Aneta Skoniecka
grupa: 3I4

Projekt 2 na przedmiot ARKO dla procesora Intel 32 bitowego
Wybrany temat: 2.5 Mini Enigma

Założenia:

- znak do zaszyfrowania przechodzi kolejno przez funkcje szyfrujące: wejscie->bęben3->bęben2->bęben1->bęben odwracający->bęben1 odwrotny-> bęben 2 odwrotny-> bęben3 odwrotny->wyjscie
- obracanie bębnów następuje po zaszyfrowaniu znaku
- ponieważ obsługujemy znaki z przedziału <32;95> to znaki w tablicach opisujących bębny są przedstawiane w przedziale <00;63>, np 'K' == 43
- szyfrowanie jest odwracalne
- jedna linia w pliku ciphertext.txt nie przekracza 1024 znaków
- między liniami w pliku ciphertext.txt nie ma lini pustych (linia pusta oznacza koniec szyfrowania)

Opis projektu:

main.c
Wczytanie plików init.txt, rotors.txt i przeniesienie ich zawartości do struktury EnigmaStruct:
- wartosci init ładowane są na pozycje *initpos1*, *initpos2*, *initpos3* (opisujące pozycje początkowe bębnów) oraz odpowiadające im *pos1*, *pos2*, *pos3* (opisujące aktualne pozycje bębnów)
- bębny(rotor) 1, 2, 3 są zapisywane sposób taki, że połączenie (w pliku rotors.txt) lewa_wartosc
- prawa_wartosc oznacza szyfrowanie najpierw z lewej na prawą, a po przejsciu przez bęben odwracający z prawej na lewą,
-przykład indeksowania dla bębna 1: lewa wartość jest *indeksem* tablicy *rotor1*, a prawa wartość *wartością* (rotor1[index] == **rotor1[lewa_wartosc] = prawa_wartosc**), za to dla rotor1rev indexem jest prawa wartość, a wartością lewa wartość (czyli odwrotnie),
- bęben odwracający został przedstawiony za pomocą dwóch tablic rotorrev i rotorrev2 o wielkosci 32 kazda, które reprezentują przejscia ze znaku przed konwersją do po konwersji
Wczytanie jednej lini z plaintext.txt do tablicy znaków *bufin* i wywołanie na niej funkcji enigma. Otrzymanie zwrotnej tablicy znaków bufout i zapisanie jej do kolejnego wiersza w pliku ciphertext.txt. Przesuwanie się, linia po lini do napotkania lini pustej zawierającej tylko znak '\n'.

enigma.asm
Do funkcji ładowana jest tablica bufin w której mamy ciąg do zaszyfrowania, tablica bufout, która ma na koniec funkcji zawierac zaszyfrowany ciąg i struktura zawierająca informacje na temat szyfrowania.
Funkcja w pętli wczytuje kolejno każdy znak z bufin, poddaje go wielu zamianom opisanym niżej i zapisuje do bufout. Po każdym zaszyfrowaniu znaku dokonuje się obrót conajmniej jednego

bębna. Wykonanie pętli dzieje się do napotkania znaku ‘\n’ lub innego nieprzewidzianego znaku białego z przedziału <0;31>dec. Litery spoza przedziału <32;95> dec z wyjątkiem ‘\n’ i białych znaków są ignorowane z poziomu funkcji asemblerowej.
Podstawienie w każdym bębnie działa na zasadzie: wczytany znak odpowiada konkretnej wartości ASCII, co po odjęciu 32 daje nam indeks dla tablicy opisującej dany bęben, która zwraca nam wartość jaką należy podstawić pod dany znak, np rotor1[index] = podstawienie.Aby szyfrowanie było zależne od aktualnej pozycji bębna to przez podstawieniem do danego indexu dodawana(lub odejmowana) jest pozycja odpowiadającego bębna. Znak po przejsciu przez bębny szyfrujące zostaje zapisany do bufout.

Opis struktury:

```
struct EnigmaStruct {
int rotor1[64];        // bębny przyjmujące lewą_wartosc za index, prawą_wartosc za wartosc
int rotor2[64];        // tablicy
int rotor3[64];        //
int rotorrev[32];      // beben odwracajacy, zawiera opisy przejsc dla znakow <32; 63>dec
int pos1;              //
int pos2;              // aktualne pozycje bebnow
int pos3;              //
int rotorrev2[32];     // beben odwracajacy, zawira opisy przejsc dla znakow <64;95> dec
int rotor1rev[64];     // bębny przyjmujące prawą_wartosc za index, lewa_wartosc za wartosc
int rotor2rev[64];     //
int rotor3rev[64];     //
int initpos1;          // initpos czyli pozycje poczatkowe wczytane bezposrednio z init.txt
int initpos2;          //
int initpos3;          //
};
```

Przykład szyfrowania:
Znak do zaszyfrowania: ‘K’
Znak ‘K’ ma w ASCII znak 75, po odjęciu 32 mamy liczę 43. Liczba ta jest indeksem dla tablicy rotor3. Mamy rotor3[43] = 56, co po konwersji daje nam litere ‘X’. Liczba 56 jest indeksem dla tablicy rotor2. Mamy rotor2[56] = 54, co po konwersji daje nam litere ‘V’. Liczba 54 jest indeksem dla tablicy rotor1. Mamy rotor1[54] = 41, czyli litere ‘I’. Teraz litera ‘I’ przechodzi przez bęben odwracający i jest zamieniana na litere ‘P’ czyli liczbe 48. Teraz analogicznie idziemy w drugą strone tylko że przemieszczamy się przez tablice rotor1rev, rotor2rev, rotor3rev, które reprezentują te same połączenia co w rotor1, rotor2, rotor3, tylko, że mamy tu zamiane wartosci i indeksu. Litera P przechodzi kolejno dalej przez podstawienia: P->T, T->N, N->N.
Czyli na wyjściu otrzymujemy litere N.

Przykład obrotu bębnów i ich wpływu na szyfrowanie:
Załóżmy, że podajemy na wejście litere ‘K’. Jeśli zachodzi szyfrowanie w miejscu

wejscie->bęben3, a bęben 3 jest ustawiony na pozycji 01, to bęben szyfruje nie litere ‘K’ a litere o 1 pozycje dalej czyli ‘L’. Dla litery ‘L’ odpowiada szyfr ‘V’ i ta litera wychodzi zaszyfrowana z bębna 3.

Opis działania bębnów w konkretnych miejscach:

	miejsce:	wpływ na index:
1.	wejscie->beben3	index+=pos3
2.	beben3->beben2	index+=(pos2-pos3)
3.	beben2->beben1	index+=(pos1-pos3)
4.	beben1->beben_odwracajacy	index+=(-pos1)
5.	beben_odwracajacy->beben1	index+=pos1
6.	beben1->beben2	index+=(pos2-pos1)
7.	beben2->beben3	index+=(pos3-pos2)
8.	beben3->wyjscie	index+=(-pos3)

Bębny: opis bębnów zawarty w rotors.txt został napisany na podstawie opisu bębnów ze strony http://edu.i-lo.tarnow.pl/inf/hist/006_col/0002.php , rozszerzony został o opis znaków ASCII z przedziału <32;64>SUM<91;95>, których podstawowy szyfr enigmy nie zawiera.