

Plecak

Generated by Doxygen 1.9.8

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	5
3.1 Backpack Struct Reference	5
3.1.1 Detailed Description	5
3.1.2 Member Data Documentation	5
3.1.2.1 capacity	5
3.1.2.2 items	5
3.1.2.3 value	6
3.2 Item Struct Reference	6
3.2.1 Detailed Description	6
3.2.2 Member Data Documentation	6
3.2.2.1 name	6
3.2.2.2 price	6
3.2.2.3 weight	6
3.3 Params Struct Reference	7
3.3.1 Detailed Description	7
3.3.2 Member Data Documentation	7
3.3.2.1 amountOfSpecimens	7
3.3.2.2 capacity	7
3.3.2.3 generations	7
3.3.2.4 itemsFile	7
3.3.2.5 solutionFile	8
4 File Documentation	9
4.1 main.cpp File Reference	9
4.1.1 Function Documentation	9
4.1.1.1 main()	9
4.2 plecak.cpp File Reference	9
4.2.1 Function Documentation	10
4.2.1.1 createGeneration()	10
4.2.1.2 drawItem()	10
4.2.1.3 findBest()	11
4.2.1.4 findFittingItems()	11
4.2.1.5 findSolution()	11
4.2.1.6 generateInitialBackpacks()	12
4.2.1.7 Load_parameters()	12
4.2.1.8 mixBackpacks()	12
4.2.1.9 move_items()	13

4.2.1.10 to_file()	13
4.3 plecak.h File Reference	13
4.3.1 Function Documentation	14
4.3.1.1 createGeneration()	14
4.3.1.2 drawItem()	15
4.3.1.3 findBest()	15
4.3.1.4 findFittingItems()	15
4.3.1.5 findSolution()	16
4.3.1.6 generateInitialBackpacks()	16
4.3.1.7 Load_parameters()	16
4.3.1.8 mixBackpacks()	17
4.3.1.9 move_items()	17
4.3.1.10 to_file()	17
4.4 plecak.h	18
Index	21

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Backpack	Struktura pelniaca role plecaka z przedmiotami	5
Item	Struktura przechowujaca dane z pliku powiazanego z przelacznikiem -i	6
Params	Struktura przyjmujaca wprowadzone dane - odpowiednie dla przelacznikow nazwy plikow i wartosci - do programu	7

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

main.cpp	9
plecak.cpp	9
plecak.h	13

Chapter 3

Class Documentation

3.1 Backpack Struct Reference

Struktura pelniaca role plecaka z przedmiotami.

```
#include <plecak.h>
```

Public Attributes

- `std::list< Item > items` = `std::list<Item>()`
- `float capacity` = 0
- `int value` = 0

3.1.1 Detailed Description

Struktura pelniaca role plecaka z przedmiotami.

Parameters

<i>items</i>	Lista typu Items przechowujaca przedmioty znajdujace sie w plecaku.
<i>capacity</i>	Zmienna float okreslajaca pojemnosc plecaka.
<i>value</i>	Zmienna int okreslajaca wartosc plecaka.

3.1.2 Member Data Documentation

3.1.2.1 capacity

```
float Backpack::capacity = 0
```

3.1.2.2 items

```
std::list<Item> Backpack::items = std::list<Item>()
```

3.1.2.3 value

```
int Backpack::value = 0
```

The documentation for this struct was generated from the following file:

- [plecak.h](#)

3.2 Item Struct Reference

struktura przechowująca dane z pliku powiazanego z przełącznikiem -i.

```
#include <plecak.h>
```

Public Attributes

- std::string [name](#) = ""
- float [weight](#) = 0
- float [price](#) = 0

3.2.1 Detailed Description

struktura przechowująca dane z pliku powiazanego z przełącznikiem -i.

Parameters

<i>name</i>	Zmienna typu string przyjmująca nazwe przedmiotu.
<i>weight</i>	Zmienna typu float przyjmująca wage przedmiotu.
<i>price</i>	Zmienna typu float przyjmująca cene przedmiotu.

