

Politechnika Śląska w Gliwicach
Wydział Automatyki, Elektroniki i Informatyki



Politechnika
Śląska

Podstawy Programowania Komputerów

Loty

autor	Natalia Cheba
prowadzący	dr inż. Adam Gudyś
rok akademicki	2018/2019
kierunek	informatyka
rodzaj studiów	SSI
semestr	1
termin laboratorium / ćwiczeń	poniedziałek, 10:15 – 11:45
grupa	1
sekcja	1
termin oddania sprawozdania	2019-01-19
data oddania sprawozdania	2019-01-12

1 Treść zadania

Korzystając z dynamicznych struktur danych napisać program, który przygotowuje listy pasażerów. Pasażerowie mogą rezerwować bilety na różne loty w różnych biurach i przez internet. Wszystkie rezerwacje są zapisywane w biurze centralnym. Mają one następującą postać:

(symbol lotu) (lotnisko startowe) (data lotu) (nazwisko pasażera) (nr miejsca)

Przykładowy plik z rezerwacjami:

```
KR54R Katowice 2011-12-13 Jaworek 33
TY340 London 2012-02-03 Hastings 2
TY340 London 2012-02-03 Poirot 23
KR54R Katowice 2011-12-13 Matianek 02
TY340 London 2012-02-03 Holmes 11
KR54R Katowice 2011-12-13 Lopez 12
TY340 London 2012-02-03 Lemon 43
KR54R Katowice 2011-12-13 Chavez 43
```

Na podstawie pliku z rezerwacjami należy stworzyć plik z listą pasażerów dla każdego lotu. Każda lista jest tworzona w odrębnym pliku. Nazwą pliku jest symbol lotu. W pliku umieszczona jest nazwa lotniska i data. W kolejnych liniach umieszczone są numery siedzeń i nazwiska pasażerów, posortowane wg numerów. Dla powyższego pliku zostaną utworzone pliki: `KR54R.txt` i `TY340.txt`.

Plik `KR54R.txt`:

```
symbol lotu: KR54R
lotnisko:    Katowice
data lotu:   2011-12-13
```

lista pasazerow:

```
02 Matianek
11 Lopez
33 Jaworek
43 Chavez
```

liczba rezerwacji: 4.

Program uruchamiany jest z linii poleceń z wykorzystaniem następującego przełącznika: `-i` plik wejściowy z rezerwacjami.

2 Analiza zadania

Program przedstawia problem sortowania lotów z pliku wejściowego oraz przydzielanie do odpowiednich lotów pasażerów w sposób posortowany według miejsc.

2.1 Struktury danych

W programie wykorzystano drzewo binarne i listę jednokierunkową. Lista przechowuje parametry lotu tj. identyfikator lotu, lotnisko startowe oraz datę lotu, natomiast drzewo posiada wszystkie informacje dotyczące pasażerów tj. nazwisko pasażera i numer miejsca. Wybranie takiej zagnieżdżonej struktury danych (lista drzew) powoduje szybkie sortowanie danych.

2.2 Algorytmy

Program pobiera z pliku wejściowego wszystkie parametry lotu przeszukując czy lot o podanych parametrach istnieje. Jeżeli istnieje, program dopisuje pasażera do listy pasażerów dla konkretnego lotu. Jeżeli lot, który pobrał program nie znajduje się w liście lotów, program dodaje nową listę.

3 Specyfikacja zewnętrzna

Program jest uruchamiany z linii poleceń. Przy wywoływaniu programu możliwe jest użycie przełączników `-i`. Po wykorzystaniu przełącznika `-i` należy przekazać do programu nazwę pliku wejściowego. Instrukcja

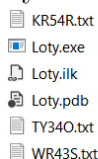
podaj dobre argumenty

jest wyświetlana w przypadku podania niepoprawnych danych.

Przykładowe wywołania programu:

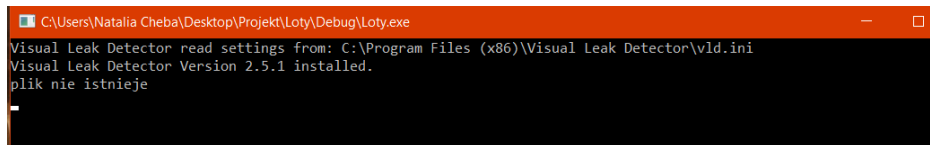
```
start Loty.exe -i ".../Loty.txt"
```

Program tworzy i zapisuje spis pasażerów w pliku tekstowym. Plik tekstowy dla każdego lotu jest nazwany symbolem lotu.



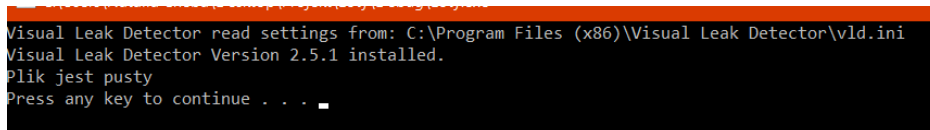
- KR54R.txt
- Loty.exe
- Loty.ilc
- Loty.pdb
- TY34O.txt
- WR43S.txt

W przypadku podania pliku, który nie istnieje wyświetlana jest instrukcja
plik nie istnieje



```
C:\Users\Natalia Cheba\Desktop\Projekt\Loty\Debug\Loty.exe
Visual Leak Detector read settings from: C:\Program Files (x86)\Visual Leak Detector\vld.ini
Visual Leak Detector Version 2.5.1 installed.
plik nie istnieje
```

W przypadku podania pliku, który jest pusty wyświetlana jest instrukcja
Plik jest pusty



```
Visual Leak Detector read settings from: C:\Program Files (x86)\Visual Leak Detector\vld.ini
Visual Leak Detector Version 2.5.1 installed.
Plik jest pusty
Press any key to continue . . .
```

4 Testowanie

Program został przetestowany na różnego rodzaju plikach: pustych i nie istniejących. Obydwa przypadki powodują zgłoszenie błędu : Plik jest pusty lub Plik nie istnieje. Dodatkowo sprawdzony został pod kątem wprowadzenia złego przełącznika. W przypadku podania błędnego pojawia się komunikat "podaj dobre argumenty". W przypadku podania błędnych danych : złe lotnisko, złe nazwisko, zły numer miejsca, zła data, program działa i nie uwzględnia podania błędnych informacji. Program został przetestowany też dla dużej ilości danych ponad 1,5Mb i kilku bajtów. Program działał trochę wolniej dla dużej ilości danych. Program został sprawdzony pod kątem wycieków pamięci tak, aby nie dopuścić do ani jednego wycieku.

