Szyfrowanie wiadomości

Projekt semestralny z programowania
Zadanie 1 - Algorytmion 2015

Jagoda Oleksiak, informatyka niestacjonarna, semestr I

Spis treści

Na czym polegało zadanie?

“Adam i Janek ustalili między sobą, że każdą wiadomość tekstową, będą szyfrować przy pomocy następującego sposobu:

1. Każdą literę i znak interpunkcyjny należy zamienić na odpowiadającą liczbę kodu ASCII (liczba naturalna od 0 do 127).
2. Pierwsza, tak powstała liczba, jest zamieniana na system dwójkowy.
3. Kolejne liczby, wynikające z kodu ASCII, są zamieniane na system liczbowy, który jest równy powiększonej o dwa reszcie z dzielenia przez osiem poprzedniej liczby.
4. Ilość cyfr zamienionej liczby, dla każdego systemu liczbowego, wynika z ilości cyfr zamiany liczby maksymalnej, czyli liczby 127 (dla systemu binarnego jest to siedem cyfr)."

Na czym polegało zadanie c.d

Moim zadaniem było napisanie programu, który szyfruje i deszyfruje wiadomości. Szyfrowanie tekstu odbywa się z pliku tekst.txt do pliku szyfr.txt, a deszyfrowanie z pliku szyfr.txt do pliku odszyfrowane.txt.

Przykład:

Tekst	Kod ASCII	System liczbowy: 2 (66 mod 8)+2=4, (105 mod 8)+2=3
Bit	66 105 116	1000010 1221 11022

Opis działania programu

01 Program służy do szyfrowania oraz deszyfrowania wiadomości znajdujących się w pliku tekstowym. Każda litera oraz znak interpunkcyjny są zamieniane na odpowiadającą liczbę kodu ASCII.

02 Powstałe w ten sposób liczby zamieniane są na system dwójkowy. Kolejne liczby, które wynikają z kodu ASCII, są zamieniane na system liczbowy, który równy jest powiększonej o dwa reszcie z dzielenia przez osiem liczby poprzedniej. Ilość cyfr zamienionej liczby wynika z ilości cyfr zamiany liczby maksymalnej.

Wejście

Na wejściu użytkownik pytany jest co chciałby zrobić - do wyboru ma trzy opcje:

a) Zaszyfrowanie pliku
b) Rozszyfrowanie pliku
c) Wyjście z programu

Wejście

Ad. a) Szyfrowanie odbywa się z pliku tekst.txt do pliku szyfr.txt.
Każda litera oraz znak interpunkcyjny są zamieniane na odpowiadającą liczbę kodu ASCII, powstałe liczby zamieniane są na system dwójkowy. Kolejne liczby wynikające z kodu ASCII, są zamieniane na system liczbowy, który równy jest powiększonej o dwa reszcie z dzielenia przez osiem liczby poprzedniej.

Ad.b) Deszyfrowanie odbywa się z piku szyfr.txt do pliku odszyfrowane.txt.

Analogicznie do szyfrowania algorytm deszyfrujący działa w drugę stronę - zapisane w systemie liczbowym równym powiększonej o dwa reszcie z dzielenia przez osiem liczby poprzedniej zamieniane są z powrotem na system binarny, a następnie liczby kodu ASCII zamieniane są na odpowiadające im litery i znaki interpunkcyjne.

Ad.c) Wyjście z programu jak sama nazwa wskazuje, powoduje zakończenie procesu.

Wyjście

Wynik na wyjściu programu jest zależny od początkowego wyboru użytkownika.

Ad. a) Szyfrowanie

W przypadku, gdy użytkownik wybierze opcję szyfrowania programu, zostanie utworzony plik o nazwie szyfr.txt, w którym znajduje się odpowiednio zaszyfrowany tekst.

Ad. b) Deszyfrowanie

W przypadku, gdy użytkownik wybierze, aby rozszyfrować dany tekst, program utworzy plik o nazwie odszyfrowane.txt w którym będzie znajdować się oryginalny tekst, który wcześniej został zaszyfrowany przy użyciu opcji numer jeden.

