22.06.2024, Gliwice

# Sprawozdanie końcowe

## Programowanie komputerów 2

## Temat projektu: Rezerwacja stolików w restauracji

AEI INF SSI, sem IV (Warunkowo sem II gr 4 sek 2)

Autor: Julia Żółty

Opiekun projektu: mgr Arkadiusz Czerwiński

Link do repozytorium: https://github.com/polsl-aei-pk2/e0f8bf12-gr42-repo/tree/main/Projekt

**Cel projektu**

Celem projektu było stworzenie konsolowej aplikacji do zarządzania rezerwacjami stolików w restauracji. Aplikacja miała umożliwiać użytkownikom przeglądanie dostępnych stolików, dokonywanie rezerwacji oraz anulowanie istniejących rezerwacji. Projekt był realizowany w języku C++ z wykorzystaniem plików do przechowywania danych o stolikach oraz rezerwacjach.

**Obsługa programu:**

- Po uruchomieniu programu, widzimy prosty interfejs. Z tego etapu możemy dokonać trzech różnych wyborów.

```
====DOSTEPNE MIASTA====
>Gliwice
>Katowice
>Warszawa

Aby zakonczyc, wprowadz '0'

Wprowadz nazwe miasta, w ktorym chcesz dokonac rezerwacji lub nacisnij '1' aby zobaczyc liste aktualnych rezerwacji
```

  - ✓ Zakończyć działanie programu wpisując ‘0’
  - ✓ Przeglądnąć listę aktualnych rezerwacji wpisując ‘1’

```
18. Miasto: Katowice; Stolik nr #1(2-osobowy) 25.07.2024, 15:00
Nazwisko:  zolty
Dodatkowe uwagi:  okno

19. Miasto: Katowice; Stolik nr #2(2-osobowy) 25.07.2024, 15:00
Nazwisko:  zolty
Dodatkowe uwagi:  okno

20. Miasto: Katowice; Stolik nr #1(2-osobowy) 25.08.2024, 16:00
Nazwisko:  zolty
Dodatkowe uwagi:  dziala

21. Miasto: Katowice; Stolik nr #4(4-osobowy) 25.08.2024, 16:00
Nazwisko:  zolty
Dodatkowe uwagi:  dziala


Wybierz numer zamowienia z ktorego chcesz zrezygnowac (jesli chcesz wrocic do poprzedniego menu wpisz 0):
```

Przeglądając listę aktualnych rezerwacji, jest możliwość ich usunięcia, poprzez wpisanie numeru rezerwacji z którego chcemy zrezygnować

```
20. Miasto: Katowice; Stolik nr #1(2-osobowy) 25.08.2024, 16:00
Nazwisko:  zolty
Dodatkowe uwagi:  dziala


Wybierz numer zamowienia z ktorego chcesz zrezygnowac (jesli chcesz wrocic do poprzedniego menu wpisz 0):

20
```

Poniższy zrzut ekranu przedstawia listę rezerwacji po usunięciu rezerwacji nr 19.

```
19. Miasto: Katowice; Stolik nr #1(2-osobowy) 25.08.2024, 16:00
Nazwisko:  zolty
Dodatkowe uwagi:  dziala


Wybierz numer zamowienia z ktorego chcesz zrezygnowac (jesli chcesz wrocic do poprzedniego menu wpisz 0):
```

  - ✓ Dokonać rezerwacji wpisując jedną nazwę miasta z tych podanych w konsoli

Jak można zauważyć, program jest zabezpieczony przed koniecznością wpisywania nazwy miasta z dużej litery.

```
====DOSTEPNE MIASTA====
>Gliwice
>Katowice
>Warszawa

Aby zakonczyc, wprowadz '0'

Wprowadz nazwe miasta, w ktorym chcesz dokonac rezerwacji lub nacisnij '1' aby zobaczyc liste aktualnych rezerwacji

GlIwiCE
```

- Na tym etapie użytkownik może przejść do dokonania rezerwacji stolika, bądź wrócić do wyboru miasta wpisując ‘0’

```
=====GLIWICE=====
Dostepne stoliki:
Stoliki 5-osobowe, ilosc: 3
Stoliki 3-osobowe, ilosc: 3
Stoliki 2-osobowe, ilosc: 2

Aby przejsc do rezerwacji godziny wybierz '1', aby wrocic do wyboru miasta wpisz '0'.
Jezeli ilosc stolikow po wybraniu godziny bedzie sie roznila to oznacza ze nie sa juz dostepne.

1
```

- Wpisanie daty i godziny rezerwacji jest zabezpieczone na kilka sposobów.
  - ✓ Dzień rezerwacji musi być rzeczywisty. Zabezpieczenie jest zaimplementowane w metodzie *bool List_res::checkdate*

```
Podaj date i godzine rezerwacji (dd-mm-rrrr, hh:min) (jesli chcesz wrocic, wybierz '0'):

32-06-2024, 16:00

Niepoprawne dane, wprowadz date jeszcze raz
```

  - ✓ Rezerwacji można dokonać od dnia obrony projektu (21.06.2024)

```
Podaj date i godzine rezerwacji (dd-mm-rrrr, hh:min) (jesli chcesz wrocic, wybierz '0'):

22-06-2023, 16:00

Wprowadz poprawny rok
```

Fragment pliku list_res.cpp :

```
186     if ((resyear == 2024 && resmonth < 6) || (resyear == 2024 && resmonth == 6 && resday < 21)) {
187         std::cout << "\nNiepoprawne dane rezerwacji\n" << std::endl;
188         std::cin.get();
189         return false;
190     }
```

  - ✓ Rezerwacji można dokonać między godziną 14 a 22

```
Podaj date i godzine rezerwacji (dd-mm-rrrr, hh:min) (jesli chcesz wrocic, wybierz '0'):

22-06-2024, 13:00

Przyjmujemy rezerwacje miedzy godzina 14 a 22.
```

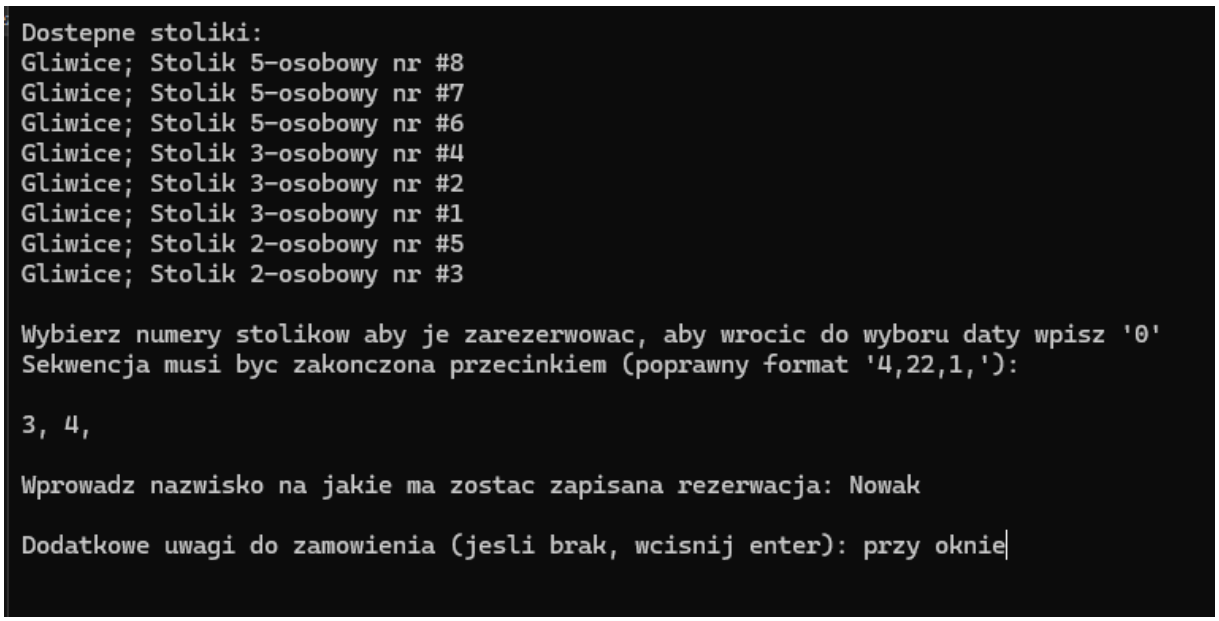
```
Podaj date i godzine rezerwacji (dd-mm-rrrr, hh:min) (jesli chcesz wrocic, wybierz '0'):

22-06-2024, 15:00
```

