Kodowanie Huffmana

Wygenerowano przez Doxygen 1.9.6

Rozdział 1

Indeks klas

1.1 Lista klas

Tutaj znajdują się klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

Rozdział 2

Indeks plików

2.1 Lista plików

Tutaj znajduje się lista wszystkich udokumentowanych plików z ich krótkimi opisami:

Rozdział 3

Dokumentacja klas

3.1 Dokumentacja struktury wezel

Struktura wezla uzywanego do tworzenia drzewa.

`#include <Funkcje.h>`

Atrybuty publiczne

- int **czestosc_wystepowania**

  ilosc danego bajtu w pliku
- int **index_tablicy**

  numer bajtu 0-256
- std::string **kod**

  kod binarny nadany w celu zakodowania
- struct wezel * **p_lewy**

  wskaznik do mlodszego lewego wezla
- struct wezel * **p_prawy**

  wskaznik do mlodszego prawego wezla

3.1.1 Opis szczegółowy

Struktura wezla uzywanego do tworzenia drzewa.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- Funkcje.h

Rozdział 4

Dokumentacja plików

4.1 Dokumentacja pliku Funkcje.h

Zalaczanie bibiliotek, naglowki funkcji oraz struktury i stale globalne.

```
#include <iostream>
#include <fstream>
#include <windows.h>
#include <cstddef>
#include <vector>
#include <string>
#include <sstream>
```

Komponenty

- struct wezel

  Struktura wezla uzywanego do tworzenia drzewa.

Funkcje

- wezel * stworzWezel (int wystepowanie)

  Alokowanie pamieci oraz tworzenie wezla, uzywanego do zbudowanie drzewa.
- bool sprawdz_argumenty (int liczba)

  Sprawdzenie czy uzytkownik wpisal poprawna liczbe argumentow.
- bool przypisz_argumenty (char *argv[ ], std::string &input, std::string &output, std::string &operacja)

  Przypisanie referencyjnie argumentow do zmiennych: wejscia, wyjscia oraz operacji.
- bool otworz_wejsciowy (std::string nazwa, std::ifstream &plik)

  Otwieranie pliku wejsciowego oraz sprawdzanie czy plik ten zostal otwarty poprawnie.
- bool otworz_wyjsciowy (std::string nazwa, std::ofstream &plik)

  Otwieranie pliku wyjsciowego oraz sprawdzanie czy plik ten zostal otwarty poprawnie.
- void zlicz_bajty_i_zapisz (std::vector< int > &kolejnosc, int tablica_bajtow[ ], std::ifstream &plik_wejsciowy)

  Przejscie calego pliku wejsciowego oraz zliczenie poszczegolnych bajtow oraz przypisanie ich kolejnosci do wektora.
- void dodaj_wezly (std::vector< wezel * > &wektor, int tablica[ ])

Tworzenie wezlow przy pomocy funckji stworzWezel() oraz dodanie do wektora tych wezlow, w ktorych czestosc wystepowanie poszczegolnych bajtow jeste wieksza od zera.

- void stworz_drzewo (std::vector< wezel * > &wektor_wezlow)

  Przeszukiwanie wektora wezlow w celu znalezienia dwoch takich, w ktorych czestosc wystepowanie jest najmniejsza. Nastepnie tworzenie nowego wezla o liczebnosci sumy tych dwoch oraz polaczenie ich z nowym wezlem. Usuniecie dwoch najmniejsz wezlow z wektora. Na koncu dodanie nowego wezla do vektora. Calosc jest powtarzana do momentu gdy wszystkie wezly utworza drzewo.
- void przypisz_kod (wezel *wezel, std::string tablica_kodow[ ])

  Rekurencyjne przypisywanie kodu do kazdego wezla. Pobieranie kodu z mlodszego wezla oraz laczenie go z kodem tego starszego.
- void oznacz_brak_kodu (std::string tablica_kodow[ ])

  Przypisanie "9" do wolnych miejsc tablicy kodow (zalozenie ze ta wartosc oznacza brak danego bajtu w pliku wejsciowym)
- void zapisz_bajty_dla_odkodowania (std::ofstream &plik_wyjsciowy, std::string tablica_kodow[ ])