Ad. c) Wyjście z programu

W przypadku, gdy użytkownik wybierze opcję wyjścia z programu, program zakończy swoje działanie wraz z kodem wyjściowym 0.

Opis algorytmu

1. main() rozpoczyna od przyjęcia inputu od użytkownika, wybierając jeden z dwóch algorytmow:
    a. Rozpoczęcie algorytmu szyfrującego poprzez wywołanie funkcji encode()
        i. Wczytanie pliku input.txt, jeśli plik istnieje, odczytaj jego zawartość w trybie binarnym
        ii. Zdefiniowanie zmiennej odpowiedzialnej za zapamiętanie poprzedniej szyfrowanej wartości.
        iii. Inicjalizacja listy przechowującej typ EncodedCharacter w celu wykonywania operacji na bufferze pobranym z pliku
        iv. Pętla dopóki jej licznik jest mniejszy niż długość wczytanego buffera

1. Jeżeli znak jest znakiem specjalnym, pomijanie iteracji
2. Określenie systemu poprzez wzór *(PoprzedniaWartość mod 8) + 2*
3. Przypisanie do zmiennej wartości w bufferze na miejscu indeksowanym poprzez licznik pętli
4. Jeżeli wartość jest większa niż 127 zwróć błąd
5. Wywołanie funkcji szyfrującej znak
    a. Jeżeli (wartość szyfrowana / system) != 0
        i. Wywołanie pkt 5. pod wartość szyfrowaną podana wartość powyższego dzielenia
    b. Dodanie aktualnego znaku do buffera
    c. Zwróć buffer zwiększony o jeden
6. Dodanie zakodowany znaku do lisy
7. Zapis w trybie tekstowym
8. Wyczyszczenie zaalokowanych miejsc pamięci

a. Rozpoczęcie algorytmu odszyfrowywania poprzez wywołania funkcji decode()
   i. Wczytanie pliku output.txt, jeśli plik istnieje, odczytanie zawartości w trybie binarnym
   ii. Zdefiniowanie aktualnego systemu na binarny
   iii. Zaalokowanie pamięci dla bufora outputowego
   iv. Pętla dopóki da się rozdzielać ciąg znaków poprzez spacje

1. Odszyfrowywanie ciągu znaków i przemiana w jeden znak:

1. Zamiana ciągu poprzez strtol() z odpowiedniego systemu na wartość liczbową
2. Jeżeli wartość liczbowa nie zawiera się w przedziale [-128; 127] zwróć błąd
3. Zwrócenie wartość znaku

   v. Zaalokowanie pamięci dla powiększonego o jeden buffora w celu zapamiętania całości zaszyfrowanego ciągu znaków
   vi. Przepisanie wartości z głównego buffera do tymczasowego
   vii. Wyczyszczenie pamięci dla głównego buffera.

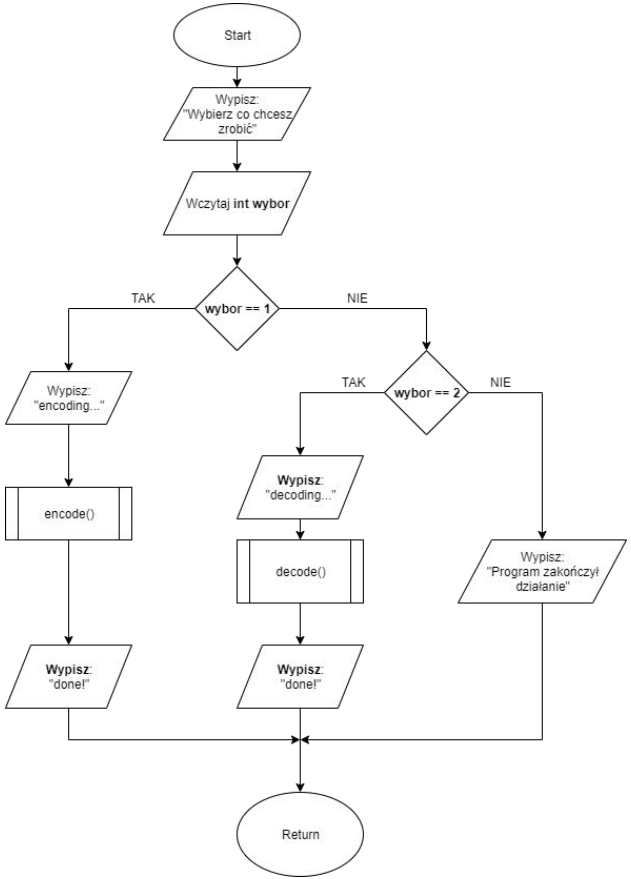
1. Powiększenie zakresu pamięci głównego buffera
2. Konkatenacja buffera tymczasowego i odszyfrowanego znadku
3. Przypisanie wartości do buffera głównego
4. Zwolnienie pamięci buffera tymczasowego
5. Zmiana wartości aktualnego systemu
    a. Zwróć *(PoprzedniaWartość mod 8) + 2*

