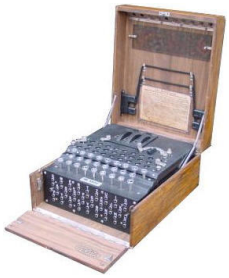
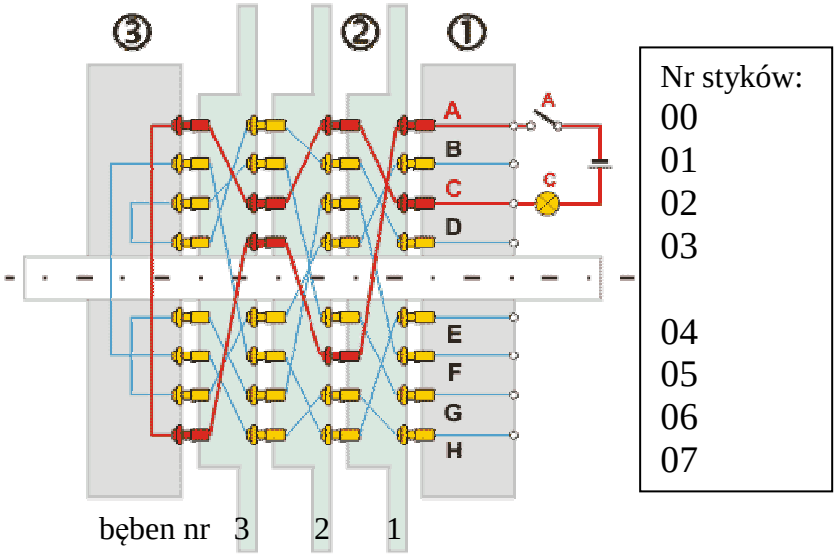


Mini Enigma

Cel programu

Zrealizować program szyfrujący ciągi znakowe (pobierane z pliku) algorytmem używanym w maszynie Enigma opisanym w następnym punkcie.

Opis algorytmu



Rys 1. Zasada działania bębnów szyfrujących maszyny Enigma [2]. Oznaczenia: 1 – styki wejściowe/wyjściowe, 2 – bębny szyfrujące, 3 – bęben odwracający

Każdy ze styków wejściowo/wyjściowych odpowiada jednemu znakowi z pewnego podzbioru znaków ASCII. Zakładamy, że styki są ponumerowane od 00. Jeden bęben szyfrujący wykonuje proste podstawienie dwóch znaków.

Na rys. 1 pokazano zasadę działania algorytmu. Sygnał odpowiadający literze A podawany jest poprzez styk wejściowy (w powyższym przykładzie o numerze 00), następnie propaguje poprzez styk (nr 00) po prawej stronie pierwszego bębna do styku (nr 05) po lewej stronie bębna. W bębnie nr 2 wchodzi na styk nr 05 po prawej stronie i przechodzi na styk nr 03 po lewej stronie. W bębnie nr 3 wchodzi na styk nr 03 po prawej stronie i przechodzi na styk nr 07 po lewej stronie. Następnie w bębnie odwracającym sygnał propaguje ze styku nr 07 na styk o numerze 00, zaś dalej z powrotem przez bębny nr 3, 2 oraz 1 (wchodząc z lewej strony i wychodząc z prawej). Zaszyfrowany znak otrzymujemy po podaniu sygnału na styk wejściowo/wyjściowy o numerze 02 (odpowiadający w przykładzie literze C).

Po zaszyfrowaniu znaku bęben nr 1 obraca się o jedną pozycję. Zatem styk o numerze 01 bębna nr 1 będzie teraz sąsiadował ze stykiem wejściowo/wyjściowym nr 00 (a styk 00 bębna nr 1, ze stykiem wej./wyj. nr 07). Gdy bęben nr 1 wykona pełen obrót, następuje obrót bębna nr 2 o jedną pozycję, itd. Bęben odwracający jest nieruchomy.

