Wojciech Matraś 80742

Sprawozdanie z zadania pierwszego z przedmiotu Teoria Informacji i Kodowania

1 Treść zadania.

Implementacja kodera/dekodera słownikowego. Metoda: LZ77.

2 Analiza zadania.

LZ77 to metoda kompresji danych, która polega na zastąpieniu powtarzających się fragmentów danych symbolami (współrzędnymi) wskazującymi na miejsce wcześniejszego wystąpienia tych danych. W procesie kompresji plik jest odczytywany, a dane są kompresowane za pomocą algorytmu LZ77 i zapisywane w pliku wyjściowym. Ten sam program może być użyty do dekompresji pliku, odczytując skompresowane dane i generując oryginalny plik.

3 Założenia:’

Do implementacji będzie wykorzystywany język C. Kodowanie i dekodowanie będzie się odbywać na plikach tekstowych. Używam słownika 4 znakowego oraz 4 znakowego buffora. Tworze jedną długą listę char w której pierwsze 8 elementów to buffor oraz słownik. Zmierzam do zakończenia tej listy, czyli momentu gdy wolny jest 5 element w górę.

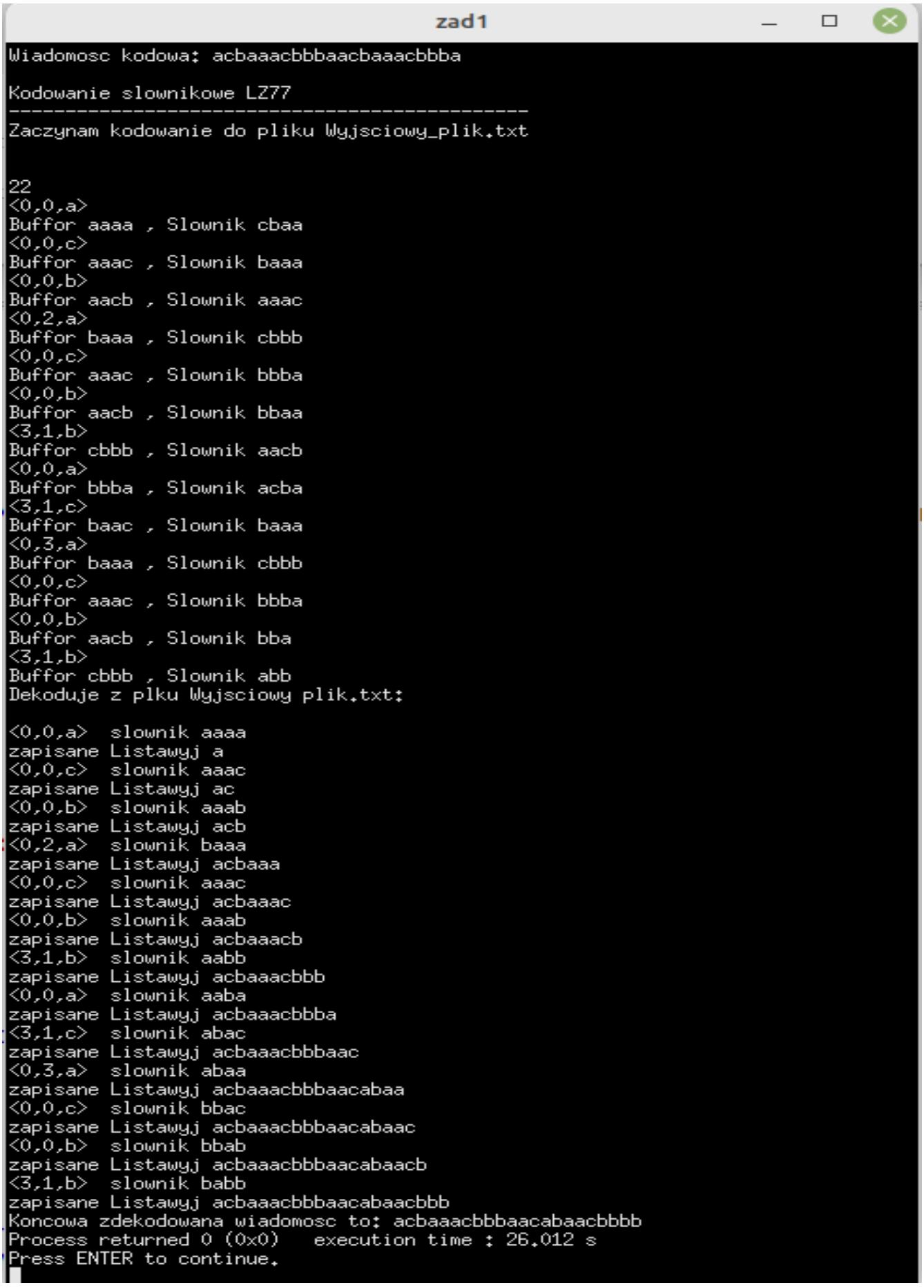
4 Algorytm:

Algorytm LZ77 polega na wyszukiwaniu powtarzających się fragmentów danych w pliku wejściowym i zastępowaniu ich symbolami (współrzędnymi) wskazującymi na miejsce i długość fragmentów danych. Symbol składa się z trzech elementów: offsetu (wskazującego na odległość od bieżącej pozycji), długości powtarzającego się fragmentu i jednego znaku, który nie jest częścią powtarzającego się fragmentu. W procesie dekompresji symbol jest rozkodowywany i odpowiednie dane są dodawane do pliku wyjściowego. Algorytm LZ77 jest efektywny w kompresji plików z dużą liczbą powtarzających się fragmentów.

5 Implementacja:

Do implementacji będzie wykorzystywany język C. Kodowanie i dekodowanie będzie się odbywać na plikach tekstowych. Używam słownika 4 znakowego oraz 4 znakowego buffora. Tworze jedną długą listę char w której pierwsze 8 elementów to buffor oraz slownik. Zmierzam do zakończenia tej listy, czyli momentu gdy wolny jest 5 element w górę. Tworze różne warunki by porównać buffor ze słownikiem, przepisuje <a,b,c> do pliku wyjsciowego przepisuje ze slownika do buffora a z listy powyżej 8 element do słownika,kodowanie koncze znakiem ‘_’.

6 Testowanie poprawności działania.


zad1
Wiadomosc kodowa: acbaaacbbbaacbaaacbbba
Kodowanie slownikowe LZ77
Zaczynam kodowanie do pliku Wyjsciowy_plik.txt
22
<0,0,a>
Buffor aaaa , Slownik cbaa
<0,0,c>
Buffor aaac , Slownik baaa
<0,0,b>
Buffor aacb , Slownik aaac
<0,2,a>
Buffor baaa , Slownik cbbb
<0,0,c>
Buffor aaac , Slownik bbba
<0,0,b>
Buffor aacb , Slownik bbaa
<3,1,b>
Buffor cbbb , Slownik aacb
<0,0,a>
Buffor bbba , Slownik acba
<3,1,c>
Buffor baac , Slownik baaa
<0,3,a>
Buffor baaa , Slownik cbbb
<0,0,c>
Buffor aaac , Slownik bbba
<0,0,b>
Buffor aacb , Slownik bba
<3,1,b>
Buffor cbbb , Slownik abb
Dekoduje z plku Wyjsciowy plik.txt:
<0,0,a> slownik aaaa
zapisane Listawyj a
<0,0,c> slownik aaac
zapisane Listawyj ac
<0,0,b> slownik aaab
zapisane Listawyj acb
<0,2,a> slownik baaa
zapisane Listawyj acbaaa
<0,0,c> slownik aaac
zapisane Listawyj acbaaac
<0,0,b> slownik aaab
zapisane Listawyj acbaaacb
<3,1,b> slownik aabb
zapisane Listawyj acbaaacbbb
<0,0,a> slownik aaba
zapisane Listawyj acbaaacbbba
<3,1,c> slownik abac
zapisane Listawyj acbaaacbbbaac
<0,3,a> slownik abaa
zapisane Listawyj acbaaacbbbaacabaa
<0,0,c> slownik bbac
zapisane Listawyj acbaaacbbbaacabaac
<0,0,b> slownik bbab
zapisane Listawyj acbaaacbbbaacabaacb
<3,1,b> slownik babb
zapisane Listawyj acbaaacbbbaacabaacbbb
Koncowa zdekodowana wiadomosc to: acbaaacbbbaacabaacbbbb
Process returned 0 (0x0) execution time : 26.012 s
Press ENTER to continue.