3.2.2 Member Data Documentation

3.2.2.1 name

```
std::string Item::name = ""
```

3.2.2.2 price

```
float Item::price = 0
```

3.2.2.3 weight

```
float Item::weight = 0
```

The documentation for this struct was generated from the following file:

- [plecak.h](#)

3.3 Params Struct Reference

Struktura przyjmująca wprowadzone dane - odpowiednie dla przełączników nazwy plików i wartości - do programu.

```
#include <plecak.h>
```

Public Attributes

- std::string [itemsFile](#)
- std::string [solutionFile](#)
- float [capacity](#)
- int [generations](#)
- int [amountOfSpecimens](#)

3.3.1 Detailed Description

Struktura przyjmująca wprowadzone dane - odpowiednie dla przełączników nazwy plików i wartości - do programu.

Parameters

<i>itemsFile</i>	Zmienna typu string przyjmująca nazwę pliku z przedmiotami.
<i>solutionFile</i>	Zmienna typu string przyjmująca nazwę pliku z zapisem wyniku.
<i>float</i>	Zmienna typu float przyjmująca pojemność plecaka.
<i>generations</i>	Zmienna typu int przyjmująca ilość generacji.
<i>amountOfSpecimens</i>	Zmienna typu int przyjmująca ilość osobników w pokoleniu.

3.3.2 Member Data Documentation

3.3.2.1 amountOfSpecimens

```
int Params::amountOfSpecimens
```

3.3.2.2 capacity

```
float Params::capacity
```

3.3.2.3 generations

```
int Params::generations
```

3.3.2.4 itemsFile

```
std::string Params::itemsFile
```

3.3.2.5 solutionFile

```
std::string Params::solutionFile
```

The documentation for this struct was generated from the following file:

- [plecak.h](#)

Chapter 4

File Documentation

4.1 main.cpp File Reference

```
#include <iostream>
#include <fstream>
#include <list>
#include "plecak.h"
```

Functions

- int [main](#) (int argc, char *argv[])

4.1.1 Function Documentation

4.1.1.1 main()

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

4.2 plecak.cpp File Reference

```
#include <iostream>
#include <vector>
#include <string>
#include <fstream>
#include <iomanip>
#include "plecak.h"
#include <random>
#include <sstream>
```

Functions

- bool [Load_parameters](#) ([Params](#) ¶m, const int &Liczba_parametrow, char *parametry[])
Funkcja sprawdzająca kontrolę nad właściwym dostarczeniem przełączników do programu.
- void [move_items](#) (std::list< [Item](#) > &items, std::ifstream &input)
Funkcja dostarczająca dane z przełącznika - i do Listy typu Items w strukturze [Backpack](#).
- void [to_file](#) ([Backpack](#) &backpack, std::ofstream &output_o, [Params](#) ¶ms)
Funkcja zapisująca wynik programu do pliku o nazwie podanej w przełączniku -o.
- std::list< [Item](#) > [findFittingItems](#) (std::list< [Item](#) > &items, float backpackCapacity)
Funkcja odrzucająca przedmioty o wadze większej niż pojemność plecaka - przygotowanie początkowych plecaków.
- [Item](#) [drawItem](#) (std::list< [Item](#) > &items)
Funkcja losująca przedmioty do plecaków initial.
- std::vector< [Backpack](#) > [generateInitialBackpacks](#) (std::list< [Item](#) > &items, int amountOfBackpacks, float backpackMaxWeight)
Funkcja odpowiadająca za przygotowanie wstępnych plecaków - populacja początkowa.
- [Backpack](#) [mixBackpacks](#) ([Backpack](#) &first, [Backpack](#) &second, float backpackCapacity)
Funkcja odpowiadająca za krzyżowanie plecaków.
- [Backpack](#) [findBest](#) (std::vector< [Backpack](#) > &generationToSearch)
Funkcja odpowiadająca za wybieranie najlepszego plecaka z poprzedniego pokolenia, szukanie najlepszych opcji.
- void [createGeneration](#) (std::vector< [Backpack](#) > &generationToMix, float backpackCapacity)
Funkcja odpowiadająca za tworzenie nowych plecaków.
- [Backpack](#) [findSolution](#) (std::vector< [Backpack](#) > &generation, int generations, float backpackCapacity)
Funkcja odpowiadająca za tworzenie nowych, coraz lepszych pokoleń.