5 Wnioski

Program Loty jest programem ciekawym, ponieważ wymaga znajomości drzew i list. Szczególnie trudne w tym projekcie okazało się porównywanie pól związanych z parametrami lotu. Najbardziej wymagające okazało się podzielenie plików na .h i .cpp. Dodatkowo dużym problemem stanowiły przełączniki. Pomimo wielu trudności projekt okazał się doskonałą okazją do poznania wielu ciekawych nowych funkcji i konstrukcji.

My Project

Generated by Doxygen 1.8.14

Contents

1	Class Index	1
1.1	Class List	1
2	File Index	3
2.1	File List	3
3	Class Documentation	5
3.1	Drzewo Struct Reference	5
3.1.1	Detailed Description	5
3.1.2	Member Data Documentation	6
3.1.2.1	lewylisc	6
3.1.2.2	pasazerPrzechowywanyWDrzewie	6
3.1.2.3	prawylisc	6
3.2	Lista Struct Reference	6
3.2.1	Detailed Description	7
3.2.2	Friends And Related Function Documentation	8
3.2.2.1	operator<<	8
3.2.3	Member Data Documentation	8
3.2.3.1	korzen	8
3.2.3.2	nastepnyElement	8
3.2.3.3	parametryLotu	8
3.3	ParametryLotu Struct Reference	9
3.3.1	Detailed Description	9
3.3.2	Friends And Related Function Documentation	9

3.3.2.1	operator<< [1/2]	10
3.3.2.2	operator<< [2/2]	10
3.3.2.3	operator==	10
3.3.3	Member Data Documentation	10
3.3.3.1	data_lotu	11
3.3.3.2	liczbaPasazerow	11
3.3.3.3	lotnisko	11
3.3.3.4	symbolLotu	11
3.4	Pasazer Struct Reference	11
3.4.1	Detailed Description	11
3.4.2	Friends And Related Function Documentation	12
3.4.2.1	operator<<	12
3.4.3	Member Data Documentation	12
3.4.3.1	nazwisko_pasazera	12
3.4.3.2	nr_miejsca	12
4	File Documentation	13
4.1	Drzewo.cpp File Reference	13
4.1.1	Function Documentation	13
4.1.1.1	dodawaniedodrzewa()	14
4.1.1.2	inorder()	14
4.1.1.3	usunDrzewo()	15
4.2	Drzewo.h File Reference	16
4.2.1	Function Documentation	17
4.2.1.1	dodawaniedodrzewa()	17
4.2.1.2	inorder()	18
4.2.1.3	usunDrzewo()	19
4.3	Lista.cpp File Reference	19
4.3.1	Function Documentation	20
4.3.1.1	dodawaniedolisty()	20
4.3.1.2	stworzPlikilZapisz()	21

4.3.1.3	usunListe()	22
4.3.1.4	wczytajzpliku()	23
4.4	Lista.h File Reference	24
4.4.1	Function Documentation	25
4.4.1.1	dodawaniedolisty()	25
4.4.1.2	stworzPlikilZapisz()	26
4.4.1.3	usunListe()	26
4.4.1.4	wczytajzpliku()	27
4.5	main.cpp File Reference	28
4.5.1	Function Documentation	29
4.5.1.1	main()	29
4.5.1.2	odczytaj_argumenty()	30
4.6	ParametryLotu.cpp File Reference	30
4.6.1	Function Documentation	31
4.6.1.1	operator<<() [1/2]	31
4.6.1.2	operator<<() [2/2]	32
4.6.1.3	operator==(())	32
4.7	ParametryLotu.h File Reference	32
4.8	Pasazer.cpp File Reference	33
4.8.1	Function Documentation	34
4.8.1.1	convertStringToTime()	34
4.8.1.2	operator<<()	34
4.9	Pasazer.h File Reference	35
4.9.1	Function Documentation	36
4.9.1.1	convertStringToTime()	36

Chapter 1

Class Index

1.1 Class List

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

Drzewo	5
Lista	6
ParametryLotu	9
Pasazer	11

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Drzewo.cpp	13
Drzewo.h	16
Lista.cpp	19
Lista.h	24
main.cpp	28
ParametryLotu.cpp	30
ParametryLotu.h	32
Pasazer.cpp	33
Pasazer.h	35

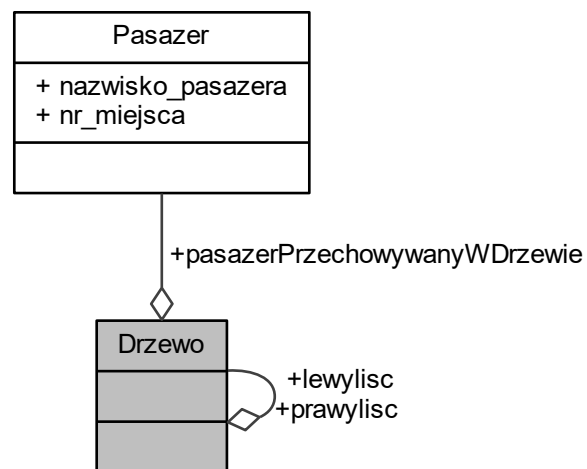
Chapter 3

Class Documentation

3.1 Drzewo Struct Reference

```
#include <Drzewo.h>
```

Collaboration diagram for Drzewo:



Public Attributes

- [Pasazer pasazerPrzechowywanyWDrzewie](#)
- [Drzewo * lewylic](#)
- [Drzewo * prawylisc](#)

