My Project

Wygenerowano przez Doxygen 1.9.3

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 plików z ich krótkimi opisami:

Rozdział 3

Dokumentacja klas

3.1 Dokumentacja struktury key

`#include <Struktury.h>`

Atrybuty publiczne

- int znak
- int lw
- std::string kod

3.1.1 Opis szczegółowy

struktura do stworzenia klucza do zakodowania

Parametry

znak	ktory pojawil sie w pliku
licznik	- zlicza ile razy dany znak sie pojawil
kod	- kod ktory powstal poprzez drzewo.

Definicja w linii 23 pliku Struktury.h.

3.1.2 Dokumentacja atrybutów składowych

3.1.2.1 kod

`std::string key::kod`

Definicja w linii 27 pliku Struktury.h.

3.1.2.2 lw

```
int key::lw
```

Definicja w linii 26 pliku Struktury.h.

3.1.2.3 znak

```
int key::znak
```

Definicja w linii 25 pliku Struktury.h.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- C:/Users/jwiec/Dropbox/Komputer/Desktop/f7139937-gr15-repo/projekt/kodowanie_huffmana/Struktury.h

3.2 Dokumentacja struktury klucz

```
#include <Struktury.h>
```

Atrybuty publiczne

- int z
- std::string k

3.2.1 Opis szczegółowy

struktura do wyczytania klucza z pliku

Parametry

z	- znak, konkretna litera z ktora pojawila sie w tekscie
k	- kod z slownika przypisany do konkretnej litery

Definicja w linii 32 pliku Struktury.h.

3.2.2 Dokumentacja atrybutów składowych

3.2.2.1 k

```
std::string klucz::k
```

Definicja w linii 35 pliku Struktury.h.

3.2.2.2 z

```
int klucz::z
```

Definicja w linii 34 pliku Struktury.h.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- C:/Users/jwiec/Dropbox/Komputer/Desktop/f7139937-gr15-repo/projekt/kodowanie_huffmana/Struktury.h

3.3 Dokumentacja struktury node

```
#include <Struktury.h>
```

Atrybuty publiczne

- int znak =NULL
- int licznik =0
- node * lewe = nullptr
- node * prawe = nullptr

3.3.1 Opis szczegółowy

file Struktura do wyczytania znakow, ktore wystapily w pliku i zliczenia ich ilosci wystapien, oraz wskazniki "lewe", "prawe", potrzebne do stworzenia drzewa.

Parametry

znak	- znak pliku
licznik	- zlicza ile razy dany znak wystapil
lewe	- wskaznik na lewa galaz
prawe	- wskaznik na prawa galaz

Definicja w linii 11 pliku Struktury.h.

3.3.2 Dokumentacja atrybutów składowych

3.3.2.1 lewe

```
node* node::lewe = nullptr
```

Definicja w linii 15 pliku Struktury.h.

3.3.2.2 licznik

```
int node::licznik =0
```

Definicja w linii 14 pliku Struktury.h.

3.3.2.3 prawe

```
node* node::prawe = nullptr
```

Definicja w linii 16 pliku Struktury.h.

3.3.2.4 znak

```
int node::znak =NULL
```

Definicja w linii 13 pliku Struktury.h.

Dokumentacja dla tej struktury została wygenerowana z pliku:

- C:/Users/jwiec/Dropbox/Komputer/Desktop/f7139937-gr15-repo/projekt/kodowanie_huffmana/Struktury.h

Rozdział 4

Dokumentacja plików

4.1 Dokumentacja pliku C:/Users/jwiec/Dropbox/Komputer/Desktop/f7139937-gr15-repo/projekt/kodowanie_huffmana/Funkcje.cpp

```
#include <iostream>
#include <iomanip>
#include <vector>
#include <sstream>
#include <string>
#include <fstream>
#include <chrono>
#include <algorithm>
#include "Funkcje.h"
#include "Struktury.h"
```

