Bezpieczeństwo systemów informatycznych – laboratorium Zachodniopomorska Szkoła Biznesu			
Grupa/Specjalność	**Imię i nazwisko**	**Data wykonania (rrrr.mm.dd)**	**Tryb studiów**
Game Design	**Sylwester Dawidowicz**	**20.06.2020**	**Z**
Nr laboratorium	**Temat**		
3	Szyfrowanie		

Na tych laboratoriach zadaniem było przygotowanie programu szyfrującego 3 określonymi algorytmami. Większość funkcji oraz wyjaśnień, za co odpowiada dana linia kodu (pomijając te naprawdę trywialne) oznaczyłem stosownym komentarzem w Visual Studio.

1) Szyfr Cezara

Jest to jeden z najprostszych szyfrów. Polega na przesunięciu wybranej litery o określoną liczbę i przypisanie jej nowej wartości. W tym przykładzie posłużyłem się wiadomością zapisaną tylko za pomocą znaków alfabetu łacińskiego. Funkcja wygląda następująco:

```
// Inicjalizacja Szyfru Cezara
caesar(int key)
{
    FILE* input_file_caesar, * output_file_caesar;
    char ch;

    input_file_caesar = fopen("input.txt", "r");
    if (input_file_caesar == NULL)
    {
        puts("Nie mozna otworzyc pliku!");
        exit(1);
    }

    output_file_caesar = fopen("encrypted.txt", "w+");
    if (output_file_caesar == NULL)
    {
        puts("Nie mozna otworzyc pliku!");
        exit(1);
    }
    printf("Wiadomosc do zaszyfrowania:\n");
    fseek(input_file_caesar, 0, SEEK_END);
    size_t length = ftell(input_file_caesar); // Ustalamy długość wiadomości
    rewind(input_file_caesar);
    int i = 0;
    do // Wyświetlamy wiadomość, którą chcemy zaszyfrować
    {
        ch = fgetc(input_file_caesar);
        printf("%c", ch);
        fputc(ch + key, output_file_caesar); // Szyfrujemy wiadomość i zapisujemy zaszyfrowane znaki do pliku
        i++;
    } while (i < length);
```

```
    printf("\n\nWiadomosc zaszyfrowana wyglada nastepujaco:\n");
    rewind(output_file_caesar); // Ustawiamy wskaźnik na początek pliku
    do // Wyświetlamy wiadomość w postaci zaszyfrowanej
    {
        ch = fgetc(output_file_caesar);
        printf("%c", ch);
    } while (ch != EOF);
    printf("\n\nOdszyfrowuje wiadomosc:\n");
    rewind(output_file_caesar);
    i = 0;
    do
    {
        ch = fgetc(output_file_caesar);
        printf("%c", ch - key); // Odszyfrowujemy wiadomość podstawiając w odwrotnym kierunku o wartość klucza
        i++;
    } while (i < length);
    fclose(output_file_caesar);
    fclose(input_file_caesar);
}
```

Rezultat działania programu:

```
C:\Users\sylwe\source\repos\qwerty\Debug\qwerty.exe
1: Szyfr Cezara
2: Homofoniczny szyfr podstawieniowy
3: Wieloalfabetowy szyfr podstawieniowy
4: Zakoncz
Wybierz jedna z powyzszych operacji: 1

Podaj przesuniecie: 5
Wiadomosc do zaszyfrowania:
This message is weird, because i wrote that one. Sylwester Dawidowicz.

Wiadomosc zaszyfrowana wyglada nastepujaco:
Ymnx%rjxxflj%nx%|jnwi1%gjhfzxj%n%|wtyj%ymfy%tsj3%X~q|jxyjw%If|nit|nh⌂3

Odszyfrowuje wiadomosc:
This message is weird, because i wrote that one. Sylwester Dawidowicz.
```

2)_ Homofoniczny szyfr podstawieniowy

Szyfr, w którym każdej literze tekstu jawnego odpowiada inny zbiór symboli kryptogramu (homofonów). Liczba homofonów powinna być zależna od częstotliwości występowania danej litery w tekście naturalnym. Przy każdym szyfrowaniu litery wybierany jest losowo jeden z jej homofonów. W ten sposób zostaje spłaszczony histogram kryptogramu, a wielokrotne szyfrowanie tego samego tekstu daje za każdym razem inny wynik. Cechy te znacząco utrudniają kryptoanalizę opartą na statystyce tekstu. (zródło: pl.wikipedia.org).