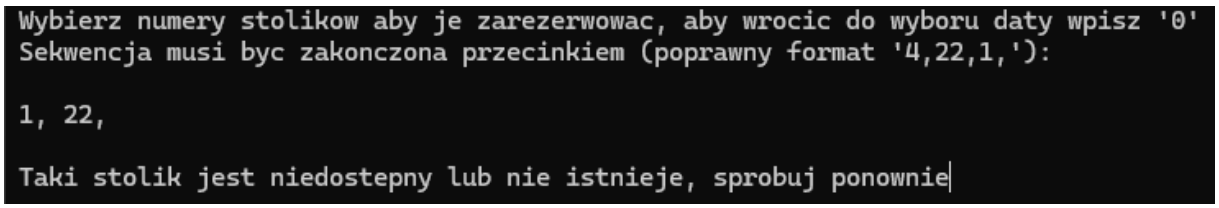
  - Po wpisaniu poprawnej daty, wyświetla się komunikat z dostępnymi stolikami

Użytkownik wybiera numery stolików, które chce zarezerwować wpisując poprawny format zamieszczony w instrukcji w konsoli

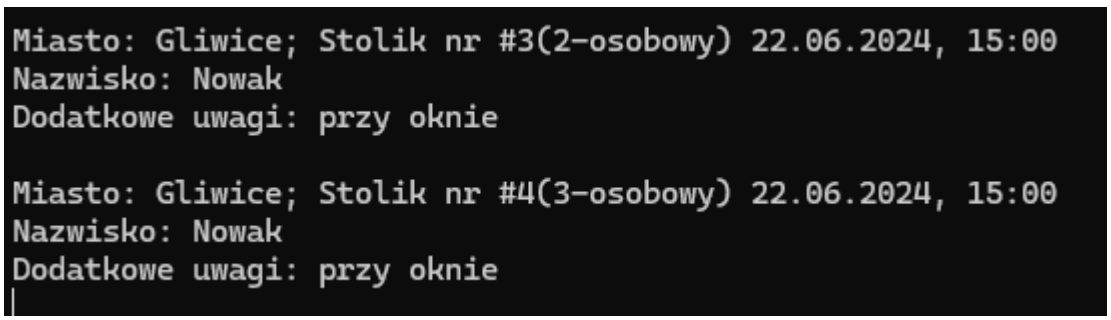
Użytkownik wprowadza swoje nazwisko, oraz dodatkowe uwagi




W przypadku wpisania złego formatu wyświetla się odpowiedni komunikat.
Użytkownik może wpisać numery stolików raz jeszcze.




Po dokonaniu prawidłowego wyboru stolika, wyświetla się podsumowanie zamówienia.




- Następnie użytkownik może dokonać następnej rezerwacji (wpisując ‘t’), bądź zakończyć działanie programu wpisując ‘n’

**Czego się nauczyłam**

- Zarządzanie pamięcią dynamiczną

Nauczyłam się, jak efektywnie korzystać z dynamicznej alokacji pamięci za pomocą operatorów *new* i *delete*. To umożliwiło mi tworzenie obiektów w trakcie działania programu oraz zarządzanie ich żywotnością.

Przykład użycia operatora *new* w pliku list_table.cpp

```
296     std::vector<Table>* returnedTables = new std::vector<Table>();
```

- Praca z plikami

Utrwaliłam techniki wczytywania i zapisywania danych z plików tekstowych. Było to kluczowe dla utrzymania stanu aplikacji pomiędzy uruchomieniami.

Metoda *savetofile* odpowiedzialna za zapis do pliku. Odwołanie do metody w pliku list_res.cpp

```
/** Metoda zapisujaca liste rezerwacji do pliku
@param name Nazwa pliku do ktorego maja zostac zapisane rezerwacje */
void savetofile(const std::string& name);
```

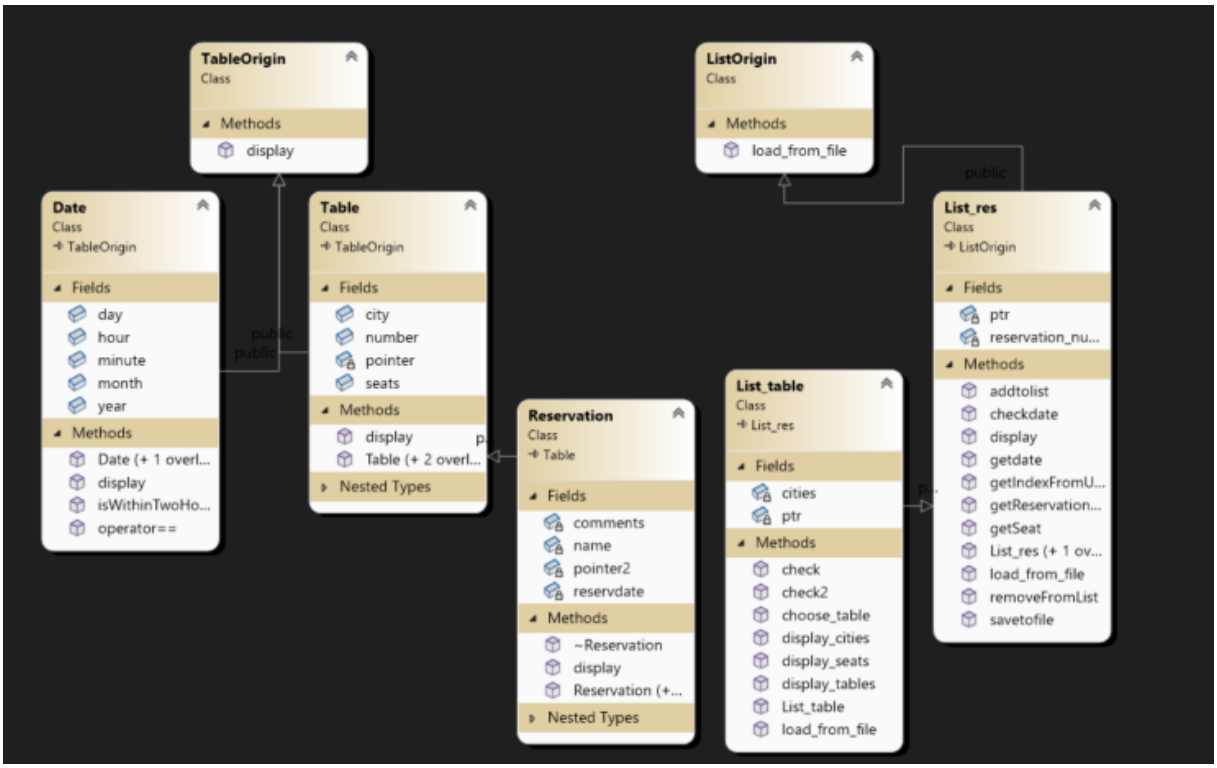
Metoda *load_from_file* odpowiedzialna za odczyt z pliku *baza.txt* bądź *rezerwacje.txt.* Odwołanie do metody w pliku list_res.cpp

```
/** Metoda wczytujaca z pliku dokonane rezerwacje
@param name Nazwa pliku z ktorego maja byc wczytane rezerwacje */
void load_from_file(const std::string& name);
```