Funkcje

- void countchar (ifstream &plik, int *tab)
- void przep (int *tab, vector< node > &vec)
- void drzewo (vector< node > &tab)
- void rek (node *korz, string code, vector< key > &kod)
- void wypK (ofstream &plik, vector< key >klucz)
- void koduj (ifstream &plik, ofstream &in, vector< key > klucz)
- void przepklucz (ifstream &plik, vector< klucz > &k)
- void odszyfr (ifstream &plik, ofstream &in, vector< klucz > k)

  funkcja bada po kolei dlugosci zero-jedynkowych tekstow, jak odpowiednia dlugosc bedzie sie zgadzac z kluczem, to wypisze ja do pliku wyjsciowego

4.1.1 Dokumentacja funkcji

4.1.1.1 countchar()

```
void countchar (
            ifstream & plik,
            int * tab )
```

Funkcja majaca na celu zczytac birarnie litery z pliku do tablicy poprzez bufor. Kazda litera jest zapisana z tablicy ascii jako int.

Parametry

tab	- tablica wczytujaca znaki jako inty na podstwawie tablicy ascii.

Definicja w linii 17 pliku Funkcje.cpp.

4.1.1.2 drzewo()

```
void drzewo (
            vector< node > & tab )
```

Funckja tworzy drzewo z vectora node. Do wskaznikow skruktury sa przypisane ostatnie wartosci vectora, po czym sa one sumowane i wrzucane do zmiennej pomocniczej "temp". Nastepnie ostatnie te wartosci sa usuwane z listy i jest wrzucany temp do wektora. czynnosc sie ta powtarza tak dlugo, az w vectorze nie zostanie ostatnia wartosc, czyli glowny korzen.

Definicja w linii 56 pliku Funkcje.cpp.

4.1.1.3 koduj()

```
void koduj (
            ifstream & plik,
            ofstream & inp,
            vector< key > klucz )
```

funkcja ma analizowac pojedynczo litery z pliku wejsciowego, porownywac z vectorem klucz i na podstawie tego przepisywac do pliku wyjsciowego kod zawarty w vectorze

Parametry

plik	- plik wejsciowy
inp	- plik wyjsciowy
klucz	- vector struktury, zawiera w zobie znaki i kody do znakow

Definicja w linii 102 pliku Funkcje.cpp.

4.1.1.4 odszyfr()

```
void odszyfr (
            ifstream & plik,
            ofstream & inp,
            vector< klucz > k )
```

funkcja bada po kolei dlugosci zero-jedynkowych tekstow, jak odpowiednia dlugosc bedzie sie zgadzac z kluczem, to wypisze ja do pliku wyjsciowego

Parametry

plik	plik wejsciowy
inp	plik wyjsciowy
k	vector klucza

Definicja w linii 141 pliku Funkcje.cpp.

4.1.1.5 przep()

```
void przep (
            int * tab,
            vector< node > & vec )
```

funckja przepisujaca zawartosc tablicy do vectora node. Zawartosc przepisuje jako licznik a indeks komorki jako znak

Parametry

∗tab	tablica int
vec	- vector struktury node

Definicja w linii 42 pliku Funkcje.cpp.

4.1.1.6 przepklucz()

```
void przepklucz (
            ifstream & plik,
            vector< klucz > & k )
```

Funckja przepisuje zawartosc z pliku wejsciowego do vectora, wykorzystywane go dekodowania

Parametry

plik	- plik wejsciowy
k	- vector struktury, majacy przechowywac dane z pliku

Definicja w linii 126 pliku Funkcje.cpp.

4.1.1.7 rek()

```
void rek (
            node * korz,
            string code,
            vector< key > & kod )
```

Funckja nadajaca rekurencyjnie kod zero-jedynkowy typu string kazdemu znakowi. przechodzi od korzenia tak dlugo, dopuki galaz lewa i prawa nie beda nullptr.

Parametry

korz	pierwsza wartosc vectora
code	poczatkowy kod string
kod	vector do ktorego sa zapisywane informacje o kodzie i literach

Definicja w linii 72 pliku Funkcje.cpp.

4.1.1.8 wypK()

```
void wypK (
            ofstream & plik,
            vector< key > klucz )
```

Funckja wpisyjaca do pliku zawartosc vectora klucz.