  Zapisanie na poczatku pliku wyjsciowego wszystkich kodow bajtow w kolejnosci aby bylo mozliwe dekodowanie.
- void wpisz_zakodowane_bajty (std::vector< int > kolejnosc, std::ofstream &plik_wyjsciowy, std::string tablica_kodow[ ])

  Wpisywanie zakodowanych bajtow do pliku wyjsciowego w odpowiedniej kolejnosci.
- void zapisz_kody_do_bufora (std::string kody_bajtow_pobrane[ ], std::ifstream &plik_wejsciowy)

  Pobranie 256 kodow z pierwszych linijek zakodowanego pliku aby ustalic jak bajty zostaly zakodowane.
- void dekoduj (std::ifstream &plik_wejsciowy, std::ofstream &plik_wyjsciowy, std::string kody_bajtow_↩ pobrane[ ])

  Czytanie pojedynczo wszystkich kodow, sprawdzanie do ktorego bajta odnosi sie dany kod oraz zapisanie odczytanego bajta do pliku wyjsciowego.
- void zwolnij_pamiec (wezel *wezel)

  Zwalnianie pamieci, przechodzac po calym drzewie, zaczynajac od najmlodszego wezla.
- void pozytywny_komunikat (std::string komunikat)

  Wyswietlenie komunikatu w konsoli na zielono.
- void negatywny_komunikat (std::string komunikat)

  Wyswietlenie komunikatu w konsoli na czerwono.

Zmienne

- const int **liczba_bajtow** = 256

  stala globalna, liczba wszystkich mozliwych bajtow

4.1.1 Opis szczegółowy

Zalaczanie bibiliotek, naglowki funkcji oraz struktury i stale globalne.

4.1.2 Dokumentacja funkcji

4.1.2.1 dekoduj()

```
void dekoduj (
            std::ifstream & plik_wejsciowy,
            std::ofstream & plik_wyjsciowy,
            std::string kody_bajtow_pobrane[] )
```

Czytanie pojedynczo wszystkich kodow, sprawdzanie do ktorego bajta odnosi sie dany kod oraz zapisanie odczytanego bajta do pliku wyjsciowego.

Parametry

plik_wejsciowy	referencyjne przekazanie pliku wejsciowego
plik_wyjsciowiy	referencyjne przekazanie pliku wyjsciowego
kody_bajtow_pobrane	tablica w ktorej zapisywane sa kody kazdego z bajtow

4.1.2.2 dodaj_wezly()

```
void dodaj_wezly (
            std::vector< wezel * > & wektor,
            int tablica[] )
```

Tworzenie wezlow przy pomocy funckji stworzWezel() oraz dodanie do wektora tych wezlow, w ktorych czestosc wystepowanie poszczegolnych bajtow jeste wieksza od zera.

Parametry

wektor	wektor wskaznikow wezlow, w ktorym przychowywane sa wszyskie wezly
tablica	tablica z iloscia wystepowania kazdego bajtu w pliku wejsciowym

4.1.2.3 negatywny_komunikat()

```
void negatywny_komunikat (
            std::string komunikat )
```

Wyswietlenie komunikatu w konsoli na czerwono.

Parametry

komunikat	wiadomosc, ktora chcemy wyswietlic

4.1.2.4 otworz_wejsciowy()

```
bool otworz_wejsciowy (
            std::string nazwa,
            std::ifstream & plik )
```

Otwieranie pliku wejsciowego oraz sprawdzanie czy plik ten zostal otwarty poprawnie.

Parametry

nazwa	nazwa pliku wejsciowego
plik	referencyjne przekazanie pliku

Zwracane wartości

TRUE	funckja zwraca referencyjnie otwarty plik, program dziala dalej
FALSE	program sie konczy oraz wypisana jest wiadomosc

4.1.2.5 otworz_wyjsciowy()

```
bool otworz_wyjsciowy (
            std::string nazwa,
            std::ofstream & plik )
```

Otwieranie pliku wyjsciowego oraz sprawdzanie czy plik ten zostal otwarty poprawnie.