- Używanie listy jednokierunkowej oraz wektorów
- Polimorfizm i dziedziczenie

Mimo, że już wcześniej miałam styczność z wykorzystaniem polimorfizmu i dziedziczenia, projekt pozwolił mi utrwalić ich wykorzystywanie. Po obronie projektu wiem że metody savetofile oraz load_from_file również mogłam zawrzeć w polimorfizmie, oprócz zaimplenetowanej metody display która wyświetla dane plików tekstowych na ekranie.

Poniższy zrzut ekranu przedstawia diagram klas wygenerowany w programie VS 2022. Jest na nim pokazane dziedziczenie klas.




Poniższe dwa zrzuty ekranu przedstawiają polimorfizm programu.

Plik list_origin.h

```cpp
/**@file*/
#ifndef LIST_ORIGIN_H
#define LIST_ORIGIN_H

#pragma once
#include <string>

/**Klasa bazowa dla klasy list_res, zawierajaca wirtualna metode load_from_file sluzaca do wczytywania danych z pliku */
class ListOrigin
{
public:

    //wirtualna metoda (=0) z argumentem z referencją do name czyli nazwa pliku

    virtual void load_from_file(const std::string& name) = 0;
};

#endif
```

Plik table_origin.h

```cpp
/** @file */
#pragma once
#ifndef TABLE_ORIGIN_H
#define TABLE_ORIGIN_H

/**Klasa bazowa dla klasy Table, zawierajaca wirtualna metode display, sluzaca do wyswietlania na ekranie */
class TableOrigin
{
public:
    virtual void display() = 0;
};
#endif
```

**Problemy i wyzwania :**

- Obsługa błędów

Kolejnym wyzwaniem była obsługa błędów podczas operacji wejścia/wyjścia z plikami oraz przetwarzania danych wejściowych od użytkownika. Musiałam upewnić się, że program nie zakończy działania w przypadku nieoczekiwanych błędów. Na różnym etapie pisania projektu pojawiały się różne błędy przy kompilacji. Czasami naprawa wymagała zmiany jednej linijki kodu, w innych wypadkach musiałam implementować daną metodę w zupełnie nowy sposób.

- Składnia i logika C++

Nauczenie się składni i specyficznych koncepcji C++ takich jak konstruktor kopiujący, destruktor, przeciążanie operatorów. Największym problemem okazało się wielokrotne użycie przeciążenia operatorów. Udało mi się dokonać tylko jednej takiej implementacji, mimo że w wymaganiach projektu było użycie minimum trzech takich operatorów.

Fragment pliku date.h :

```
/** @brief Przeciążony operator porównania dla daty.
*
* Ten operator porownuje dwie daty i zwraca true, jesli są identyczne,
* tj. jesli dzień, miesiąc, rok, godzina i minuta są identyczne.
* W przeciwnym razie zwraca false.
*
* @param other Druga data do porownania.
* @return true, jesli daty są identyczne.
* @return false, jesli daty są rożne.
*/
bool operator == (const Date& d) const;
```

- Interfejs i zabezpieczenia działania programu

Temat projektu wydaje się dość prosty i intuicyjny, natomiast trzeba było zmierzyć się z zabezpieczeniami przed tym co użytkownik wpisuje do konsoli. Np. co gdy wpisze datę 31-06-2024, gdy taka data nie istnieje ponieważ czerwiec ma 30 dni.

# rezerwacja stolikow w restauracji

Wygenerowano za pomocą Doxygen 1.11.0

# 1 Indeks hierarchiczny

## 1.1 Hierarchia klas

Ta lista dziedziczenia posortowana jest z grubsza, choć nie całkowicie, alfabetycznie:



# 2 Indeks klas

## 2.1 Lista klas

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



# 3 Indeks plików

## 3.1 Lista plików

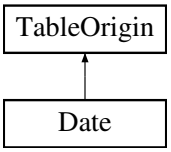
Tutaj znajduje się lista wszystkich plików wraz z ich krótkimi opisami:

# 4 Dokumentacja klas

## 4.1 Dokumentacja klasy Date

`#include <date.h>`

Diagram dziedziczenia dla Date




### Metody publiczne

- void **display** ()
- bool **isWithinTwoHours** (const **Date** &d) const

  *Sprawdza, czy rezerwacja dla danej daty miesci sikresie dwoch godzin.*
- bool **operator==** (const **Date** &d) const

  *Przeciony operator pornia dla daty.*
- **Date** (int &y, int &m, int &d, int &h, int &min)
- **Date** ()

### Atrybuty publiczne

- int **year**
- int **month**
- int **day**
- int **hour**
- int **minute**

**Przyjaciele**

- class **List_res**
- class **Reservation**

### 4.1.1 Opis szczegółowy

Klasa **Date** (str. **??**) zawierajaca dane dotyczace daty rezerwacji

### 4.1.2 Dokumentacja konstruktora i destruktora

**Date() [1/2]**

```
Date::Date (
            int & y,
            int & m,
            int & d,
            int & h,
            int & min)
```

Konstruktor piecioargumentowy

**Parametry**

| | |
|---|---|
| *y* | Rok rezerwacji stolika |
| *d* | Dzien rezerwacji stolika |
| *h* | Godzina rezerwacji stolika |
| *min* | Minuta rezerwacji stolika |

**Date() [2/2]**

```
Date::Date ()
```

Konstruktor bezargumentowy

### 4.1.3 Dokumentacja funkcji składowych

**display()**

```
void Date::display ()  [virtual]
```

Metoda sluzaca do wyswietlania daty

Implementuje **TableOrigin** (str. **??**).

**isWithinTwoHours()**

```
bool Date::isWithinTwoHours (
            const  Date & d) const
```

Sprawdza, czy rezerwacja dla danej daty miesci sikresie dwoch godzin.

Funkcja ta porownuje date wywoujacego obiektu z data podan jako parametr i zwraca true, jesli miy nimi nie ma wij ni 2 godzin roznicy czasu. W przeciwnym razie zwraca false.