4.2.1 Function Documentation

4.2.1.1 createGeneration()

```
void createGeneration (
    std::vector< Backpack > & generationToMix,
    float backpackCapacity )
```

Funkcja odpowiadająca za tworzenie nowych plecaków.

Parameters

<i>generationToMix</i>	Vector zawierający osobniki - plecaki do krzyżowania.
<i>backpackCapacity</i>	Zmienna typu float określająca pojemność plecaka.

4.2.1.2 drawItem()

```
Item drawItem (
    std::list< Item > & items )
```

Funkcja losująca przedmioty do plecaków initial.

Parameters

<i>items</i>	Lista typu Items przechowująca przedmioty znajdujące się w plecaku.
--------------	---

Returns

zwraca losowo wybrany przedmiot z listy rzeczy pasujących do plecaka.

4.2.1.3 findBest()

```
Backpack findBest (
    std::vector< Backpack > & generationToSearch )
```

Funkcja odpowiadająca za wybieranie najlepszego plecaka z poprzedniego pokolenia, szukanie najlepszych opcji.

Parameters

Vector	zawierający przygotowane plecaki - osobniki.
--------	--

Returns

zwraca plecak o największej wartości - algorytm max.

4.2.1.4 findFittingItems()

```
std::list< Item > findFittingItems (
    std::list< Item > & items,
    float backpackCapacity )
```

Funkcja odrzucająca przedmioty o wadze większej niż pojemność plecaka - przygotowanie początkowych plecaków.

Parameters

<i>items</i>	Lista typu Items przechowująca przedmioty znajdujące się w plecaku.
<i>backpackCapacity</i>	Zmienna typu float określająca pojemność plecaka.

Returns

zwraca listę przedmiotów pasujących do plecaka

4.2.1.5 findSolution()

```
Backpack findSolution (
    std::vector< Backpack > & generation,
    int generations,
    float backpackCapacity )
```

Funkcja odpowiadająca za tworzenie nowych, coraz lepszych pokoleń.

Parameters

<i>generation</i>	Vector zawierający pokolenie plecaków do krzyżowania - pokolenie initial.
<i>generations</i>	Zmienna typu int określająca ilość pokoleń.
<i>backpackCapacity</i>	Zmienna typu float określająca pojemność plecaka.

Returns

zwraca najlepsze rozwiązanie dla uzyskanych generacji.

4.2.1.6 generateInitialBackpacks()

```
std::vector< Backpack > generateInitialBackpacks (
    std::list< Item > & items,
    int amountOfBackpacks,
    float backpackWeight )
```

Funkcja odpowiadająca za przygotowanie wstępnych plecakow - populacja początkowa.

Parameters

<i>items</i>	Lista typu Items przechowująca przedmioty znajdujące się w plecaku.
<i>amountOfBackpack</i>	Zmienna typu int określająca ilość osobników w pokoleniu.
<i>backpackWeight</i>	Zmienna typu float określająca wagę przedmiotów w plecaku.

Returns

zwraca vector plecakow przygotowanych pod krzyżowanie.

4.2.1.7 Load_parameters()

```
bool Load_parameters (
    Params & param,
    const int & Liczba_parametrow,
    char * parametry[] )
```

Funkcja sprawująca kontrolę nad właściwym dostarczeniem przełączników do programu.

Parameters

<i>param</i>	Zmienna struktury pozwalająca na posługiwanie się plikami i wartościami, które były dostarczone przez przełączniki.
<i>Liczba_parametrow</i>	Zmienna przyjmująca wartość od argc - ilość parametrow.
<i>parametry</i>	Zmienna przyjmująca wartość od argv - wartości parametrow.

Returns

false błąd przy czytaniu przełączników - niewłaściwy format / kolejność.

true brak błędów

4.2.1.8 mixBackpacks()

```
Backpack mixBackpacks (
    Backpack & first,
```



```

    Backpack & second,
    float backpackCapacity )

```

Funkcja odpowiadająca za krzyzowanie plecakov.

Parameters

<i>first</i>	Zmienna struktury Backpack stworzona w celu porownywania 2 plecakww.
<i>second</i>	Zmienna struktury Backpack stworzona w celu porownywania 2 plecakov.
<i>backpackCapacity</i>	Zmienna typu float okreslajaca ilosc wolnego miejsca w plecaku temporary.

