Phòng thí nghiệm An toàn Thông tin InsecLab
Trường Đại học Công nghệ Thông tin (UIT)

BÁO CÁO THỰC HÀNH

Môn học: CRYPTOGRAPHY – Mật mã học

Tên chủ đề: Classical Crytography

GVHD: Tô Trọng Nghĩa

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: **NT219.N21.ANTT.1**

STT	Họ và tên	MSSV	Email
1	Lê Đoàn Trà My	21521149	21521149@gm.uit.edu.vn

2. NỘI DUNG THỰC HIỆN:[1]

STT	Nội dung	Tình trạng	Trang
1	Kickoff: Crack the code	100%	2 - 4
2	Caesar cipher	100%	4 - 7
3	Mono-alphabetic substitution cipher and frequency analysis	100%	7-10
4	Playfair cipher	100%	10 – 16
5	Vigenére cipher	100%	17 - 19
6	Other ciphers	100%	19 - 22
Điểm tự đánh giá			**9.5 - 10/10**

Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.

[1] Ghi nội dung công việc, các kịch bản trong bài Thực hành

BÁO CÁO CHI TIẾT

1. Kickoff: Crack the code

a, Tìm mật khẩu của khoá:

- Giả sử, mật khẩu có dạng: ABC.
- Các gợi ý: (1) {6, 8, 2}: 1 số đúng và đúng vị trí.
  (2) {6, 1, 4}: 1 số đúng nhưng sai vị trí.
  (3) {2, 0, 6}: 2 số đúng nhưng sai vị trí.
  (4) {7, 3, 8}: Không số nào đúng.
  (5) {7, 8, 0}: 1 số đúng nhưng sai vị trí.

Figure 4: Crack the code to open the lock

- *Giải mã:*
  + Từ các gợi ý, ta có các số có thể là chữ số của mật khẩu là: 0, 1, 2, 3, 4, 5, 6, 7, 8.
  + Từ (4) loại các số: 7, 3, 8.
  + Kết hợp (4), (5) → 0 là 1 chữ số trong mật khẩu nhưng 0 không phải là C. (6)
  + Kết hợp (1), (2) → loại số 6.
  + Kết hợp (1), (4), (6) → C = 2. (7)
  + Kết hợp (3), (6), (7) → A = 0. (8)
  + Kết hợp (2), (7), (8) → B = 4.
- *Kết luận:* Mật khẩu của khoá là: **042**.

b, Tìm mã hoá tương ứng cho mỗi số từ 1 đến 9:

★	★	★	★	?
★	★	★	★	★★
?	?	★	★	★★
?	★	★	★	★★
★★	★★	★★	★★	

Table 1: Find the corresponding encoding for each number

Giải mã:

- Mỗi ký hiệu mã hoá cho 1 số bất kỳ từ 1 đến 9.
- Có tổng của 4 số bất kỳ luôn ≤ 36 (vì tổng lớn nhất có thể có: 9+9+9+9 = 36).
- Các ô có 2 ký hiệu giống nhau như ★★ , ★★ chỉ có thể là: 11 hoặc 22 hoặc 33.
- Xét cột 3, có: 2★ + 2★ = ★★ → ★★= 22, ★ = **2** (Tổng của 2 số chẵn là 1 số chẵn)

- 🌟🌟 chỉ có thể là 11 hoặc 33:

*** Trường hợp 1:** 🌟🌟 = 11, 🌟 = 1.

+ Xét hàng 2, có: 3🌟 + 🌟 = 11.

🌟	1	2	3	4
🌟	8	5	2	-1

Vì 🌟 = 2, 🌟 = 1 nên loại tất cả trường hợp.

*** Trường hợp 2:** 🌟🌟 = 33, 🌟 = 3.

+ Xét hàng 2, có: 3🌟 + 🌟 = 33 và xét cột 3: 2🌟 + 2🌟 = 22

🌟	1	2	…	7	8	9
🌟	30	27	…	12	9	6
🌟	-	-	…	-	2	5

Vì 🌟 = 2 nên loại trường hợp 🌟 = 8, 🌟 = 9

Vậy: 🌟 = 3, 🌟 = 9, 🌟 = 6, 🌟 = 5

+ Xét cột 4, có: 🌟 + 🌟 + 2🌟 = 🌟🌟 ↔ 🌟 + 🌟 + 2*9 = 23

🌟	1	2	3	4
🌟	4	3	2	1

Vì 🌟 = 2 nên loại trường hợp {🌟, 🌟} = {2, 3} và {🌟, 🌟} = {3, 2}

Vậy: {🌟, 🌟} có thể là {1, 4} hoặc {4, 1}; {🌟, 🌟} chỉ có thể là {7, 8} hoặc {8, 7}

+ Xét trường hợp {🌟, 🌟} = {7, 8}:

Cột 2, có: 🌟 + 2*🌟 + $?_{3,2}$ = 🌟🌟 ↔ 7 + 2*9 + $?_{3,2}$ = 33 → $?_{3,2}$ = 8 = 🌟

Hàng 4, có: $?_{4,1}$ + 2*🌟 + 🌟 = 🌟🌟 ↔ $?_{4,1}$ + 2*9 + 6 = 28 → $?_{4,1}$ = 4

Cột 1, có: 🌟 + 🌟 + $?_{3,1}$ + $?_{4,1}$ = 🌟🌟 ↔ 7 + 9 + $?_{3,1}$ + 4 = 29 → $?_{3,1}$ = 9 = 🌟

Hàng 3, có: $?_{3,1}$ + $?_{3,2}$ + 🌟 + 🌟 = 🌟🌟 ↔ 9 + 8 + 5 + 🌟 = 22 → 🌟 = 0

Vậy loại trường hợp {🌟, 🌟} = {7, 8}

+ Xét trường hợp {🌟, 🌟} = {8, 7}:

Cột 2, có: 🌟 + 2*🌟 + $?_{3,2}$ = 🌟🌟 ↔ 8 + 2*9 + $?_{3,2}$ = 33 → $?_{3,2}$ = **7** = 🌟

Hàng 4, có: $?_{4,1}$ + 2*🌟 + 🌟 = 🌟🌟 ↔ $?_{4,1}$ + 2*9 + 6 = 27 → $?_{4,1}$ = 3 = 🌟

Cột 1, có: 🌟 + 🌟 + $?_{3,1}$ + $?_{4,1}$ = 🌟🌟 ↔ 8 + 9 + $?_{3,1}$ + 3 = 29 → $?_{3,1}$ = 9 = 🌟

Hàng 3, có: ? $_{3,1}$ + ? $_{3,2}$ + ✪ + ✪ = ✪✪ ↔ 9 + 7 + 5 + ✪ = 22 → ✪ = 1; ✪ = 4

Hàng 1, có: 2* ✪ + ✪ + ✪ = ? $_{1,4}$ ↔ 2*8 + 5 + 4 = ? $_{1,4}$ → ? $_{1,4}$ = 25 = ✪✪

Kết luận: ✪ = 1, ✪ = 2, ✪ = 3, ✪ = 4, ✪ = 5, ✪ = 6, ✪ = 7, ✪ =8, ✪ = 9.

2. Caesar cipher

- **Mật mã Caesar** là một dạng mật mã thay thế, mỗi ký tự trên văn bản thô sẽ được thay bằng một ký tự khác, có vị trí cách nó một khoảng xác định trong bảng chữ cái.