Parametry

nazwa	nazwa pliku wyjsciowego
plik	referencyjne przekazanie pliku

Zwracane wartości

TRUE	funckja zwraca referencyjnie otwarty plik, program dziala dalej
FALSE	program sie konczy oraz wypisana jest wiadomosc

4.1.2.6 oznacz_brak_kodu()

```
void oznacz_brak_kodu (
            std::string tablica_kodow[] )
```

Przypisanie "9" do wolnych miejsc tablicy kodow (zalozenie ze ta wartosc oznacza brak danego bajtu w pliku wejsciowym)

Parametry

tablica_kodow	tablica w ktorej zapisywane sa przekonwertowane bajty

4.1.2.7 pozytywny_komunikat()

```
void pozytywny_komunikat (
            std::string komunikat )
```

Wyswietlenie komunikatu w konsoli na zielono.

Parametry

komunikat	wiadomosc, ktora chcemy wyswietlic

4.1.2.8 przypisz_argumenty()

```
bool przypisz_argumenty (
            char * argv[],
            std::string & input,
            std::string & output,
            std::string & operacja )
```

Przypisanie referencyjnie argumentow do zmiennych: wejscia, wyjscia oraz operacji.

Parametry

argv	tablica argumentow wpisanych podczas startowania programu
input	nazwa pliku wejsciowego
output	nazwa pliku wyjsciowego
operacja	operacja programu wpisana jako agrument (kodowanie, dekodowanie)

Zwracane wartości

TRUE	program dziala dalej
FALSE	program sie konczy oraz wypisywana jest wiadomosc

4.1.2.9 przypisz_kod()

```
void przypisz_kod (
            wezel * wezel,
            std::string tablica_kodow[] )
```

Rekurencyjne przypisywanie kodu do kazdego wezla. Pobieranie kodu z mlodszego wezla oraz laczenie go z kodem tego starszego.

Parametry

wezel	najstarszy wezel drzewa od ktorego zaczyna sie funkcja
tablica_kodow	tablica w ktorej zapisywane sa przekonwertowane bajty

4.1.2.10 sprawdz_argumenty()

```
bool sprawdz_argumenty (
            int liczba )
```

Sprawdzenie czy uzytkownik wpisal poprawna liczbe argumentow.

Parametry

liczba	ilosc agrumentow wpisana podczas startowania programu

Zwracane wartości

TRUE	program dziala dalej
FALSE	program sie konczy oraz wypisywana jest wiadomosc

4.1.2.11 stworz_drzewo()

```
void stworz_drzewo (
            std::vector< wezel * > & wektor_wezlow )
```

Przeszukiwanie wektora wezlow w celu znalezienia dwoch takich, w ktorych czestosc wystepowanie jest najmniejsza. Nastepnie tworzenie nowego wezla o liczebnosci sumy tych dwoch oraz polaczenie ich z nowym wezlem. Usuniecie dwoch najmniejsz wezlow z wektora. Na koncu dodanie nowego wezla do vektora. Calosc jest powtarzana do momentu gdy wszystkie wezly utworza drzewo.

Parametry

wektor_wezlow	wektor wskaznikow wezlow w ktorym znajduja sie wszystkie wezly z liczebnoscia bajtow

4.1.2.12 stworzWezel()

```
wezel * stworzWezel (
            int wystepowanie )
```

Alokowanie pamieci oraz tworzenie wezla, uzywanego do zbudowanie drzewa.

Parametry

wystepowanie	ilosc danego bajtu w pliku

Zwraca

wskaznik stworzonego wezla

4.1.2.13 wpisz_zakodowane_bajty()

```
void wpisz_zakodowane_bajty (
            std::vector< int > kolejnosc,
            std::ofstream & plik_wyjsciowy,
            std::string tablica_kodow[] )
```

Wpisywanie zakodowanych bajtow do pliku wyjsciowego w odpowiedniej kolejnosci.

Parametry

kolejnosc	wektor, w ktorym zapisana jest kolejnosc wystepowania bajtow z pliku wejsciowego
plik_wyjsciowy	referencyjne przekazanie pliku wyjsciowego
tablica_kodow	tablica w ktorej zapisywane sa przekonwertowane bajty

4.1.2.14 zapisz_bajty_dla_odkodowania()

```
void zapisz_bajty_dla_odkodowania (
            std::ofstream & plik_wyjsciowy,
            std::string tablica_kodow[] )
```

Zapisanie na poczatku pliku wyjsciowego wszystkich kodow bajtow w kolejnosci aby bylo mozliwe dekodowanie.