Wejście

Ciągi znakowe do zaszyfrowania:

Ciągi znakowe do zaszyfrowania (tekst jawny) znajdują się w kolejnych wierszach pliku o nazwie **`plaintext.txt`**. Zakładamy, że:

- długość wiersza tekstu razem ze znakiem końca wiersza (o kodzie 10_{dec}) jest nie większa niż 1024 znaki.
- tekst jawny zawiera znaki o kodach ASCII z przedziału $<32_{dec}, 95_{dec}>$
- 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 10_{dec}

Połączenia bębnów szyfrujących:

Plik o nazwie **`rotors.txt`** zawiera opis połączeń bębnów szyfrujących. Plik składa się z czterech sekcji – trzy pierwsze opisują połączenia bębnów standardowych (w stałej kolejności: bęben nr 1, 2, 3), ostatnia bębna odwracającego. Każda sekcja składa się z etykiety (dowolny ciąg znakowy) oraz 64 linii opisujących połączenia styków po prawej i po lewej stronie bębna (dla bębnów standardowych) i 32 linie (dla bębna odwracającego).

Przykład sekcji opisującej połączenia bębna nr 1

```
Rotor 1;nr styku po lewej stronie- nr styku po prawej stronie
00-32
01-34
02-36
....
62-03
63-01
```

Powyższe dane oznaczają, że np. lewy styk 00 jest połączony z prawym stykiem 32.

Początkowe położenie bębnów szyfrujących:

Początkowe położenie bębnów szyfrujących podane jest w pliku **`init.txt`**. W kolejnych wierszach podane jest początkowe położenie bębnów nr 1, 2, 3.

Przykładowa zawartość pliku init.txt

```
00
17
28
```

Dane z powyższego przykładu oznaczają, że:

- styk 00 pierwszego bębna sąsiaduje ze stykiem wej./wyj. o numerze 00,
- styk 17 drugiego bębna sąsiaduje ze stykiem 00 pierwszego bębna i stykiem 28 trzeciego bębna
- styk 28 trzeciego bębna sąsiaduje ze stykiem 17 drugiego bębna i stykiem 00 bębna odwracającego.

Wyjście

Zaszyfrowane ciągi znakowe powinny być umieszczone w kolejnych wierszach pliku o nazwie **`ciphertext.txt`**. Znak końca wiersza ma kod 10_{dec}. Plik powinien kończyć się pustym wierszem zawierającym wyłącznie znak końca wiersza.

Uwagi

1. Powyższy algorytm szyfruje w sposób odwracalny. Zatem:
   F(F(***X***))=***X***, gdzie ***X*** – ciąg wejściowy, F – operacja szyfrowania
2. Program SPIM można uruchomić z wiersza poleceń w następujący sposób:

```
/usr/local/spim/bin/spim -f turtle.asm
```

3. Fragmenty kodu służącego do odczytu danych z pliku [1]:

```
#otwarcie pliku
      li $v0, 13        #system call for file_open
      la $a0, filen     #address of filename string
      li $a1, 0x8000    #flags - binary, read
      li $a2, 0444      #UNIX octal mode 0444 r--r--r--
      syscall           #file descriptor of opened file in v0

#sprawdzenie czy plik udalo sie otworzyc file_descriptor<>-1
      ...

#odczyt danych z pliku
      li $v0, 14        #system call for file_read
      move $a0, $...    #move file descr from ... to a0
      la $a1, buf       #address of data buffer
      li $a2, 4048      #amount to read (bytes)
      syscall

#sprawdzenie ile danych wczytano
      beq $zero,$v0, fclose   #branch if no data is read
      ...

#zamkniecie pliku
      li $v0, 16              #system call for file_close
      move $a0, $...          #move file descr from ... to a0
      syscall
```

Dokumentacja

Wraz z działającym programem należy przedstawić dokumentację projektu. Powinna zawierać m.in.:

- opis struktury programu,
- opis implementacji algorytmu,
- wyniki testowania programu.

Literatura

1. "SPIM From Wikipedia, the free encyclopedia", http://en.wikipedia.org/wiki/SPIM
2. T. Sale, "Komputery Colossus", http://edu.i-lo.tarnow.pl/inf/hist/006_col/0001.php