3.1.1 Detailed Description

Parameters

<i>pasazerPrzechowywanyWDrzewie</i>	zmienna stuktury pasazer zawieraja informacje o pasazerze
<i>lewylic</i>	wskaznik na lewa strone drzewa
<i>prawylic</i>	wskaznik na prawa strone drzewa

3.1.2 Member Data Documentation**3.1.2.1 lewylic**

`Drzewo* Drzewo::lewylic`

3.1.2.2 pasazerPrzechowywanyWDrzewie

`Pasazer Drzewo::pasazerPrzechowywanyWDrzewie`

3.1.2.3 prawylic

`Drzewo* Drzewo::prawylic`

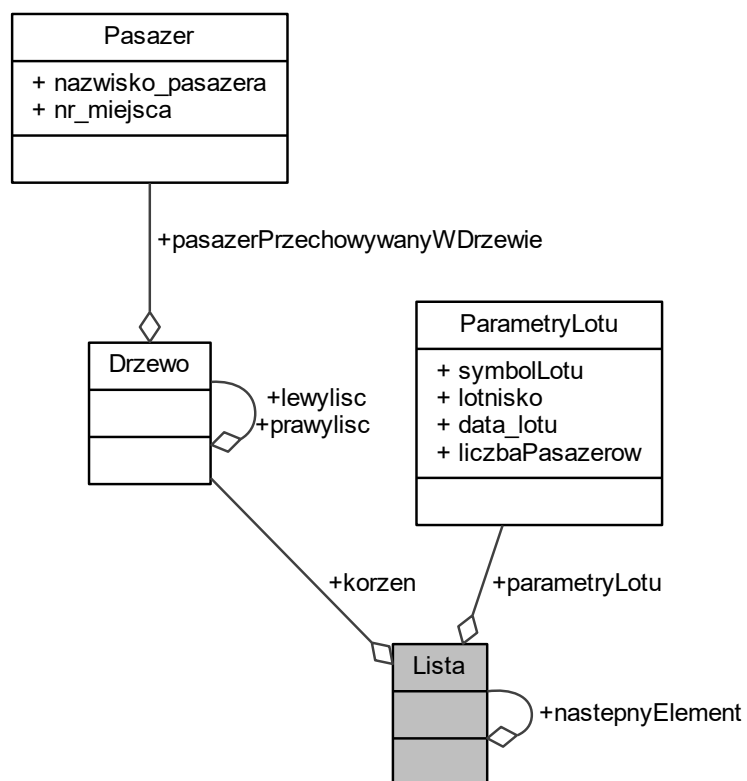
The documentation for this struct was generated from the following file:

- [Drzewo.h](#)

3.2 Lista Struct Reference

```
#include <Lista.h>
```


Collaboration diagram for Lista:



Public Attributes

- [Lista](#) * [nastepnyElement](#)
- [Drzewo](#) * [korzen](#)
- [ParametryLotu](#) [parametryLotu](#)

Friends

- `std::ostream & operator<< (std::ostream &os, const std::tm &t)`

3.2.1 Detailed Description

Parameters

<i>nastepnyElement</i>	wskaznik do nastpnego elementu
<i>korzen</i>	korzen drzewa
<i>parametryLotu</i>	parametryLotu

3.2.2 Friends And Related Function Documentation

3.2.2.1 operator<<

```
std::ostream& operator<< (  
    std::ostream & os,  
    const std::tm & t ) [friend]
```

Funkcja do wywietlania danych ze struktury tm - opisujcej dat

/*

Parameters

<i>os</i>	os to jest strumie ostreamowy
<i>t</i>	zmienna czasu

Returns

os zwracamy strumie os

3.2.3 Member Data Documentation

3.2.3.1 korzen

[Drzewo*](#) Lista::korzen

3.2.3.2 nastepnyElement

[Lista*](#) Lista::nastepnyElement

3.2.3.3 parametryLotu

[ParametryLotu](#) Lista::parametryLotu

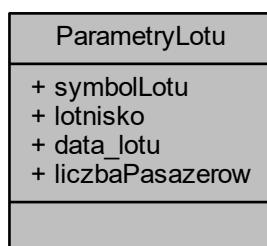
The documentation for this struct was generated from the following file:

- [Lista.h](#)

3.3 ParametryLotu Struct Reference

```
#include <ParametryLotu.h>
```

Collaboration diagram for ParametryLotu:



Public Attributes

- std::string [symbolLotu](#)
- std::string [lotnisko](#)
- struct std::tm [data_lotu](#)
- int [liczbaPasazerow](#)

Friends

- std::ostream & [operator<<](#) (std::ostream &os, const std::tm &t)
- bool [operator==](#) (const [ParametryLotu](#) &p1, const [ParametryLotu](#) &p2)
- std::ostream & [operator<<](#) (std::ostream &os, const [ParametryLotu](#) ¶metryLotu)

3.3.1 Detailed Description

Parameters

<i>symbolLotu</i>	symbol lotu
<i>lotnisko</i>	lotnisko
<i>data_lotu</i>	data lotu
<i>liczbaPasazerow</i>	liczba Pasazerow

3.3.2 Friends And Related Function Documentation

3.3.2.1 operator<< [1/2]

```
std::ostream& operator<< (  
    std::ostream & os,  
    const std::tm & t ) [friend]
```

Funkcja do wyświetlania danych ze struktury tm - opisującej dat

/*

Parameters

<i>os</i>	os to jest strumie ostreamowy
<i>t</i>	zmienna czasu

Returns

os zwracamy strumie os

3.3.2.2 operator<< [2/2]

```
std::ostream& operator<< (  
    std::ostream & os,  
    const ParametryLotu & parametryLotu ) [friend]
```

3.3.2.3 operator==

```
bool operator== (  
    const ParametryLotu & p1,  
    const ParametryLotu & p2 ) [friend]
```