**Parametry**

| | |
|---|---|
| *d* | Data, ktora naley porownac z data wywokujacego obiektu. |

**Zwraca**

true, jesli miedzy datami nie ma wij ni 2 godzin roznicy czasu.

false, jesli miedzy datami jest wij ni 2 godziny roznicy czasu.

### operator==()

```
bool Date::operator== (
            const  Date & d) const
```

Przeciony operator pornia dla daty.

Ten operator porownuje dwie daty i zwraca true, jesli s identyczne, tj. jesli dzieesic, rok, godzina i minuta s identyczne. W przeciwnym razie zwraca false.

**Parametry**

| | |
|---|---|
| *other* | Druga data do porownania. |

**Zwraca**

true, jesli daty s identyczne.

false, jesli daty s rone.

## 4.1.4 Dokumentacja przyjaciół i powiązanych symboli

### List_res

```
friend class  List_res  [friend]
```

### Reservation

```
friend class  Reservation  [friend]
```

## 4.1.5 Dokumentacja atrybutów składowych

### day

```
int Date::day
```

#### hour

```
int Date::hour
```

#### minute

```
int Date::minute
```

#### month

```
int Date::month
```

#### year

```
int Date::year
```

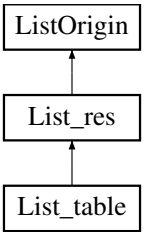
Dokumentacja dla tej klasy została wygenerowana z plików:

- C:/Users/Julia/Documents/PK2/e0f8bf12-gr42-repo/Projekt/rezerwacja/ **date.h**
- C:/Users/Julia/Documents/PK2/e0f8bf12-gr42-repo/Projekt/rezerwacja/ **date.cpp**

## 4.2 Dokumentacja klasy List_res

```
#include <list_res.h>
```

Diagram dziedziczenia dla List_res




#### Metody publiczne

- int **getReservationNumber** ()
- void **load_from_file** (const std::string &name)
- bool **checkdate** (std::string &city, std::string &stringdate)
- void **getSeat** (std::string &chosenSeat)
- void **addtolist** ( **Reservation** &res)
- bool **removeFromList** (int primaryKey)
- void **display** ()
- void **savetofile** (const std::string &name)
- **Date** **getdate** (std::string &stringdate)
- **List_res** ( **smart2** &respt r)
- **List_res** ()
- int **getIndexFromUser** ()

**Przyjaciele**

- class **List_table**
- std::ostream & **operator**<< (std::ostream &s, **List_res** &list)

### 4.2.1 Opis szczegółowy

Klasa bedaca lista rezerwacji wczytanych z pliku i wprowadzonych przez uzytkownika

### 4.2.2 Dokumentacja konstruktora i destruktora

**List_res() [1/2]**

```
List_res::List_res (
              smart2 & resptr)
```

Konstruktor jednoargumentowy

**Parametry**

| | |
|---|---|
| *resptr* | Wskaznik na obiekt typu **Reservation** (str. **??**) |

**List_res() [2/2]**

```
List_res::List_res ()
```

Konstruktor bezargumentowy

### 4.2.3 Dokumentacja funkcji składowych

**addtolist()**

```
void List_res::addtolist (
              Reservation & res)
```

Metoda dodajaca rezerwacje do listy

**Parametry**

| | |
|---|---|
| *res* | Rezerwacja ktora ma zostac dodana do listy |

**checkdate()**

```
bool List_res::checkdate (
             std::string & city,
             std::string & stringdate)
```

Metoda sprawdzajaca poprawnosc wprowadzonej przez uzytkownika daty

**Parametry**

| | |
|---|---|
| *city* | Wybor miasta wprowadzony przez uzytkownika |
| *seats* | Liczba siedzen przy stoliku wprowadzona przez uzytkownika |
| *stringdate* | Wybor daty wprowadzony przez uzytkownika return True jesli data jest poprawna, w przeciwnym wypadku wartosc false |

### display()

```
void List_res::display ()
```

Metoda wypisujaca zawartosc listy

### getdate()

```
Date List_res::getdate (
            std::string & stringdate)
```

Metoda pobierajaca wartosc daty ze znakow wprowadzonych przez uzytkownika

**Parametry**

| | |
|---|---|
| *stringdate* | Wybor daty wprowadzony przez uzytkownika |

**Zwraca**

Obiekt typu **Date** (str. **??**) bedacy data o danych wprowadzonych przez uzytkownika

### getIndexFromUser()

```
int List_res::getIndexFromUser ()
```

Metoda pobierajaca numer rezerwacji do usuniecia i zamieniajaca go z liczbe

**Zwraca**

Wybrany numer rezerwacji

### getReservationNumber()

```
int List_res::getReservationNumber ()
```

Metoda pobierajaca liczbe rezerwacji

**Zwraca**

Liczbe rezerwacji

### getSeat()

```
void List_res::getSeat (
            std::string & chosenSeat)
```

Metoda pobierajaca numery stolikow do rezerwacji od uzytkownika a nastepnie zamienia je w liczbe

**Parametry**

| | |
|---|---|
| *chosenSeat* | Numer stolika wprowadzony przez uzytkownika |

### load_from_file()

```
void List_res::load_from_file (
            const std::string & name)  [virtual]
```

Metoda wczytujaca z pliku dokonane rezerwacje

**Parametry**

| | |
|---|---|
| *name* | Nazwa pliku z ktorego maja byc wczytane rezerwacje |

Implementuje **ListOrigin** (str. **??**).

Reimplementowana w **List_table** (str. **??**).

### removeFromList()

```
bool List_res::removeFromList (
            int primaryKey)
```

Metoda usuwajaca rezerwacje z listy

**Parametry**

| | |
|---|---|
| *primaryKey* | Numer rezerwacji do usuniecia wprowadzony przez uzytkownika |

**Zwraca**

True gdy rezerwacja zostanie poprawnie usunieta, przeciwnym wypadku False

### savetofile()

```
void List_res::savetofile (
            const std::string & name)
```

Metoda zapisujaca liste rezerwacji do pliku

**Parametry**

| | |
|---|---|
| *name* | Nazwa pliku do ktorego maja zostac zapisane rezerwacje |

#### 4.2.4 Dokumentacja przyjaciół i powiązanych symboli

**List_table**

```
friend class List_table [friend]
```

**operator<<**

```
std::ostream & operator<< (
             std::ostream & s,
              List_res & list) [friend]
```

Operator strumieniowy wyjscia wypisujacy zawartosc listy

**Parametry**

| | |
|---|---|
| *s* | Operator strumieniowy wyjscia |
| *list* | Lista do wypisania |

**Zwraca**

Operator strumieniowy wyjscia

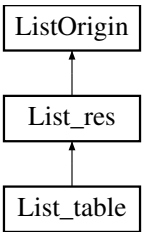
Dokumentacja dla tej klasy została wygenerowana z plików:

- C:/Users/Julia/Documents/PK2/e0f8bf12-gr42-repo/Projekt/rezerwacja/ **list_res.h**
- C:/Users/Julia/Documents/PK2/e0f8bf12-gr42-repo/Projekt/rezerwacja/ **list_res.cpp**

