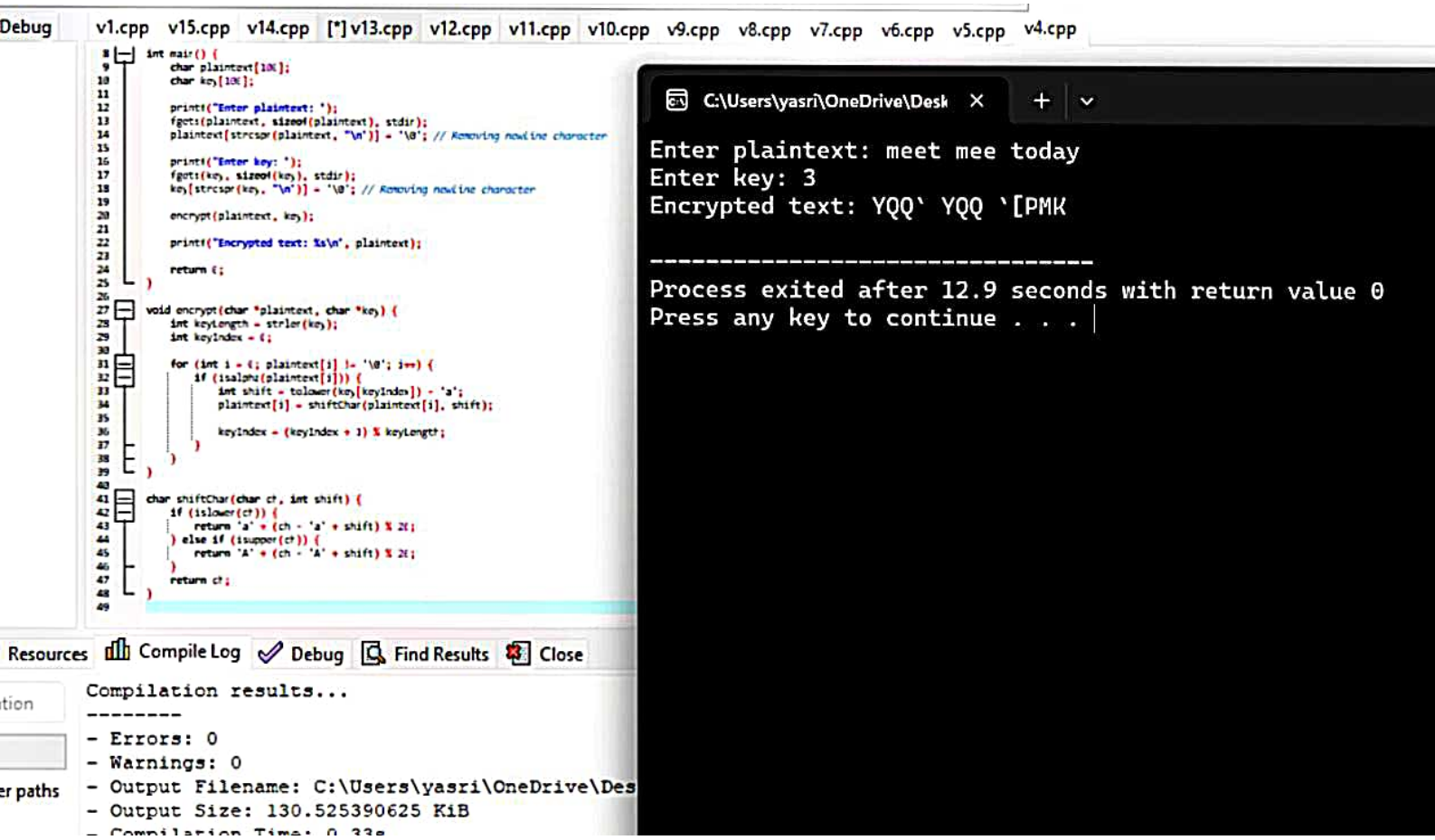
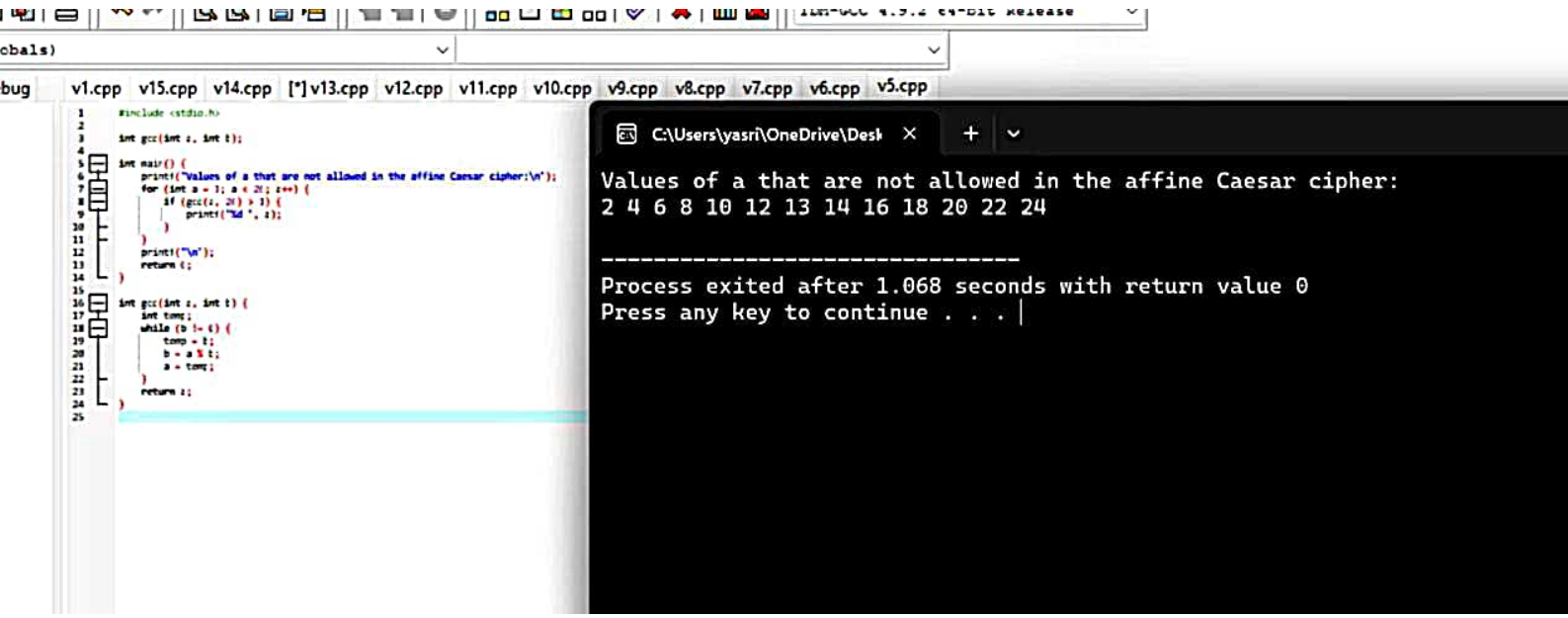