Funkcja zwraca czy lotnisko i symbol lotu jest takie same

Parameters

<i>p1</i>	parametry pierwszego lotu
<i>p2</i>	parametry drugiego lotu

3.3.3 Member Data Documentation

3.3.3.1 data_lotu

```
struct std::tm ParametryLotu::data_lotu
```

3.3.3.2 liczbaPasazerow

```
int ParametryLotu::liczbaPasazerow
```

3.3.3.3 lotnisko

```
std::string ParametryLotu::lotnisko
```

3.3.3.4 symbolLotu

```
std::string ParametryLotu::symbolLotu
```

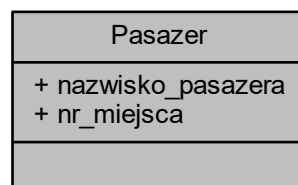
The documentation for this struct was generated from the following file:

- [ParametryLotu.h](#)

3.4 Pasazer Struct Reference

```
#include <Pasazer.h>
```

Collaboration diagram for Pasazer:



Public Attributes

- std::string [nazwisko_pasazera](#)
- int [nr_miejsca](#)

Friends

- std::ostream & [operator<<](#) (std::ostream &os, const [Pasazer](#) &p)

3.4.1 Detailed Description

Parameters

<i>nazwisko_pasazera</i>	nazwisko pasazera
<i>nr_miejsca</i>	numer miejsca, gdzie siedzi dany pasazer

3.4.2 Friends And Related Function Documentation**3.4.2.1 operator<<**

```
std::ostream& operator<< (
    std::ostream & os,
    const Pasazer & p ) [friend]
```

Funkcja do wywietlania struktury pasazer

Parameters

<i>os</i>	strumie
<i>p</i>	zmienna o strukturze Pasazer

Returns

os zwracamy strumie

3.4.3 Member Data Documentation**3.4.3.1 nazwisko_pasazera**

```
std::string Pasazer::nazwisko_pasazera
```

3.4.3.2 nr_miejsca

```
int Pasazer::nr_miejsca
```

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

- [Pasazer.h](#)

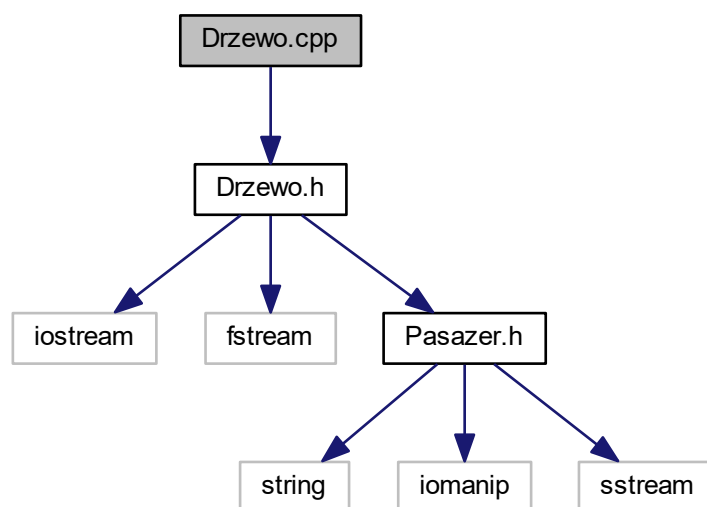
Chapter 4

File Documentation

4.1 Drzewo.cpp File Reference

```
#include "Drzewo.h"
```

Include dependency graph for Drzewo.cpp:



Functions

- void [inorder](#) ([Drzewo](#) *node, std::ofstream &outfile)
- void [usunDrzewo](#) ([Drzewo](#) *&korzen)
- [Drzewo](#) * [dodawanieDodrzewa](#) ([Drzewo](#) *korzen, [Pasazer](#) &ktos)

4.1.1 Function Documentation

4.1.1.1 dodawanie do drzewa()

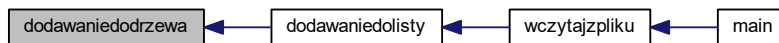
```
Drzewo* dodawanie do drzewa (
    Drzewo * korzen,
    Pasazer & ktos )
```

Funcja dodaje elementy do drzewa

Parameters

<i>korzen</i>	jeli korzenia nie ma to drzewo jest puste i nie ma pasaera, tworzymy element lewy i prawy li, nastpny elementy ustawiamy na nullptr
<i>przechowywator korzenia</i>	wskanik do przechowywania korzenia

Here is the caller graph for this function:



4.1.1.2 inorder()

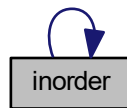
```
void inorder (
    Drzewo * node,
    std::ofstream & outfile )
```

Funcja wypisujca do pliku rekurencyjnie drzewa wartoci

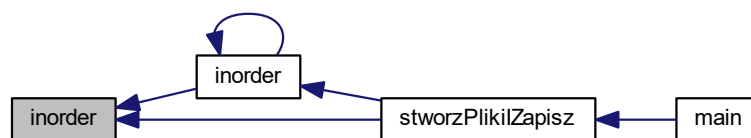
Parameters

<i>node</i>	li drzewa
<i>lewyliśc</i>	lewa strona drzewa
<i>prawyliśc</i>	prawa strona drzewa

Here is the call graph for this function:



Here is the caller graph for this function:



4.1.1.3 usunDrzewo()

```
void usunDrzewo (  
    Drzewo *& korzen )
```

Funcja usuwajca drzewo

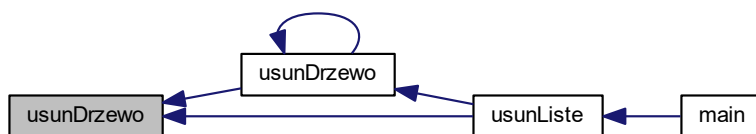
Parameters

<i>korzen</i>	korze drzewa
---------------	--------------

Here is the call graph for this function:

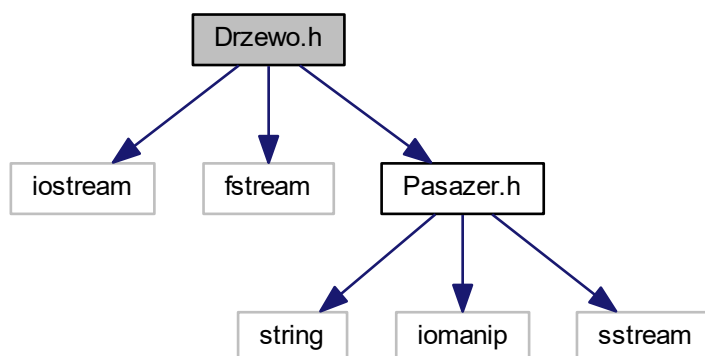


Here is the caller graph for this function:

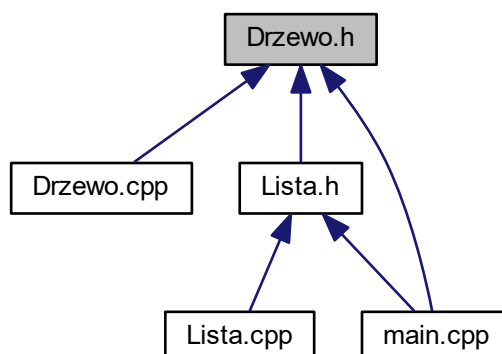


4.2 Drzewo.h File Reference

```
#include <iostream>
#include <fstream>
#include "Pasazer.h"
Include dependency graph for Drzewo.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- struct [Drzewo](#)

Functions

- [Drzewo](#) * [dodawaniedodrzewa](#) ([Drzewo](#) *korzen, [Pasazer](#) &ktos)
- void [usunDrzewo](#) ([Drzewo](#) *&korzen)
- void [inorder](#) ([Drzewo](#) *node, std::ofstream &outfile)

4.2.1 Function Documentation

4.2.1.1 dodawaniedodrzewa()

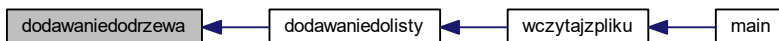
```
Drzewo* dodawaniedodrzewa (  
    Drzewo * korzen,  
    Pasazer & ktos )
```

Funcja dodaje elementy do drzewa

Parameters

<i>korzen</i>	jeli korzenia nie ma to drzewo jest puste i nie ma pasaera, tworzymy element lewy i prawy li, nastpny elementy ustawiamy na nullptr
<i>przechowywatorkorzenia</i>	wskanik do przechowywania korzenia

Here is the caller graph for this function:



4.2.1.2 inorder()

```

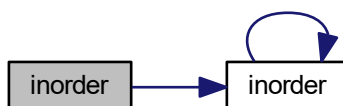
void inorder (
    Drzewo * node,
    std::ofstream & outfile )
  
```

Funkcja wypisująca do pliku rekurencyjnie drzewa wartości

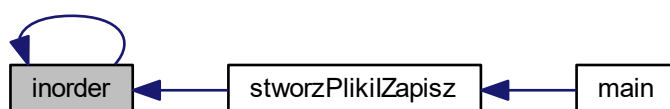
Parameters

<i>node</i>	li drzewa
<i>lewyLisc</i>	lewa strona drzewa
<i>prawyLisc</i>	prawa strona drzewa

Here is the call graph for this function:



Here is the caller graph for this function:



4.2.1.3 usunDrzewo()

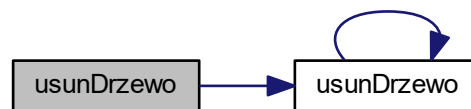
```
void usunDrzewo (
    Drzewo *& korzen )
```

Funcja usuwajca drzewo

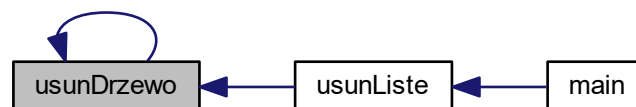
Parameters

<i>korzen</i>	korze drzewa
---------------	--------------

Here is the call graph for this function:



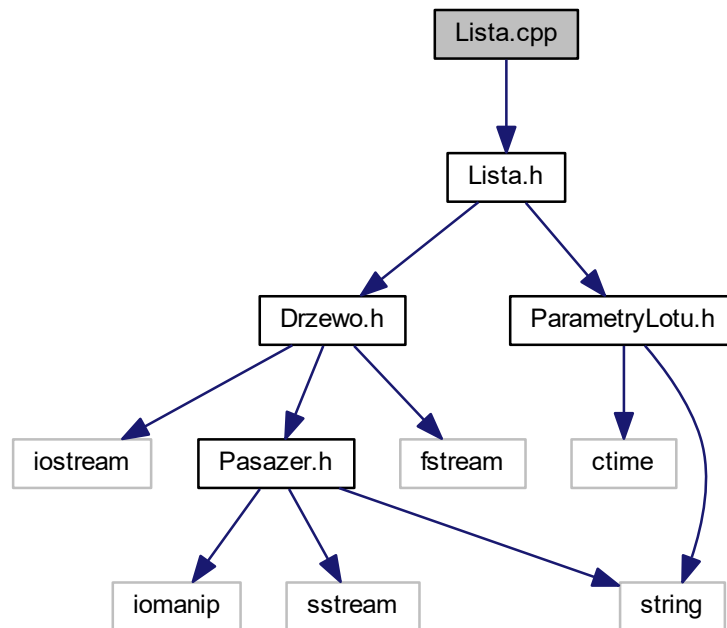
Here is the caller graph for this function:



4.3 Lista.cpp File Reference

```
#include "Lista.h"
```

Include dependency graph for Lista.cpp:



Functions

- void `stworzPlikilZapisz` (`Lista *phead`)
- void `usunListe` (`Lista *glowka`)
- `Lista * wczytajzpliku` (`const std::string &nazwaPliku`)
- `Lista * dodawaniedolisty` (`Lista *pHead`, `ParametryLotu ¶metryLotu`, `Pasazer &pasazer`)

4.3.1 Function Documentation

4.3.1.1 dodawaniedolisty()