ii. Zapis do pliku tekstowego odszyfrowanego tekstu

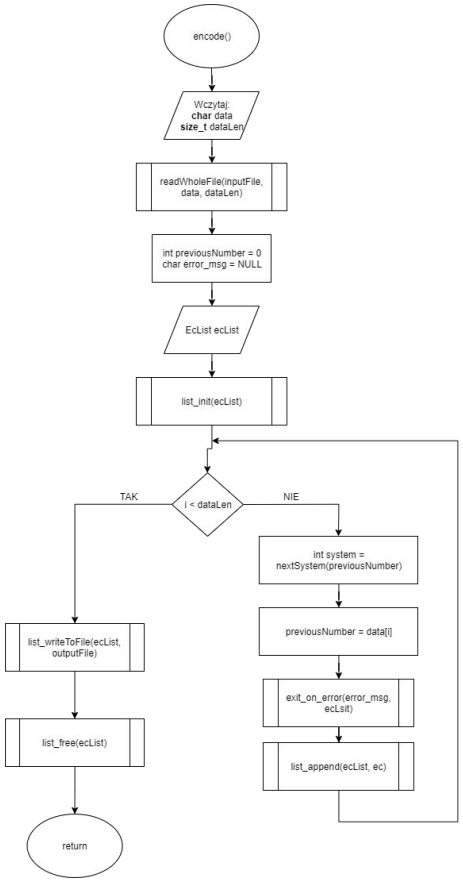
iii. Wyczyszczenie zaalokowanych miejsc pamięci

