Warszawa, 30.11.2015
Politechnika Warszawska
Wydział Elektroniki i Technik Informacyjnych

Projekt MIPS

Mini Enigma

(3.5)

Architektura komputerów

prowadzący: mgr inż. Zbigniew Szymański

Tatsiana Lukashevich

Spis treści:

1. Opis struktury programu

Program powinien zaszyfrować tekst i zdeszyfrować tekst o odpowiednio podanych ustawieniach wirników. Po zaszyfrowaniu/zdeszyfrowaniu jednego znaku jeden z wirników obraca się o jedną pozycję.
Tekst do zaszyfrowania/zdeszyfrowania zapisuje się w pliku o nazwie *plaintext.txt.* Ustawienie wirników zmienia się w pliku *init.txt*. Zaszyfrowany/zdeszyfrowany tekst zapisuje się do pliku *ciphertext.txt.* Jak połączone wirniki są zapisane w pliku *rotors.txt*.

Wymagania dotyczące pliku *plaintext.txt*:
-tekst jawny zawiera znaki o kodach ASCII z przedziału <32dec, 95dec> wszystkie znaki, których kody ASCII nie mieszczą się w przedziale wymienionym w poprzednim punkcie są ignorowane
- plik będzie wczytywany wiersz po wierszu (a nie na jeden raz w całości) po wczytaniu kolejnego wiersza algorytm szyfrujący kontynuuje pracę (nie powraca do stanu początkowego)
-dane kończą się pustym wierszem zawierającym tylko znak o kodzie 10dec

Plik o nazwie *rotors.txt* zawiera opis połączeń bębnów szyfrujących:
Przykład:
Rotorjeden;nr styku wejscia - nr styku prawa strona
00-04
01-10
02-12
….
Powyższe dane oznaczają, że np. styk wejścia 00 jest połączony z prawym stykiem wirnika pierwszego 04. Każda sekcja składa się z etykiety (dowolny ciąg znakowy) oraz 52 linii opisujących połączenia styków po prawej i po lewej stronie bębna (dla bębnów standardowych) i 26 linie dla bębna odwracającego(alfabet łaciński). Połączenia bębnów można odnaleźć na stronie http://habrahabr.ru/post/217331/

Początkowe położenie bębnów szyfrujących podane jest w pliku *init.txt*:
Przykład:
22
05
12
Dane z powyższego przykładu oznaczają, że:

styk 22 pierwszego bębna sąsiaduje ze stykiem wej./wyj. o numerze 00,
styk 05 drugiego bębna sąsiaduje ze stykiem 22 pierwszego bębna i stykiem 12 trzeciego bębna
styk 12 trzeciego bębna sąsiaduje ze stykiem 05 drugiego bębna i stykiem 00 bębna odwracającego.

Zaszyfrowane ciągi znakowe powinny być umieszczone w kolejnych wierszach pliku o nazwie *ciphertext.txt*.

2.Opis struktur danych i ich implementacji, implementacji algorytmu

tablica0: .space 20 -da ustawienia bebnow
tablicapom: .space 2000 tablica pomocnicza dla zapisania wirników
tablicapomoc: .space 1024 -tablica pomocnicza dla zapisania tekstu do zaszyfrowania/zdeszyfrowania
tablica2: .space 2000 -tablica przechowuje miejsce i numer wirnika, z którym jest połączony(od prawej do lewej)
tablica3: .space 400 -tablica przechowuje miejsce i numer wirnika, z którym jest połączony(od lewej do prawej)
tablica4: .space 1024 -ciąg zaszyfrowany/zdeszyfrowany

$s0 - deskryptor pliku init, rotors, plaintext

$s1 - przechowuje położenie 1 wirnika
$s2 - przechowuje położenie 2 wirnika
$s3 - przechowuje położenie 3 wirnika
$s5 - deskryptor pliku ciphertext
$s6 - iterator po *tablicapomoc*
$s7- iterator po *tablica4*
$t0 - iterator po tablica0, tablica2
$t1 - przechowuje znak pobrany z bufora
$t2 - przechowuje znak pobrany z bufora
$t3 - iterator po tablica2, tablicapom
$t4 - obrót wirnika 1 odnośnie położenia ustawionego
$t5 - obrót wirnika 2 odnośnie położenia ustawionego
$t6 - obrót wirnika 3 odnośnie położenia ustawionego
$t7 - przechowuje w sobie 26, potrzebny do pobrania moduło

read_init - odczytuje położenia wirników i zapamiętuje w rejestrach

- **zmien_liczbe -** Zamieniamy na liczbę ciąg znaków
- **powrot -** Zapisujemy do rejestrów z ciągu ustawienia wirników

read_rotors - Odczytuje plik z połączeniem wirników
rotory_zapisz - Zapisuje do tablicy2 tylko cyfry z pliku rotors
zamknij_rotors - Zamykamy plik rotors
zapisz_ciag - Zapisujemy położenia wirników w dzwie tablicy : tablica2 (tablica przechowuje miejsce i numer wirnika, z którym jest połączony(od prawej do lewej)) i tablica3(tablica przechowuje miejsce i numer wirnika, z którym jest połączony(od lewej do prawej))

- **puste_miejsce -** Pozostałe miejsce tablicy2 wypełniamy pustym miejscem, bo tylko powinna przechowywać liczby, które przerobiliśmy w funkcji *zmien_na_liczbe*
- **zmien_na_liczbe -** Zamienia na liczbę ciąg znaków
- **zapisz_drugatab -** Zapisuje do tablica3 zamieniony ciąg na liczby

otworz_plik - Otwiera plik do zaszyfrowania/zdeszyfrowania
zapisz_wiersz - Zapisuje wiersz po wierszu do ciągu wyjściowego tablica4

- **newline -** Do ciągu wyjściowego zapisujemy nową linię
- **zapisz_wyjscie -** zapisujemy w ciąg wyjściowy znaki, które wchodzą w przedział od 32 - 95, ale nie będą zakodowane przez enigmę
- **kodowanie -** Kodujemy otrzymaną literę, po zakończeniu przesuwamy rotor o jedną pozycję

  Kodowanie odbywa się w taki sposób,że otrzymamy z pliku literę do zakodowania z pliku plaintext

  Następnie w tablicy2 pod miejscem, na które wskazuje liczba, otrzymujemy liczbę, z którą jest powiązane wejscie i roter1

Dalej przechodzimy dodając 26(bo alfabet łaciński) do następnego rotera, sprawdzając na ile on jest obrócony itd.

Na wyjściu otrzymamy zaszyfrowaną/zdeszyfrowaną literę

- **zamien -** Moduł z ujemnej liczby
- **rotate -** Obraca wirniki
  - **obroc2** - Obraca 2-i wirnik
  - **obroc3** - Obraca 3-i wirnik
  - **obroc1** - Obraca 1-y wirnik

- **close_plaintext** - Zamykanie pliku z tekstem do szyfrowania/deszyfrowania
- **open_cipher -** Otwieranie pliku do zapisania teksty zaszyfrowanego/zdeszyfrowanego
- **write_cipher -** Zapisanie tekstu zaszyfrowanego/zdeszyfrowanego
- **close_cipher -** Zamykanie pliku z tekstem zaszyfrowanym/zdeszyfrowanym

3.Opis testów

Testy zostały prowadzony za pomocą wpisania do pliku plaintext.txt tekstu do zaszyfrowania, wynik został zapisany do pliku ciphertext.txt. Zatem do sprawdzenia zakodowanego tekstu umieszczam go w plik plaintext.txt, porównując wyniki, w pliku ciphertext.txt i tekstu jawnego, możemy decydować o poprawności zaszyfrowanych danych za pomocą programu. Poniższa tabela przedstawia wyniki testowania:

Tekst do zaszyfrowania	Zaszyfrowany tekst	Zdeszyfrowany tekst	Ustawienie	Komentarz

			wirników w pliku init.txt	
ATLX MUN Y HB	OLZL UGS U RC	ATLX MUN Y HB	22 05 12	Jest dobrze
KNLABCIYJ NHFRYU IUSJKDLDJFJ BSFD	PMMTHQHEY UWMVTB SFRPWQGXLYT HBHT	KNLABCIYJ NHFRYU IUSJKDLDJFJ BSFD	15 10 05	Jest dobrze
ABNDHYRLFJFH F KSJ J KKS SJD UUT J LLKSHahs JsuJ	GPBXAKMQLOTM A FMV Y SYU HOZ ROG T NFEGJ VB	ABNDHYRLFJFH F KSJ J KKS SJD UUT J LLKSH JJ	07 24 18	Jest dobrze Pomijamy przy kodowaniu małe litery,bo nie wchodzą w obszar podany w wymaganiach(32dec, 95dec)
LSJDHHU SJ SKKI J	JGHRFTT TP NWFU O	LSJDHHU SJ SKKI J	03 20 10	Jest dobrze
JGHRH5SG GSJ6JS SGY N	VFTJG5JE AKD6PH ELS I	JGHRH5SG GSJ6JS SGY N	22 11 06	Jest dobrze Dopuszczam przechodzenie znaku bez kodowania(32dec-65dec;90dec-95dec)
KDDHHFNCBSJ SJDJ DJKEIEJD SJSLOS DJD D DJ JDKLLDJEUU	QREMZEVBCXQ DDLV PXPCNWXR YOGKUU YVX R EG EUIZEOYQQO	KDDHHFNCBSJ SJDJ DJKEIEJD SJSLOS DJD D DJ JDKLLDJEUU	23 20 24	Jest dobrze

Te testy pokazują,że program działa poprawnie.