```

Lista* dodawaniedolisty (
    Lista * pHead,
    ParametryLotu & parametryLotu,
    Pasazer & pasazer )

```

Funkcja dodająca do listy

Parameters

<code>pNowy</code>	wskanik na element, ktrego jeszcze nie bylo
<code>przechowywatorglowy</code>	przechowuje pHead
<code>temp</code>	kopia przechowywatoraglowy po to, aby lista moga wej do ostatniej iteracji
<code>nowyElement</code>	jeli nie ma symbolu lotu tworzymy nowe drzewo

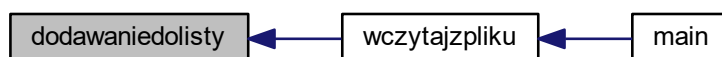
Returns

pHead zwraca pHead

Here is the call graph for this function:



Here is the caller graph for this function:



4.3.1.2 stworzPlikIzZapisz()

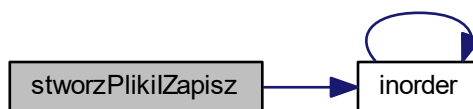
```
void stworzPlikIzZapisz (  
    Lista * phead )
```

Funkcja tworzy pliki. Jako parametr przyjmuje gow listy. Sprawdzamy czy przekazana lista istnieje.

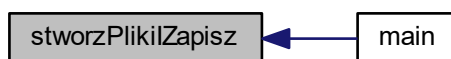
Parameters

<i>pHead</i>	gowa listy
<i>przechowywatorgowy</i>	dajemy gow do przechowywatora, eby nie niszczy listy i mc swobodnie j iterowa
<i>outfile</i>	stworzono strumie outfile do wywietlania

Here is the call graph for this function:



Here is the caller graph for this function:



4.3.1.3 usunListe()

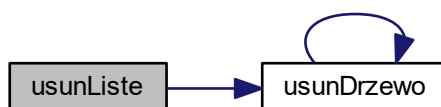
```
void usunListe (
    Lista * glowka )
```

Funkcja usuwa list.

Parameters

<i>glowka</i>	gowa listy
<i>*nastpny</i>	wskanik na nastpny element

Here is the call graph for this function:



Here is the caller graph for this function:



4.3.1.4 wczytajzpliku()

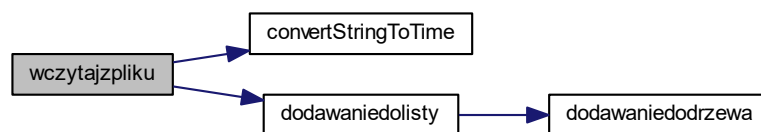
```
Lista* wczytajzpliku (  
    const std::string & nazwaPliku )
```

Funkcja wczytajca dane z pliku.

Parameters

<i>pHead</i>	gowa listy ustawiona na nullptr
<i>parametrylotu</i>	zmienna o strukturze parametry lotu
<i>data</i>	data lotu
<i>nazwisko</i>	nazwisko pasaera
<i>miejsce</i>	miejsce pasaera

Here is the call graph for this function:

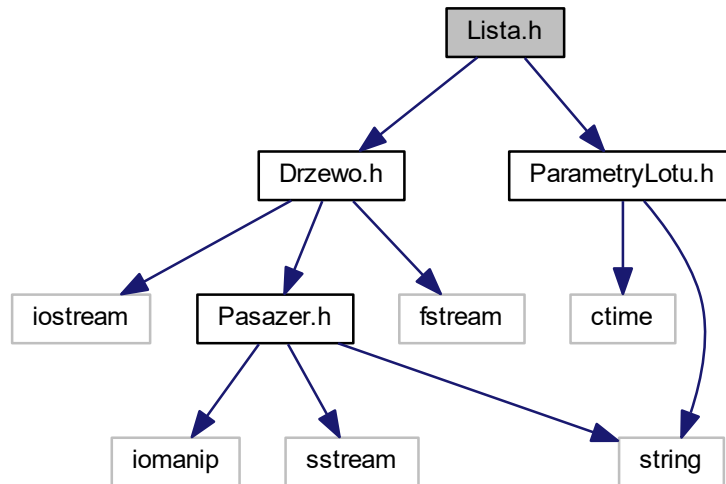


Here is the caller graph for this function:

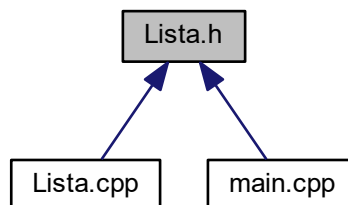


4.4 Lista.h File Reference

```
#include "Drzewo.h"
#include "ParametryLotu.h"
Include dependency graph for Lista.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- struct [Lista](#)

Functions

- void [stworzPlikilZapisz](#) ([Lista](#) *phead)
- void [usunListe](#) ([Lista](#) *glowka)
- [Lista](#) * [wczytajzpliku](#) (const std::string &nazwaPliku)
- [Lista](#) * [dodawanieDolisty](#) ([Lista](#) *pHead, [ParametryLotu](#) ¶metryLotu, [Pasazer](#) &pasazer)

4.4.1 Function Documentation

4.4.1.1 dodawaniedolisty()

```
Lista* dodawaniedolisty (
    Lista * pHead,
    ParametryLotu & parametryLotu,
    Pasazer & pasazer )
```

Funkcja dodajca do listy

Parameters

<i>pNowy</i>	wskanik na element, ktrego jeszcze nie bylo
<i>przechowywatorglowy</i>	przechowuje pHead
<i>temp</i>	kopia przechowywatoraglowy po to, aby lista mogala wejść do ostatniej iteracji
<i>nowyElement</i>	jeli nie ma symbolu lotu tworzymy nowe drzewo

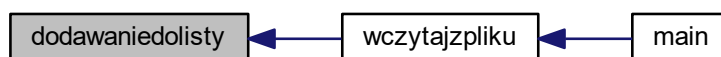
Returns

pHead zwraca pHead

Here is the call graph for this function:



Here is the caller graph for this function:



4.4.1.2 stworzPlikilZapisz()

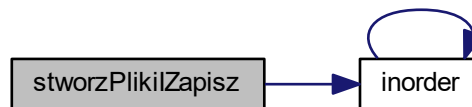
```
void stworzPlikilZapisz (
    Lista * phead )
```

Funkcja tworzy pliki. Jako parametr przyjmuje gow listy. Sprawdzamy czy przekazana lista istnieje.