7. Stopień kompresji (bez printfow oraz bez sleep jakich użyłem w kodzie do przejrzystszej prezentacji)

znaki	sekundy
50	0,008
500	0,327
1000	0,333
2000	0,41
3000	0,46
4000	0,48
8000	0,713

7 Wnioski

Dzięki algorytmowi LZ77 możemy zaszyfrować nasz tekst. Jedynie znając długość słownika i buffora będziemy mogli go odczytać.

8 Kod źródłowy:

```
#include <iostream>
#include <string>
#include <array>
#include<fstream>
#include <thread>
#include <unistd.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <string>

using namespace std;

void kodowanie_lz77(char we[])
{
```

```
ofstream plik("Wyjsciowy_plik.txt");
int dotylu = 0;
char lista_kolejnych_wczytanych[100] = { ' ' };
int nastepny_argument = 0;
int t = strlen(we)-2;
printf("%d\n", t);
for (int i = 0; i < t; i++)
{
   lista_kolejnych_wczytanych[8 + i] = we[i];
   //cout << lista_kolejnych_wczytanych[8 + i]; dobrxze
}

for (int k = 0; k <= 3; k++)
{
   lista_kolejnych_wczytanych[k] = lista_kolejnych_wczytanych[8];
}

cout << "<0,0," << lista_kolejnych_wczytanych[8] << ">" << endl;
plik << "<0,0," << lista_kolejnych_wczytanych[8] << ">";
for (int i = 0; i <= 3; i++)
{
   lista_kolejnych_wczytanych[4 + i] = lista_kolejnych_wczytanych[9 + i];
}

for (int i = 0; i <= t-5; i++)
{
   lista_kolejnych_wczytanych[8 + i] = lista_kolejnych_wczytanych[13 + i];
}

nastepny_argument++;
```

```
   while (nastepny_argument != t-4 || nastepny_argument != t-3 || nastepny_argument !=t-2 || nastepny_argument !=t-1)

   {  sleep(1);

      printf("Buffor %c%c%c%c , Slownik %c%c%c%c\n", lista_kolejnych_wczytanych[0], lista_kolejnych_wczytanych[1], lista_kolejnych_wczytanych[2], lista_kolejnych_wczytanych[3], lista_kolejnych_wczytanych[4], lista_kolejnych_wczytanych[5], lista_kolejnych_wczytanych[6], lista_kolejnych_wczytanych[7]);

      if(lista_kolejnych_wczytanych[5]==NULL){

      plik << "<0,4,_>";

      break;

      }

      else if (lista_kolejnych_wczytanych[0] == lista_kolejnych_wczytanych[4]) // mozemy sprawdzic caly slownik

      {

         if (lista_kolejnych_wczytanych[1] == lista_kolejnych_wczytanych[5])

         {

            if (lista_kolejnych_wczytanych[2] == lista_kolejnych_wczytanych[6])

            {

               if (lista_kolejnych_wczytanych[3] == lista_kolejnych_wczytanych[7]) // pokrywaja sie //?

               {

                  dotylu = 5;

                  if (nastepny_argument == 1)

                  {


                     nastepny_argument=nastepny_argument+3;

                     cout << "<0,4," << lista_kolejnych_wczytanych[8] << ">" << endl;

                     plik << "<0,4," << lista_kolejnych_wczytanych[8] << ">";

                  }

                  else

                  {

                     nastepny_argument=nastepny_argument+4;

                     cout << "<0,4," << lista_kolejnych_wczytanych[8] << ">" << endl;
```

```
                plik << "<0,4," << lista_kolejnych_wczytanych[8] << ">";
            }
        }
        else
        {
            dotylu = 4;

            nastepny_argument=nastepny_argument+4;
            cout << "<0,3," << lista_kolejnych_wczytanych[7] << ">" << endl;
            plik << "<0,3," << lista_kolejnych_wczytanych[7] << ">";
        }
    }
    else
    {
        dotylu = 3;
        //  Listawyj[nastepny_argument] = lista_kolejnych_wczytanych[4];