2. Zakończenie programu.

Schemat blokowy



Schemat blokowy funkcji encode



Testy na poprawność

input.txt

```
1    Bardzo lubie przegladac zdjecia slodkich kotkww i pieskww w internecie.
```

```
Wybierz, co chcesz zrobić:
1 - Chcę zaszyfrować plik
2 - Chcę rozszyfrować plik
3-Chcę wyjść z programu
1
encoding...
EncodedCharacter{B (66), 2, 7, 1000010}
EncodedCharacter{a (97), 4, 4, 12010□}
EncodedCharacter{r (114), 3, 5, 11020□}
EncodedCharacter{d (100), 4, 4, 12100□}
EncodedCharacter{z (122), 6, 3, 32200□}
EncodedCharacter{o (111), 4, 4, 12330□}
EncodedCharacter{  (32), 9, 2, 35000□}
EncodedCharacter{l (108), 2, 7, 1101100}
EncodedCharacter{u (117), 6, 3, 31300□}
EncodedCharacter{b (98), 7, 3, 20000□}
EncodedCharacter{i (105), 4, 4, 12210□}
EncodedCharacter{e (101), 3, 5, 10202□}
EncodedCharacter{  (32), 7, 2, 44000□}
EncodedCharacter{p (112), 2, 7, 1110000}
EncodedCharacter{r (114), 2, 7, 1110010}
EncodedCharacter{z (122), 4, 4, 13220□}
EncodedCharacter{e (101), 4, 4, 12110□}
EncodedCharacter{g (103), 7, 3, 20500□}
EncodedCharacter{l (108), 9, 3, 13000□}
EncodedCharacter{a (97), 6, 3, 24100□}
EncodedCharacter{d (100), 3, 5, 10201□}
EncodedCharacter{a (97), 6, 3, 24100□}
EncodedCharacter{c (99), 3, 5, 10200□}
EncodedCharacter{  (32), 5, 3, 11200□}
EncodedCharacter{z (122), 2, 7, 1111010}
EncodedCharacter{d (100), 4, 4, 12100□}
EncodedCharacter{j (106), 6, 3, 25400□}
EncodedCharacter{e (101), 4, 4, 12110□}
EncodedCharacter{c (99), 7, 3, 20100□}
EncodedCharacter{i (105), 5, 3, 41000□}
EncodedCharacter{a (97), 3, 5, 10121□}
EncodedCharacter{  (32), 3, 4, 10120□}
EncodedCharacter{s (115), 2, 7, 1110011}
EncodedCharacter{l (108), 5, 3, 41300□}
EncodedCharacter{o (111), 6, 3, 30300□}
EncodedCharacter{d (100), 9, 3, 12100□}
EncodedCharacter{k (107), 6, 3, 25500□}
EncodedCharacter{i (105), 5, 3, 41000□}
EncodedCharacter{c (99), 3, 5, 10200□}
EncodedCharacter{h (104), 5, 3, 40400□}
EncodedCharacter{  (32), 2, 6, 100000}
EncodedCharacter{k (107), 2, 7, 1101011}
EncodedCharacter{o (111), 5, 3, 42100□}
EncodedCharacter{t (116), 9, 3, 13800□}
EncodedCharacter{k (107), 6, 3, 25500□}
EncodedCharacter{w (119), 5, 3, 43400□}
EncodedCharacter{w (119), 9, 3, 14200□}
EncodedCharacter{  (32), 9, 2, 35000□}
```

input.txt × output.txt ×

```
1000010 1201 11020 1210 322 1233 35 1101100 313 200 1221 10202 44 1110000 1110010 1322 1211 205 130 241 10201 241 10200 112 1111010 1210 254 1211 201 410 10121 1012 1110011 413 303 121 255
```

```
EncodedCharacter{  (32), 9, 2, 35[illegible]}
EncodedCharacter{i (105), 2, 7, 1101001}
EncodedCharacter{  (32), 3, 4, 1012[illegible]}
EncodedCharacter{p (112), 2, 7, 1110000}
EncodedCharacter{i (105), 2, 7, 1101001}
EncodedCharacter{e (101), 3, 5, 10202}
EncodedCharacter{s (115), 7, 3, 223}
EncodedCharacter{k (107), 5, 3, 412}
EncodedCharacter{w (119), 5, 3, 434}
EncodedCharacter{w (119), 9, 3, 142}
EncodedCharacter{  (32), 9, 2, 35}
EncodedCharacter{w (119), 2, 7, 1110111}
EncodedCharacter{  (32), 9, 2, 35}
EncodedCharacter{i (105), 2, 7, 1101001}
EncodedCharacter{n (110), 3, 5, 11002}
EncodedCharacter{t (116), 8, 3, 164}
EncodedCharacter{e (101), 6, 3, 245}
EncodedCharacter{r (114), 7, 3, 222}
EncodedCharacter{n (110), 4, 4, 1232}
EncodedCharacter{e (101), 8, 3, 145}
EncodedCharacter{c (99), 7, 3, 201}
EncodedCharacter{i (105), 5, 3, 410}
EncodedCharacter{e (101), 3, 5, 10202}
EncodedCharacter{. (46), 7, 2, 64}
done!

Process finished with exit code 0
```

input.txt × output.txt ×

```
1  1000010 1201 11020 1210 322 1233 35 1101100 313 200 1221 10202 44 1110000 1110010 1322 1211 205 130 241 10201 241 10200 112 1111010 1210 254 1211 201 410 10121 1012 1110011 413 303 121 255
```