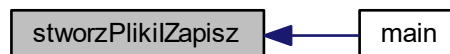
Parameters

<i>pHead</i>	gowa listy
<i>przechowywatorgowy</i>	dajemy gow do przechowywatora, eby nie niszczy listy i mc swobodnie j iterowa
<i>outfile</i>	stworzono strumie outfile do wywietlania

Here is the call graph for this function:



Here is the caller graph for this function:



4.4.1.3 usunListe()

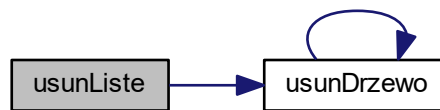
```
void usunListe (
    Lista * glowka )
```

Funkcja usuwa list.

Parameters

<i>glowka</i>	gowa listy
<i>*nastpny</i>	wskanik na nastpny element

Here is the call graph for this function:



Here is the caller graph for this function:



4.4.1.4 wczytajzpliku()

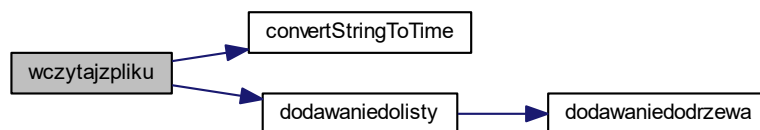
```
Lista* wczytajzpliku (
    const std::string & nazwaPliku )
```

Funkcja wczytajca dane z pliku.

Parameters

<i>pHead</i>	gowa listy ustawiona na nullptr
<i>parametrylotu</i>	zmienna o strukturze parametry lotu
<i>data</i>	data lotu
<i>nazwisko</i>	nazwisko pasaera
<i>miejsce</i>	miejsce pasaera

Here is the call graph for this function:



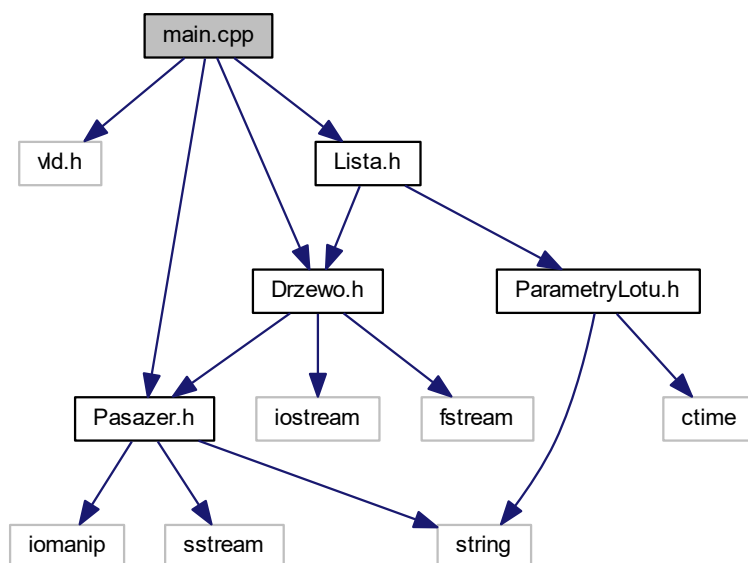
Here is the caller graph for this function:



4.5 main.cpp File Reference

```
#include "vld.h"
#include "Pasazer.h"
#include "Drzewo.h"
#include "Lista.h"
```

Include dependency graph for main.cpp:



Functions

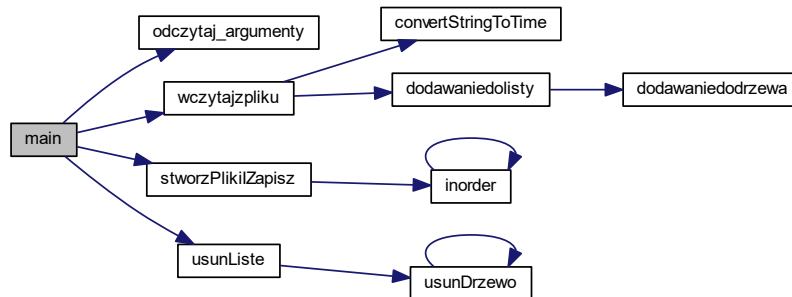
- bool [odczytaj_argumenty](#) (int ile, char **argumenty, std::string &inputFile)
- int [main](#) (int ile, char **argumenty)

4.5.1 Function Documentation

4.5.1.1 main()

```
int main (  
    int ile,  
    char ** argumenty )
```

Here is the call graph for this function:



4.5.1.2 odczytaj_argumenty()

```

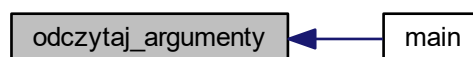
bool odczytaj_argumenty (
    int ile,
    char ** argumenty,
    std::string & inputFile )
  
```

Funkcja odczytująca argumenty

Parameters

<i>ile</i>	ile jest parametrów funkcji
<i>argumenty</i>	argumenty przekazywana w linii komend
<i>ETYKIETA_INPUT</i>	
<i>arg</i>	przechowuje dany argument

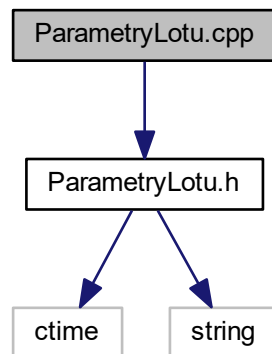
Here is the caller graph for this function:



4.6 ParametryLotu.cpp File Reference

```
#include "ParametryLotu.h"
```


Include dependency graph for ParametryLotu.cpp:



Functions

- bool `operator==` (const [ParametryLotu](#) &p1, const [ParametryLotu](#) &p2)
- std::ostream & `operator<<` (std::ostream &os, const std::tm &t)
- std::ostream & `operator<<` (std::ostream &os, const [ParametryLotu](#) ¶metryLotu)