W moim przypadku zapisałem literę oraz na następnej pozycji odpowiadającą jej literę szyfru (wszystko w jednej linii). Prezentacja kodu tego szyfru:

```c
// Inicjalizacja homofonicznego szyfru podstawieniowego
homophonic_cipher()
{
    FILE* input_file, * output_file, * dictionary_file;
    char ch, x;

    input_file = fopen("input.txt", "r");
    if (input_file == NULL)
    {
        puts("Nie znaleziono pliku!");
        exit(1);
    }

    output_file = fopen("homophonic.txt", "w+");
    if (output_file == NULL)
    {
        puts("Nie znaleziono pliku!");
        exit(1);
    }
    printf("Wiadomosc do zaszyfrowania:\n");
    int i = 0;
    dictionary_file = fopen("dictionary.txt", "r");
    do
    {
        ch = fgetc(input_file);
        printf("%c", ch); // Wyświetlamy wiadomość, którą chcemy zaszyfrować
        do
        {
            x = fgetc(dictionary_file); // Pobieramy znak z pliku ze słownikiem

            if (x == ch) // Sprawdzamy, czy znaleziono znak w kluczu
            {
                break; // Jeżeli znaleziono, przerywamy pętle; wskaźnik ustawia się na pozycji
            }          // znalezionej litery
        } while (x != EOF);
        x = fgetc(dictionary_file);  // Wywołując kolejne fgetc pobieramy znak o jeden dalej w prawo
        fputc(x, output_file);       // zapisujemy zmienną do pliku
        rewind(dictionary_file);
        i = 0;
    } while (ch != EOF);
    printf("\n\nWiadomosc zaszyfrowana:\n");
    rewind(output_file);
    rewind(dictionary_file);
    fclose(input_file);
    do
    {
        ch = fgetc(output_file);
        printf("%c", ch); // Wyświetlamy wiadomość w postaci zaszyfrowanej
    } while (ch != EOF);
    rewind(output_file);
    printf("\n\nOdszyfrowuje wiadomosc:\n");

    // Odszyfrowujemy wiadomość
    do
    {
        ch = fgetc(output_file);
        do
        {
            x = fgetc(dictionary_file);
            if (x == ch)
            {
                break;
            }
        } while (x != EOF);
        i = ftell(dictionary_file);
        fseek(dictionary_file, i - 2, SEEK_SET); // Przesuwamy wskaźnik pliku o dwa w lewo
        x = fgetc(dictionary_file); // fgetc przesuwa wskaźnik o jeden w prawo, czyli finalnie
        rewind(dictionary_file);    // mamy przesunięcie o jeden w lewo
        printf("%c", x); // Wyświetlamy odszyfrowaną wiadomość
        i = 0;
    } while (ch != EOF);
    fclose(dictionary_file);
    fclose(output_file);
    fclose(input_file);
}
```

Rezultat działania szyfru:

```
C:\Users\sylwe\source\repos\qwerty\Debug\qwerty.exe
1: Szyfr Cezara
2: Homofoniczny szyfr podstawieniowy
3: Wieloalfabetowy szyfr podstawieniowy
4: Zakoncz
Wybierz jedna z powyzszych operacji: 2

Wiadomosc do zaszyfrowania:
This message is weird, because i wrote that one. Sylwester Dawidowicz.

Wiadomosc zaszyfrowana:
Iu15QRD5548DQ15QSD1f6EQ<D&4J5DQ1QSf2nDQnu4nQ2yDWQehLSD5nDfQr4S162S1&PW

Odszyfrowuje wiadomosc:
This message is weird, because i wrote that one. Sylwester Dawidowicz.
```

3) Wieloalfabetowy szyfr podstawieniowy

Każdy znak tekstu jawnego jest podmieniany na inny znak. Sposób podmiany zmienia się cyklicznie w zależności od aktualnej pozycji podmienianego znaku. W tym programie skorzystałem z Szyfru Vigenère'a, polegającego na szyfrowaniu i odkodowywaniu za pomocą tabula recta, czyli specjalnej tablicy o wymiarach 26x26 zawierającej w pierwszym wierszu litery alfabetu w oryginalnej kolejności, a następnie w każdym kolejnym wierszu litery alfabetu przesunięte o jedną kolejną pozycję w lewo:

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

Wyszukujemy pierwszy znak tekstu jawnego w wierszu oraz pierwszy znak klucza w kolumnie i na przecięciu otrzymujemy zakodowany znak. Analogicznie postępujemy dla wszystkich kolejnych znaków. W praktyce kod szyfru wygląda następująco:

```
// Inicjalizacja wieloalfabetowego szyfru podstawieniowego
running_key_cipher()
{
    size_t size = 1;
    FILE* input_file, * output_file, * keystream_file, * code_table;
    input_file = fopen("message.txt", "r");
    char* message;
    fseek(input_file, 0, SEEK_END);
    size_t length = ftell(input_file); // Ustalamy długość wiadomości
    message = malloc(length); // alokacja pamięci dla tablicy stringów
    rewind(input_file);
    char tabula_recta[NUMBER_OF_STRING][MAX_STRING_SIZE] = { "" }; // Określenie wymiarów tablicy
    char ch, x;
    int i = 0;
    int j = 0;
    int k = 0;
    int z = 0;
    printf("Wiadomosc do zaszyfrowania:\n");
    do
    {
        ch = fgetc(input_file);
        message[i] = ch;
        printf("%c", message[i]);
        i++;
    } while (ch != EOF);
    printf("\n");
    code_table = fopen("tabula_recta.txt", "r");
    for (i = 0; i <= 26; i++)
    {
        for (j = 0; j <= 26; j++)
        {
            ch = fgetc(code_table);
            tabula_recta[i][j] = ch; // Uzupełniamy naszą tablicę "tabula recta"
        }
    }
    rewind(code_table);
    fclose(code_table);
    fclose(input_file);
    output_file = fopen("encrypted_message.txt", "w+");
    keystream_file = fopen("keystream.txt", "r");
    int row = 0, column = 0;
    k = 0;
    printf("\n\nWiadomosc zaszyfrowana: \n");
```

(Ciąg dalszy funkcji)

(ciąg dalszy funkcji)

```
/* Szyfrujemy wiadomość odszukując kolejne litery klucza w pierwszym wierszu oraz
litery wiadomości w pierwszej kolumnie. Następnie na przecięciu wiersza z kolumną
mamy nasz zaszyfrowany znak */

// SZYFROWANIE //
do
{
    x = fgetc(keystream_file);
    for (i = 0; i < 26; i++)
    {
        if (tabula_recta[0][i] == x)
        {
            row = i;
            break;
        }
    }
    for (j = 0; j < 26; j++)
    {
        if (tabula_recta[j][0] == message[k])
        {
            column = j;
            k++;
            break;
        }
    }
    fputc(tabula_recta[row][column], output_file); // Zapisujemy szyfr do pliku
    z++;
    printf("%c", tabula_recta[row][column]);
} while (z < strlen(message) - sizeof(int));

printf("\n\nOdszyfrowanie wiadomosci: \n");
rewind(keystream_file);
rewind(output_file);
int calc = 0, key_value = 0, cipher_value = 0;
k = 0;
/* Odszyfrowanie polega na odjęciu wartości INT klucza od szyfru. Każda litera ma
swoją pozycję w tabeli (np. D ma wartość 3). Jeżeli różnica wychodzi ujemna dodajemy
liczbę 26. Otrzymana liczba odpowiada danej literze w pierwszym pierszu, czyli literze
naszej wiadomości */
```

(ciąg dalszy funkcji)