        nastepny_argument++;
        //    Listawyj[nastepny_argument] = lista_kolejnych_wczytanych[5];
        nastepny_argument++;
        //        Listawyj[nastepny_argument] = lista_kolejnych_wczytanych[6];
        nastepny_argument++;
        //cout << "Bufor " << lista_kolejnych_wczytanych[4] << " znaleziono w slowniku na 1
pozycji i kolejny element" << endl;
        cout << "<0,2," << lista_kolejnych_wczytanych[6] << ">" << endl;
        plik << "<0,2," << lista_kolejnych_wczytanych[6] << ">";
    }
}
else
{
    dotylu = 2;
    nastepny_argument=nastepny_argument+2;
```

```
            cout << "<0,1," << lista_kolejnych_wczytanych[5] << ">" << endl;
            plik << "<0,1," << lista_kolejnych_wczytanych[5] << ">";
        }
    }
    else if (lista_kolejnych_wczytanych[1] == lista_kolejnych_wczytanych[4]) // 3 ostatnie elem slownika
    {
        if (lista_kolejnych_wczytanych[2] == lista_kolejnych_wczytanych[5])
        {
            if (lista_kolejnych_wczytanych[3] == lista_kolejnych_wczytanych[6])
            {
                dotylu = 4;
                nastepny_argument=nastepny_argument+4;
                cout << "<1,3," << lista_kolejnych_wczytanych[7] << ">" << endl;
                plik << "<1,3," << lista_kolejnych_wczytanych[7] << ">";
            }
            else
            {
                dotylu = 3;
                nastepny_argument=nastepny_argument+3;
                cout << "<1,2," << lista_kolejnych_wczytanych[6] << ">" << endl;
                plik << "<1,2," << lista_kolejnych_wczytanych[6] << ">";
            }
        }
        else
        {
            dotylu = 2;
            nastepny_argument=nastepny_argument+2;
            cout << "<1,1," << lista_kolejnych_wczytanych[5] << ">" << endl;
            plik << "<1,1," << lista_kolejnych_wczytanych[5] << ">";
        }
```

```
}
else if (lista_kolejnych_wczytanych[2] == lista_kolejnych_wczytanych[4]) // 2 os
{
   if (lista_kolejnych_wczytanych[3] == lista_kolejnych_wczytanych[5])
   {
      dotylu = 3;
      nastepny_argument=nastepny_argument+3;
      cout << "<2,2," << lista_kolejnych_wczytanych[6] << ">" << endl;
      plik << "<2,2," << lista_kolejnych_wczytanych[6] << ">";
   }
   else
   {
      dotylu = 2;
      nastepny_argument=nastepny_argument+2;
      cout << "<2,1," << lista_kolejnych_wczytanych[5] << ">" << endl;
      plik << "<2,1," << lista_kolejnych_wczytanych[5] << ">";
   }
}
else if (lista_kolejnych_wczytanych[3] == lista_kolejnych_wczytanych[4]) // 1
{
   dotylu = 2;
   nastepny_argument=nastepny_argument+2;
   cout << "<3,1," << lista_kolejnych_wczytanych[5] << ">" << endl;
   plik << "<3,1," << lista_kolejnych_wczytanych[5] << ">";
}
else
{
   dotylu = 1;
   nastepny_argument++;
   cout << "<0,0," << lista_kolejnych_wczytanych[4] << ">" << endl;
   plik << "<0,0," << lista_kolejnych_wczytanych[4] << ">";
```

```
        }


        if (lista_kolejnych_wczytanych[0] == lista_kolejnych_wczytanych[4] &&
lista_kolejnych_wczytanych[1] == lista_kolejnych_wczytanych[5] &&
lista_kolejnych_wczytanych[2] == lista_kolejnych_wczytanych[6] &&
lista_kolejnych_wczytanych[3] == lista_kolejnych_wczytanych[7])
        {
            for (int i = 0; i <= t+7; i++)
            {
                lista_kolejnych_wczytanych[i] = lista_kolejnych_wczytanych[i + 4];
            }
        }


        else
        {
            for (int i = 0; i <= t+7; i++)
            {
                lista_kolejnych_wczytanych[i] = lista_kolejnych_wczytanych[i + dotylu];
            }
        }


    }
    plik.close();


}


void dekodowanie_lz77()
{


    fstream plik;
    char dekodowane[100] = { 0 };
    int tmp = 0;
```

```
char slownik[4] = { ' ' };
char Listawyj[100] = { ' ' };
int nastepny_argument = 0;
int kolejka = 0;
int dotylu = 0;
plik.open("zakodowane.txt", ios::in);
if (plik.is_open())
{
   while (!plik.eof())
   {
         plik >> dekodowane[tmp];
         tmp++;

      }
      cout << endl;
   }

plik.close();
int dlugoscWyrazenia = strlen(dekodowane)/7;
 char poz, dl, nastepny;
while (kolejka != dlugoscWyrazenia)
{   poz = dekodowane[1];
   dl = dekodowane[3];
   nastepny = dekodowane[5];
   int pozycja, dlugosc;
   pozycja = poz - 48;
   dlugosc = dl - 48;
   if(nastepny == '_'){
  Listawyj[nastepny_argument]= slownik[0];
   break;
   }
```

```
else if (nastepny_argument == 0)
{
   for (int k = 0; k <= 3; k++)
   {
      slownik[k] = nastepny;
   }
   Listawyj[nastepny_argument] = nastepny;
   nastepny_argument++;
}
else
{
   if (dlugosc == 0)
   {
      Listawyj[nastepny_argument] = nastepny;
      nastepny_argument++;
      slownik[0] = slownik[1];
      slownik[3] = nastepny;
   }
   else if (dlugosc == 1)
   {
      char szukana = slownik[pozycja];
      Listawyj[nastepny_argument] = szukana;
      nastepny_argument++;
      Listawyj[nastepny_argument] = nastepny;
      nastepny_argument++;
      for (int i = 0; i <= 1; i++)
      {
         slownik[i] = slownik[i + 1];
      }
      slownik[pozycja - 1] = szukana;
      slownik[3] = nastepny;
```

```
        }
        else
        {
           for (int k = 0; k < dlugosc; k++)
           {
              Listawyj[nastepny_argument] = slownik[pozycja + k];
              nastepny_argument++;
           }
           Listawyj[nastepny_argument] = nastepny;
           int l = 0;
           while (l != 4 - dlugosc - 1)
           {
              slownik[pozycja] = slownik[pozycja + dlugosc + 1];
              l++;
           }
           slownik[3] = nastepny;
           nastepny_argument++;
        }

     }
     for (int k = 0; k <= 100; k++)
     {
        dekodowane[k] = dekodowane[k + 7];
     }
        sleep(1);
        printf("<%d,%d,%c>  slownik %c%c%c%c\nzapisane Listawyj %s
\n",pozycja,dlugosc,nastepny, slownik[0],slownik[1],slownik[2],slownik[3], &Listawyj);
        kolejka++;
   }
   cout<<"Koncowa zdekodowana wiadomosc to: ";
   for (int i = 0; i <= 99; i++)
```

```
    {
        cout << Listawyj[i];
    }
}

int main()
{   char arr[100];
  ifstream ifs = ifstream("we12A.txt", ios_base::in);
        if (ifs.is_open() == true) {
            char c; int i = 0;
            while (ifs.get(c)) {
arr[i] = c;
i++;
            }
ifs.close(); arr[i] = '\0'; cout<<"Wiadomosc kodowa: "<<arr<<endl;
        }
        else
cout<< "File could not be open!" <<endl;


    cout << "Kodowanie slownikowe LZ77" << endl;
    cout << "--------------------------------------------" << endl;
    cout << "Zaczynam kodowanie do pliku Wyjsciowy_plik.txt" << endl<<endl<<endl;
    kodowanie_lz77(arr);
    cout << "Dekoduje z plku Wyjsciowy plik.txt: " << endl;
    dekodowanie_lz77();
    return 0;
}
```