Parametry

plik	- plik do ktorego jest zapisywany klucz, znaki sa zapisane kodem ascii
klucz	- vector w ktorym sa zapisane kody

Definicja w linii 93 pliku Funkcje.cpp.

4.2 Funkcje.cpp

Idź do dokumentacji tego pliku.

```
00001
00002 #include <iostream>
00003 #include <iomanip>
00004 #include <vector>
00005 #include <sstream>
00006 #include <string>
00007 #include <fstream>
00008 #include <chrono>
00009 #include <algorithm>
```

```
00010
00011 #include "Funkcje.h"
00012 #include "Struktury.h"
00013
00014
00015 using namespace std;
00016
00017 void countchar(ifstream& plik,int *tab)
00018 {
00019     plik.seekg(0, plik.end);
00020     int d = plik.tellg();
00021     plik.seekg(0, plik.beg);
00022
00023     char buf[1000];
00024
00025     for (int k=0;k<(d/1000);k++)
00026     {
00027         plik.read(buf, 1000);
00028         for (int i = 0; i < 1000; i++)
00029         {
00030             tab[buf[i]]++;
00031         }
00032     }
00033     int licz = d - plik.gcount();
00034     plik.read(buf, licz);
00035     for (int i = 0; i < licz; i++)
00036     {
00037         tab[buf[i]]++;
00038     }
00039
00040 }
00041
00042 void przep(int* tab, vector<node>& vec)
00043 {
00044     for (int i = 0; i < 256; i++)
00045     {
00046         if (tab[i] != 0)
00047         {
00048             node z;
00049             z.znak = i;
00050             z.licznik = tab[i];
00051             vec.push_back(z);
00052         }
00053     }
00054 }
00055
00056 void drzewo(vector<node> &tab)
00057 {
00058     while (tab.size() > 1)
00059     {
00060         node temp;
00061         temp.lewe=new node(tab[tab.size()-1]);
00062         temp.prawe=new node(tab[tab.size()-2]);
00063         temp.licznik = tab[tab.size() - 1].licznik+ tab[tab.size() - 2].licznik;
00064         tab.pop_back();
00065         tab.pop_back();
00066         tab.push_back(temp);
00067         sort(tab.begin(), tab.end(), [](const node& l, const node& p) {return l.licznik > p.licznik;
      });
00068
00069     }
00070 }
00071
00072 void rek(node *korz, string code, vector<key>&kod)
00073 {
00074     if (korz->znak != NULL)
00075     {
00076         key temp;
00077         temp.znak=korz->znak;
00078         temp.kod=code;
00079         temp.lw=korz->licznik;
00080         kod.push_back(temp);
00081         return;
00082     }
00083     rek(korz->lewe, code + "0", kod);
00084     rek(korz->prawe, code + "1", kod);
00085
00086     delete korz->lewe;
00087     delete korz->prawe;
00088     korz->lewe = nullptr;
00089     korz->prawe = nullptr;
00090     sort(kod.begin(), kod.end(), [](const key& lewy, const key& prawy) { return lewy.lw > prawy.lw;
      });
00091 }
00092
00093 void wypK(ofstream &plik,vector<key>klucz)
00094 {
```

```
00095     for (int i = 0; i < klucz.size(); i++)
00096     {
00097         plik  « klucz[i].znak « '\t' « klucz[i].kod « endl;
00098     }
00099     plik.close();
00100  }
00101
00102 void koduj(ifstream& plik,ofstream &in,vector<key> klucz)
00103 {
00104     string tekst;
00105     plik.seekg(0, plik.beg);
00106     while (getline(plik, tekst))
00107     {
00108
00109         for (int i = 0; i < tekst.length(); i++)
00110         {
00111             bool jest = false;
00112             for (int j = 0; j < klucz.size() && !jest; j++)
00113             {
00114                 if (tekst[i] == (char)klucz[j].znak)
00115                 {
00116                     in « klucz[j].kod;
00117                     jest = true;
00118                 }
00119             }
00120         }
00121     }
00122     plik.close();
00123     in.close();
00124 }
00125
00126 void przepklucz(ifstream& plik, vector<klucz> &k)
00127 {
00128     int z;
00129     string kod;
00130     while (plik » z » kod)
00131     {
00132         klucz temp;
00133         temp.z = z;
00134         temp.k = kod;
00135         k.push_back(temp);
00136     }
00137     sort(k.begin(), k.end(), [](const klucz& l, const klucz& p) {return l.k.length() <
       p.k.length();});
00138     plik.close();
00139 }
00140
00141 void odszyfr(ifstream& plik, ofstream& in, vector<klucz> k)
00142 {
00143     plik.seekg(0, plik.beg);
00144     string tekst;
00145     vector<klucz> temp = k;
00146     while (getline(plik,tekst))
00147     {
00148
00149         string str{};
00150         for (int i = 0; i < tekst.length(); i++)
00151         {
00152             str += tekst[i];
00153             bool jest = false;
00154             for (int j = 0; j < k.size() ; j++)
00155             {
00156                 if (str == temp[j].k)
00157                 {
00158                     in « (char)k[j].z;
00159                     str = {};
00160                 }
00161             }
00162
00163         }
00164     }
00165     plik.close();
00166     in.close();
00167
00168 }
```