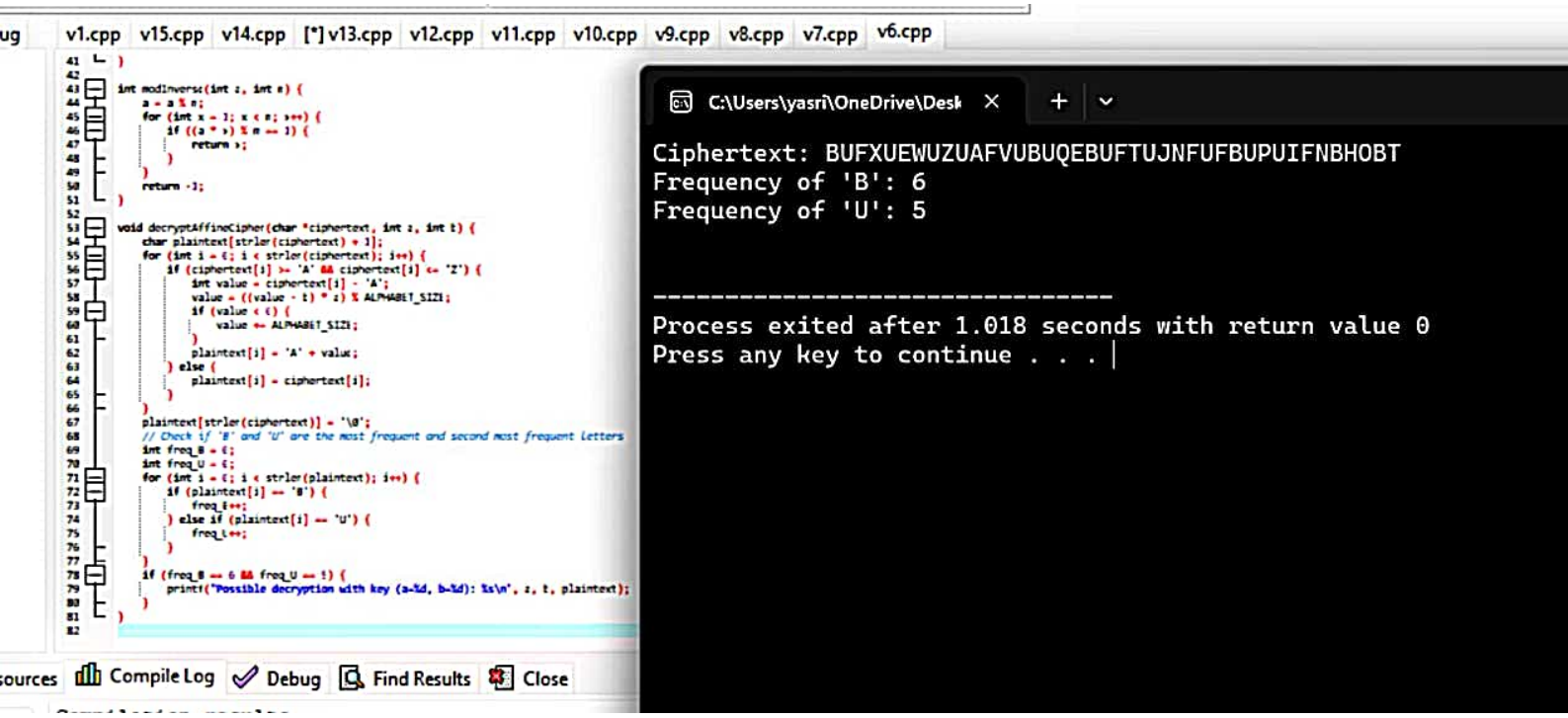

```
Enter plaintext: hello world
Enter key (must be a permutation of the alphabet):  ZEBRACDFGHIJ
KLMNOPQSTUVWXY
Encrypted text: DRIIL ULOIB
Decrypted text: hello world

--------------------------------
Process exited after 11.5 seconds with return value 0
Press any key to continue . . .
```

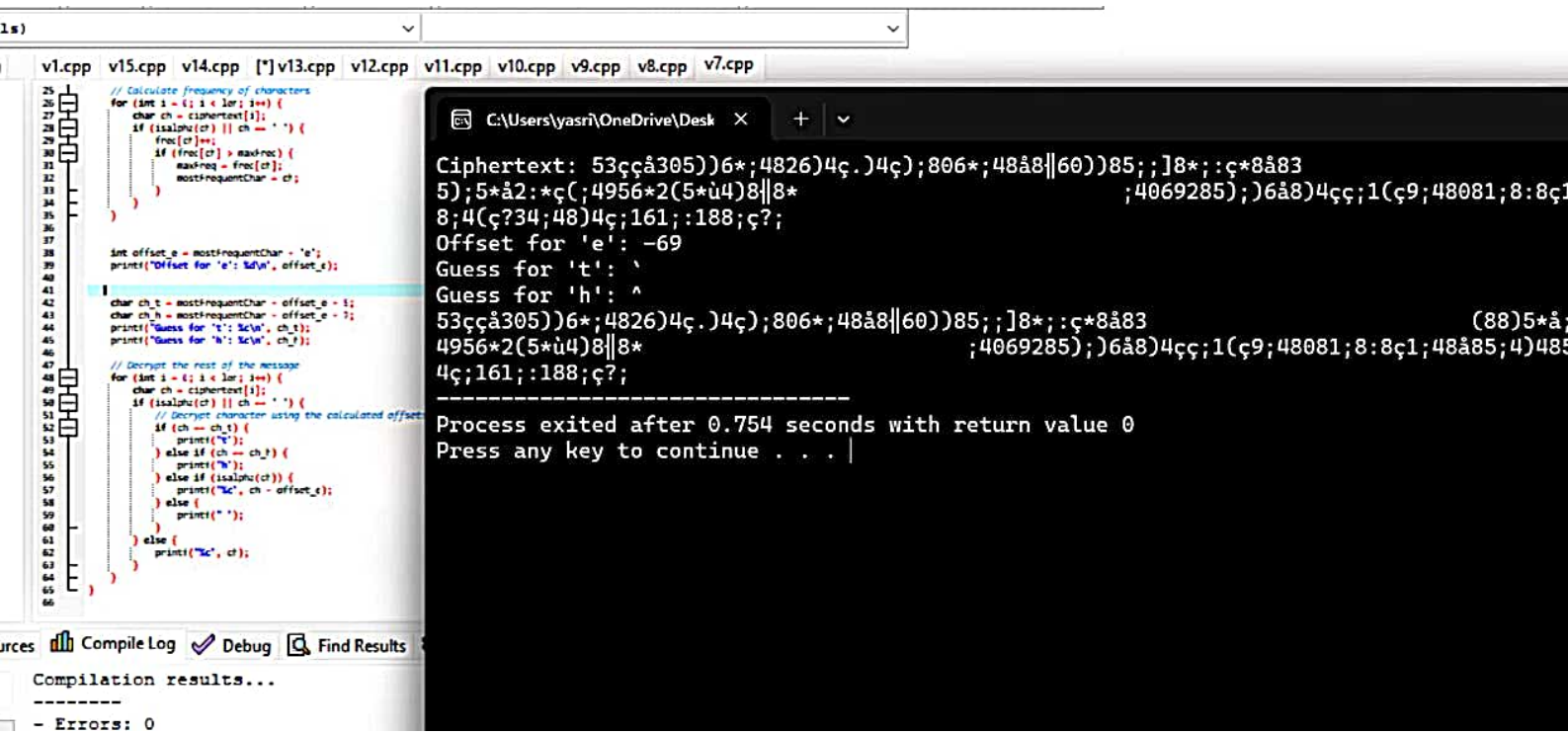

pp v15.cpp v14.cpp [*] v13.cpp v12.cpp v11.cpp v10.cpp v9.cpp v8.cpp v7.cpp v6.cpp v5.cpp v4.cpp v3.cpp
for (int i = 0; i < len; i += 2) {
int row1, col1, row2, col2;
findPosition(matrix, plaintext[i], &row1, &col1);
findPosition(matrix, plaintext[i + 1], &row2, &col2);
if (row1 == row2) {
handleSameRow(matrix, plaintext, row1, col1, col2);
} else if (col1 == col2) {
handleSameColumn(matrix, plaintext, col1, row1, row2);
} else {
handleRectangle(matrix, plaintext, row1, col1, row2, col2);
}
}
}
void findPosition(char matrix[MATRIX_SIZE][MATRIX_SIZE], char letter, int *row, int *col) {
for (int i = 0; i < MATRIX_SIZE; i++) {
for (int j = 0; j < MATRIX_SIZE; j++) {
if (matrix[i][j] == letter) {
*row = i;
*col = j;
return;
}
}
}
}
void handleSameRow(char matrix[MATRIX_SIZE][MATRIX_SIZE], char *ciphertext, int row, int col1, int col2) {
ciphertext[col1] = matrix[row][(col1 + 1) % MATRIX_SIZE];
ciphertext[col2] = matrix[row][(col2 + 1) % MATRIX_SIZE];
}
void handleSameColumn(char matrix[MATRIX_SIZE][MATRIX_SIZE], char *ciphertext, int col, int row1, int row2)
ciphertext[row1] = matrix[(row1 + 1) % MATRIX_SIZE][col];
ciphertext[row2] = matrix[(row2 + 1) % MATRIX_SIZE][col];
}
void handleRectangle(char matrix[MATRIX_SIZE][MATRIX_SIZE], char *ciphertext, int row1, int col1, int row2,
ciphertext[row1] = matrix[row1][col2];
ciphertext[row2] = matrix[row2][col1];
}
Compile Log
Debug
Find Results
Close
ilation results...
C:\Users\yasri\OneDrive\Desk
Enter key: 2
Enter plaintext: meet mee night
Matrix generated from key:
A B C D E
F G H I K
L M N O P
Q R S T U
V W X Y Z
Encrypted text: CIPSMEENIGHT
Process exited after 13.5 seconds with return value 0
Press any key to continue . . .