Parametry

plik_wyjsciowy	przekazanie pliku wyjsciowego
tablica_kodow	tablica w ktorej zapisywane sa przekonwertowane bajty

4.1.2.15 zapisz_kody_do_bufora()

```
void zapisz_kody_do_bufora (
            std::string kody_bajtow_pobrane[],
            std::ifstream & plik_wejsciowy )
```

Pobranie 256 kodow z pierwszych linijek zakodowanego pliku aby ustalic jak bajty zostaly zakodowane.

Parametry

kody_bajtow_pobrane	tablica w ktorej zapisywane sa kody kazdego z bajtow
plik_wejsciowy	referencyjne przekazanie pliku wejsciowego

4.1.2.16 zlicz_bajty_i_zapisz()

```
void zlicz_bajty_i_zapisz (
            std::vector< int > & kolejnosc,
            int tablica_bajtow[],
            std::ifstream & plik_wejsciowy )
```

Przejscie calego pliku wejsciowego oraz zliczenie poszczegolnych bajtow oraz przypisanie ich kolejnosci do wektora.

Parametry

kolejnosc	wektor w ktorym jest zapisywana kolejnosc wystepowania bajtow w pliku wejsciowym
tablica_bajtow	tablica z iloscia wystepowania kazdego bajtu w pliku wejsciowym
plik_wejsciowy	referencyjne przekazanie pliku wejsciowego

4.1.2.17 zwolnij_pamiec()

```
void zwolnij_pamiec (
            wezel * wezel )
```

Zwalnianie pamieci, przechodzac po calym drzewie, zaczynajac od najmlodszego wezla.

Parametry

wezel	wskaznik najmlodszego wezla w drzewie

4.2 Funkcje.h

Idź do dokumentacji tego pliku.

```
00001
00006 #pragma once
00007
00008 #include <iostream>
00009 #include <fstream>
00010 #include <windows.h>
00011 #include <cstddef>
00012 #include <vector>
00013 #include <string>
00014 #include <sstream>
00015
00016 const int liczba_bajtow = 256;
00017
00019 struct wezel
00020 {
00021     int czestosc_wystepowania;
00022     int index_tablicy;
00023     std::string kod;
00024     struct wezel* p_lewy;
00025     struct wezel* p_prawy;
00026 };
00027
00032 wezel* stworzWezel(int wystepowanie);
00033
00039 bool sprawdz_argumenty(int liczba);
00040
00049 bool przypisz_argumenty(char* argv[], std::string& input, std::string& output, std::string& operacja);
```

```
00050
00057 bool otworz_wejsciowy(std::string nazwa, std::ifstream& plik);
00058
00065 bool otworz_wyjsciowy(std::string nazwa, std::ofstream& plik);
00066
00072 void zlicz_bajty_i_zapisz(std::vector<int>& kolejnosc, int tablica_bajtow[], std::ifstream&
      plik_wejsciowy);
00073
00078 void dodaj_wezly(std::vector<wezel*>& wektor, int tablica[]);
00079
00086 void stworz_drzewo(std::vector<wezel*>& wektor_wezlow);
00087
00092 void przypisz_kod(wezel* wezel, std::string tablica_kodow[]);
00093
00097 void oznacz_brak_kodu(std::string tablica_kodow[]);
00098
00103 void zapisz_bajty_dla_odkodowania(std::ofstream& plik_wyjsciowy, std::string tablica_kodow[]);
00104
00110 void wpisz_zakodowane_bajty(std::vector<int> kolejnosc, std::ofstream& plik_wyjsciowy, std::string
      tablica_kodow[]);
00111
00116 void zapisz_kody_do_bufora(std::string kody_bajtow_pobrane[], std::ifstream& plik_wejsciowy);
00117
00123 void dekoduj(std::ifstream& plik_wejsciowy, std::ofstream& plik_wyjsciowy, std::string
      kody_bajtow_pobrane[]);
00124
00128 void zwolnij_pamiec(wezel* wezel);
00129
00133 void pozytywny_komunikat(std::string komunikat);
00134
00138 void negatywny_komunikat(std::string komunikat);
```

Skorowidz