- **Thuật toán Caeser:**

+ Encryption (mã hoá): C = E(k, p) = (p + k) mod 26

+ Decryption (giải mã): p = D(k, C) = (C - k) mod 26

với: C: bản mã, p: bản rõ, k: khoá (khoảng dịch chuyển)

- **Các đoạn code trong Caeser:**

+ Đoạn code Caeser Encryption:

```
void encrypt(string text, int key) //ma hoa voi key
{
    char alphabet[26] = { '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' };
    char alphabet1[26] = { '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' };
    string output = text;
    for (int j = 0; j < text.length(); j++)
    {
        for (int i = 0; i < 26; i++)
        {
            if (text[j] == alphabet[i])
            {
                output[j] = alphabet[abs(i + key) % 26];
                break;
            }
            else if (text[j] == alphabet1[i])
            {
                output[j] = alphabet1[abs(i + key) % 26];
                break;
            }
        }
    }
    cout << "\nCiphertext: " << output;
}
```

Hình. Đoạn code Caeser Encryption

+ Đoạn code Caeser Decryption có key và không key:

```
void decrypt(string text, int key) //giai ma voi key
{
    char alphabet[26] = { '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' };
    char alphabet1[26] = { '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' };
    string output = text;
    for (int j = 0; j < text.length(); j++)
    {
        for (int i = 0; i < 26; i++)
        {
            if (text[j] == alphabet[i])
            {
                output[j] = alphabet[abs(i - key + 26) % 26];
                break;
            }
            else if (text[j] == alphabet1[i])
            {
                output[j] = alphabet1[abs(i - key + 26) % 26];
                break;
            }
        }
    }
    cout << "\nPlaintext: " << output;
}
```

Hình. Đoạn code Caeser Decryption sử dụng key

```
void decrypt(string text) //giai ma brute-force
{
    char alphabet[26] = { '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' };
    char alphabet1[26] = { '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' };
    string output = text;
    for (int k = 0; k <= 25; k++)
    {
        for (int j = 0; j < text.length(); j++)
        {
            for (int i = 0; i < 26; i++)
            {
                if (text[j] == alphabet[i])
                {
                    output[j] = alphabet[abs(i - k + 26) % 26];
                    break;
                }
                else if (text[j] == alphabet1[i])
                {
                    output[j] = alphabet1[abs(i - k + 26) % 26];
                    break;
                }
            }
        }
        cout << "\nPlaintext with key = " << k << " : " << output<<endl;
    }
}
```

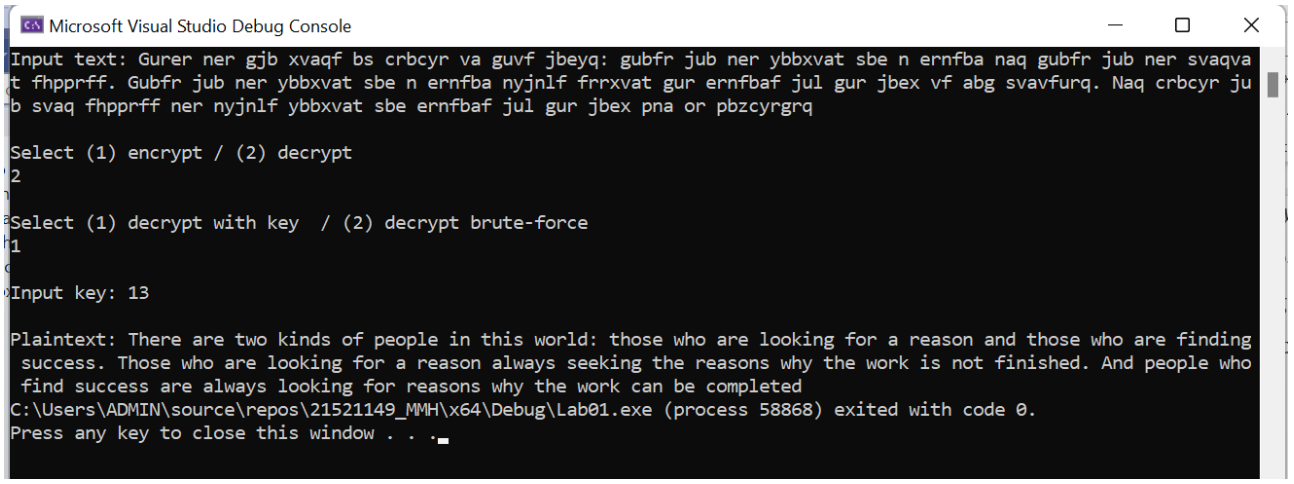
Hình. Đoạn code Caeser Decryption không sử dụng key (brute-force)

- Tiến hành kiểm tra giải mã đoạn mã:

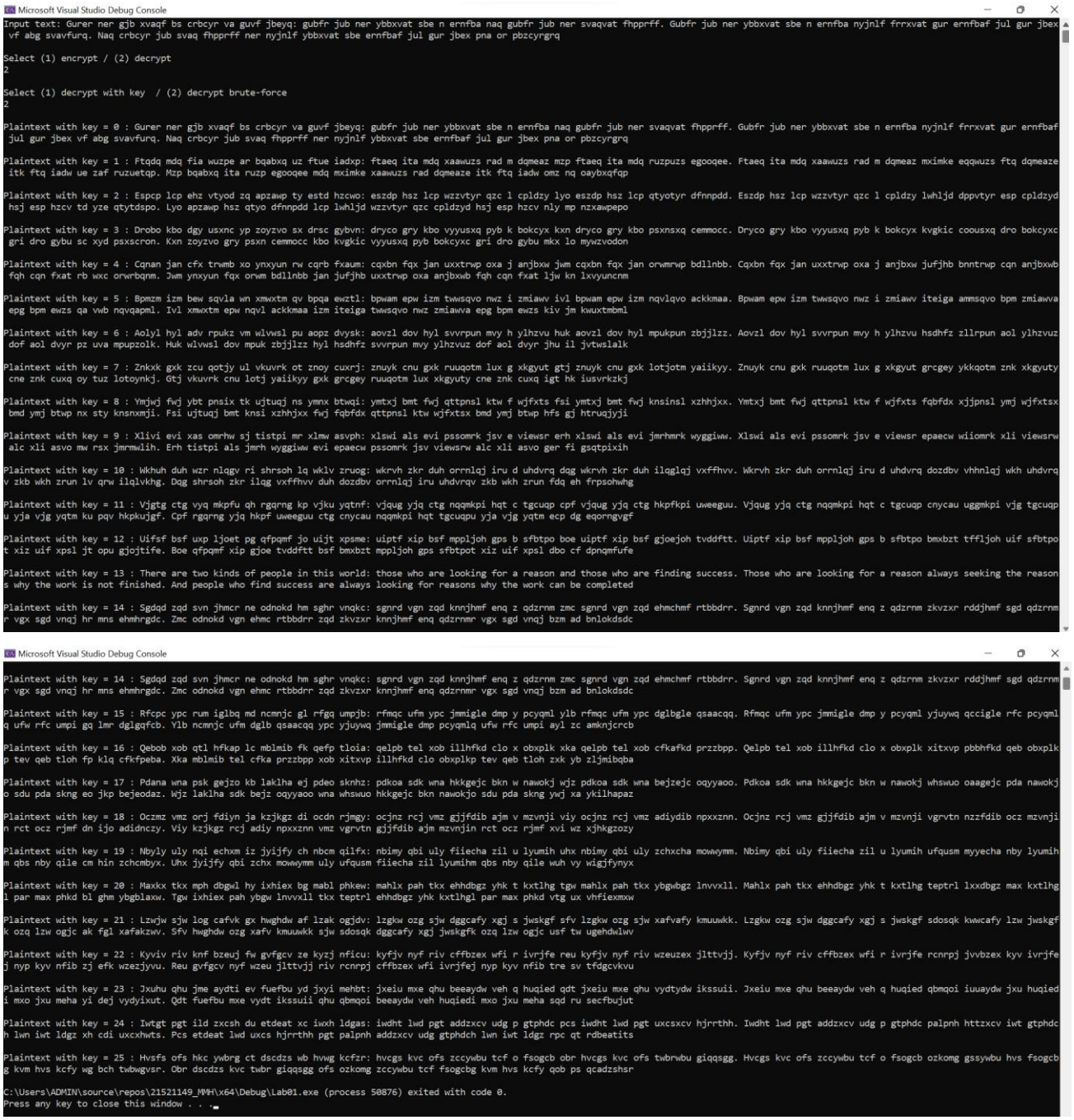
Gurer ner gjb xvaqf bs crbcyr va guvf jbeyq: gubfr jub ner ybbxvat sbe n ernfba naq gubfr jub ner svaqvat fhpprff. Gubfr jub ner ybbxvat sbe n ernfba nyjnlf frrxvat gur ernfbaf jul gur jbex vf abg svavfurq. Naq crbcyr jub svaq fhpprff ner nyjnlf ybbxvat sbe ernfbaf jul gur jbex pna or pbzcyrgrq.

+ Trong đoạn văn trên, có ký tự 'n' đứng riêng lẻ, trong tiếng Anh các ký tự đơn đứng riêng lẻ có nghĩa thường là I, a tuy nhiên I ít khi đứng giữa câu và không đứng cuối câu → 'n' trong bản mã sẽ là 'a' trong bản rõ → key = 13.