## 4.3 Dokumentacja klasy List_table

```
#include <list_table.h>
```

Diagram dziedziczenia dla List_table




**Metody publiczne**

- void **load_from_file** (const std::string &name)
- void **display_cities** ()
- void **display_seats** (std::string &city)
- void **display_tables** (std::string &city)
- std::vector< **Table** > * **choose_table** ( **Date** &dat, std::string &city, std::string &seats, **List_res** &thelist)
- bool **check** (std::string &city)
- bool **check2** (std::string &city, std::string &seats)
- **List_table** ( **smart** &tableptr)

**Metody publiczne dziedziczone z List_res**

- int **getReservationNumber** ()
- bool **checkdate** (std::string &city, std::string &stringdate)
- void **getSeat** (std::string &chosenSeat)
- void **addtolist** ( **Reservation** &res)
- bool **removeFromList** (int primaryKey)
- void **display** ()
- void **savetofile** (const std::string &name)
- **Date** **getdate** (std::string &stringdate)
- **List_res** ( **smart2** &respt r)
- **List_res** ()
- int **getIndexFromUser** ()

**Przyjaciele**

- std::ostream & **operator**<< (std::ostream &s, std::set< std::string > c)
- std::ostream & **operator**<< (std::ostream &s, **List_table** &list)

### 4.3.1 Opis szczegółowy

Klasa bedaca lista stolikow wczytanych z pliku

### 4.3.2 Dokumentacja konstruktora i destruktora

**List_table()**

```
List_table::List_table (
              smart & tableptr)
```

Konstruktor jednoargumentowy

**Parametry**

| | |
|---|---|
| *tableptr* | Wskaznik na obiekt typu **Table** (str. **??**) |

### 4.3.3 Dokumentacja funkcji składowych

**check()**

```
bool List_table::check (
             std::string & city)
```

Metoda sprawdzajca poprawnosc wprowadzonego miasta

**Parametry**

| | |
|---|---|
| *city* | Ciag znakow wprowadzony od uzytkownika |

**Zwraca**

True gdy miasto zostalo wprowadzone poprawnie, w przeciwnym wypadku wartosc False

### check2()

```
bool List_table::check2 (
            std::string & city,
            std::string & seats)
```

Metoda sprawdzajaca czy ilosc wprowadzonych miejsc przy stoliku jest poprawna

**Parametry**

| | |
|---|---|
| *city* | Wybor miasta wprowadzony przez uzytkownika |
| *seats* | Ciag znakow wprowadzony przez uzytkownika |

**Zwraca**

True gdy liczba miejsc zostala wprowadzona porawnie, w przeciwnym wypadku wartosc False

### choose_table()

```
std::vector<  Table > * List_table::choose_table (
             Date & dat,
            std::string & city,
            std::string & seats,
             List_res & thelist)
```

Metoda pozwalajaca zarezerwowac dowolna liczbe stolikow w danym miescie

**Parametry**

| | |
|---|---|
| *dat* | Data wprowadzona przez uzytkownika |
| *city* | Miasto wprowadzone przez uzytkownika |
| *seats* | Liczba siedzen przy stoliku |
| *thelist* | Lista zarezerwowanych stolikow, sluzaca do porownania daty rezerwacji |

**Zwraca**

Wektor przechowujacy obiekty typu **Table** (str. **??**) ktore zostaly zarezerwowane

### display_cities()

```
void List_table::display_cities ()
```

Metoda wyswietlajaca miasta

### display_seats()

```
void List_table::display_seats (
            std::string & city)
```

Metoda wyswietlajaca ilosc siedzen przy stolikach, dostepnych w danym miescie

**Parametry**

| | |
|---|---|
| *city* | Miasto, ktorego stoliki sa wyswietlane |

### display_tables()

```
void List_table::display_tables (
            std::string & city)
```

Metoda wyswietlajace wszystkie stoliki dostepne do zarezerwowania w danym mieiscie

**Parametry**

| | |
|---|---|
| *city* | Miasto dla ktorego stoliki sa wyswietlane |

### load_from_file()

```
void List_table::load_from_file (
            const std::string & name)  [virtual]
```

Metoda wczytujaca dane dotyczace stolikow z pliku

**Parametry**

| | |
|---|---|
| *name* | Nazwa pliku do wczytania |

Reimplementowana z **List_res** (str. **??**).

#### 4.3.4 Dokumentacja przyjaciół i powiązanych symboli

### operator<< [1/2]

```
std::ostream & operator<< (
            std::ostream & s,
             List_table & list)  [friend]
```

Operator strumieniowy wyjscia sluzacy do wypisania zawartosci listy (stolikow wczytanych)

**Parametry**

| | |
|---|---|
| *s* | Operator strumieniowy wyjscia @list Lista stolikow wczytanych z pliku |

**Zwraca**

Operator strumieniowy wyjscia

### operator<< [2/2]

```
std::ostream & operator<< (
            std::ostream & s,
            std::set< std::string > c)  [friend]
```

Operator strumieniowy wyjscia sluzacy do wypisania zawartosci zbioru miast

**Parametry**

| | |
|---|---|
| *s* | Operator strumieniowy wyjscia |
| *c* | Zbior miast |

**Zwraca**

Operator strumieniowy wyjscia

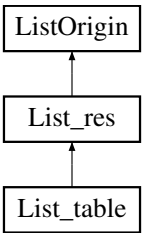
Dokumentacja dla tej klasy została wygenerowana z plików:

- C:/Users/Julia/Documents/PK2/e0f8bf12-gr42-repo/Projekt/rezerwacja/ **list_table.h**
- C:/Users/Julia/Documents/PK2/e0f8bf12-gr42-repo/Projekt/rezerwacja/ **list_table.cpp**

## 4.4 Dokumentacja klasy ListOrigin

`#include <list_origin.h>`

Diagram dziedziczenia dla ListOrigin




**Metody publiczne**

- virtual void **load_from_file** (const std::string &name)=0

### 4.4.1 Opis szczegółowy

Klasa bazowa dla klasy list_res, zawierajaca wirtualna metode load_from_file sluzaca do wczytywania danych z pliku

### 4.4.2 Dokumentacja funkcji składowych

**load_from_file()**

```
virtual void ListOrigin::load_from_file (
             const std::string & name)  [pure virtual]
```

Implementowany w **List_res** (str. **??**) i **List_table** (str. **??**).