4.3 Dokumentacja pliku C:/Users/jwiec/Dropbox/Komputer/↩ Desktop/f7139937-gr15-repo/projekt/kodowanie_huffmana/Funkcje.h

```
#include <iostream>
#include <iomanip>
```

```
#include <vector>
#include <sstream>
#include <string>
#include <fstream>
#include "Struktury.h"
```

Funkcje

- void countchar (ifstream &plik, int *tab)
- void przep (int *tab, vector< node > &vec)
- void drzewo (vector< node > &tab)
- void rek (node *korz, string code, vector< key > &kod)
- void wypK (ofstream &plik, vector< key >klucz)
- void koduj (ifstream &plik, ofstream &inp, vector< key >klucz)
- void przepklucz (ifstream &plik, vector< klucz > &k)
- void odszyfr (ifstream &plik, ofstream &inp, vector< klucz > k)

  funkcja bada po kolei dlugosci zero-jedynkowych tekstow, jak odpowiednia dlugosc bedzie sie zgadzac z kluczem, to wypisze ja do pliku wyjsciowego

4.3.1 Dokumentacja funkcji

4.3.1.1 countchar()

```
void countchar (
            ifstream & plik,
            int * tab )
```

Funkcja majaca na celu zczytac birarnie litery z pliku do tablicy poprzez bufor. Kazda litera jest zapisana z tablicy ascii jako int.

Parametry

tab	- tablica wczytujaca znaki jako inty na podstwawie tablicy ascii.

Definicja w linii 17 pliku Funkcje.cpp.

4.3.1.2 drzewo()

```
void drzewo (
            vector< node > & tab )
```

Funckja tworzy drzewo z vectora node. Do wskaznikow skruktury sa przypisane ostatnie wartosci vectora, po czym sa one sumowane i wrzucane do zmiennej pomocniczej "temp". Nastepnie ostatnie te wartosci sa usuwane z listy

i jest wrzucany temp do wektora. czynnosc sie ta powtarza tak dlugo, az w vectorze nie zostanie ostatnia wartosc, czyli glowny korzen.

Definicja w linii 56 pliku Funkcje.cpp.

4.3.1.3 koduj()

```
void koduj (
            ifstream & plik,
            ofstream & inp,
            vector< key > klucz )
```

funkcja ma analizowac pojedynczo litery z pliku wejsciowego, porownywac z vectorem klucz i na podstawie tego przepisywac do pliku wyjsciowego kod zawarty w vectorze

Parametry

plik	- plik wejsciowy
inp	- plik wyjsciowy
klucz	- vector struktury, zawiera w zobie znaki i kody do znakow

Definicja w linii 102 pliku Funkcje.cpp.

4.3.1.4 odszyfr()

```
void odszyfr (
            ifstream & plik,
            ofstream & inp,
            vector< klucz > k )
```

funkcja bada po kolei dlugosci zero-jedynkowych tekstow, jak odpowiednia dlugosc bedzie sie zgadzac z kluczem, to wypisze ja do pliku wyjsciowego

Parametry

plik	plik wejsciowy
inp	plik wyjsciowy
k	vector klucza

Definicja w linii 141 pliku Funkcje.cpp.

4.3.1.5 przep()

```
void przep (
```

```
            int * tab,
            vector< node > & vec )
```

funckja przepisujaca zawartosc tablicy do vectora node. Zawartosc przepisuje jako licznik a indeks komorki jako znak

Parametry

**tab*	tablica int
vec	- vector struktury node

Definicja w linii 42 pliku Funkcje.cpp.

4.3.1.6 przepklucz()

```
void przepklucz (
            ifstream & plik,
            vector< klucz > & k )
```

Funckja przepisuje zawartosc z pliku wejsciowego do vectora, wykorzystywane go dekodowania

Parametry

plik	- plik wejsciowy
k	- vector struktury, majacy przechowywac dane z pliku

Definicja w linii 126 pliku Funkcje.cpp.

4.3.1.7 rek()

```
void rek (
            node * korz,
            string code,
            vector< key > & kod )
```

Funckja nadajaca rekurencyjnie kod zero-jedynkowy typu string kazdemu znakowi. przechodzi od korzenia tak dlugo, dopuki galaz lewa i prawa nie beda nullptr.

Parametry

korz	pierwsza wartosc vectora
code	poczatkowy kod string
kod	vector do ktorego sa zapisywane informacje o kodzie i literach

Definicja w linii 72 pliku Funkcje.cpp.

4.3.1.8 wypK()

```
void wypK (
            ofstream & plik,
            vector< key > klucz )
```

Funckja wpisyjaca do pliku zawartosc vectora klucz.

Parametry

plik	- plik do ktorego jest zapisywany klucz, znaki sa zapisane kodem ascii
klucz	- vector w ktorym sa zapisane kody

Definicja w linii 93 pliku Funkcje.cpp.

4.4 Funkcje.h

Idź do dokumentacji tego pliku.

```
00001
00002 #ifndef FUNKCJE_H
00003 #define FUNKCJE_H
00004
00005 #include <iostream>
00006 #include <iomanip>
00007 #include <vector>
00008 #include <sstream>
00009 #include <string>
00010 #include <fstream>
00011 #include "Struktury.h"
00012
00013 using namespace std;
00014
00015
00018 void countchar(ifstream& plik, int* tab);
00019
00024 void przep(int* tab, vector<node>& vec);
00025
00029 void drzewo(vector<node>& tab);
00030
00036 void rek(node* korz, string code, vector<key>& kod);
00037
00041 void wypK(ofstream& plik, vector<key>klucz);
00042
00048 void koduj(ifstream& plik, ofstream& inp, vector<key>klucz);
00049
00050
00054 void przepklucz(ifstream& plik, vector<klucz>& k);
00055
00062 void odszyfr(ifstream& plik, ofstream& inp, vector<klucz> k);
00063
00064 #endif
```

4.5 Dokumentacja pliku C:/Users/jwiec/Dropbox/Komputer/↩ Desktop/f7139937-gr15-repo/projekt/kodowanie_huffmana/main.cpp

```
#include <iostream>
#include <iomanip>
#include <vector>
```

```
#include <sstream>
#include <string>
#include <fstream>
#include <algorithm>
#include <cstdlib>
#include "Funkcje.h"
#include "Struktury.h"
```

Funkcje

- int main (int argc, const char ∗argv[ ])

4.5.1 Dokumentacja funkcji

4.5.1.1 main()

```
int main (
            int argc,
            const char * argv[] )
```

Funkcja main, punkt wyjcia do wykonywania programu

Parametry

argc	liczba parametrow przy uruchomieniu programu
argv	tablica wskaznikow na lancuchy uzytych przy uruchomienieu programu

Zwraca

jeli program wykona siawidowo zwracane jest 0

Definicja w linii 25 pliku main.cpp.

4.6 main.cpp

Idź do dokumentacji tego pliku.

```
00001
00002 #include <iostream>
00003 #include <iomanip>
00004 #include <vector>
00005 #include <sstream>
00006 #include <string>
00007 #include <fstream>
00008 #include <algorithm>
00009 #include <cstdlib>
00010
00011 #include "Funkcje.h"
00012 #include "Struktury.h"
00013
```

```
00014
00015 using namespace std;
00016
00025 int main(int argc, const char* argv[])
00026 {
00027
00028
00029     cout « "--------------------------"«endl;
00030     string file_in, file_out, mod, cod;
00031     string napis;
00032     if (argc ==9)
00033     {
00034         for (int i = 0; i < argc-1; i++)
00035         {
00036             napis = argv[i];
00037             if (napis == "-o")
00038             {
00039                 file_out = argv[i + 1];
00040             }
00041             else if (napis == "-i")
00042             {
00043                 file_in = argv[i + 1];
00044             }
00045             else if (napis == "-t")
00046             {
00047                 mod = argv[i + 1];
00048             }
00049             else if (napis == "-s")
00050             {
00051                 cod = argv[i+1];
00052             }
00053         }
00054
00055     }
00056     else
00057     {
00058         cout « "Nie prawidlowa ilosc wprowadzonych parametrow" « endl;
00059         return 0;
00060     }
00061
00062
00063
00064     //KODOWANIE
00065     if (mod == "k")
00066     {
00067         ifstream out(file_in, ios_base::binary);
00068         if (out)
00069         {
00070             vector<node> cojest;
00071             int tablica[256]{ 0 };
00072             countchar(out, tablica);
00073             przep(tablica, cojest);
00074             drzewo(cojest);
00075
00076             vector<key>kl;
00077             rek(&cojest[0],"", kl);
00078             //cout « cojest[0].licznik;
00079             ofstream in(cod);
00080             if (in.is_open())
00081             {
00082                 wypK(in, kl);
00083             }
00084             else
00085                 cerr « "\nblad otwarcia pliku ink\n";
00086             ofstream wp(file_out);
00087             if (wp.is_open()) {
00088                 koduj(out, wp, kl);
00089             }
00090             else
00091                 cerr « "blad otwarcia pliku wp\n";
00092
00093
00094         }
00095         else
00096             cerr « "blad otwarcia pliku out";
00097     }
00098     //DEKODOWANIE
00099     else if (mod == "d")
00100     {
00101         cout « "hej";
00102         ifstream od(file_in);
00103         if (od)
00104         {
00105             ifstream ke(cod);
00106             if (ke)
00107             {
00108                 vector<klucz>key;
```

```
00109                 przepklucz(ke, key);
00110                 ofstream odkod(file_out);
00111                 if (odkod)
00112                 {
00113                     odszyfr(od, odkod, key);
00114                 }
00115                 else
00116                     cerr«"blad otwrcia pliku of\n";
00117             }
00118             else
00119                 cerr « "blad otwarcia pliku key\n";
00120         }
00121         else
00122             cerr « "blad otwarcia pliku ink\n";
00123
00124     }
00125
00126     return 0;
00127 }
```

4.7 Dokumentacja pliku C:/Users/jwiec/Dropbox/Komputer/Desktop/f7139937-gr15-repo/projekt/kodowanie_huffmana/Struktury.h

```
#include <string>
```

Komponenty

- struct node
- struct key
- struct klucz

4.8 Struktury.h

Idź do dokumentacji tego pliku.

```
00001
00002 #include <string>
00003 #ifndef STRUKTURY_H
00004 #define STRUKTURY_H
00005
00011 struct node {
00012
00013     int znak=NULL;
00014     int licznik=0;
00015     node* lewe = nullptr;
00016     node* prawe = nullptr;
00017 };
00018
00023 struct key {
00024
00025     int znak;
00026     int lw;
00027     std:: string kod;
00028 };
00032 struct klucz {
00033
00034     int z;
00035     std::string k;
00036 };
00037 #endif
```

Indeks