+ Thu được các kết quả khi chạy chương trình giải mã đoạn bản mã trên:

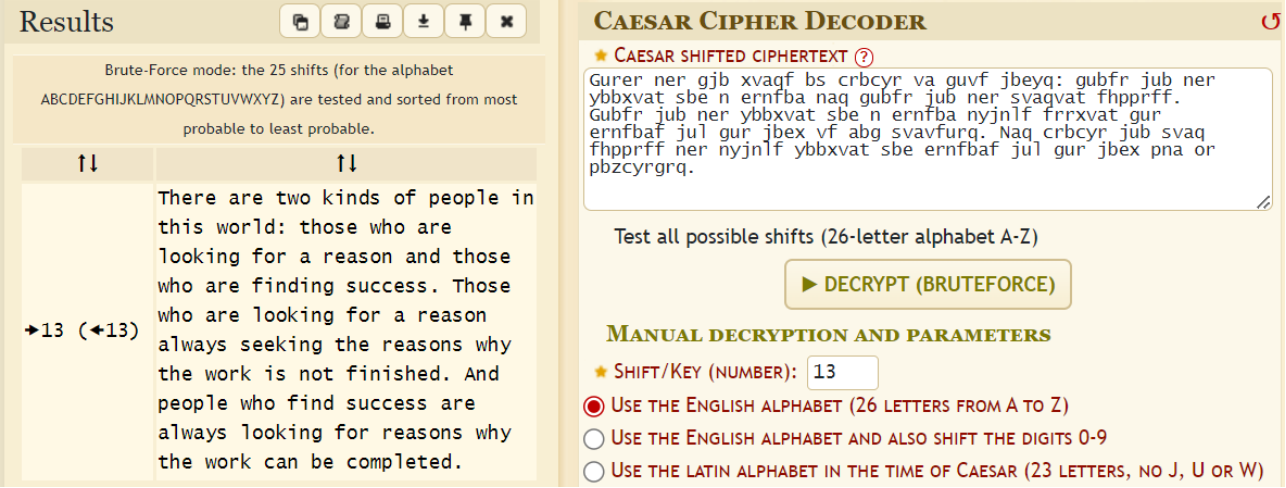


Hình. Kết quả giải mã đoạn bản mã bằng chương trình vừa viết với key = 13



Hình. Kết quả giải mã đoạn bản mã bằng chương trình vừa viết bằng brute-force

+ Kiểm tra, so sánh lại với bản rõ được giải mã bằng dcode:



Hình. Kết quả giải mã đoạn bản mã bằng https://www.dcode.fr/caesar-cipher

3. Mono-alphabetic substitution cipher and frequency analysis

- Sử dụng https://www.cryptool.org/en/cto/n-gram-analysis để tiến hành phân tích đoạn bản mã, dựa vào kết quả phân tích có:

+ Trong các ký tự, 'n' xuất hiện nhiều nhất với tần suất 12,16% → 'n' khả năng cao là 'e'.

+ Trong các từ có 3 ký tự, 'ytn' xuất hiện nhiều nhất và thường ở đầu đoạn văn bản → 'ytn' khả năng cao là 'the' → y = t, t = h, n = e.

+ 'v' đứng một mình và xuất hiện tại đầu và giữa câu → khả năng cao v = a

+ 'y[x]' ↔ 't[x]' → khả năng cao là 'to'→ x = o

+ 'v[q] a' ↔ 'a[q] a' → q = s (q ≠ n vì không thể là an a)

+ 'x[u]n', 'x[u]', '[u]x' ↔ 'o[u]e', 'o[u]', '[u]o' → u = n

+ '[m]u' ↔ '[m]n' và đứng trước mạo từ the → có thể là giới từ in → m = i

+ 'x[b]' ↔ 'o[b]' và đứng trước mạo từ the → có thể là giới từ of → b = f

+ '[l]myt' ↔ '[l]ith' và đứng trước mạo từ the → có thể là giới từ with → l = w

+ 'yx [g]n' ↔ 'to [g]e' → có thể là to be, g = b

+ 'vu[p]' ↔ 'an[p]' với tần suất xuất hiện đứng thứ 2 trong các chữ có 3 ký tự → có thể là and → p = d

+ 'dnv[h]', 'ln[h]n'↔ 'pea[h]', 'we[h]e' → có thể là pear, were → h = r

+ 'xu[a]n' ↔ 'on[a]e' → có thể là once → a = c

+ 'v[ii]', 'av[ii], 'av[ii]np' ↔ 'a[ii]', 'ca[ii]', 'ca[ii]ed' → ii = ll, i = l

+ 'hnvii[d]' ↔ 'reall[d]' → có thể là really → d = y

+ ‘ym[c]n’, ‘lx[c]nu’ ↔ ‘ti[c]e’, ‘wo[c]en’ → c = m

+ ‘gv[rr]nh’, ‘ixu[r]’, ‘hm[r]ty’, ‘gm[r]’ ↔ ‘ba[rr]er’, ‘lon[r]’, ‘ri[r]ht’, ‘bi[r]’ → r = g

+ ‘[e]vd’, ‘[e]xlnh’, ‘td[e]n’, ‘tni[e]’ ↔ ‘[e]ay’, ‘[e]ower’, ‘hy[e]e’, hel[e] → e = p

+ ‘[z]e’, ‘x[z]y’ ↔ ‘[z]p’, ‘o[z]t’ → z = u

+ ‘bn[f]nh’, ‘cx[f]np’, ‘v[f]xmp’, ‘un[f]nh’ ↔ ‘fe[f]er’, ‘mo[f]ed’, ‘a[f]oid’, ‘ne[f]er’ → f = v

+ ‘[o]zqy’, ‘[o]zpp’, ‘[o]zgmivuy’ ↔ ‘[o]ust’, ‘[o]udd’, ‘[o]ubilant’ → o = j

+ ‘n[k]yhv’, ‘qn[k]mqy’, ‘n[k]enhyq’ ↔ ‘e[k]tra’, ‘se[k]ist’, ‘e[k]perts’ → k = x

+ ‘cv[s]n’, ‘im[s]n’, ‘giva[s]’↔ ‘ma[s]e’, ‘li[s]e’, ‘blac[s]’ → s = k

+ ‘[j]zmy’ ↔ ‘[j]uit’, vì chỉ còn z và q → j = q

+ ‘ehm[w]n’ ↔ ‘pri[w]e’ → w = z

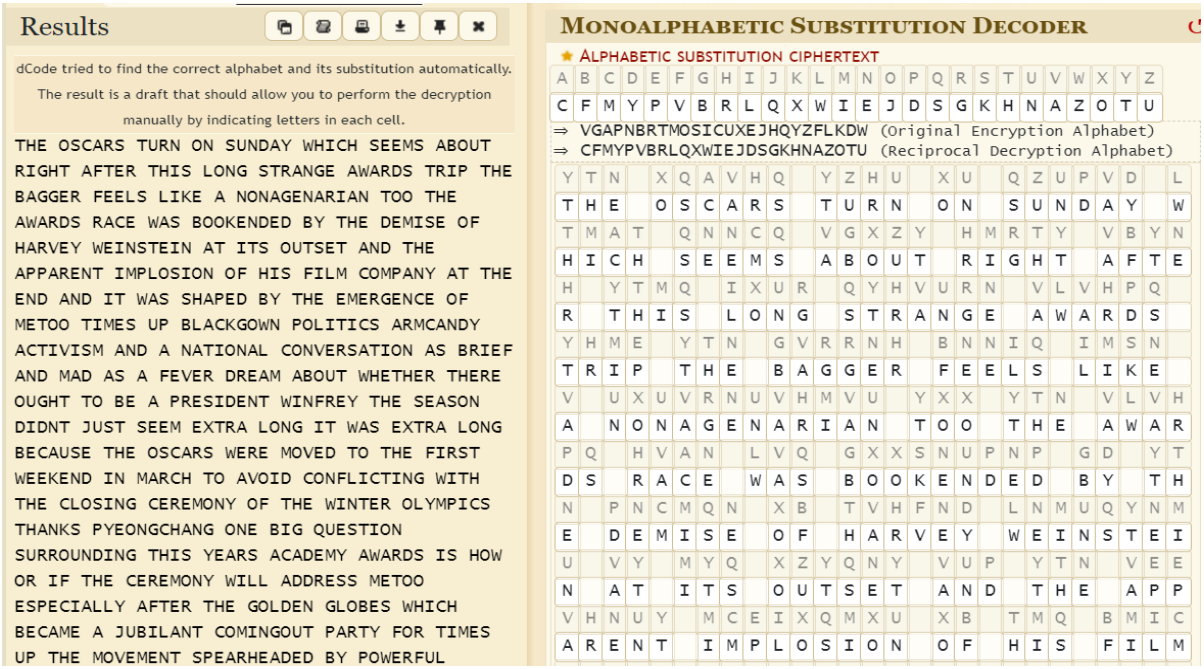
Key mã hoá:

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

Key giải mã:

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

- Kiểm tra lại kết quả (dùng https://www.dcode.fr/monoalphabetic-substitution):



Hình. Kết quả giải mã đoạn text bằng dcode

- Bản rõ hoàn chỉnh được gửi cùng báo cáo Lab 01 với tên ***task3_Decrypt.txt***.

*** Bài nâng cao:** Giải mã đoạn bản mã trong cuốn sách của Edgar Allan Poe – Con bọ hung vàng

53‡‡†305))6;4826)4‡.)4‡);806*;48†8¶60))85;1‡(;:‡*8†83(88)*
5*†;46(;88*96*?;8)*‡(;485);5*†2:*‡(;4956*2(5*-4)8¶8*;40692*
85);)6†8)4‡‡;1(‡9;48081;8:8‡1;48†85;4)485†528806*81(‡9;48*
;(88;4(‡?34;48)4‡;161;:188;‡?;*

- Sử dụng https://www.cryptool.org/en/cto/n-gram-analysis để tiến hành phân tích đoạn bản mã, dựa vào kết quả phân tích có:

+ 3 ký tự có tần suất cao nhất: '8', ';', '4' → khả năng cao là 'e'.

+ 'ee' là cụm chữ thường gặp trong tiếng anh, xét tần suất xuất hiện các cặp 2 ký tự '88', ';;', '44' có tần suất của '88' là cao nhất → '8' = e

+ 'the' là một từ 3 chữ thường xuất hiện nhiều nhất trong tiếng anh, xét bảng tuần suất thấy '[;4]8' ↔ '[;4]e' suất hiện 7 lần → ';' = t , '4' = h

+ Dựa vào tần số, thấy ')', '‡' , '*', '5', '6' có khả năng là thay thế của 'a', 'o', 'i', 'n', 's', 'r'

+ Các từ 2 ký tự thường gặp trong tiếng anh còn có 'an, 'in', và chúng đều kết thúc bằng 'n' và 'oo', 'ss' thường hay gặp

→ xét các tần suất của các chữ 2 ký tự được tạo bởi ')', '‡' , '*', '5', '6' thấy '6*' (5 lần), '5*' (3 lần), '5)' (3 lần), '‡‡' (2 lần), '))' (2 lần)

→ khả năng cao '*' = n, '5' = a, '6' = i

+ Giả sử '5' ↔ 'a' đầu tiên là một mạo từ → 'a 3‡ ‡ ' → '‡' = o, ')' = s

+ 'an[†]' → khả năng cao là 'an[d]' → † = d

+ 'thi[(]teen', 'no[(]theast' ↔ 'thi[r]teen', 'no[r]theast' → '(' = r

+ '[3]ood', 'de[3]ree' ↔ '[g]ood', 'de[g]ree' → '3' = g

+ 'thro[?]gh' ↔ 'thro[u]gh' → '?' = u

+ '[9]inutes' ↔ '[m]inutes' → '9' = m

+ 'g[0]ass' ↔ 'g[l]ass' → '0' = l

+ 'de[¶]ils' ↔ 'de[v]ils' → '¶' = v

+ '[1]rom', 'le[1]t' ↔ '[f]rom', 'le[f]t' → '1' = f

+ 'fort[:]', 'e[:]e', 'fift[:]' ↔ 'fort[y]', 'e[y]e', 'fift[y]' → ':' = y

+ '[2]y' ↔ '[b]y' → '2' = b

+ 'bran[-]h' ↔ 'bran[c]h' → '-' = c

+ 'bisho[.]' ↔ 'bisho[p]' → '.' = p

	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
key	5	2	-	†	8	1	3	4	6			0	9	*	‡		.	(	)	;	?	¶			:	

- Bản rõ:

Agoodglassinthebishopshostelinthedevilsseattwentyonedegreesandthirteenminutesnortheastandbynorthmainbranchseventhlimbeastsideshootfromthelefteyeofthedeathsheadabeelinefromthetreethroughtheshotfiftyfeetout

(*A good glass in the bishop's hostel in the devil's seat forty-one degrees and thirteen minutes northeast and by north main branch seventh limb eastside shoot from the left eye of the deaths head a beeline from the tree through the shot fifty feet out*)

4. Playfair cipher

- **Mật mã Playfair** là một hệ mã hóa nhiều chữ.

- **Cơ chế hoạt động:** sử dụng một ma trận chữ cái 5x5 trên cơ sở một từ khóa, điền các chữ cái của từ khóa (bỏ các chữ trùng), điền những vị trí còn lại của ma trận với các chữ cái khác của bảng chữ cái; I, J có thể ở trên cùng một ô của ma trận (hoặc thường bỏ đi J, nếu trong bản rõ chứa J thì nó được thay bằng I).

- **Thuật toán mã hoá:**

+ Bản rõ được chia thành các cặp gồm 2 chữ cái, nếu số chữ cái lẻ thì thêm một chữ cái không có thật vào cuối. *(Em sử dụng một chữ cái không có thật là chữ 'X')*

+ Không được có cặp chữ giống nhau, nếu có thì chia ra và ghép với một chữ cái không có thật vào chữ cái trước. *(Em sử dụng một chữ cái không có thật là chữ 'X')*

+ Cả 2 chữ cái trong cặp nằm trong cùng một cột: lấy chữ cái bên dưới mỗi chữ (quay trở lại đầu nếu ở dưới cùng).

+ Cả 2 chữ cái trong cặp nằm trong cùng một hàng: lấy chữ cái bên phải mỗi chữ (quay trở lại ngoài cùng bên trái nếu ở ở vị trí ngoài cùng bên phải).

+ 2 chữ cái không cùng hàng/cột: Tạo một hình chữ nhật với hai chữ cái và lấy các chữ cái ở góc đối diện nằm ngang của hình chữ nhật.

- Các đoạn code trong Playfair:

+ Đoạn code xử lý đoạn text đưa vào: chuyển tất cả các ký tự về chữ in hoa, xoá các ký tự không là chữ cái (các ký tự đặc biệt, khoảng trắng,...).

```
char keyMatrix[5][5];   //tạo ma trận khoá

void changeText(string& text)
{
    for (int i = 0; i < text.length(); i++)
    {
        text[i] = toupper(text[i]);  //chuyển tất cả kí tự trong text về chữ in hoa
    }
    int i, count = 0;
    for (i = 0; i < text.length(); i++)
    {
        if ((text[i] >= 'A') && (text[i] <= 'Z'))
        {
            text[count++] = text[i];
            count;
        }
        else //xoá các ký tự không là chữ
        {
            text.erase(text.begin() + i);
            i--;
        }
    }
}
```

Hình. Đoạn code xử lý chuyển các ký tự sang chữ in hoa, xoá các ký tự không là chữ

+ Đoạn code tạo ma trận khoá 5x5 dựa trên key và bảng chữ cái (tiếng anh):

```
void createKeyMatrix(string key, char keyMatrix[5][5])
{
    int flag[26] = {}; //tạo mảng để đánh dấu sự xuất hiện của ký tự, mặc định = 0
    int row = 0, col = 0;
    for (int i = 0; i < key.length(); i++) //thêm các chữ trong key vào ma trận trước
    {
        if (key[i] == 'J') //vì lược bỏ J, I/J ở cùng 1 ô nên các vị trí có j sẽ được thay bằng i
            key[i] = 'I';
        if (flag[key[i] - 'A'] == 0) //thêm kí tự chưa có vào ma trận và bật cờ của chữ cái đó
        {
            flag[key[i] - 'A'] = 1;
            keyMatrix[row][col++] = key[i];
        }
        if (col == 5)//hết hàng xuống hàng mới
        {
            row++;
            col= 0;
        }
    }
    for (char c='A'; c <= 'Z'; c++) //thêm các chữ cái trong bảng chữ cái vào ma trận
    {
        if (c == 'J') //vì lược bỏ J
            continue;
        if (flag[c - 'A'] == 0) //thêm kí tự chưa có vào ma trận và bật cờ của chữ cái đó
        {
            flag[c - 'A'] = 1;
            keyMatrix[row][col++] = c;
        }
        if (col == 5)//hết hàng xuống hàng mới
        {
            row++;
            col = 0;
        }
    }

}
```

Hình. Đoạn code tạo ma trận khoá 5x5

+ Đoạn code khai báo cấu trúc của chữ cái và hàm trả về vị trí (hàng, cột) của chữ cái:

```
typedef struct //khai bao cau truc vi tri (hàng, cột) cua chu cai
{
    int row;
    int column;
} pos;

pos getPosition(char c)
{
    for (int i = 0; i < 5; i++)
    {
        for (int j = 0; j < 5; j++)
        {
            if (c == keyMatrix[i][j])
            {
                pos position = { i, j };
                return position;
            }
        }
    }
}
```