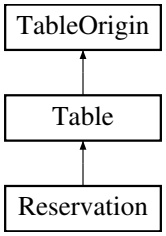
Dokumentacja dla tej klasy została wygenerowana z pliku:

- C:/Users/Julia/Documents/PK2/e0f8bf12-gr42-repo/Projekt/rezerwacja/ **list_origin.h**

## 4.5 Dokumentacja klasy Reservation

`#include <reservation.h>`

Diagram dziedziczenia dla Reservation




### Metody publiczne

- **Reservation** (std::string &c, int &n, int &s, std::string &resname, std::string &com, **Date** &dat, smart2 next2)
- **Reservation** (std::string &c, int &n, int &s, std::string &resname, std::string &com, **Date** &dat)
- **Reservation** (const **Reservation** &r)
- ∼**Reservation** ()
- void **display** ()

### Metody publiczne dziedziczone z Table

- **Table** (std::string &c, int &n, int &s, **smart** &next)
- **Table** (std::string &c, int &n, int &s)
- **Table** ()
- void **display** ()

### Przyjaciele

- class **List_res**
- class **List_table**
- std::ostream & **operator**<< (std::ostream &s, **List_res** &list)
- std::ostream & **operator**<< (std::ostream &s, **Reservation** &res)

### Dodatkowe dziedziczone składowe

### Typy publiczne dziedziczone z Table

- typedef std::shared_ptr< **Table** > **smart**

  *Typydef dla wskanika na kolejny obiekt typu **Reservation** (*str. **??**).*

### Atrybuty publiczne dziedziczone z Table

- std::string **city**

  *Zmienna przechowujca miasto, w kt znajduje siolik.*
- int **number**

  *Zmienna przechowujca numer stolika.*
- int **seats**

  *Zmienna przechowujca liczbejsc przy stoliku.*

### 4.5.1 Opis szczegółowy

Klasa bedaca rezerwacja stolika dokonana przez uzytkownika

### 4.5.2 Dokumentacja konstruktora i destruktora

#### Reservation() [1/3]

```
Reservation::Reservation (
            std::string & c,
            int & n,
            int & s,
            std::string & resname,
            std::string & com,
             Date & dat,
            smart2 next2)
```

Konstruktor siedmioargumentowy

**Parametry**

| | |
|---|---|
| *c* | Miasto |
| *n* | Numer stolika |
| *resname* | Nazwisko rezerwacji |
| *com* | Dodatkowe uwagi |
| *dat* | Data rezerwacji |
| *next2* | Wskaznik na nastepny obiekt |

#### Reservation() [2/3]

```
Reservation::Reservation (
            std::string & c,
            int & n,
            int & s,
            std::string & resname,
            std::string & com,
             Date & dat)
```

Konstruktor szescioargumentowy

**Parametry**

| | |
|---|---|
| *c* | Miasto |
| *n* | Numer stolika |
| *resname* | Nazwisko rezerwacji |
| *com* | Dodatkowe uwagi |
| *dat* | Data rezerwacji |

#### Reservation() [3/3]

```
Reservation::Reservation (
            const  Reservation & r)
```

Konstruktor kopiujacy

**Parametry**

| | |
|---|---|
| *r* | Obiekt ktorego dane sa kopiowane |

### ~Reservation()

```
Reservation::~Reservation ()
```

Destruktor bezargumentowy klasy **Reservation** (str. **??**)

### 4.5.3 Dokumentacja funkcji składowych

### display()

```
void Reservation::display ()  [virtual]
```

Metoda wypisujaca dane klasy

Implementuje **TableOrigin** (str. **??**).

### 4.5.4 Dokumentacja przyjaciół i powiązanych symboli

### List_res

```
friend class List_res [friend]
```

### List_table

```
friend class List_table [friend]
```

### operator<< [1/2]

```
std::ostream & operator<< (
            std::ostream & s,
             List_res & list)  [friend]
```

Operator strumieniowy wyjscia

**Parametry**

| | |
|---|---|
| *s* | Operator strumieniowy wyjscia |
| *list* | Lista do wypisania |

**Zwraca**

Operator strumieniowy wyjscia

Operator strumieniowy wyjscia wypisujacy zawartosc listy

**Parametry**

| | |
|---|---|
| *s* | Operator strumieniowy wyjscia |
| *list* | Lista do wypisania |

**Zwraca**

Operator strumieniowy wyjscia

**operator<< [2/2]**

```
std::ostream & operator<< (
             std::ostream & s,
              Reservation & res)  [friend]
```

Operator strumieniowy wyjscia

**Parametry**

| | |
|---|---|
| *s* | Operator strumieniowy wyjscia |
| *res* | Rezerwacja do wypisania |

**Zwraca**

Operator strumieniowy wyjscia

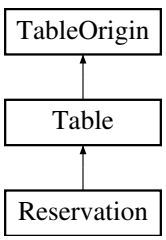
Dokumentacja dla tej klasy została wygenerowana z plików:

- C:/Users/Julia/Documents/PK2/e0f8bf12-gr42-repo/Projekt/rezerwacja/ **reservation.h**
- C:/Users/Julia/Documents/PK2/e0f8bf12-gr42-repo/Projekt/rezerwacja/ **reservation.cpp**

## 4.6 Dokumentacja klasy Table

```
#include <table.h>
```

Diagram dziedziczenia dla Table




### Typy publiczne

- typedef std::shared_ptr< **Table** > **smart**

  *Typydef dla wskanika na kolejny obiekt typu* ***Reservation*** *(*str. **??***).*

#### Metody publiczne

- **Table** (std::string &c, int &n, int &s, **smart** &next)
- **Table** (std::string &c, int &n, int &s)
- **Table** ()
- void **display** ()

#### Atrybuty publiczne

- std::string **city**

  *Zmienna przechowujca miasto, w kt znajduje siolik.*
- int **number**

  *Zmienna przechowujca numer stolika.*
- int **seats**

  *Zmienna przechowujca liczbejsc przy stoliku.*

#### Przyjaciele

- class **List_table**
- std::ostream & **operator**<< (std::ostream &s, **List_table** &list)
- std::ostream & **operator**<< (std::ostream &s, **Table** &t)

### 4.6.1 Opis szczegółowy

Klasa **Table** (str. **??**) zawierajaca dane dotyczace stolika

### 4.6.2 Dokumentacja składowych definicji typu

#### smart

```
typedef std::shared_ptr< Table>  Table::smart
```

Typydef dla wskanika na kolejny obiekt typu **Reservation** (str. **??**).

### 4.6.3 Dokumentacja konstruktora i destruktora

#### Table() [1/3]

```
Table::Table (
            std::string & c,
            int & n,
            int & s,
             smart & next)
```

Konstruktor czteroargumentowy

**Parametry**

| | |
|---|---|
| *c* | Miasto |
| *n* | Numer stolika |
| *s* | Liczba miejsc przy stoliku |
| *next* | Wskaznik na nastepny obiekt typu **Table** (str. **??**) |

#### Table() [2/3]

```
Table::Table (
            std::string & c,
            int & n,
            int & s)
```

#### Table() [3/3]

```
Table::Table ()
```

Konstruktor bezargumentowy

### 4.6.4 Dokumentacja funkcji składowych

#### display()

```
void Table::display ()  [virtual]
```

Metoda wyswietlajaca dane stolika

Implementuje **TableOrigin** (str. **??**).

### 4.6.5 Dokumentacja przyjaciół i powiązanych symboli

#### List_table

```
friend class  List_table  [friend]
```

#### operator<< [1/2]

```
std::ostream & operator<< (
            std::ostream & s,
             List_table & list)  [friend]
```

Operator strumieniowy wyjscia

**Parametry**

| | |
|---|---|
| *s* | Operator strumieniowy wyjscia |
| *list* | Lista stolikow wczytanych z pliku |

**Zwraca**

Operator strumieniowy wyjscia

Operator strumieniowy wyjscia sluzacy do wypisania zawartosci listy (stolikow wczytanych)

**Parametry**

| | |
|---|---|
| *s* | Operator strumieniowy wyjscia @list Lista stolikow wczytanych z pliku |

**Zwraca**

Operator strumieniowy wyjscia

### operator<< [2/2]

```
std::ostream & operator<< (
            std::ostream & s,
             Table & t)  [friend]
```

Operator strumieniowy wyjscia

**Parametry**

| | |
|---|---|
| *s* | operator strumieniowy wyjscia |
| *t* | Stolik ktorego dane zostana wypisane |

**Zwraca**

Operator strumieniowy wyjscia

### 4.6.6 Dokumentacja atrybutów składowych

### city

```
std::string Table::city
```

Zmienna przechowujca miasto, w kt znajduje siolik.