Debug
v1.cpp
v15.cpp
v14.cpp
[*] v13.cpp
v12.cpp
v11.cpp
v10.cpp
v9.cpp
v8.cpp
v7.cpp
v6.cpp
v5.cpp
v4.cpp
int main() {
char plaintext[100];
char key[100];
printf("Enter plaintext: ");
fgets(plaintext, sizeof(plaintext), stdin);
plaintext[strcspn(plaintext, "\n")] = '\0'; // Removing newline character
printf("Enter key: ");
fgets(key, sizeof(key), stdin);
key[strcspn(key, "\n")] = '\0'; // Removing newline character
encrypt(plaintext, key);
printf("Encrypted text: %s\n", plaintext);
return 0;
}
void encrypt(char *plaintext, char *key) {
int keyLength = strlen(key);
int keyIndex = 0;
for (int i = 0; plaintext[i] != '\0'; i++) {
if (isalpha(plaintext[i])) {
int shift = tolower(key[keyIndex]) - 'a';
plaintext[i] = shiftChar(plaintext[i], shift);
keyIndex = (keyIndex + 1) % keyLength;
}
}
}
char shiftChar(char ch, int shift) {
if (islower(ch)) {
return 'a' + (ch - 'a' + shift) % 26;
} else if (isupper(ch)) {
return 'A' + (ch - 'A' + shift) % 26;
}
return ch;
}
C:\Users\yasri\OneDrive\Desk
Enter plaintext: meet mee today
Enter key: 3
Encrypted text: YQQ` YQQ `[PMK
Process exited after 12.9 seconds with return value 0
Press any key to continue . . .
Resources
Compile Log
Debug
Find Results
Close
Compilation results...
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\yasri\OneDrive\Des
- Output Size: 130.525390625 KiB


v1.cpp
v15.cpp
v14.cpp
[*] v13.cpp
v12.cpp
v11.cpp
v10.cpp
v9.cpp
v8.cpp
v7.cpp
v6.cpp
v5.cpp
#include <stdio.h>
int gcd(int a, int b);
int main() {
printf("Values of a that are not allowed in the affine Caesar cipher:\n");
for (int a = 1; a < 26; a++) {
if (gcd(a, 26) > 1) {
printf("%d ", a);
}
}
printf("\n");
return 0;
}
int gcd(int a, int b) {
int temp;
while (b != 0) {
temp = b;
b = a % b;
a = temp;
}
return a;
}
C:\Users\yasri\OneDrive\Desk
Values of a that are not allowed in the affine Caesar cipher:
2 4 6 8 10 12 13 14 16 18 20 22 24

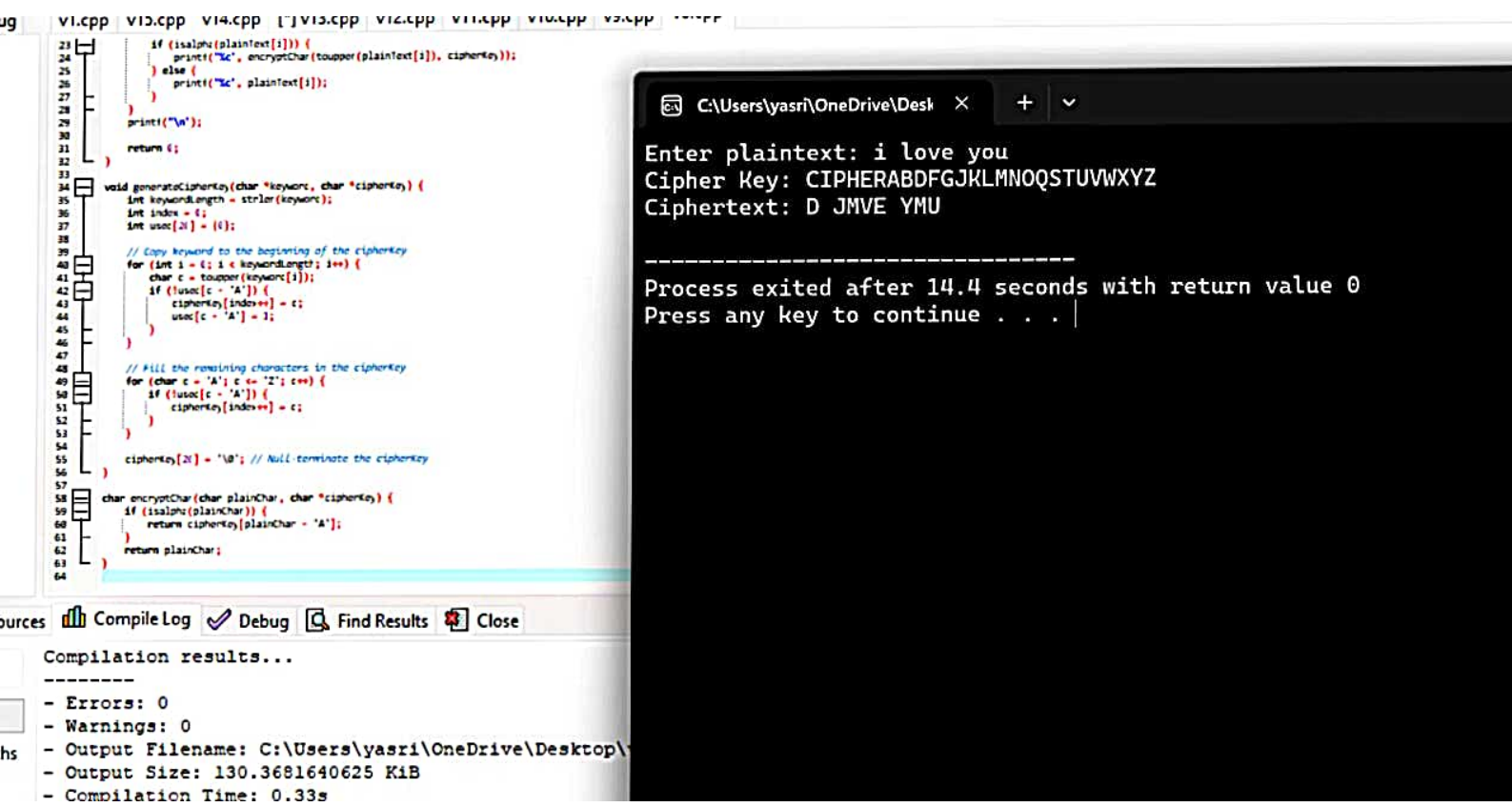
Process exited after 1.068 seconds with return value 0
Press any key to continue . . .