Returns

zwraca nowy plecak, otrzymany w wyniku krzyzowania 2 sasiednich.

4.2.1.9 move_items()

```

void move_items (
    std::list< Item > & items,
    std::ifstream & input )

```

Funkcja dostarczająca dane z przełącznika -i do Listy typu Items w strukturze [Backpack](#).

Parameters

<i>items</i>	Lista typu Items przechowująca przedmioty znajdujące się w plecaku.
<i>input</i>	Strumień ifstream pobierający dane z pliku z przełącznika -i.

4.2.1.10 to_file()

```

void to_file (
    Backpack & backpack,
    std::ofstream & output_o,
    Params & params )

```

Funkcja zapisująca wynik programu do pliku o nazwie podanej w przełączniku -o.

Parameters

<i>backpack</i>	Zmienna typu Backpack umożliwiająca obsługę struktury Backpack .
<i>output_o</i>	Strumień typu ofstream zapisujący zmienne do pliku.
<i>params</i>	Zmienna typu Param - zapis danych plecaka do pliku.

4.3 plecak.h File Reference

```
#include <iostream>
```

```
#include <vector>
#include <string>
#include <list>
```

Classes

- struct [Params](#)
Struktura przyjmująca wprowadzone dane - odpowiednie dla przełączników nazwy plików i wartości - do programu.
- struct [Item](#)
struktura przechowująca dane z pliku powiązanego z przełącznikiem -i.
- struct [Backpack](#)
Struktura pełniąca rolę plecaka z przedmiotami.

Functions

- bool [Load_parameters](#) ([Params](#) ¶m, const int &Liczba_parametrow, char *parametry[])
Funkcja sprawująca kontrolę nad właściwym dostarczeniem przełączników do programu.
- void [move_items](#) (std::list< [Item](#) > &items, std::ifstream &input)
Funkcja dostarczająca dane z przełącznika - i do Listy typu Items w strukturze [Backpack](#).
- void [to_file](#) ([Backpack](#) &backpack, std::ofstream &output_o, [Params](#) ¶ms)
Funkcja zapisująca wynik programu do pliku o nazwie podanej w przełączniku -o.
- std::list< [Item](#) > [findFittingItems](#) (std::list< [Item](#) > &items, float backpackCapacity)
Funkcja odrzucająca przedmioty o wadze większej niż pojemność plecaka - przygotowanie początkowych plecaków.
- [Item](#) [drawItem](#) (std::list< [Item](#) > &items)
Funkcja losująca przedmioty do plecaków initial.
- std::vector< [Backpack](#) > [generateInitialBackpacks](#) (std::list< [Item](#) > &items, int amountOfBackpacks, float backpackWeight)
Funkcja odpowiadająca za przygotowanie wstępnych plecaków - populacja początkowa.
- [Backpack](#) [mixBackpacks](#) ([Backpack](#) &first, [Backpack](#) &second, float backpackCapacity)
Funkcja odpowiadająca za krzyżowanie plecaków.
- [Backpack](#) [findBest](#) (std::vector< [Backpack](#) > &generationToSearch)
Funkcja odpowiadająca za wybieranie najlepszego plecaka z poprzedniego pokolenia, szukanie najlepszych opcji.
- void [createGeneration](#) (std::vector< [Backpack](#) > &generationToMix, float backpackCapacity)
Funkcja odpowiadająca za tworzenie nowych plecaków.
- [Backpack](#) [findSolution](#) (std::vector< [Backpack](#) > &generation, int generations, float backpackCapacity)
Funkcja odpowiadająca za tworzenie nowych, coraz lepszych pokoleń.

4.3.1 Function Documentation

4.3.1.1 createGeneration()

```
void createGeneration (
    std::vector< Backpack > & generationToMix,
    float backpackCapacity )
```

Funkcja odpowiadająca za tworzenie nowych plecaków.

Parameters

<i>generationToMix</i>	Vector zawierający osobniki - plecaki do krzyżowania.
<i>backpackCapacity</i>	Zmienna typu float określająca pojemność plecaka.

4.3.1.2 drawItem()

```
Item drawItem (
    std::list< Item > & items )
```

Funkcja losująca przedmioty do plecaków initial.