### number

```
int Table::number
```

Zmienna przechowujca numer stolika.

#### seats

```
int Table::seats
```

Zmienna przechowujca liczbejsc przy stoliku.

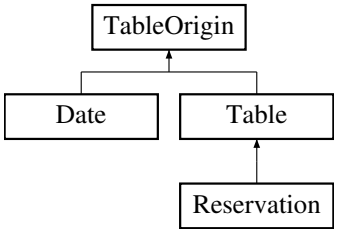
Dokumentacja dla tej klasy została wygenerowana z plików:

- C:/Users/Julia/Documents/PK2/e0f8bf12-gr42-repo/Projekt/rezerwacja/ **table.h**
- C:/Users/Julia/Documents/PK2/e0f8bf12-gr42-repo/Projekt/rezerwacja/ **table.cpp**

## 4.7 Dokumentacja klasy TableOrigin

```
#include <table_origin.h>
```

Diagram dziedziczenia dla TableOrigin




#### Metody publiczne

- virtual void **display** ()=0

### 4.7.1 Opis szczegółowy

Klasa bazowa dla klasy **Table** (str. **??**), zawierajaca wirtualna metode display, sluzaca do wyswietlania na ekranie

### 4.7.2 Dokumentacja funkcji składowych

#### display()

```
virtual void TableOrigin::display ()  [pure virtual]
```

Implementowany w **Date** (str. **??**), **Reservation** (str. **??**) i **Table** (str. **??**).

Dokumentacja dla tej klasy została wygenerowana z pliku:

- C:/Users/Julia/Documents/PK2/e0f8bf12-gr42-repo/Projekt/rezerwacja/ **table_origin.h**

# 5 Dokumentacja plików

## 5.1 Dokumentacja pliku C:/Users/Julia/Documents/PK2/e0f8bf12-gr42-repo/Projekt/rezerwacja/date.cpp

```
#include "date.h"
#include <iostream>
```

## 5.2 Dokumentacja pliku C:/Users/Julia/Documents/PK2/e0f8bf12-gr42-repo/Projekt/rezerwacja/date.h

```
#include "table_origin.h"
```

### Komponenty

- class **Date**

### Definicje

- #define **DATE_H**

#### 5.2.1 Dokumentacja definicji

#### DATE_H

```
#define DATE_H
```

## 5.3 date.h

**Idź do dokumentacji tego pliku.**

```
00001
00002 #pragma once
00003 #ifndef DATE_H
00004 #define DATE_H
00005
00006 #include "table_origin.h"
00007
00009 class Date : public TableOrigin {
00010 public:
00011     int year; //Zmienna przechowujaca rok rezerwacji stolika
00012
00013     int month; //Zmienna przechowuajca miesiac rezerwacji stolika
00014
00015     int day; //Zmienna przechowujaca dzien rezerwacji stolika
00016
00017     int hour; //Zmienna przechowujaca godzine rezerwacji stolika
00018
00019     int minute; //Zmienna przechowujaca minute rezerwacji stolika
00020
00021 public:
00022
00024     void display();
00025
00037     bool isWithinTwoHours(const Date& d) const;
```

```
00038
00049     bool operator == (const Date& d) const;
00050
00056     Date(int& y, int& m, int& d, int& h, int& min);
00057
00059     Date();
00060
00061     //zaprzyjaznione klasy maja dostep do prywatnych i chronioncyh skladowych klasy Date przez klasy
      list_res i reservation
00062     friend class List_res;
00063
00064     friend class Reservation;
00065
00066 };
00067
00068 #endif
```

## 5.4 Dokumentacja pliku C:/Users/Julia/Documents/PK2/e0f8bf12-gr42-repo/Projekt/rezerwacja/list_origin.h

```
#include <string>
```

### Komponenty

- class **ListOrigin**

## 5.5 list_origin.h

**Idź do dokumentacji tego pliku.**

```
00001
00002 #ifndef LIST_ORIGIN_H
00003 #define LIST_ORIGIN_H
00004
00005 #pragma once
00006 #include <string>
00007
00009 class ListOrigin
00010 {
00011 public:
00012
00013     //wirtualna metoda (=0) z argumentem z referencj do name czyli nazwa pliku
00014
00015     virtual void load_from_file(const std::string& name) = 0;
00016 };
00017
00018 #endif
```

## 5.6 Dokumentacja pliku C:/Users/Julia/Documents/PK2/e0f8bf12-gr42-repo/Projekt/rezerwacja/list_res.cpp

```
#include "list_res.h"
#include <fstream>
#include <iostream>
#include <windows.h>
#include <sstream>
#include <string>
```

### Funkcje

- std::ostream & **operator**<< (std::ostream &s, **List_res** &list)

### 5.6.1 Dokumentacja funkcji

#### operator<<()

```
std::ostream & operator<< (
             std::ostream & s,
              List_res & list)
```

Operator strumieniowy wyjscia wypisujacy zawartosc listy

**Parametry**

| | |
|---|---|
| *s* | Operator strumieniowy wyjscia |
| *list* | Lista do wypisania |

**Zwraca**

Operator strumieniowy wyjscia

## 5.7 Dokumentacja pliku C:/Users/Julia/Documents/PK2/e0f8bf12-gr42-repo/Projekt/rezerwacja/list_res.h

```
#include "list_origin.h"
#include "reservation.h"
#include <string>
```

**Komponenty**

- class **List_res**

**Definicje**

- #define **LIST_RES_H**

### 5.7.1 Dokumentacja definicji

#### LIST_RES_H

```
#define LIST_RES_H
```

## 5.8 list_res.h

**Idź do dokumentacji tego pliku.**

```
00001 #pragma once
00002 #ifndef LIST_RES_H
00003 #define LIST_RES_H
00004
00005 #include "list_origin.h"
00006 #include "reservation.h"
00007 #include <string>
00008
00010 class List_res : public ListOrigin
00011 {
00012 private:
00013     smart2 ptr;
00014     int reservation_number;
00015 public:
00016
00019     int getReservationNumber(); //
00020
00023     void load_from_file(const std::string& name);
00024
00030     bool checkdate(std::string& city, std::string& stringdate);
00031
00034     void getSeat(std::string& chosenSeat);
00035
00038     void addtolist(Reservation& res);
00039
00043     bool removeFromList(int primaryKey);
00044
00046     void display();
00047
00050     void savetofile(const std::string& name);
00051
00055     Date getdate(std::string& stringdate);
00056
00059     List_res(smart2& resptr);
00060
00062     List_res();
00063
00064     friend class List_table;
00065
00070     friend std::ostream& operator«(std::ostream& s, List_res& list);
00071
00074     int getIndexFromUser();
00075 };
00076
00077 #endif
```