4.6.1 Function Documentation

4.6.1.1 `operator<<()` [1/2]

```
std::ostream& operator<< (
    std::ostream & os,
    const std::tm & t )
```

Funkcja do wyświetlania danych ze struktury tm - opisującej dat

/*

Parameters

<i>os</i>	os to jest strumie ostreamowy
<i>t</i>	zmienna czasu

Returns

os zwracamy strumie os

4.6.1.2 operator<<() [2/2]

```
std::ostream& operator<< (
    std::ostream & os,
    const ParametryLotu & parametryLotu )
```

4.6.1.3 operator==()

```
bool operator== (
    const ParametryLotu & p1,
    const ParametryLotu & p2 )
```

Funkcja zwraca czy lotnisko i symbol lotu jest takie same

Parameters

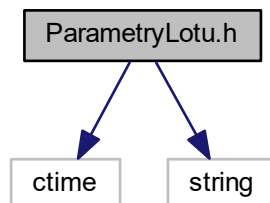
<i>p1</i>	parametry pierwszego lotu
<i>p2</i>	parametry drugiego lotu

4.7 ParametryLotu.h File Reference

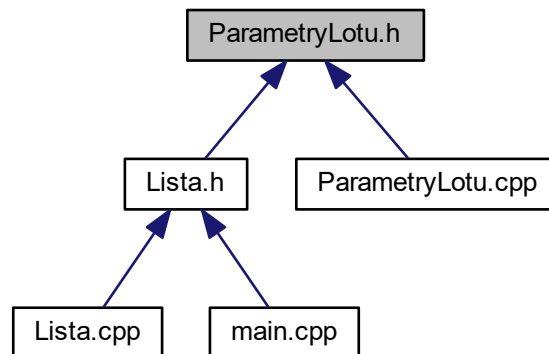
```
#include <ctime>
```

```
#include <string>
```

Include dependency graph for ParametryLotu.h:



This graph shows which files directly or indirectly include this file:



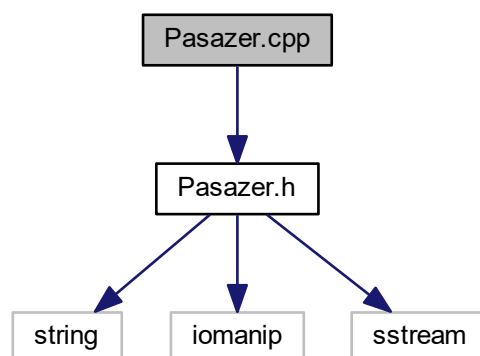
Classes

- struct [ParametryLotu](#)

4.8 Pasazer.cpp File Reference

```
#include "Pasazer.h"
```

Include dependency graph for Pasazer.cpp:



Functions

- `std::ostream & operator<< (std::ostream &os, const Pasazer &p)`
- `struct std::tm convertStringToTime (const std::string &str)`

4.8.1 Function Documentation

4.8.1.1 convertStringToTime()

```
struct std::tm convertStringToTime (
    const std::string & str )
```

Funcja konwertujca ciąg znaków na czas

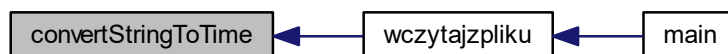
Parameters

<i>str</i>	string wejściowy
<i>ss</i>	strumień, który dzielimy tak jak wskazujemy w funkcji <code>get_time</code>

Returns

tm

Here is the caller graph for this function:



4.8.1.2 operator<<()

```
std::ostream& operator<< (
    std::ostream & os,
    const Pasazer & p )
```

Funcja do wyświetlania struktury `Pasazer`

Parameters

<i>os</i>	strumień
<i>p</i>	zmienna o strukturze <code>Pasazer</code>

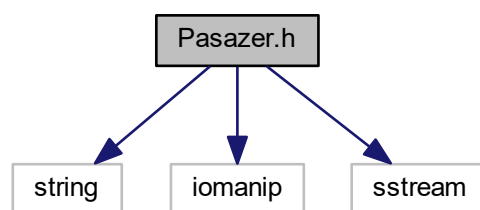
Returns

os zwracamy strumie

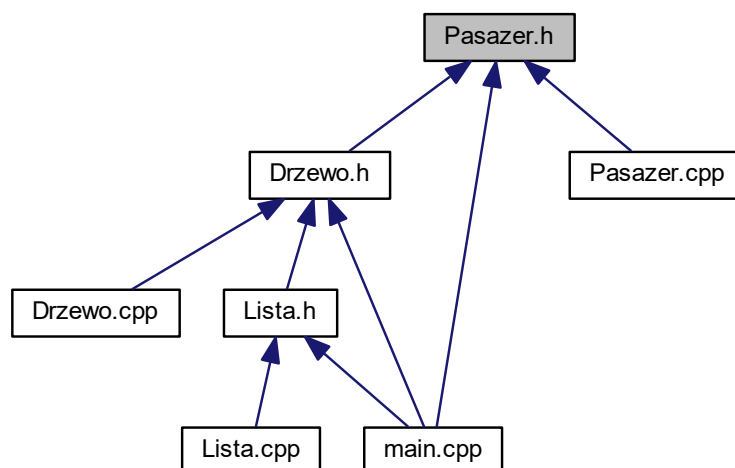
4.9 Pasazer.h File Reference

```
#include <string>
#include <iomanip>
#include <sstream>
```

Include dependency graph for Pasazer.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct [Pasazer](#)

Functions

- std::tm [convertStringToTime](#) (const std::string &str)

4.9.1 Function Documentation

4.9.1.1 convertStringToTime()

```
std::tm convertStringToTime (  
    const std::string & str )
```

Funkcja konwertująca ciąg znaków na czas

Parameters

<i>str</i>	string wejściowy
<i>ss</i>	strumień, który dzielimy tak jak wskazujemy w funkcji <code>get_time</code>

Returns

tm

Here is the caller graph for this function:

