Data Security Programs

المكتبات

سوف نحتاج إلى كل من المكتبات التالية في جميع البرامج تقريباً

```
#include <iostream.h>
#include <string.h>
#include <stdio.h>
#include <conio.h>
#include<math.h>
```

لعمليات الإدخال والطباعة
دالة strlen ودالة strchar
دالة gets ودالة puts
دالة clrscr ودالة getch
للدوال الرياضية

ملاحظات عامة

1:- يتم طرح ٩٧ من كل حرف لتحويلهُ من ASCII إلى تسلسل.
ثم يتم تحويلهُ من تسلسل إلى ASCII بإضافة ٩٧ إلى الحرف.
٢:- يتم أعطاء قيمة فارغة لكل string عند التعريف.
٣:- تكون عملية فك الشفرة عكس عملية التشفير أي أن الجمع يصبح طرح والضرب قسمة وهكذا.
٤:-يتم التعامل مع النص حرف تلوا الأخر عن طريق for loop بعدد أحرف ذلك النص.
٥:- الغاية من استعمال الأقواس في العمليات الحسابية هي لزيادة الأسبقية.

البرنامج الرئيسي
يكون البرنامج الرئيسي عادةً نفسهُ لأغلب خوارزميات وطرق التشفير، ويكتب كالتالي

```
void main()
{
        clrscr ( );
        char *st = "";
        int k;
        cout<<"enter the Key: ";        cin>> k;
        cout<<"Enter E for Encryption\nEnter D for Decryption\n";
        int ch = getch ( );
        if(ch = = 'E' | | ch = = 'e')
        {
                cout<<"enter plain text: ";
                gets (st);
                Enc (st,k);
        }
        if(ch = = 'D' || ch = = 'd')
        {
                cout<<"enter cipher text: ";
                gets (st);
                Dec (st, k);
        }
        getch ( );
}
```

طرق التشفير

سوف نتطرق لمجموعة من طرق التشفير وكتابة كل من دالة التشفير ودالة فك الشفرة لكل من تلك الطرق

1- Column Transposition

```
void Enc(char*p,char Key[10])
{
	float col = strlen (Key);
	float Plen = strlen (p);
	int row = (Plen / col = = (int) (Plen / col) ) ? Plen / col : Plen / col + 1;
	char A1[10][10] = {"\0"}, A2[10][10] = {"\0"};
	int i = 0;
	for (int x = 0; x < col; x++)
		for (int y = 0; y < row; y++)
		{
			if(i = = Plen)
				A1[y][x] = 'x';
			else
				A1[y][x] = p[i++];
		}
	for (i = 0; i < col; i++)
		Key[i] -= 49;
	for (i = 0; i < col; i++)
		for (int j = 0; j < row; j++)
			A2[j][i] = A1[j][Key[i]];
	for (i = 0; i < row; i++)
		for (int j = 0; j < col; j++)
			cout<<A2[i][j];
}
```

```
void Dec (char*c, char Key[10])
{
	float col = strlen (Key);
	float Clen = strlen (c);
	int row = (Clen / col = = (int) (Clen / col) ) ? Clen / col : Clen / col + 1;
	char A1[10][10] = {"\0"}, A2[10][10] = {"\0"};
	int i = 0;
	for (int x = 0; x < row; x++)
		for (int y = 0; y < col; y++)
			A1[x][y] = c[i++];
	for (i = 0; i < col; i++)
		Key[i] -= 49;
	for (i = 0; i < col; i++)
		for (int j = 0; j < row; j++)
			A2[j][Key[i]] = A1[j][i];
	for(i = 0; i < col; i++)
		for (int j = 0; j < row; j++)
			cout<<A2[j][i];
}
```

2- Addition Cipher (Direct Standard Or Caesar Cipher)

Encryption Formula C = (M + K) Mod 26

Decryption Formula M = (C – K)

```
 void Enc(char *p, int k)
{
        char *c;                //cipher text string
        for (int i = 0; i < strlen (p); i++)
                c[i] = ( p[i] – 97 + k ) % 26 + 97;        // C = ( M + K ) Mod 26
        c[i] = '\0';            // لكي لا يأخذ رموز إضافية عدا أحرفهُ ( String ) قفل الـ
        cout<<"the cipher Text are: ";
        puts(c);                // طباعة النص المشفر
}

void Dec (char *c, int k)
{
        char *p;                //plain text string
        for (int i = 0; i < strlen(c); i++)
        {
                if (( p[i] = c[i] – 97 – k ) < 0)     // M = ( C – K )
                        p[i] += 26;             //إذا كان الناتج سالب يتم إضافة ٢٦ ليعود حرف
                p[i] += 97;             // ASCII إعادة التسلسل إلى
        }
        p[i] = '\0';            // لكي لا يأخذ رموز إضافية عدا أحرفهُ ( String ) قفل الـ
        cout<<"the Plan Text are: ";
        puts (p);
}
```

3- Zigzag Transposition Cipher

```
void Enc(char *p)
{
	char *ci = "\0";
	int n = strlen (p) – strlen (p) / 2;
	for (int i = 0, x = 0; i < strlen (p); i++, x++)
	{
		ci[x] = p[i];
		ci[x + n] = p[++i];
	}
	ci[x + n] = '\0';
	puts (ci);
}

void Dec(char *ci)
{
	char *p = "\0";
	int n = strlen (ci) – strlen (ci) / 2;
	for(int i = 0, x = 0; i < strlen (ci); i++, x++)
	{
		p[i] = ci[x];
		p[++i] = ci[x + n];
	}
	p[x+n] = '\0';
	puts (p);
}
```

4- Multiplicative Cipher

Encryption Formula C = (M * K) Mod 26

Decryption Formula M = (C / K)

```
void Enc ( char *p,  int k)
{
        char *c;
        for (int i = 0; i < strlen(p); i++)
                c[i] = (( p[i] – 97) * k) % 26 + 97;
        c[i] = '\0';
        cout<<"the cipher Text are: ";
        puts (c);
}

void Dec ( char *c, int k)
{
        char *p;
        for(int i = 0; i < strlen (c); i++)
        {
                p[i] = c[i] – 97;
                while ( p[i] % k != 0)      // Mod عكس لعملية
                        p[i] += 26;//طالما أن العدد لا يقبل القسمة على المفتاح نضيف ٢٦ إلى العدد
                p[i] = p[i] / k + 97;
        }
        p[i] = '\0';
        cout<<"the Plan Text are: ";
        puts (p);
}
```

5- Standard Reverse Cipher

Encryption Formula C = (25 – M) Mod 26

Decryption Formula M = (25 – C) Mod 26

كما نلاحظ فإن هذه الطريقة تستخدم نفس القانون للتشفير ولفك الشفرة

```
void revers(char *p)
{
        char *c;
        for (int i = 0; i < strlen (p); i++)
        c[i] = (25 – (p[i] – 97 )) % 26 + 97;
        c[i] = '\0';
        cout<<"the cipher Text are: ";
        puts (c);
}


void main()
{
        clrscr ();
        char *st = "\0";
        cout<<"enter The text to Enc or Dec: ";
        gets (st);
        revers (st);
        getch ();
}
```

6- Affine Cipher

Encryption Formula C = (M * K1 + K2) Mod 26

Decryption Formula M = ((C – K2) / K1) Mod 26

من شروط هذه الطريقة أن القاسم المشترك الأعظم بين K1 و ٢٦ يساوي واحد، لذا نحتاج إلى دالة ليجاد القاسم المشترك الأعظم.

```
int GCD ( int x, int y)          // دالة ليجاد العامل المشترك الأعظم
{
        if (y = = 0)
                return (x);
        return ( GCD( y, x % y ));
}
```

طريقة ثانية

```
int GCD ( int x, int y)
{
        while( y != 0 )
        {
                int z = x % y;
                x = y;
                y = z;
        }
        return (x);
}

Enc( char *p, int k1, int k2)
{
        if( GCD ( k1, 26) = = 1)
        {
                char *c;
                for( int i = 0; i < strlen(p); i++)
                        c[i] = (( p[i] – 97 ) * k1 + k2 ) % 26 + 97;
                c[i] = '\0';
                cout<<"the cipher Text are: ";
                puts (c);
        }
        else
                cout<<" unvalued key";
}
```

```
Dec(char *c, int k1, int k2)
{
        if (GCD( k1, 26) = = 1)
        {
                char *p;
                for(int i = 0; i < strlen (c); i++)
                {
                        if (( p[i] = c[i] – 97 – k2) < 0)
                                p[i] += 26;
                        while ( p[i] % k1 != 0)
                                p[i] += 26;
                        p[i] = p[i] / k1 + 97;
                }
                p[i] = '\0';
                cout<<"the Plan Text are: ";
                puts (p);
        }
        else
                cout<<"unvalued Key";
}
```

7- Keyword Mixed

```
char* mixed_array ( char* key1, char kl)
{
	char *key = "";
	int x = 0;
	for (int i = 0; i < strlen (key1); i++)
		if (!strchr ( key, key1[i] ))
		{
			key[x++] = key1[i];
			key[x] = '\0';
		}
	char mix[27], a = 'a';
	x = 0;
	for ( i = kl – 97; x < strlen (key); i++)
		mix[i] = key[x++];
	for (; i < 26; i++)
	{
		if( !strchr ( key, a ))
			mix[i] = a;
		else i--;
		a++;
	}
	for( i = 0; i < kl – 97; i++)
	{
		if (!strchr ( key, a ))
			mix[i] = a;
		else i--;
		a++;
	}
	return mix;
}
```

```
void Enc ( char* p, char* key, char k)
{
        char *c;
        char *mix = mixed_array ( key, k);
        for (int i = 0; i < strlen (p); i++)
                c[i] = mix[p[i] – 97];
        c[i] = '\0';
        puts(c);
}

void KeywordMix::Dec(char *c,char *key,char k)
{
        char* p;
        char* mix = mixed_array ( key, k);
        for (int i = 0; i < strlen (c); i++)
                for (int j = 0; j < 26; j++)
                        if (mix[j] = = c[i])
                        {
                                p[i] = j + 97;
                                break;
                        }
        p[i] = '\0';
        puts (p);
}
```

8- Vigenere Cipher

Encryption Formula	C = (M + K) % 26
Decryption Formula	M = (C – K)

```
void Enc (char *p, char *k)
{
        char *c = "\0";
        int x = 0;
        for (int i = 0; i < strlen (p); i++)
        {
                x = ( x = = strlen (k))? 0 : x;
                c[i] = (( p[i] – 97) + (k[x++] – 97)) % 26 + 97;
        }
        c[i] = '\0';
        puts(c);
}
```

```
void Dec (char *c, char *k)
{
        char *p;
        int x=0;
        for (int i = 0; i < strlen (c); i++)
        {
                x = ( x = = strlen (k))? 0 : x;
                p[i] = (( c[i] – 97) –  (k[x++] – 97));
                p[i] = (p[i] < 0)? p[i] + 26 + 97 : p[i] + 97;
        }
        p[i] = '\0';
        puts(p);
}
```

9- Beaufort Cipher

Encryption Formula C = (K – M)

Decryption Formula M = (K – C)

```
void Enc (char *p, char *k)
{
        char *c = "\0";
        int x = 0;
        for (int i = 0; i < strlen (p); i++)
        {
                x = ( x == strlen (k))? 0 : x;
                c[i] = (( k[x++] – 97) – (p[i] – 97));
                if (c[i] < 0)
                        c[i] += 26;
                c[i] += 97;
        }
        c[i] = '\0';
        puts(c);
}
```

```
void main ( )
{
        clrscr ( );
        char *p = "\0";
        cout<<"Enter E for Encrption\nEnter D for Decrption\n";
        int c = getch( );
        cout<<"enter the keyword: ";
        char *k = "\0";
        gets (k);
        if(c = = 'E' || c = = 'e')
                cout<<"enter plinte text: ";
        else if(c = = 'D' || c = = 'd')
                cout<<"enter cipher text: ";
        gets (p);
        Enc (p, k);
        getch ( );
}
```

10- LFSR(Linear Feedback Shift Register)

```
void Binary (int N, int b[8])
{
        for (int i = 0; i < 8; i++)
        {
                b[i] = N % 2;
                N /= 2;
        }
}

int Digit (int N, int Index)
{
        for (int i = 0; i < Index; i++)
                N /= 10;
        return N % 10;
}
```

```
int Count (int N)
{
	int Counter = 0;
	while (N != 0)
	{
		Counter++;
		N /= 10;
	}
	return Counter;
}

int Shift (int N, int size)
{
	int K = 1;
	for (int i = 1; i < size; i++)
		K *= 10;
	N = N % K * 10;
	return (N);
}

void Enc (char* p, int m[15], int k)
{
	int num2[8], num1[8], x=0;
	for (int i = 0; i < strlen (p); i++)
	{
		Binary (p[i], num1);
		for (int j = 7; j >= 0; j--)
		{
			if (x = = k)
				x = 0;
			num2[j] = num1[j] ^ m[x++];
		}
		for (int h = 7; h >= 0; h--)
			cout<<num1[h];
		cout<<"\t\t";
		for (h = 7; h >= 0;h--)
			cout<<num2[h];
		cout<<"\n";
	}
}
```

```
void main()
{
	int init, con, d = 0, K;
	int ms[15];
	char *st = "";
	cout<<"please enter init value";
	cin>>init;
	cout<<"please enter fun value";
	cin>>con;
	int init1 = init, N = Count(init);
	ms[0] = Digit(init,N-1);
	for (int i = 1; i <= pow (2, N); i++)
	{
		int init2 = Shift (init, N);
		for (int j = 0; j < N; j++)
		{
			if (Digit (con, j))
				if (d = = 0)
				{
					K = Digit (init,j);
					d = 1;
				}
				else
					K ^= Digit (init, j);
		}
		init2 += K;
		init = init2;
		if (init = = init1)
			break;
		ms[i] = Digit(init,N-1);
		d = 0;
	}
	cin>>st;
	Enc (st, ms, i);
}
```

حساب قيمة Euler

```
int prime (int N)
{
	for (int i = 2; i < N / 2; i++)
		if (N % i = = 0)
			return 0;
	return 1;
}

long Euler (int N)
{
	if (prime (N))
		return (N - 1);
	int m = 2, k = 2;
	while (pow (k, m) <= N)
	{
		while (pow(k,m) <= N)
		{
			if (( pow (k, m) = = N) && (prime (k) ))
				return (pow (k, m – 1) * (k – 1));
			m++;
		}
		m = 2;
		k++;
	}
	int p = 2, q = 2;
	while (p * q <= N)
	{
		while( p * q <= N)
		{
			if (( p * q = = N) && (prime (p)) && (prime (q) ))
				return (( p – 1) * (q – 1 ));
			q++;
		}
		q = 2;
		p++;
	}
```

```
        long e = 1;
        int n = N;
        for (int i = 2; i < N / 2; i++)
        {
                int j = 0;
                while (n % i = = 0 && n > 1)
                {
                        j++;
                        n /=i;
                }
                if(j!=0)
                        e *= pow(i,j-1)*(i-1);
        }
        return(e);
}
void main()
{
        clrscr ( );
        int n;
        cin>>n;
        cout<< Euler (n);
        getch ( );
}
```