v1.cpp
v15.cpp
v14.cpp
[*] v13.cpp
v12.cpp
v11.cpp
v10.cpp
v9.cpp
v8.cpp
v7.cpp
v6.cpp
int modInverse(int a, int m) {
a = a % m;
for (int x = 1; x < m; x++) {
if ((a * x) % m == 1) {
return x;
}
}
return -1;
}
void decryptAffineCipher(char *ciphertext, int a, int b) {
char plaintext[strlen(ciphertext) + 1];
for (int i = 0; i < strlen(ciphertext); i++) {
if (ciphertext[i] >= 'A' && ciphertext[i] <= 'Z') {
int value = ciphertext[i] - 'A';
value = ((value - b) * a) % ALPHABET_SIZE;
if (value < 0) {
value += ALPHABET_SIZE;
}
plaintext[i] = 'A' + value;
} else {
plaintext[i] = ciphertext[i];
}
}
plaintext[strlen(ciphertext)] = '\0';
// Check if 'B' and 'U' are the most frequent and second most frequent letters
int freq_B = 0;
int freq_U = 0;
for (int i = 0; i < strlen(plaintext); i++) {
if (plaintext[i] == 'B') {
freq_B++;
} else if (plaintext[i] == 'U') {
freq_U++;
}
}
if (freq_B == 6 && freq_U == 5) {
printf("Possible decryption with key (a=%d, b=%d): %s\n", a, b, plaintext);
}
}
C:\Users\yasri\OneDrive\Desk
Ciphertext: BUFXUEWUZUAFVUBUQEBUFTUJNFUFBUPUIFNBHOBT
Frequency of 'B': 6
Frequency of 'U': 5
Process exited after 1.018 seconds with return value 0
Press any key to continue . . .
sources
Compile Log
Debug
Find Results
Close


v1.cpp
v15.cpp
v14.cpp
[*] v13.cpp
v12.cpp
v11.cpp
v10.cpp
v9.cpp
v8.cpp
v7.cpp
// Calculate frequency of characters
for (int i = 0; i < len; i++) {
char ch = ciphertext[i];
if (isalpha(ch) || ch == ' ') {
freq[ch]++;
if (freq[ch] > maxFreq) {
maxFreq = freq[ch];
mostFrequentChar = ch;
}
}
}
int offset_e = mostFrequentChar - 'e';
printf("Offset for 'e': %d\n", offset_e);
char ch_t = mostFrequentChar - offset_e + 1;
char ch_h = mostFrequentChar - offset_e + 7;
printf("Guess for 't': %c\n", ch_t);
printf("Guess for 'h': %c\n", ch_h);
// Decrypt the rest of the message
for (int i = 0; i < len; i++) {
char ch = ciphertext[i];
if (isalpha(ch) || ch == ' ') {
// Decrypt character using the calculated offset
if (ch == ch_t) {
printf("t");
} else if (ch == ch_h) {
printf("h");
} else if (isalpha(ch)) {
printf("%c", ch - offset_e);
} else {
printf(" ");
}
} else {
printf("%c", ch);
}
}
}
Compile Log
Debug
Find Results
Compilation results...

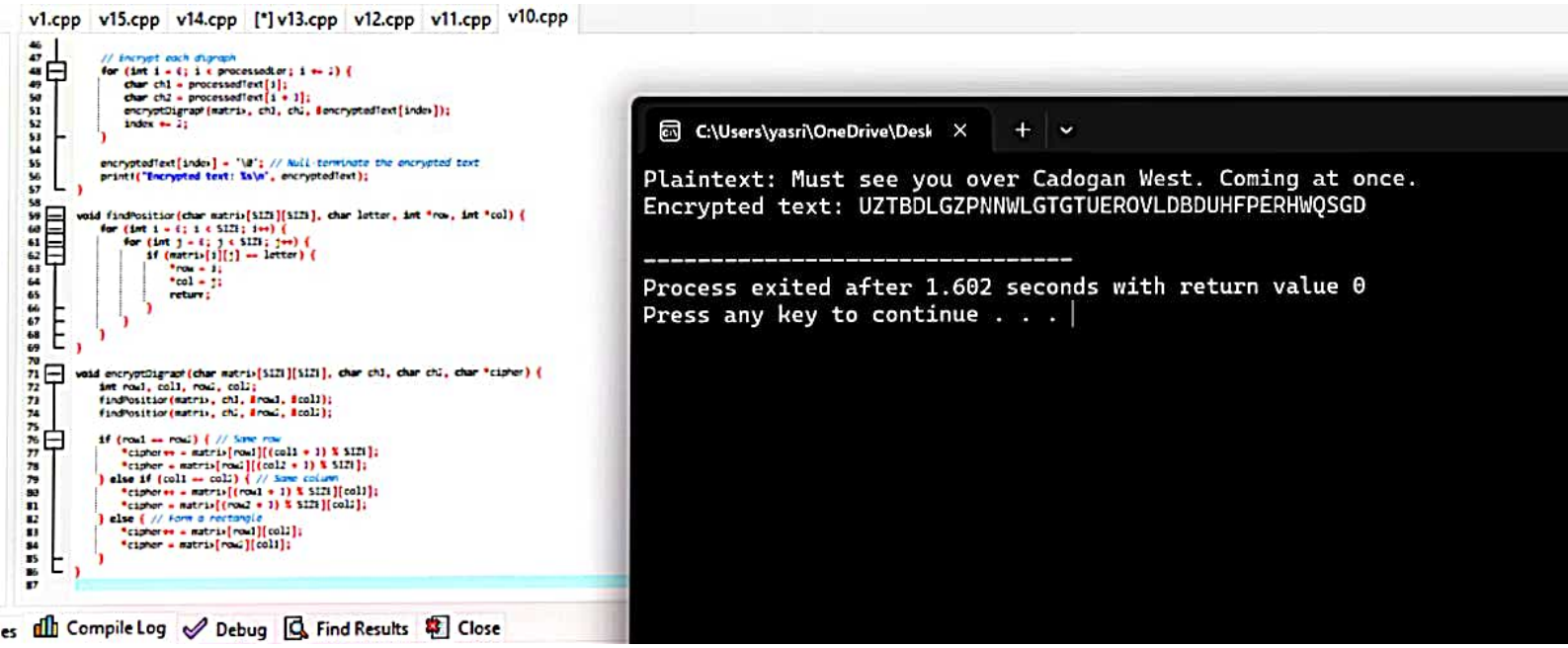
- Errors: 0
C:\Users\yasri\OneDrive\Desk
Ciphertext: 53ççå305))6*;4826)4ç.)4ç);806*;48å8‖60))85;;]8*;:ç*8å83
5);5*å2:*ç(;4956*2(5*ù4)8‖8*
;4069285);)6å8)4çç;1(ç9;48081;8:8ç1
8;4(ç?34;48)4ç;161;:188;ç?;
Offset for 'e': -69
Guess for 't': `
Guess for 'h': ^
53ççå305))6*;4826)4ç.)4ç);806*;48å8‖60))85;;]8*;:ç*8å83
(88)5*å;
4956*2(5*ù4)8‖8*
;4069285);)6å8)4çç;1(ç9;48081;8:8ç1;48å85;4)485
4ç;161;:188;ç?;