Hình. Đoạn code khai báo cấu trúc và hàm trả về vị trí của chữ cái

+ Đoạn code xử lý chuỗi đưa vào theo thuật toán Playfair:

```
void handleText(string& text)
{

    for (int i = 0; i < text.length(); i++) // thay J bằng I
    {
        if (text[i] == 'J')
            text[i] = 'I';
    }
    for (int j = 1; j < text.length(); j = j + 2) // biến đổi text theo 2 ý đầu của thuận toán mã hoá
    {
        if (text[j] == text[j - 1])
            text.insert(text.begin() + j, 'X');
    }
    if (text.length() % 2 == 1) //nếu chuỗi lẻ, thêm ký tự X
        text = text + 'X';
}
```

Hình. Đoạn code xử lý đoạn text theo thuật toán Playfair

+ Đoạn code Playfair Encryption:

```
void encrypt(string text)
{
    string output = "";
    for (int i = 0; i < text.length(); i += 2)
    {
        //lấy vị trí của 2 ký tự kề nhau
        pos p1 = getPosition(text[i]);
        pos p2 = getPosition(text[i + 1]);
        int x1 = p1.row; int y1 = p1.column;
        int x2 = p2.row; int y2 = p2.column;

        if (x1 == x2) // cùng hàng dịch phải 1, mod 5 nếu quá vòng
        {
            output = output + keyMatrix[x1][(y1 + 1) % 5];
            output = output + keyMatrix[x2][(y2 + 1) % 5];
        }
        else if (y1 == y2) // cùng cột dịch xuống 1, mod 5 nếu quá vòng
        {
            output = output + keyMatrix[(x1 + 1) % 5][y1];
            output = output + keyMatrix[(x2 + 1) % 5][y2];
        }
        else //khác, đổi cột
        {
            output = output + keyMatrix[x1][y2];
            output = output + keyMatrix[x2][y1];
        }
    }
    cout << "\nCiphertext: " << output;
}
```

Hình. Đoạn code Playfair Encryption

+ Đoạn code Playfair Decryption:

```
void decrypt(string text)
{
    string output = "";
    for (int i = 0; i < text.length(); i += 2)
    {
        //lấy vị trí của 2 ký tự kề nhau
        pos p1 = getPosition(text[i]);
        pos p2 = getPosition(text[i + 1]);
        int x1 = p1.row; int y1 = p1.column;
        int x2 = p2.row; int y2 = p2.column;

        if (x1 == x2) // cùng hàng dịch phải 1, mod 5 nếu quá vòng
        {
            output = output + keyMatrix[x1][abs(y1 - 1 + 5) % 5];
            output = output + keyMatrix[x2][abs(y2 - 1 + 5) % 5];
        }
        else if (y1 == y2) // cùng cột dịch xuống 1, mod 5 nếu quá vòng
        {
            output = output + keyMatrix[abs(x1 - 1 + 5) % 5][y1];
            output = output + keyMatrix[abs(x2 - 1 + 5) % 5][y2];
        }
        else //khác, đổi cột
        {
            output = output + keyMatrix[x1][y2];
            output = output + keyMatrix[x2][y1];
        }
    }
    cout << "\nPlaintext: " << output;
}
```

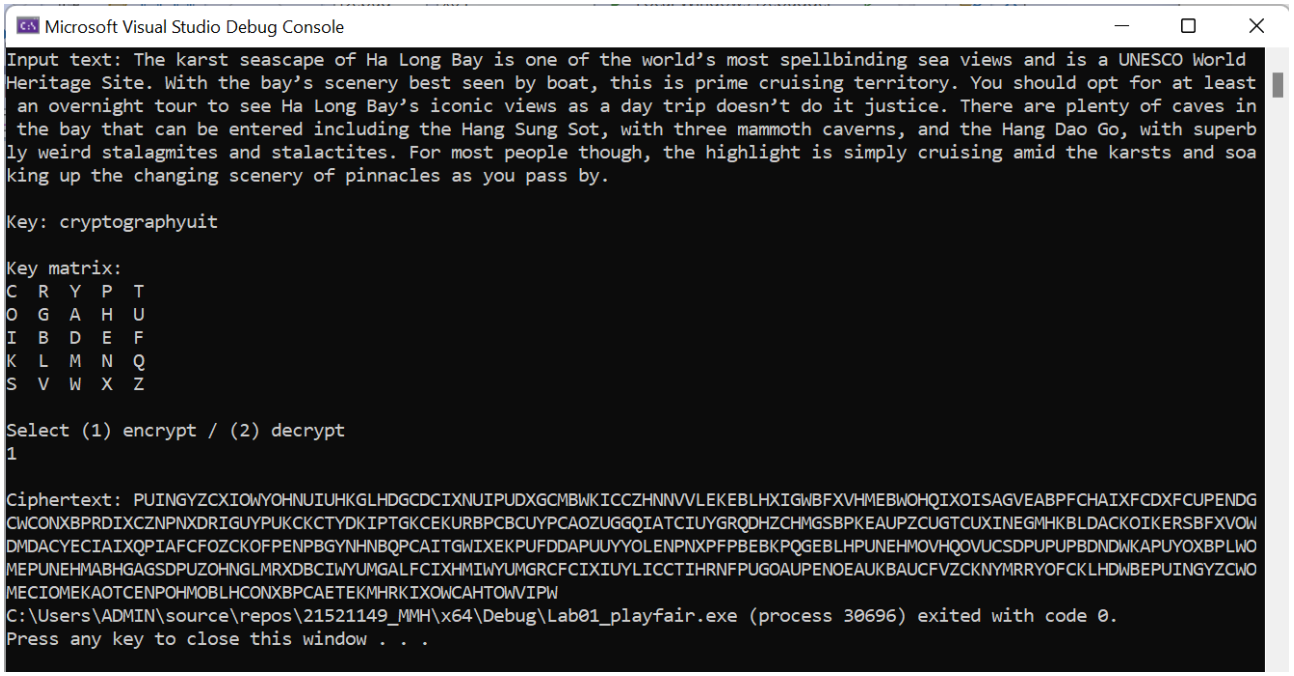
Hình. Đoạn code Playfair Encryption

a, Kiểm tra lại chương trình vừa viết:

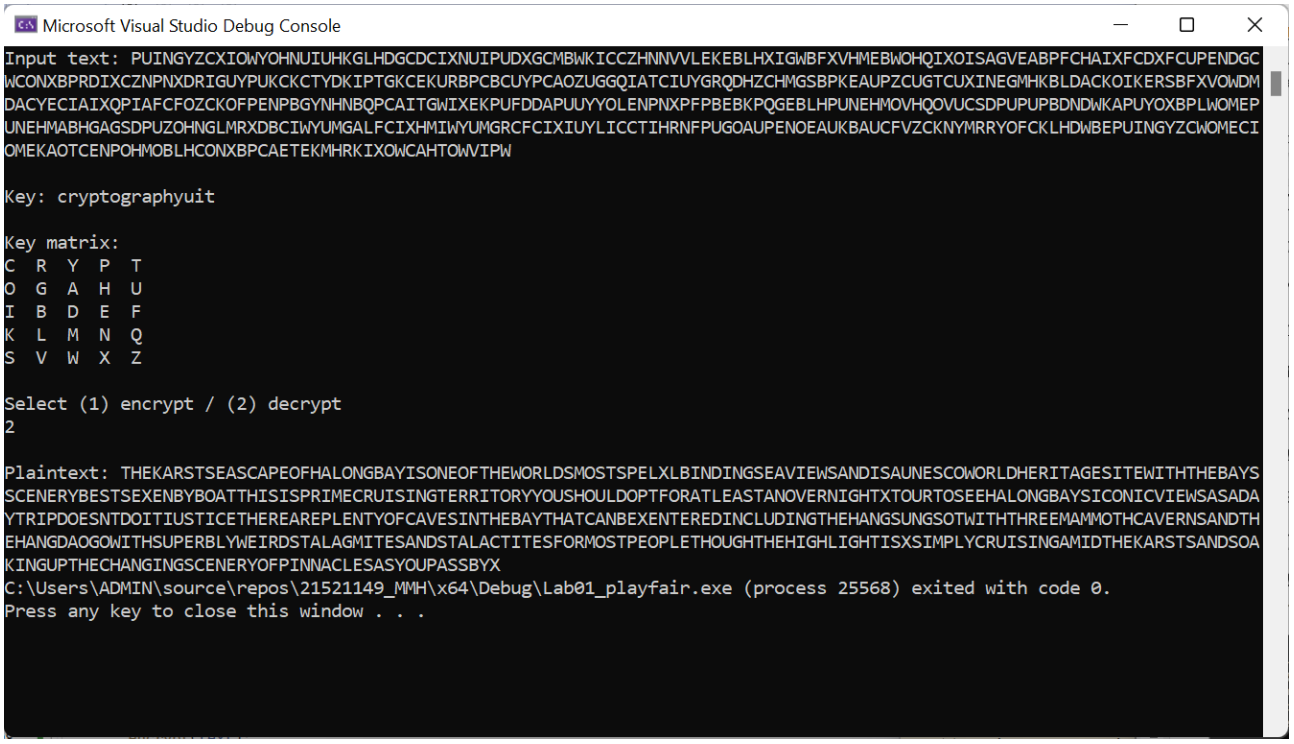
- Text: *The karst seascape of Ha Long Bay is one of the world's most spellbinding sea views and is a UNESCO World Heritage Site. With the bay's scenery best seen by boat, this is prime cruising territory. You should opt for at least an overnight tour to see Ha Long Bay's iconic views as a day trip doesn't do it justice. There are plenty of caves in the bay that can be entered including the Hang Sung Sot, with three mammoth caverns, and the Hang Dao Go, with superbly weird stalagmites and stalactites. For most people though, the highlight is simply cruising amid the karsts and soaking up the changing scenery of pinnacles as you pass by.*

- Key*: cryptographyuit*

- Các kết quả thu được khi chạy chương trình:

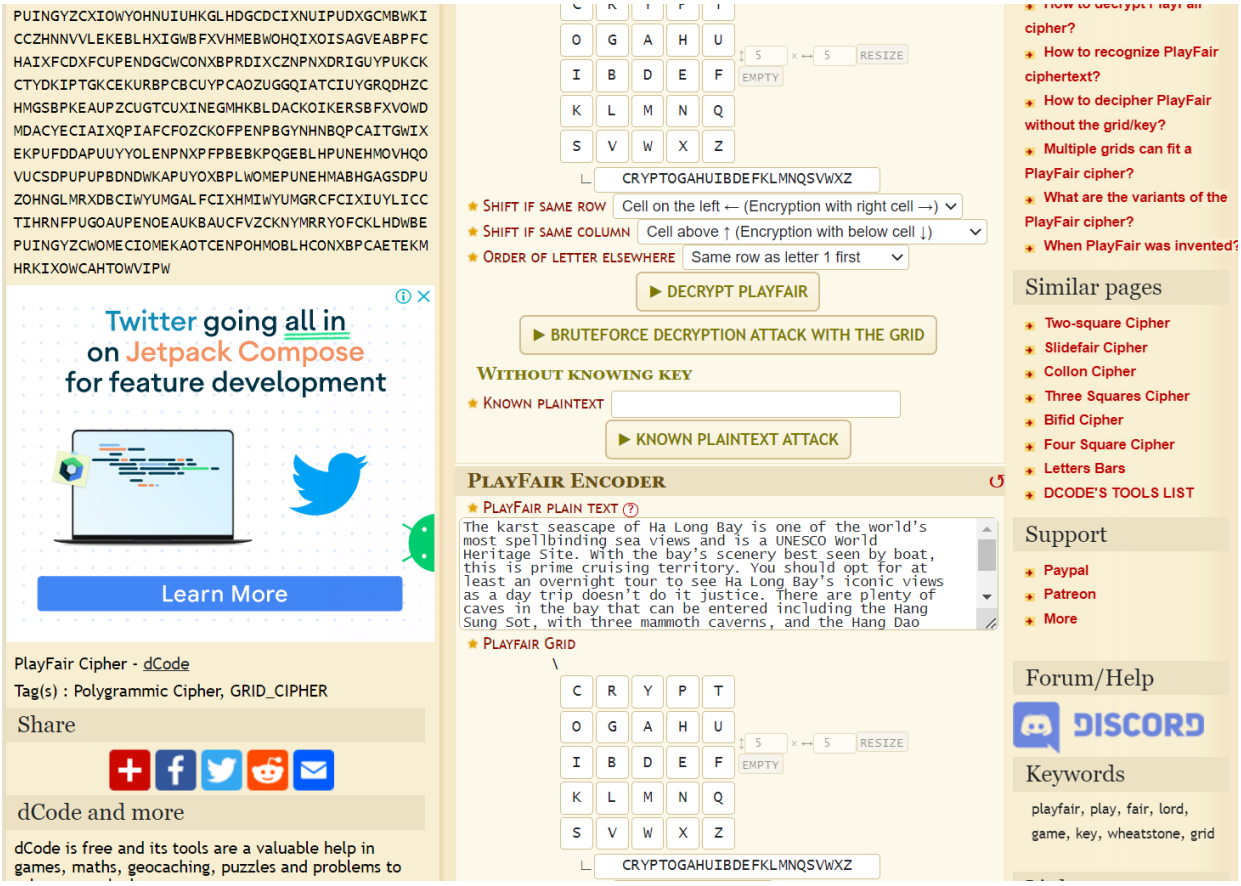


Hình. Kết quả mã hoá đoạn text bằng chương trình vừa viết



Hình. Kết quả giải mã đoạn text bằng chương trình vừa viết

- Kiểm tra lại kết quả (dùng https://www.dcode.fr/vigenere-cipher):



Hình. Kết quả mã hoá đoạn text bằng dcode

Hình. Kết quả giải mã đoạn text bằng dcode

b, Mã hoá bản rõ dựa trên ma trận khoá cho trước

- Đoạn text: LEDOANTRAMYATTTUITCOBONLACAMTUCAUTUDANGDAQUYBINGAN

- Ma trận khoá:

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

Figure 5:Playfair matrix for task

- Mã hoá:

+ Tiến hành xử lý đoạn text trên, tách thành các cặp 2 ký tự và xử lý chuỗi lặp, lẻ nếu có

LE | DO | AN | TR | AM | YA | TX | TX | TU | IT | CO | BO | NL | AC |AM | TU |CA | UT | UD | AN | GD | AQ | UY | BI |NG | AN

+ Xử lý mã hoá theo từng cặp, thu được các cặp chữ tương ứng:

GD | FL | IV | BL | OI | VO | HW | HW | BP | TH | FA | MR | PA | IN | OI | BP | NI | PB | PK | IV | LE | GN | SZ | IT | QA | IV

+ Kết quả: GDFLIVBLOIVOHWHWBPTHFAMRPAINOIBPNIPBPKIVLEGNSZITQAIV

5. Polyalphabetic cipher – Vigenère

- **Mật mã Vigenère** là phương pháp mã hoá thay thế đa bảng chữ cái bằng cách sử dụng xen kẽ một số phép mã hóa Caesar khác nhau dựa trên các chữ cái của từ khóa.

- **Thuật toán Vigenère:**

+ Encryption: Ei = (Pi + Ki) mod 26

+ Decryption: Pi = (Ei – Ki + 26) mod 26

- Các đoạn code trong Vigenère:

+ Khi tiến hành giải mã/mã hoá cần tạo một key mới có chiều dài bằng với đoạn text đã cho bằng cách lặp lại key trong chuỗi key mới:

```
void createNewKey(string text, string key, string& newkey) //tạo key mới có độ dài bằng độ dài text bằng cách lặp lại key
{
    int i, j;
    for (i = 0, j = 0; i < text.length(); i++, j++)
    {
        if (((text[i]>='A') && (text[i]<='Z'))|| ((text[i] >= 'a') && (text[i] <= 'z')))//nếu ký tự là chữ thì lặp lại trong key mới, còn không giữ nguyên
        {
            if (j == key.length())
                j = 0; //khi j = độ dài chuỗi key, lặp lại key trong chuỗi mới
            newkey[i] = key[j];
        }
        else
        {
            j--;
        }
    }
}
```

Hình. Đoạn code tạo key mới

+ Đoạn code Vigenére Encrypt:

```
void encrypt(string text, string key)
{
    char alphabet[26] = { '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' };
    string output = text;

    for (int j = 0; j < text.length(); j++)
    {
        for (int i = 0; i < 26; i++)
        {
            if (text[j] == alphabet[i])
            {
                output[j] = alphabet[abs(text[j]+key[j]) % 26];
                break;
            }
        }
    }
    cout << "\nCiphertext: " << output;
}
```

Hình. Đoạn code Vigenére Encrypt

+ Đoạn code Vigenére Decrypt:

```
void decrypt(string text, string key)
{
    char alphabet[26] = { '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' };
    string output = text;

    for (int j = 0; j < text.length(); j++)
    {
        for (int i = 0; i < 26; i++)
        {
            if (text[j] == alphabet[i])
            {
                output[j] = alphabet[abs(text[j] - key[j] + 26) % 26];
                break;
            }
        }
    }
    cout << "\nPlaintext: " << output;
}
```

