|  |
| --- |
| **Programowanie Obiektowe** |
| **Autor: Hubert Zając, Elektronika 283056** |
| https://github.com/huuubertz/file-operations.git |

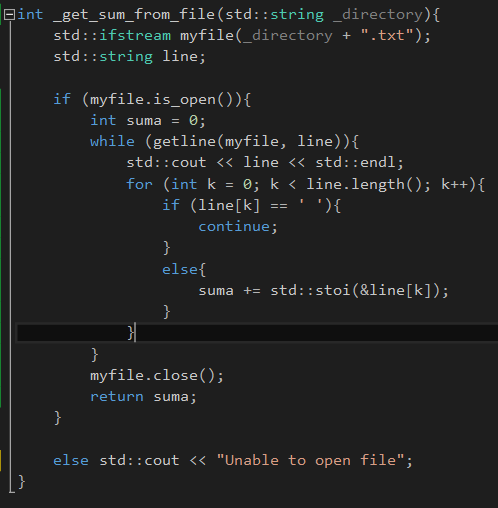
**Zadanie 8.2.7**

**OPIS:**

W zadaniu korzystamy z bibliotek: String i Fstream.

Algorytm wczytuje i otwiera plik z ścieżki, którą podajemy. Następnie wczytuje linie z pliku i sprawdza gdzie występują białe znaki. Jeżeli jest to biały znak to nie robi nic jeżeli jest to inny znak, konwertuje go na int i dodaje do zmiennej suma.

**KOD:**



**UWAGI:**

- argument bool do funkcji, który (if true) wyświetlałby zawartość pliku.

- Dodatkową opcją mogło by być branie pod uwagę każdej nowej linii czyli sumujemy liniami, i dodatkowa opcja (parametr bool), która pozwalałby najpierw zsumować linami, a później w zależności czy true czy false sumuj sume każdej linii ze sobą.

- W zadaniu moglibyśmy również dodać warunek sprawdzający czy dany wyraz jest liczbą czy innym znakiem.

**Zadanie 8.2.8**

**OPIS:**

W zadaniu korzystamy z bibliotek: String i Fstream, Vector, Algorithm.

Funkcja otwiera plik .txt i odczytuje z niego dane liniami. Następnie algorytm sprawdza gdzie mamy białe znaki, a gdzie występuje jakiś znak. Znam inni niż biały zostaje przekonwertowany funkcja stoi z biblioteki string na int i dodany do wektora typu int. Następnie gdy już mamy wszystkie dane wektor jest sortowany przy użyciu funkcji sort() z biblioteki algorithm i zwracamy pierwszy wyraz bo jest on zgodnie z zasadzą działania funkcji sort() najmniejszy.

**KOD:**

int \_get\_the\_lowest\_number\_from\_file(std::string \_directory){

std::ifstream myfile(\_directory + ".txt");

std::string line;

if (myfile.is\_open()){

std::vector<int> \_data\_from\_file;

while (getline(myfile, line)){

std::cout << line << std::endl;

for (int k = 0; k < line.length(); k++){

if (line[k] == ' '){

continue;

}

else{

\_data\_from\_file.push\_back(std::stoi(&line[k]));

}

}

}

std::sort(begin(\_data\_from\_file), end(\_data\_from\_file));

myfile.close();

return \_data\_from\_file.front();

}

else std::cout << "Unable to open file";

}

**UWAGI:**

**-** Podobnie jak w zadaniu poprzednim można by dodać jakieś zabezpieczenie, które sprawdzałoby czy liczba konwertowania na int nie jest jakimś innym znakiem np. literą

**Zadanie 8.2.19\***

**OPIS:**

W zadaniu korzystam z bibliotek: String, Fstream i Bitset.

Funkcja przyjmuje, 4 argumenty: scieżke do pliku, tablice 2D, oraz jej wymiary.

Algorytm zamienia mi zawartość tablicy 2D z zapisu 10-tnego na binarny i zapisuje go do pliku wraz z wymiarami tablic. Dodatkowo w celu przetestowania dodałem zabieg odwrotny z zapisu binarnego wracamy na 10-tny.