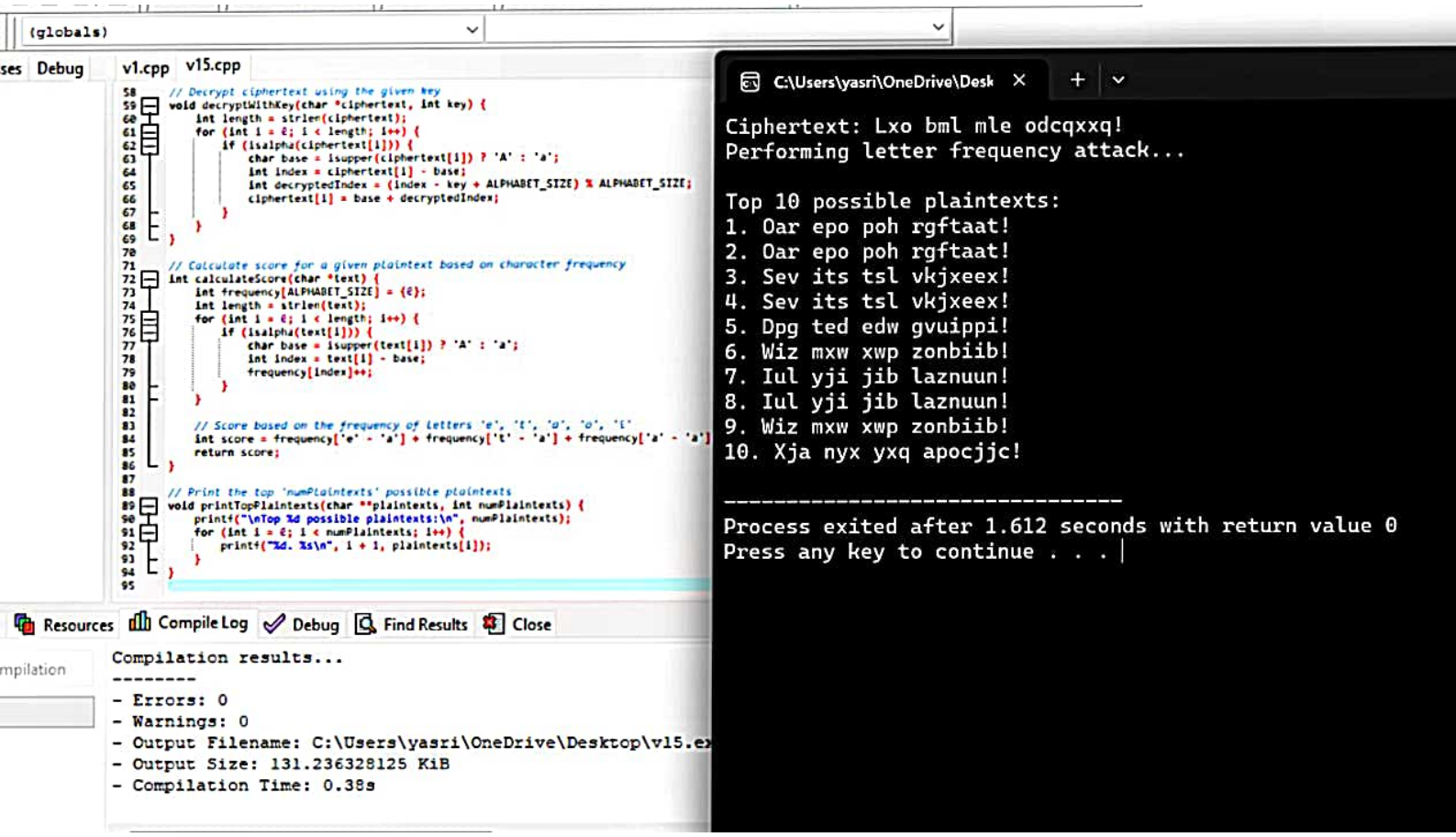
Process exited after 0.754 seconds with return value 0
Press any key to continue . . .


v1.cpp v13.cpp v14.cpp v13.cpp v12.cpp v11.cpp v10.cpp v9.cpp
if (isalpha(plaintext[i])) {
printf("%c", encryptChar(toupper(plaintext[i]), cipherKey));
} else {
printf("%c", plaintext[i]);
}
}
printf("\n");
return 0;
}
void generateCipherKey(char *keyword, char *cipherKey) {
int keywordLength = strlen(keyword);
int index = 0;
int used[26] = {0};
// Copy keyword to the beginning of the cipherKey
for (int i = 0; i < keywordLength; i++) {
char c = toupper(keyword[i]);
if (!used[c - 'A']) {
cipherKey[index++] = c;
used[c - 'A'] = 1;
}
}
// Fill the remaining characters in the cipherKey
for (char c = 'A'; c <= 'Z'; c++) {
if (!used[c - 'A']) {
cipherKey[index++] = c;
}
}
cipherKey[26] = '\0'; // Null-terminate the cipherKey
}
char encryptChar(char plainChar, char *cipherKey) {
if (isalpha(plainChar)) {
return cipherKey[plainChar - 'A'];
}
return plainChar;
}
Compile Log
Debug
Find Results
Close
Compilation results...

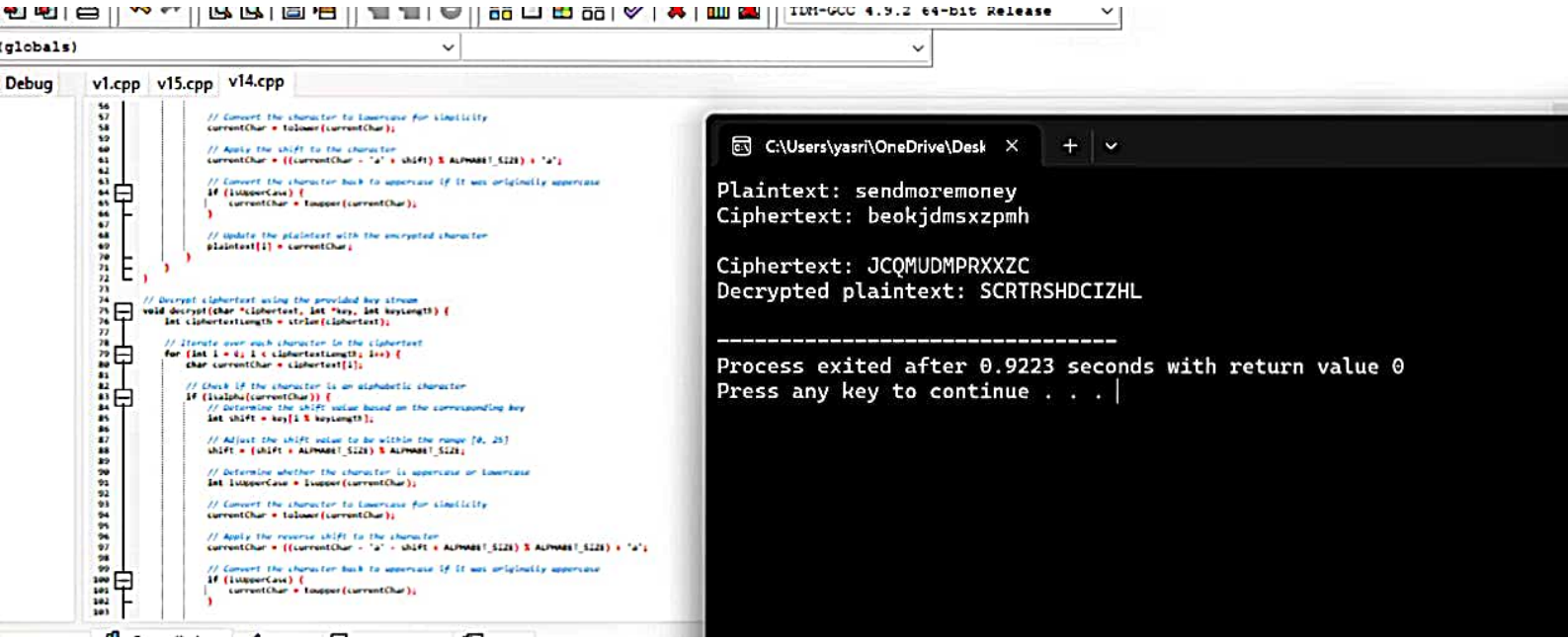
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\yasri\OneDrive\Desktop\
- Output Size: 130.3681640625 KiB
- Compilation Time: 0.33s
C:\Users\yasri\OneDrive\Desk
Enter plaintext: i love you
Cipher Key: CIPHERABDFGJKLMNOQSTUVWXYZ
Ciphertext: D JMVE YMU

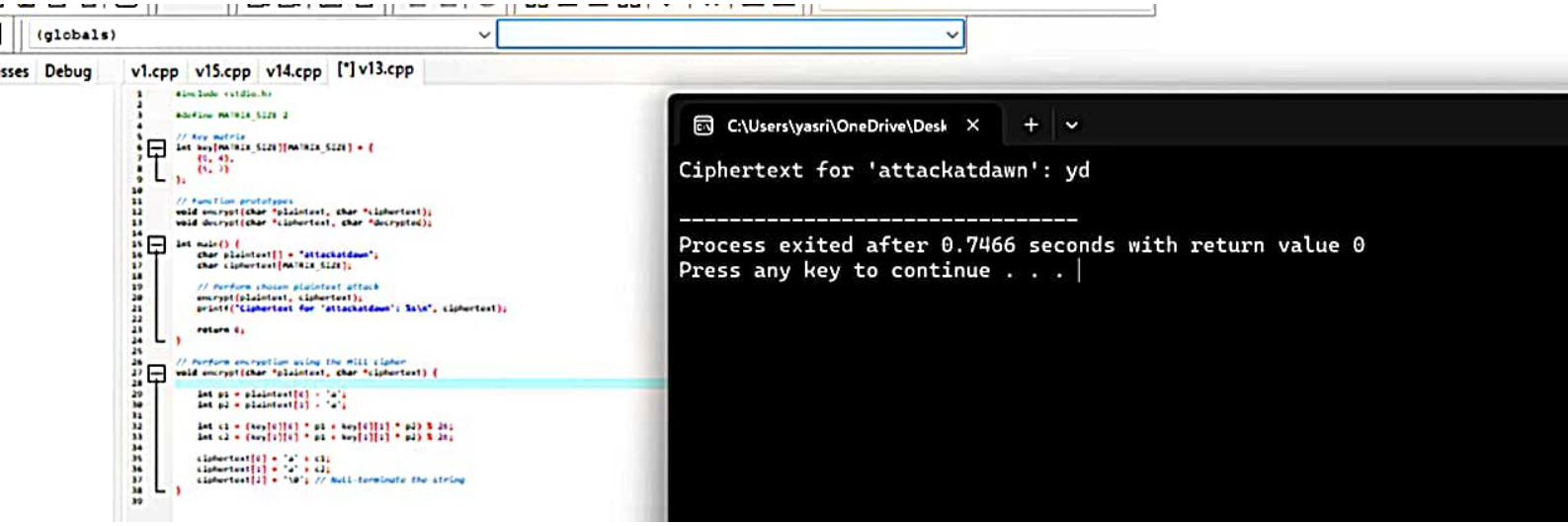
Process exited after 14.4 seconds with return value 0
Press any key to continue . . .


v1.cpp
v15.cpp
v14.cpp
[*] v13.cpp
v12.cpp
v11.cpp
v10.cpp
// Encrypt each digraph
for (int i = 0; i < processedLen; i += 2) {
char ch1 = processedText[i];
char ch2 = processedText[i + 1];
encryptDigraph(matrix, ch1, ch2, &encryptedText[index]);
index += 2;
}
encryptedText[index] = '\0'; // Null-terminate the encrypted text
printf("Encrypted text: %s\n", encryptedText);
}
void findPosition(char matrix[SIZE][SIZE], char letter, int *row, int *col) {
for (int i = 0; i < SIZE; i++) {
for (int j = 0; j < SIZE; j++) {
if (matrix[i][j] == letter) {
*row = i;
*col = j;
return;
}
}
}
}
void encryptDigraph(char matrix[SIZE][SIZE], char ch1, char ch2, char *cipher) {
int row1, col1, row2, col2;
findPosition(matrix, ch1, &row1, &col1);
findPosition(matrix, ch2, &row2, &col2);
if (row1 == row2) { // Same row
*cipher++ = matrix[row1][(col1 + 1) % SIZE];
*cipher = matrix[row2][(col2 + 1) % SIZE];
} else if (col1 == col2) { // Same column
*cipher++ = matrix[(row1 + 1) % SIZE][col1];
*cipher = matrix[(row2 + 1) % SIZE][col2];
} else { // Form a rectangle
*cipher++ = matrix[row1][col2];
*cipher = matrix[row2][col1];
}
}
Compile Log
Debug
Find Results
Close
C:\Users\yasri\OneDrive\Desk
Plaintext: Must see you over Cadogan West. Coming at once.
Encrypted text: UZTBDLGZPNNWLGTGTUEROVLDBDUHFPERHWQSGD
Process exited after 1.602 seconds with return value 0
Press any key to continue . . .