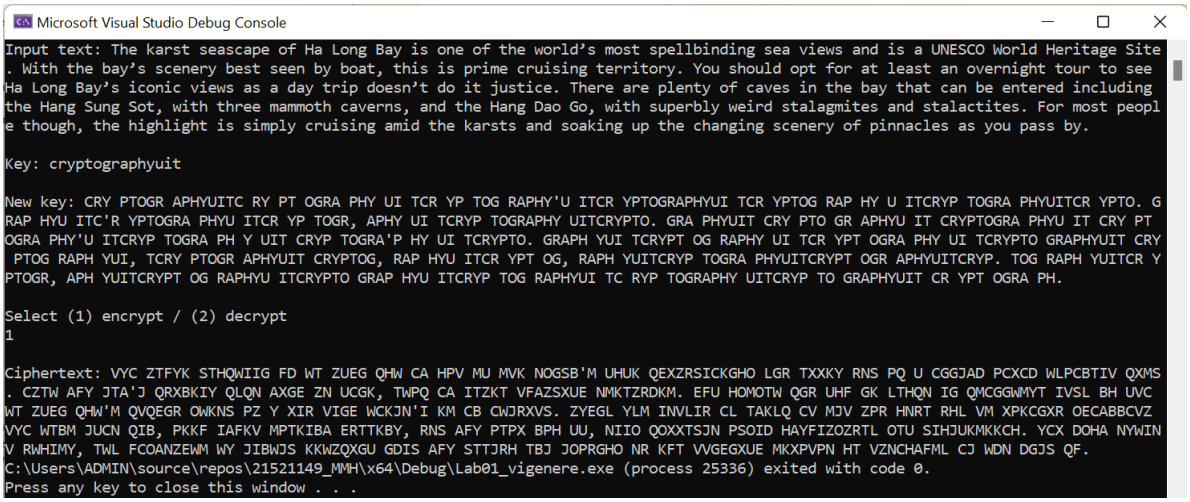
Hình. Đoạn code Vigenére Decrypt

- Tiến hành kiểm tra chương trình vừa viết:

+ Text: *The karst seascape of Ha Long Bay is one of the world's most spellbinding sea views and is a UNESCO World Heritage Site. With the bay's scenery best seen by boat, this is prime cruising territory. You should opt for at least an overnight tour to see Ha Long Bay's iconic views as a day trip doesn't do it justice. There are plenty of caves in the bay that can be entered including the Hang Sung Sot, with three mammoth caverns, and the Hang Dao Go, with superbly weird stalagmites and stalactites. For most people though, the highlight is simply cruising amid the karsts and soaking up the changing scenery of pinnacles as you pass by.*

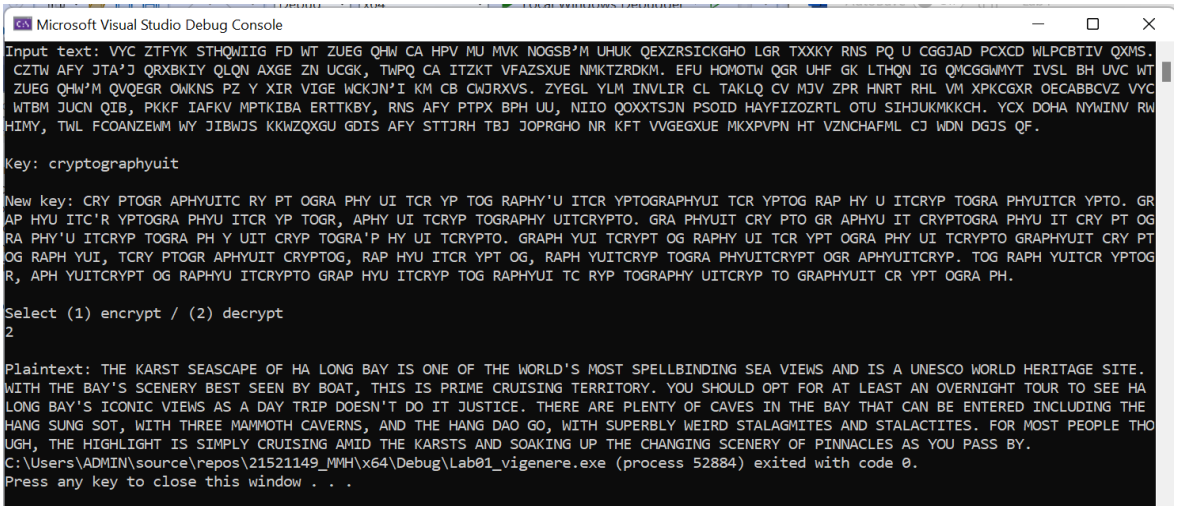
+ Key*: cryptographyuit*

+ Kết quả thu được khi chạy chương trình:



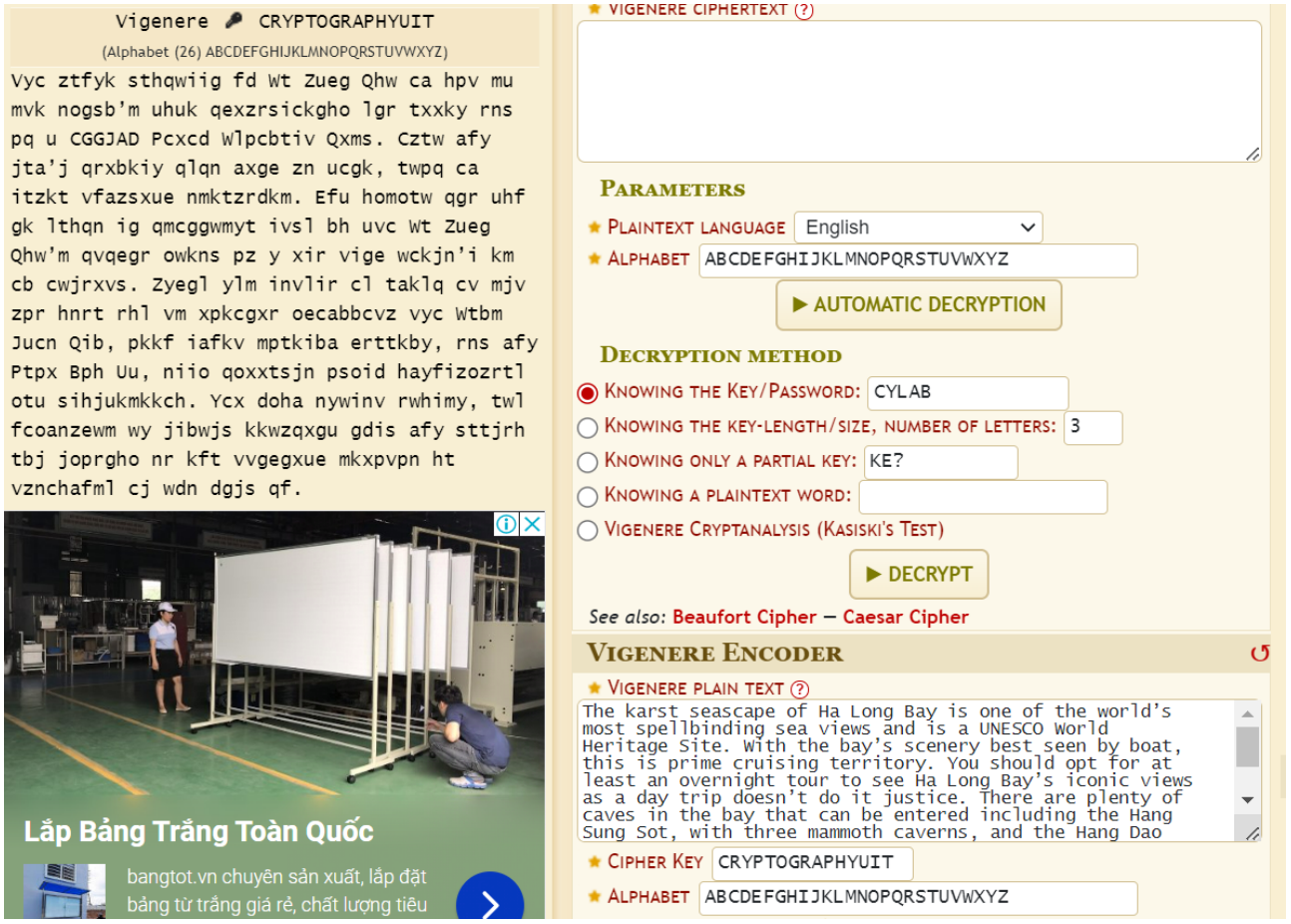
Hình. Kết quả mã hoá đoạn text bằng chương trình vừa viết

+ Giải mã ngược lại đoạn kết quả trên:



Hình. Kết quả giải mã đoạn text bằng chương trình vừa viết

- Kiểm tra lại kết quả (dùng https://www.dcode.fr/vigenere-cipher):



Hình. Kết quả giải mã đoạn text bằng dcode

6. Other ciphers:

a, Tìm flag của thông điệp

- Thông điệp:

TXpNek5ETXpNek16TXpNMU16TXpNak16TXpVek16TTVNek16TlRNek16QXpNek0xTXpNek5ETXpNelF6TkRNMk16TXpORE16TXpjek16TTFNek16TWpNek16UXpNek14TXpNek5UTXpNekF6TXpNME16TXpPRE16TXpVek16TTU=

- Giải mã:

+ Nhận thấy dấu '=' ở cuối thông điệp, ta giải mã thông điệp sang Base64 (sử dụng https://www.base64decode.org/), thu được thông điệp mới:

MzMzNDMzMzMzMzM1MzMzMjMzMzUzMzM5MzMzNTMzMzAzMzM1MzMzNDMzMzQzNDM2MzMzNDMzMzczMzM1MzMzMjMzMzQzMzMxMzMzNTMzMzAzMzM0MzMzODMzMzUzMzM5

+ Tiếp tục giải mã thông điệp mới trên sang Base64 (sử dụng https://www.base64decode.org/), thu được thông điệp mới:

333433333335333233353339333533303335333433343436333433373335333233343331333533303334333833353339

+ Thông điệp mới thu được ở trên là mã Thập lục phân (hexadecimal), ta chuyển thông điệp đó sang chữ - mã ASCII (sử dụng https://anytexteditor.com/vi/hex-to-ascii), thu được thông điệp mới: *34333532353935303534344634373532343135303438353 9*

+ Tiếp tục chuyển thông điệp trên sang chữ – mã ASCII (sử dụng https://anytexteditor.com/vi/hex-to-ascii), thu được thông điệp mới: *43525950544F475241504859*

+ Tiếp tục chuyển thông điệp trên sang chữ – mã ASCII (sử dụng https://anytexteditor.com/vi/hex-to-ascii), thu được thông điệp mới: *CRYPTOGRAPHY*

- *Kết luận:* Vậy flag của thông điệp là **CRYPTOGRAPHY**.

b, Mật mã cổ điển khác – Affine:

- **Mật mã Affine** là một loại mật mã thay thế một chữ cái, trong đó mỗi chữ cái trong bảng chữ cái được ánh xạ thành số tương đương của nó, được mã hóa bằng một hàm toán học đơn giản và được chuyển đổi trở lại thành một chữ cái.

- Thuật toán Affine:

+ Encryption: $E(p, k) = (a*p + b) \bmod 26$

+ Decryption: $P(c, k)= (a^{-1}(c - b)) \bmod 26$

Với: k là một bộ gồm 2 thành phần k = (a, b), a^{-1} là module nghịch đảo của a.

- Các đoạn code trong Affine:

+ Đoạn code Affine Encrypt:

```
void encrypt(string text, int a, int b)
{
    char alphabet[26] = { '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' };
    char alphabet1[26] = { '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' };
    string output = text;
    for (int j = 0; j < text.length(); j++)
    {
        for (int i = 0; i < 26; i++)
        {
            if (text[j] == alphabet[i])
            {
                output[j] = alphabet[abs(a * i + b) % 26];
                break;
            }
            else if (text[j] == alphabet1[i])
            {
                output[j] = alphabet1[abs(a * i + b) % 26];
                break;
            }
        }
    }
    cout << "\nCiphertext: " << output;
}
```

Hình. Đoạn code Affine Encrypt

+ Đoạn code Affine Decrypt:

```
void decrypt(string text, int a, int b)
{
    char alphabet[26] = { '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' };
    char alphabet1[26] = { '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' };
    int modulDN = 0, check = 0;
    string output = text;

    for (int i = 0; i <= 25; i++)//tìm modul đảo nghịch của a
    {
        check = (a * i) % 26;
        if (check == 1)
        {
            modulDN = i;
        }
    }

    for (int j = 0; j < text.length(); j++)
    {
        for (int i = 0; i < 26; i++)
        {
            if (text[j] == alphabet[i])
            {
                output[j] = alphabet[abs(modulDN * (i - b + 26))% 26];
                break;
            }
            else if (text[j] == alphabet1[i])
            {
                output[j] = alphabet1[abs(modulDN * (i - b + 26)) % 26];
                break;
            }
        }
    }
    cout << "\nPlaintext: " << output;
}
```

Hình. Đoạn code Affine Decrypt

- Tiến hành kiểm tra chương trình vừa viết:

+ Text: *I'm from VietNam*

+ Key*: a = 3, b = 9*

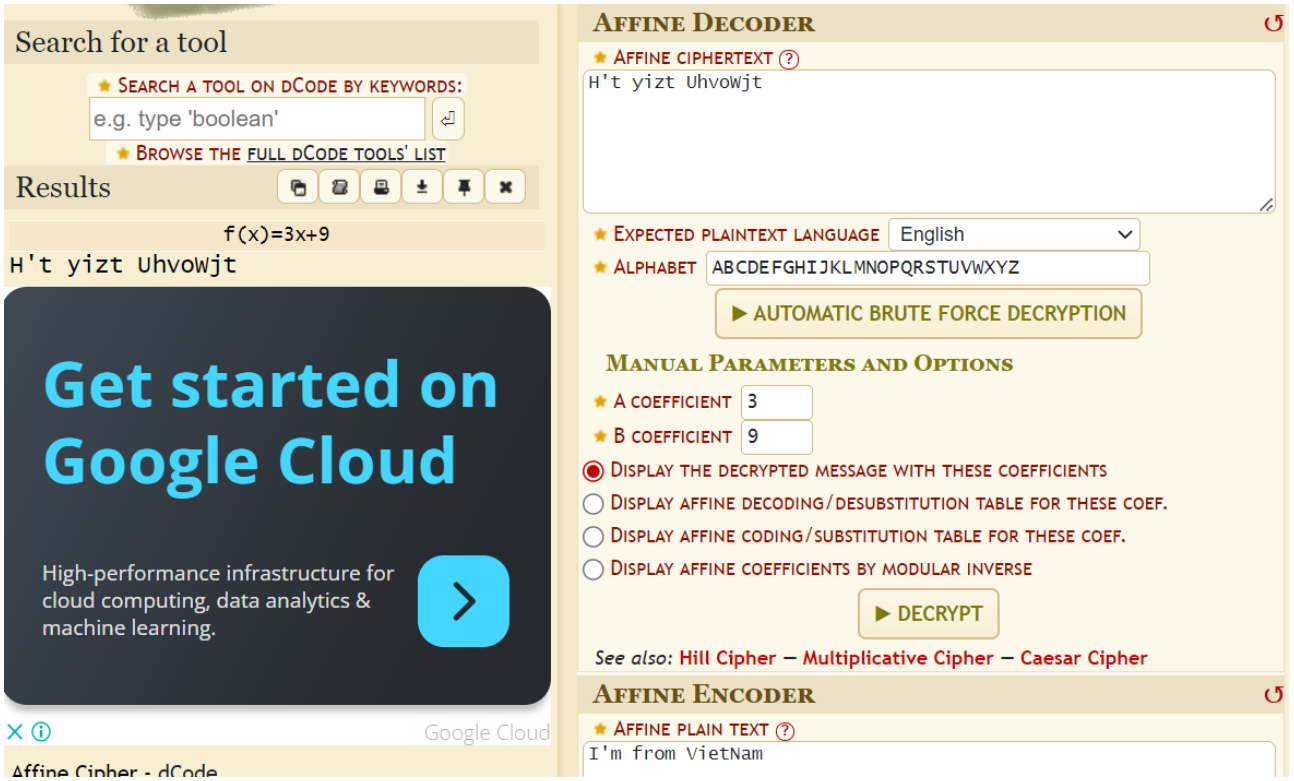
+ Kết quả thu được:

Hình. Kết quả mã hoá đoạn text bằng chương trình vừa viết

+ Giải mã ngược lại đoạn kết quả trên:

Hình. Kết quả giải mã đoạn text bằng chương trình vừa viết

- Kiểm tra lại kết quả (dùng https://www.dcode.fr/affine-cipher):



Hình. Kết quả giải mã đoạn text bằng dcode