**KOD:**

void \_save\_date\_to\_file(std::string \_directory, int\*\* \_2d\_array\_data, int \_row\_count, int \_col\_count){

// OPEN file from directory

std::ofstream myfile(\_directory + ".txt");

// INITIALIZE 2D array

//\_2d\_array\_data = new int\*[\_row\_count];

//for (int i = 0; i < \_row\_count; i++){

// \_2d\_array\_data[i] = new int[\_col\_count];

//}

// SAVE data to file

if (myfile.is\_open())

{

int \_int\_bit;

myfile << \_row\_count << ", " << \_col\_count << std::endl;

for (int i = 0; i < \_row\_count; i++){

for (int j = 0; j < \_row\_count; j++){

std::bitset<8> x(\_2d\_array\_data[i][j]);

myfile << x << std::endl;

myfile << (int)(x.to\_ulong()) << std::endl;

//myfile << \_2d\_array\_data[i][j] << std::endl;

}

}

myfile.close();

}

else std::cout << "Unable to open file";

}

**UWAGI:**

- Można by dodać warunki i inteligentny wybór na ilu bitach wystarczy nam zapisać zawartość tej tablicy lub dodac argument do funkcji, który mówlby na ilu bitów nas interesuje.

- Można by dodać warunki, które pilnowaly by żeby podane wymiary tablicy nie były ujemne, lub w ogóle wykluczyć podawanie rozmaru i robić to za pomocą funkcji length().

**Zadanie 8.2.20\***

**OPIS:**

W zadaniu korzystam z bibliotek: String, Fstream, Bitset i Vector.

Funkcja wczytuje plik, który jej podajemy. Następnie wczytuje plik liniami. Pierwsza linia mówi nam o wymiarach tablicy 2D. Pobieramy ją i przypisujemy zmiennym, które później pozwolą nam zainicjalizować tablice 2D i jeszcze później przypisać jej odpowiednie wartości. Dane, które pobieramy w liniach są typu string dlatego przy użyciu biblioteki Bitset rzutujemy je na typ bitset i następnie na int. Wektor z biblioteki wektor używamy do pobrania najpierw wymiarów tablicy później do przechowywania danych, które następnie po zainicjalizowaniu tablicy 2D wkładamy do niej.

**KOD:**

int\*\* \_save\_date\_from\_file\_to\_2D\_array(std::string \_directory){

std::ifstream\* myfile = new std::ifstream(\_directory);

//myfile->open(\_directory);

int \_idx = 0;

int x, y;

int\*\* \_2d\_array\_data=nullptr;

std::vector <int> dimension;

std::string line;

if (myfile->is\_open()){

while (std::getline(\*myfile, line)){

//std::cout << line << std::endl;

if (\_idx == 0){

for (int k = 0; k < line.length(); k++){

if (line[k] == ','){

continue;

}

else{

dimension.push\_back(std::stoi(&line[k]));

}

if (k == line.length() - 1){

if (!(dimension.empty())){

x = dimension.front();

y = dimension.back();

dimension.clear();

//std::cout << x << ", " << y << std::endl;

//std::cout << dimension.empty() << std::endl;

// INITIALIZE 2D array

\_2d\_array\_data = new int\*[x];

for (int i = 0; i < x; i++){

\_2d\_array\_data[i] = new int[y];

}

}

else std::cout << "dimension array's are 0" << std::endl;

}

}

}

else{

std::bitset<8> x(line);

dimension.push\_back((int)(x.to\_ulong()));

//std::cout << (int)(x.to\_ulong()) << std::endl;

}

\_idx++;

}

for (int i = x - 1; i >= 0; i--){

for (int j = y - 1; j >= 0; j--){

\_2d\_array\_data[i][j] = dimension.back();

dimension.pop\_back();

//std::cout << \_2d\_array\_data[i][j] << std::endl;

}

}

return \_2d\_array\_data;

myfile->close();

}

else std::cout << "Unable to open file" << std::endl;

}

**UWAGI:**

**Zadanie 8.2.1**

**OPIS:**

W zadaniu korzystamy z bibliotek: String i Fstream.

Funkcja tworzy obiekt klasy ifstream, który wchodzi w dziedzine Fstream i jednocześnie otwiera plik.

Funkcja jest typu ifstream i zwraca adres, w którym znajduje się nasz otwarty plik.

**KOD:**

std::ifstream\* \_get\_descriptor\_from\_file\_to\_read(std::string \_directory){

static std::ifstream myfile(\_directory + ".txt");

return &myfile;

}

**UWAGI:**

**Zadanie 8.2.2**

**OPIS:**

W zadaniu korzystamy z bibliotek: String i Fstream.

Funkcja przyjmuje wskaźnik do obiektu typu ifstream. Następnie sprawdza czy plik jest otwarty i jeżeli tak to wczytuje każdą linie jaka w nim jest. W przeciwnym wypadku wyskakuje nam komunikat o niepowodzeniu.

**KOD:**

void \_get\_data\_from\_file\_to\_read(std::ifstream\* \_file){

std::string line;

if (\_file->is\_open())

{

while (getline(\*\_file, line))

{

std::cout << line << '\n';

}

\_file->close();

}

else std::cout << "Unable to open file";

}

**UWAGI:**