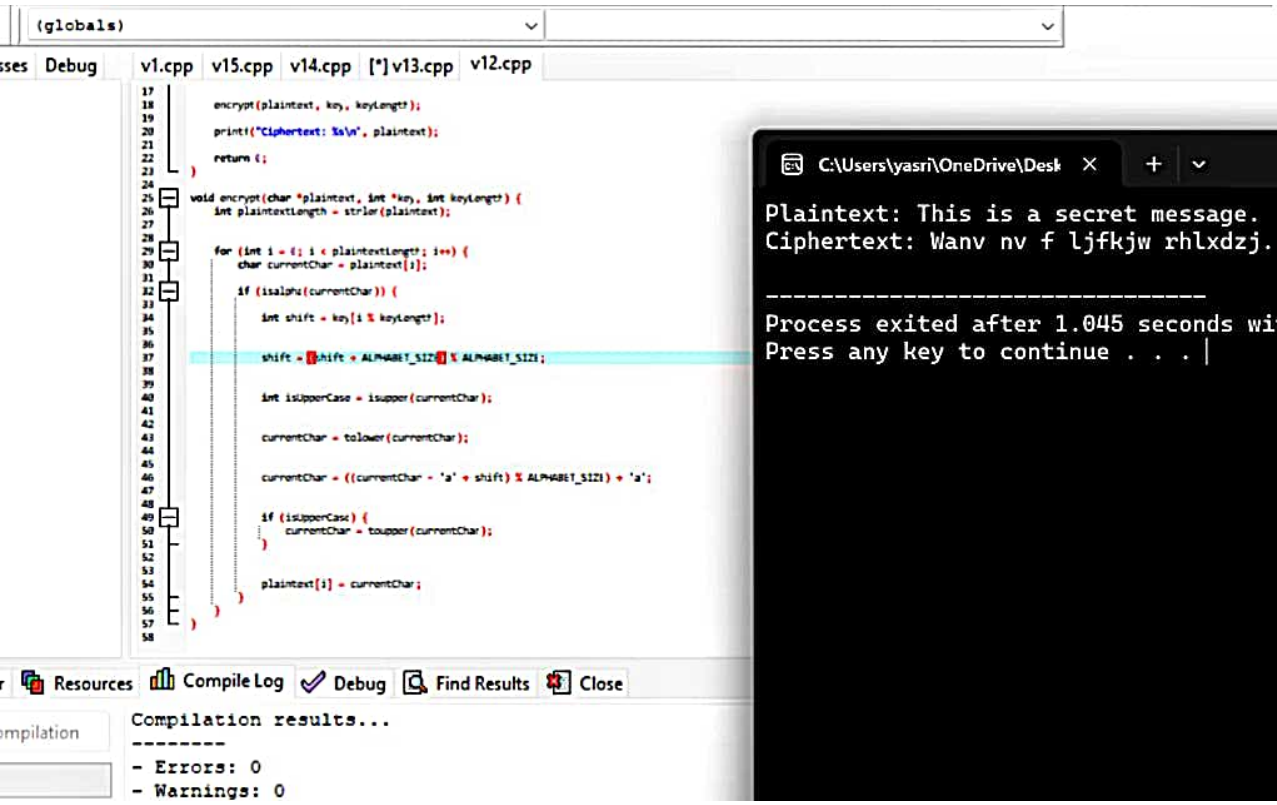

(globals)
Debug
v1.cpp
v15.cpp
// Decrypt ciphertext using the given key
void decryptWithKey(char *ciphertext, int key) {
int length = strlen(ciphertext);
for (int i = 0; i < length; i++) {
if (isalpha(ciphertext[i])) {
char base = isupper(ciphertext[i]) ? 'A' : 'a';
int index = ciphertext[i] - base;
int decryptedIndex = (index - key + ALPHABET_SIZE) % ALPHABET_SIZE;
ciphertext[i] = base + decryptedIndex;
}
}
}
// Calculate score for a given plaintext based on character frequency
int calculateScore(char *text) {
int frequency[ALPHABET_SIZE] = {0};
int length = strlen(text);
for (int i = 0; i < length; i++) {
if (isalpha(text[i])) {
char base = isupper(text[i]) ? 'A' : 'a';
int index = text[i] - base;
frequency[index]++;
}
}
// Score based on the frequency of letters 'e', 't', 'a', 'o', 'i'
int score = frequency['e' - 'a'] + frequency['t' - 'a'] + frequency['a' - 'a']
return score;
}
// Print the top 'numPlaintexts' possible plaintexts
void printTopPlaintexts(char **plaintexts, int numPlaintexts) {
printf("\nTop %d possible plaintexts:\n", numPlaintexts);
for (int i = 0; i < numPlaintexts; i++) {
printf("%d. %s\n", i + 1, plaintexts[i]);
}
}
Resources
Compile Log
Debug
Find Results
Close
Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\yasri\OneDrive\Desktop\v15.ex
- Output Size: 131.236328125 KiB
- Compilation Time: 0.38s
C:\Users\yasri\OneDrive\Desk
Ciphertext: Lxo bml mle odcqxxq!
Performing letter frequency attack...
Top 10 possible plaintexts:
1. Oar epo poh rgftaat!
2. Oar epo poh rgftaat!
3. Sev its tsl vkjxeex!
4. Sev its tsl vkjxeex!
5. Dpg ted edw gvuippi!
6. Wiz mxw xwp zonbiib!
7. Iul yji jib laznuun!
8. Iul yji jib laznuun!
9. Wiz mxw xwp zonbiib!
10. Xja nyx yxq apocjjc!
Process exited after 1.612 seconds with return value 0
Press any key to continue . . .


(globals)
Debug
v1.cpp
v15.cpp
v14.cpp
// Convert the character to lowercase for simplicity
currentChar = tolower(currentChar);
// Apply the shift to the character
currentChar = ((currentChar - 'a' + shift) % ALPHABET_SIZE) + 'a';
// Convert the character back to uppercase if it was originally uppercase
if (isUpperCase) {
currentChar = toupper(currentChar);
}
// Update the plaintext with the encrypted character
plaintext[i] = currentChar;
}
}
}
// Decrypt ciphertext using the provided key stream
void decrypt(char *ciphertext, int *key, int keyLength) {
int ciphertextLength = strlen(ciphertext);
// Iterate over each character in the ciphertext
for (int i = 0; i < ciphertextLength; i++) {
char currentChar = ciphertext[i];
// Check if the character is an alphabetic character
if (isalpha(currentChar)) {
// Determine the shift value based on the corresponding key
int shift = key[i % keyLength];
// Adjust the shift value to be within the range [0, 25]
shift = (shift + ALPHABET_SIZE) % ALPHABET_SIZE;
// Determine whether the character is uppercase or lowercase
int isUpperCase = isupper(currentChar);
// Convert the character to lowercase for simplicity
currentChar = tolower(currentChar);
// Apply the reverse shift to the character
currentChar = ((currentChar - 'a' - shift + ALPHABET_SIZE) % ALPHABET_SIZE) + 'a';
// Convert the character back to uppercase if it was originally uppercase
if (isUpperCase) {
currentChar = toupper(currentChar);
}
C:\Users\yasri\OneDrive\Desk
Plaintext: sendmoremoney
Ciphertext: beokjdmsxzpmh
Ciphertext: JCQMUDMPRXXZC
Decrypted plaintext: SCRTRSHDCIZHL
Process exited after 0.9223 seconds with return value 0
Press any key to continue . . .


(globals)
sses
Debug
v1.cpp
v15.cpp
v14.cpp
[*] v13.cpp
#include <stdio.h>
#define MATRIX_SIZE 2
// Key matrix
int key[MATRIX_SIZE][MATRIX_SIZE] = {
{5, 4},
{1, 3}
};
// Function prototypes
void encrypt(char *plaintext, char *ciphertext);
void decrypt(char *ciphertext, char *decrypted);
int main() {
char plaintext[] = "attackatdawn";
char ciphertext[MATRIX_SIZE];
// Perform chosen plaintext attack
encrypt(plaintext, ciphertext);
printf("Ciphertext for 'attackatdawn': %s\n", ciphertext);
return 0;
}
// Perform encryption using the Hill cipher
void encrypt(char *plaintext, char *ciphertext) {
int p1 = plaintext[0] - 'a';
int p2 = plaintext[1] - 'a';
int c1 = (key[0][0] * p1 + key[0][1] * p2) % 26;
int c2 = (key[1][0] * p1 + key[1][1] * p2) % 26;
ciphertext[0] = 'a' + c1;
ciphertext[1] = 'a' + c2;
ciphertext[2] = '\0'; // Null-terminate the string
}
C:\Users\yasri\OneDrive\Desk
Ciphertext for 'attackatdawn': yd

Process exited after 0.7466 seconds with return value 0
Press any key to continue . . .


(globals)
Debug
v1.cpp
v15.cpp
v14.cpp
[*] v13.cpp
v12.cpp
encrypt(plaintext, key, keyLength);
printf("Ciphertext: %s\n", plaintext);
return 0;
void encrypt(char *plaintext, int *key, int keyLength) {
int plaintextLength = strlen(plaintext);
for (int i = 0; i < plaintextLength; i++) {
char currentChar = plaintext[i];
if (isalpha(currentChar)) {
int shift = key[i % keyLength];
shift = (shift + ALPHABET_SIZE) % ALPHABET_SIZE;
int isUpperCase = isupper(currentChar);
currentChar = tolower(currentChar);
currentChar = ((currentChar - 'a' + shift) % ALPHABET_SIZE) + 'a';
if (isUpperCase) {
currentChar = toupper(currentChar);
}
plaintext[i] = currentChar;
}
}
}
Resources
Compile Log
Debug
Find Results
Close
Compilation results...

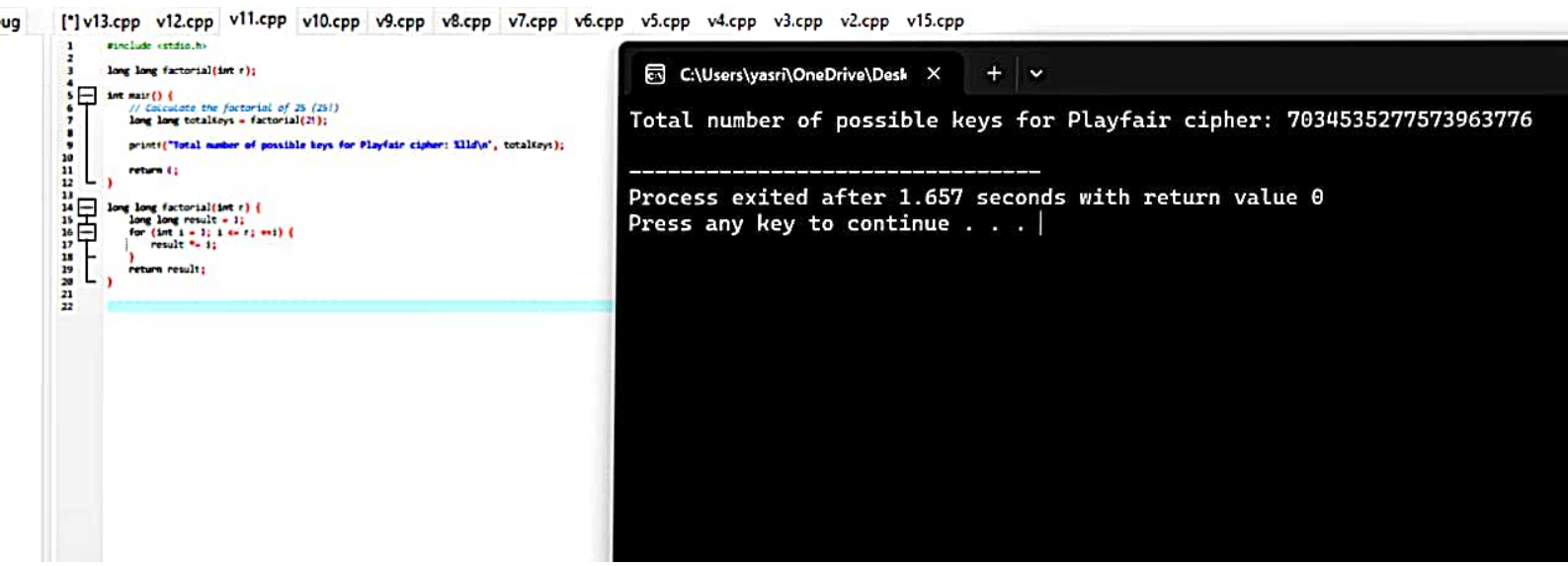
- Errors: 0
- Warnings: 0
C:\Users\yasri\OneDrive\Desk
Plaintext: This is a secret message.
Ciphertext: Wanv nv f ljfkjw rhlxdzj.