Parameters

<i>items</i>	Lista typu Items przechowująca przedmioty znajdujące się w plecaku.
--------------	---

Returns

zwraca losowo wybrany przedmiot z listy rzeczy pasujących do plecaka.

4.3.1.3 findBest()

```
Backpack findBest (
    std::vector< Backpack > & generationToSearch )
```

Funkcja odpowiadająca za wybieranie najlepszego plecaka z poprzedniego pokolenia, szukanie najlepszych opcji.

Parameters

<i>Vector</i>	zawierający przygotowane plecaki - osobniki.
---------------	--

Returns

zwraca plecak o największej wartości - algorytm max.

4.3.1.4 findFittingItems()

```
std::list< Item > findFittingItems (
    std::list< Item > & items,
    float backpackCapacity )
```

Funkcja odrzucająca przedmioty o wadze większej niż pojemność plecaka - przygotowanie początkowych plecaków.

Parameters

<i>items</i>	Lista typu Items przechowująca przedmioty znajdujące się w plecaku.
<i>backpackCapacity</i>	Zmienna typu float określająca pojemność plecaka.

Returns

zwraca liste przedmiotow pasujacych do plecaka

4.3.1.5 findSolution()

```
Backpack findSolution (
    std::vector< Backpack > & generation,
    int generations,
    float backpackCapacity )
```

Funkcja odpowiadajaca za tworzenie nowych, coraz lepszych pokolen.

Parameters

<i>generation</i>	Vector zawierajacy pokolenie plecakow do krzyzowania - pokolenie initial.
<i>generations</i>	Zmienna typu int okreslajaca ilosc pokolen.
<i>backpackCapacity</i>	Zmienna typu float okreslajaca pojemnosc plecaka.

Returns

zwraca najlepsze rozwiazanie dla uzyskanych generacji.

4.3.1.6 generateInitialBackpacks()

```
std::vector< Backpack > generateInitialBackpacks (
    std::list< Item > & items,
    int amountOfBackpacks,
    float backpackWeight )
```

Funkcja odpowiadajaca za przygotowanie wstepnych plecakow - populacja poczatkowa.

Parameters

<i>items</i>	Lista typu Items przechowujaca przedmioty znajdujace sie w plecaku.
<i>amountOfBackpack</i>	Zmienna typu int okreslajaca ilosc osobnikow w pokoleniu.
<i>backpackWeight</i>	Zmienna typu float okreslajaca wage przedmiotow w plecaku.

Returns

zwraca vector plecakow przygotowanych pod krzyzowanie.

4.3.1.7 Load_parameters()

```
bool Load_parameters (
    Params & param,
    const int & Liczba_parametrow,
    char * parametry[] )
```

Funkcja sprawujaca kontrole nad wlasciwym dostarczeniem przelacznikow do programu.

Parameters

<i>param</i>	Zmienna struktury pozwalająca na posługiwanie się plikami i wartościami, które były dostarczone przez przełączniki.
<i>Liczba_parametrow</i>	Zmienna przyjmująca wartość od argc - ilość parametrów.
<i>parametry</i>	Zmienna przyjmująca wartość od argv - wartości parametrów.

Returns

false błąd przy czytaniu przełączników - niewłaściwy format / kolejność.

true brak błędów

4.3.1.8 mixBackpacks()

```
Backpack mixBackpacks (
    Backpack & first,
    Backpack & second,
    float backpackCapacity )
```

Funkcja odpowiadająca za krzyżowanie plecaków.

Parameters

<i>first</i>	Zmienna struktury Backpack stworzona w celu porównywania 2 plecaków.
<i>second</i>	Zmienna struktury Backpack stworzona w celu porównywania 2 plecaków.
<i>backpackCapacity</i>	Zmienna typu float określająca ilość wolnego miejsca w plecaku tymczasowym.

Returns

zwraca nowy plecak, otrzymany w wyniku krzyżowania 2 sąsiadnych.

4.3.1.9 move_items()

```
void move_items (
    std::list< Item > & items,
    std::ifstream & input )
```

Funkcja dostarczająca dane z przełącznika -i do Listy typu Items w strukturze [Backpack](#).

Parameters

<i>items</i>	Lista typu Items przechowująca przedmioty znajdujące się w plecaku.
<i>input</i>	Strumień ifstream pobierający dane z pliku z przełącznika -i.

4.3.1.10 to_file()

```
void to_file (
```

```

    Backpack & backpack,
    std::ofstream & output_o,
    Params & params )

```

Funkcja zapisująca wynik programu do pliku o nazwie podanej w przełączniku -o.

Parameters

<i>backpack</i>	Zmienna typu Backpack umożliwiająca obsługę struktury Backpack .
<i>output_o</i>	Strumień typu ofstream zapisujący zmienne do pliku.
<i>params</i>	Zmienna typu Param - zapis danych plecaka do pliku.

4.4 plecak.h

[Go to the documentation of this file.](#)

```

00001 #ifndef PLECAK_H
00002 #define PLECAK_H
00003
00004 #include <iostream>
00005 #include <vector>
00006 #include <string>
00007 #include <list>
00008
00017 struct Params
00018 {
00019     std::string itemsFile;
00020     std::string solutionFile;
00021     float capacity;
00022     int generations;
00023     int amountOfSpecimens;
00024 };
00025
00032 struct Item
00033 {
00034     std::string name = "";
00035     float weight = 0;
00036     float price = 0;
00037 };
00038
00039
00046 struct Backpack
00047 {
00048     std::list<Item> items = std::list<Item>();
00049     float capacity = 0;
00050     int value = 0;
00051 };
00052
00053
00062 bool Load_parameters(Params& param, const int& Liczba_parametrow, char* parametry[]);
00063
00064
00070 void move_items(std::list<Item>& items, std::ifstream& input);
00071
00072
00079 void to_file(Backpack& backpack, std::ofstream& output_o, Params& params);
00080
00081
00088 std::list<Item> findFittingItems(std::list<Item>& items, float backpackCapacity);
00089
00090
00096 Item drawItem(std::list<Item>& items);
00097
00098
00106 std::vector<Backpack> generateInitialBackpacks(std::list<Item>& items, int amountOfBackpacks, float
    backpackWeight);
00107
00108
00116 Backpack mixBackpacks(Backpack& first, Backpack& second, float backpackCapacity);
00117
00118
00124 Backpack findBest(std::vector<Backpack>& generationToSearch);
00125

```

```
00126
00132 void createGeneration(std::vector<Backpack>& generationToMix, float backpackCapacity);
00133
00134
00142 Backpack findSolution(std::vector<Backpack>& generation, int generations, float backpackCapacity);
00143
00144
00145 #endif // !PLECAK_H
```


Index

amountOfSpecimens
Params, 7

Backpack, 5
capacity, 5
items, 5
value, 5

capacity
Backpack, 5
Params, 7

createGeneration
plecak.cpp, 10
plecak.h, 14

drawItem
plecak.cpp, 10
plecak.h, 15

findBest
plecak.cpp, 11
plecak.h, 15

findFittingItems
plecak.cpp, 11
plecak.h, 15

findSolution
plecak.cpp, 11
plecak.h, 16

generateInitialBackpacks
plecak.cpp, 12
plecak.h, 16

generations
Params, 7

Item, 6
name, 6
price, 6
weight, 6

items
Backpack, 5

itemsFile
Params, 7

Load_parameters
plecak.cpp, 12
plecak.h, 16

main
main.cpp, 9
main.cpp, 9

main, 9
mixBackpacks
plecak.cpp, 12
plecak.h, 17
move_items
plecak.cpp, 13
plecak.h, 17

name
Item, 6

Params, 7
amountOfSpecimens, 7
capacity, 7
generations, 7
itemsFile, 7
solutionFile, 7
plecak.cpp, 9
createGeneration, 10
drawItem, 10
findBest, 11
findFittingItems, 11
findSolution, 11
generateInitialBackpacks, 12
Load_parameters, 12
mixBackpacks, 12
move_items, 13
to_file, 13

plecak.h, 13
createGeneration, 14
drawItem, 15
findBest, 15
findFittingItems, 15
findSolution, 16
generateInitialBackpacks, 16
Load_parameters, 16
mixBackpacks, 17
move_items, 17
to_file, 17

price
Item, 6

solutionFile
Params, 7

to_file
plecak.cpp, 13
plecak.h, 17

value
Backpack, 5

weight

Item, [6](#)