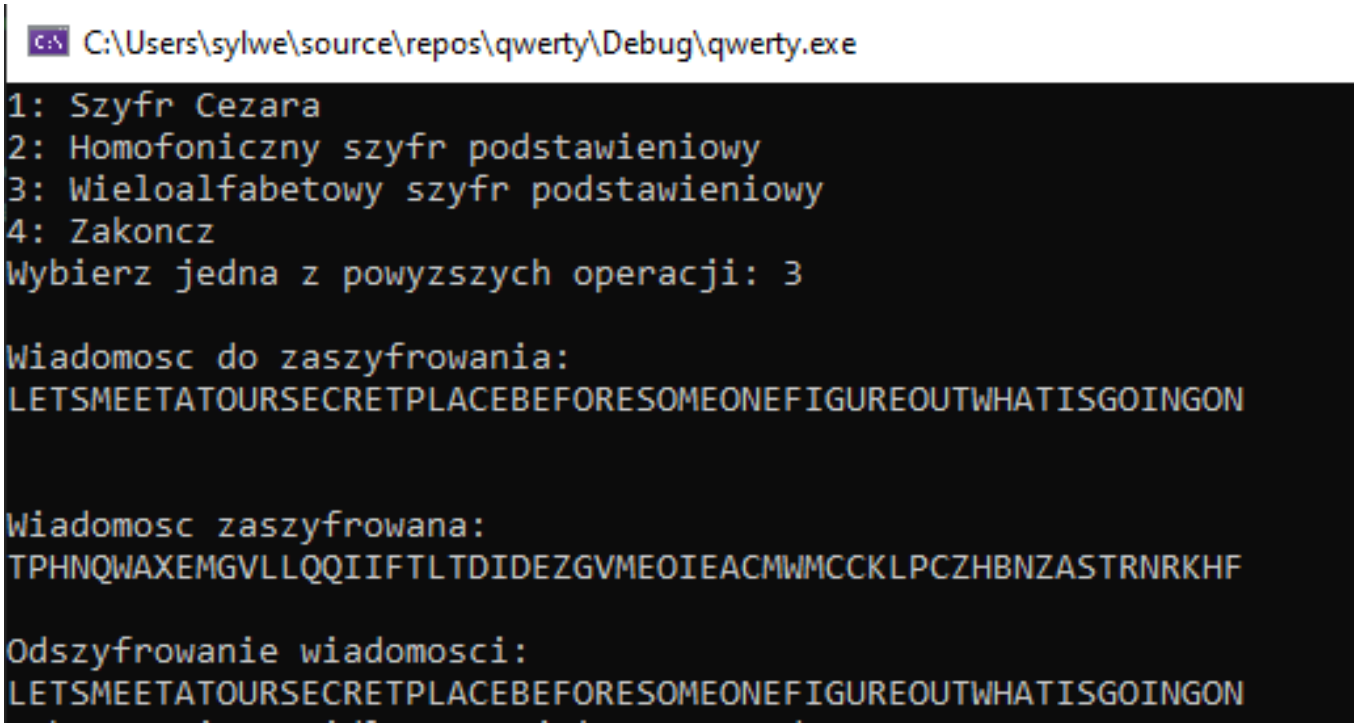
```
    // ODSZYFROWANIE //
    do
    {

        x = fgetc(keystream_file);
        ch = fgetc(output_file);
        for (j = 0; j <= 26; j++)
        {
            if (ch == tabula_recta[0][j])
            {

                cipher_value = j;
                break;
            }
        }
        for (i = 0; i <= 26; i++)
        {
            if (x == tabula_recta[0][i])
            {
                key_value = i;
                break;
            }
        }
        calc = cipher_value - key_value;
        if (calc < 0)
        {
            calc = calc + 26;
            printf("%c", tabula_recta[0][calc]);

        }
        else
        {
            printf("%c", tabula_recta[0][calc]);
        }
        k++;
    } while (k < length);
}
```

Wynik działania szyfru:



Pliki, z których korzystają szyfry:

1) Szyfr cezara
   - **input.txt** wiadomość jawna
   - **encrypted.txt** wiadomość zaszyfrowana
2) Homofoniczny szyfr podstawieniowy
   - **input.txt** wiadomość jawna
   - **homophonic.txt** wiadomość zaszyfrowana
   - **dictionary.txt** słownik liter oraz odpowiadającym im literom szyfrującym.
3) Wieloalfabetowy szyfr podstawieniowy
   - **message.txt** wiadomość jawna
   - **keystream.txt** wiadomość/kod szyfrujący
   - **tabula_recta.txt** specjalna tablica do szyfrowania
   - **encrypted_message.txt** zaszyfrowana wiadomość

Podsumowując, były to dla mnie najcięższe laboratoria z tego przedmiotu, jednakże najbardziej satysfakcjonujące, kiedy program działał prawidłowo. Zgłębiłem swoją wiedzę na temat szyfrowania oraz dowiedziałem się na temat wad i zalet poszczególnych szyfrów oraz poziomów ich bezpieczeństwa.