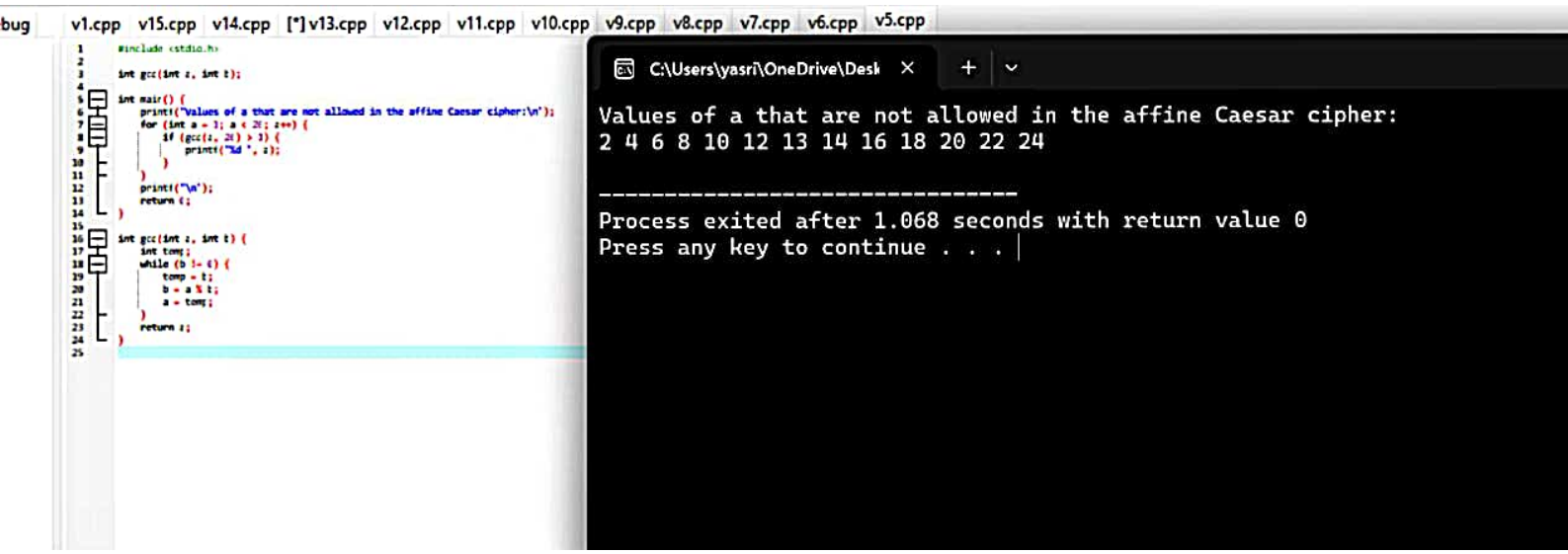
Process exited after 1.045 seconds with return value 0
Press any key to continue . . .

v1.cpp

```c
#include <stdio.h>
#include <string.h>

void caesarCipher(char *text, int shift);

int main() {
    char text[100];

    printf("Enter text to encrypt: ");
    fgets(text, sizeof(text), stdin);
    text[strcspn(text, "\n")] = 0; // Removing newline character from f

    printf("\nCaesar Cipher Encryptions:\n");

    for (int shift = 1; shift <= 25; shift++) {
        printf("Shift %d: ", shift);
        caesarCipher(text, shift);
        printf("\n");
    }

    return 0;
}

void caesarCipher(char *text, int shift) {
    int i;
    for(i = 0; text[i] != '\0'; i++) {
        if(text[i] >= 'A' && text[i] <= 'Z') {
            text[i] = ((text[i] - 'A' + shift) % 26) + 'A';
        }
        else if(text[i] >= 'a' && text[i] <= 'z') {
            text[i] = ((text[i] - 'a' + shift) % 26) + 'a';
        }
    }
    printf("%s", text);
}
```

```
C:\Users\yasri\OneDrive\Desk

Caesar Cipher Encryptions:
Shift 1: nffu nf upnpsspx
Shift 2: phhw ph wrpruurz
Shift 3: skkz sk zusuxxuc
Shift 4: wood wo dywybbyg
Shift 5: btti bt idbdggdl
Shift 6: hzzo hz ojhjmmjr
Shift 7: oggv og vqoqttqy
Shift 8: wood wo dywybbyg
Shift 9: fxxm fx mhfhkkhp
Shift 10: phhw ph wrpruurz
Shift 11: assh as hcacffck
Shift 12: meet me tomorrow
Shift 13: zrrg zr gbzbeebj
Shift 14: nffu nf upnpsspx
Shift 15: cuuj cu jecehhem
Shift 16: skkz sk zusuxxuc
Shift 17: jbbq jb qljloolt
Shift 18: btti bt idbdggdl
Shift 19: ummb um bmuwzzwo
Shift 20: oggv og vqoqttqy
Shift 21: jbbq jb qljloolt
Shift 22: fxxm fx mhfhkkhp
Shift 23: cuuj cu jecehhem
Shift 24: assh as hcacffck
Shift 25: zrrg zr gbzbeebj

--------------------------------
Process exited after 22.39 seconds with return value 0
Press any key to continue . . .
```


ug
[*] v13.cpp
v12.cpp
v11.cpp
v10.cpp
v9.cpp
v8.cpp
v7.cpp
v6.cpp
v5.cpp
v4.cpp
v3.cpp
v2.cpp
v15.cpp
#include <stdio.h>
long long factorial(int r);
int main() {
// Calculate the factorial of 25 (25!)
long long totalKeys = factorial(25);
printf("Total number of possible keys for Playfair cipher: %lld\n", totalKeys);
return 0;
}
long long factorial(int r) {
long long result = 1;
for (int i = 1; i <= r; ++i) {
result *= i;
}
return result;
}
C:\Users\yasri\OneDrive\Desk
Total number of possible keys for Playfair cipher: 7034535277573963776
Process exited after 1.657 seconds with return value 0
Press any key to continue . . .


bug
v1.cpp
v15.cpp
v14.cpp
[*] v13.cpp
v12.cpp
v11.cpp
v10.cpp
v9.cpp
v8.cpp
v7.cpp
v6.cpp
v5.cpp
#include <stdio.h>
int gcd(int a, int b);
int main() {
printf("Values of a that are not allowed in the affine Caesar cipher:\n");
for (int a = 1; a < 26; a++) {
if (gcd(a, 26) > 1) {
printf("%d ", a);
}
}
printf("\n");
return 0;
}
int gcd(int a, int b) {
int temp;
while (b != 0) {
temp = b;
b = a % b;
a = temp;
}
return a;
}
C:\Users\yasri\OneDrive\Desk
Values of a that are not allowed in the affine Caesar cipher:
2 4 6 8 10 12 13 14 16 18 20 22 24
Process exited after 1.068 seconds with return value 0
Press any key to continue . . .