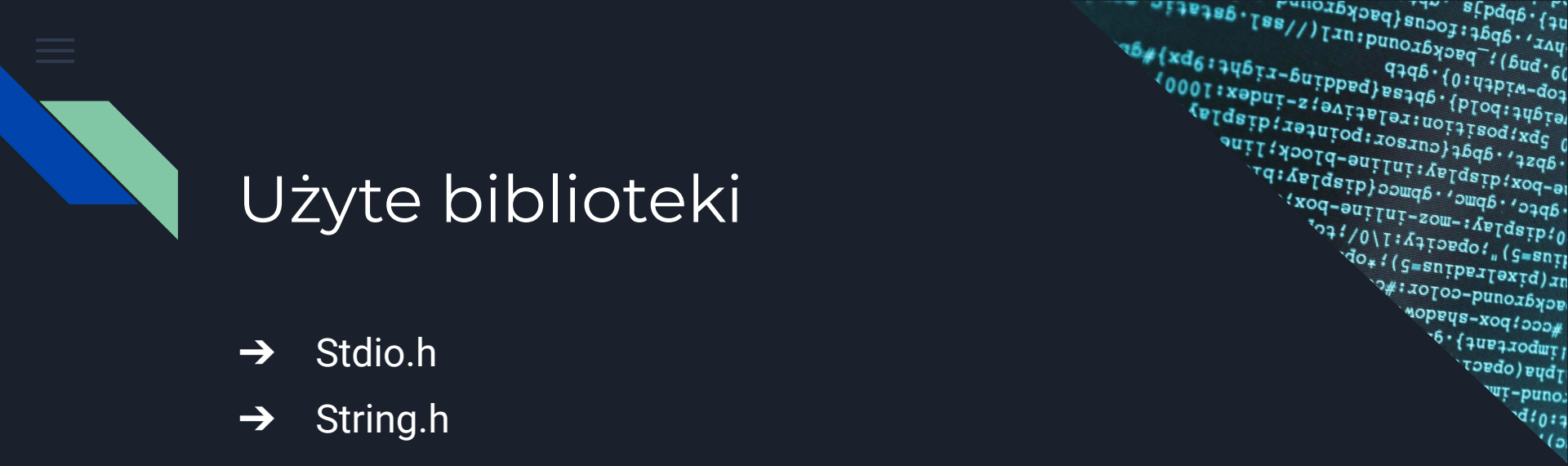
```
EncodedCharacter{e (101), 3, 5, 10202}
EncodedCharacter{s (115), 7, 3, 223}
EncodedCharacter{k (107), 5, 3, 412}
EncodedCharacter{w (119), 5, 3, 434}
EncodedCharacter{w (119), 9, 3, 142}
EncodedCharacter{  (32), 9, 2, 35}
EncodedCharacter{w (119), 2, 7, 1110111}
EncodedCharacter{  (32), 9, 2, 35}
EncodedCharacter{i (105), 2, 7, 1101001}
EncodedCharacter{n (110), 3, 5, 11002}
EncodedCharacter{t (116), 8, 3, 164}
EncodedCharacter{e (101), 6, 3, 245}
EncodedCharacter{r (114), 7, 3, 222}
EncodedCharacter{n (110), 4, 4, 1232}
EncodedCharacter{e (101), 8, 3, 145}
EncodedCharacter{c (99), 7, 3, 201}
EncodedCharacter{i (105), 5, 3, 410}
EncodedCharacter{e (101), 3, 5, 10202}
EncodedCharacter{. (46), 7, 2, 64}
done!

Process finished with exit code 0
```

input.txt | old_input.txt | output.txt

```
1  Bardzo lubie przegladac zdjecia slodkich kotkww i pieskww w internecie.
```

Użyte biblioteki

- Stdio.h
- String.h
- Stdlib.h
- Stdbool.h
- Errno.h



Źródła

- http://einstein.ms.polsl.pl/Files/Zadania2015.pdf
- https://www.programiz.com/c-programming/examples/ASCII-value-character
- https://www.thecrazyprogrammer.com/2016/11/caesar-cipher-c-c-encryption-decryption.html

Link do mojego githuba:
https://github.com/jagodaoleksiak/szyfr

Dziękuję za uwagę!