## 5.9 Dokumentacja pliku C:/Users/Julia/Documents/PK2/e0f8bf12-gr42-repo/Projekt/rezerwacja/list_table.cpp

```
#include "list_table.h"
#include <fstream>
#include <iostream>
#include <sstream>
#include <string>
#include <cctype>
#include <algorithm>
```

### Funkcje

- std::ostream & **operator**<< (std::ostream &s, std::set< std::string > c)
- std::ostream & **operator**<< (std::ostream &s, **List_table** &list)

### 5.9.1 Dokumentacja funkcji

#### operator<<() [1/2]

```
std::ostream & operator<< (
            std::ostream & s,
             List_table & list)
```

Operator strumieniowy wyjscia sluzacy do wypisania zawartosci listy (stolikow wczytanych)

**Parametry**

| | |
|---|---|
| *s* | Operator strumieniowy wyjscia @list Lista stolikow wczytanych z pliku |

**Zwraca**

Operator strumieniowy wyjscia

#### operator<<() [2/2]

```
std::ostream & operator<< (
            std::ostream & s,
            std::set< std::string > c)
```

Operator strumieniowy wyjscia sluzacy do wypisania zawartosci zbioru miast

**Parametry**

| | |
|---|---|
| *s* | Operator strumieniowy wyjscia |
| *c* | Zbior miast |

**Zwraca**

Operator strumieniowy wyjscia

## 5.10 Dokumentacja pliku C:/Users/Julia/Documents/PK2/e0f8bf12-gr42-repo/Projekt/rezerwacja/list_table.h

```
#include "list_origin.h"
#include "list_res.h"
#include "Table.h"
#include <vector>
#include <set>
#include <string>
#include <iostream>
```

**Komponenty**

- class **List_table**

**Definicje**

- #define **LIST_TABLE_H**

### 5.10.1 Dokumentacja definicji

**LIST_TABLE_H**

```
#define LIST_TABLE_H
```

## 5.11 list_table.h

**Idź do dokumentacji tego pliku.**

```
00001 #pragma once
00003 #ifndef LIST_TABLE_H
00004 #define LIST_TABLE_H
00005
00006 #include "list_origin.h"
00007 #include "list_res.h"
00008 #include "Table.h"
00009 #include <vector>
00010 #include <set>
00011 #include <string>
00012 #include <iostream>
00013
00015 class List_table : public List_res
00016 {
00017 private:
00018     smart ptr;
00019     std::set<std::string> cities;
00020 public:
00021
00024     void load_from_file(const std::string& name);
00025
00027     void display_cities();
00028
00031     void display_seats(std::string& city);
00032
00035     void display_tables(std::string& city);
00036
00043     std::vector<Table>* choose_table(Date& dat, std::string& city, std::string& seats, List_res&
      thelist);
00044
00049     friend std::ostream& operator « (std::ostream& s, std::set<std::string> c);
00050
00055     friend std::ostream& operator « (std::ostream& s, List_table& list);
00056
00060     bool check(std::string& city);
00061
00066     bool check2(std::string& city, std::string& seats);
00067
00070     List_table(smart& tableptr);
00071 };
00072
00073 #endif
```

## 5.12 Dokumentacja pliku C:/Users/Julia/Documents/PK2/e0f8bf12-gr42-repo/Projekt/rezerwacja/main.cpp

```
#include "list_table.h"
#include "list_res.h"
#include <iostream>
#include <sstream>
#include <vector>
```

**Funkcje**

- int **main** ()

### 5.12.1 Dokumentacja funkcji

**main()**

```
int main ()
```

## 5.13 Dokumentacja pliku C:/Users/Julia/Documents/PK2/e0f8bf12-gr42-repo/↩ Projekt/rezerwacja/reservation.cpp

```
#include "reservation.h"
#include <iostream>
```

**Funkcje**

- std::ostream & **operator**<< (std::ostream &s, **Reservation** &res)

### 5.13.1 Dokumentacja funkcji

**operator**<<()

```
std::ostream & operator<< (
            std::ostream & s,
             Reservation & res)
```

Operator strumieniowy wyjscia

**Parametry**

| | |
|---|---|
| *s* | Operator strumieniowy wyjscia |
| *res* | Rezerwacja do wypisania |

**Zwraca**

Operator strumieniowy wyjscia

## 5.14 Dokumentacja pliku C:/Users/Julia/Documents/PK2/e0f8bf12-gr42-repo/↩ Projekt/rezerwacja/reservation.h

```
#include "Table.h"
#include "Date.h"
```

### Komponenty

- class **Reservation**

### Definicje

- #define **RESERVATION_H**

### Definicje typów

- typedef std::shared_ptr< **Reservation** > **smart2**

### 5.14.1 Dokumentacja definicji

#### RESERVATION_H

```
#define RESERVATION_H
```

### 5.14.2 Dokumentacja definicji typów

#### smart2

```
typedef std::shared_ptr< Reservation>  smart2
```

## 5.15 reservation.h

**Idź do dokumentacji tego pliku.**

```
00001
00002 #pragma once
00003 #ifndef RESERVATION_H
00004 #define RESERVATION_H
00005
00006 #include "Table.h"
00007 #include "Date.h"
00008
00010 class Reservation : public Table
00011 {
00012 private:
00013
00014     //static int counter; ///<Zmienna statyczna wyswietlajaca ilosc rezerwacji znajdujacych sie w
      pliku
00015     std::string name;
00016     std::string comments;
00017     Date reservdate;
00018
00019     typedef std::shared_ptr<Reservation> smart2;
00020     smart2 pointer2;
00021
00022     friend class List_res;
00023
00024 public:
00025
00033     Reservation(std::string& c, int& n, int& s, std::string& resname, std::string& com, Date& dat,
      smart2 next2);
00034
00041     Reservation(std::string& c, int& n, int& s, std::string& resname, std::string& com, Date& dat);
00042
00045     Reservation(const Reservation& r);
00046
00048     ~Reservation();
00049
```

```
00051     void display();
00052
00053     friend class List_table;
00054
00059     friend std::ostream& operator « (std::ostream& s, List_res& list);
00060
00065     friend std::ostream& operator « (std::ostream& s, Reservation& res);
00066 };
00067
00068 //uzywane do uproszczenia zapisu typu std::shared_ptr<Reservation>, lepsza czytelnosc
00069 typedef std::shared_ptr<Reservation> smart2;
00070
00071 #endif
```

## 5.16 Dokumentacja pliku C:/Users/Julia/Documents/PK2/e0f8bf12-gr42-repo/Projekt/rezerwacja/table.cpp

```
#include "table.h"
#include <iostream>
```

### Funkcje

- std::ostream & **operator**<< (std::ostream &s, **Table** &t)

### 5.16.1 Dokumentacja funkcji

#### operator<<()

```
std::ostream & operator<< (
             std::ostream & s,
              Table & t)
```

Operator strumieniowy wyjscia

**Parametry**

| | |
|---|---|
| *s* | operator strumieniowy wyjscia |
| *t* | Stolik ktorego dane zostana wypisane |

**Zwraca**

Operator strumieniowy wyjscia

## 5.17 Dokumentacja pliku C:/Users/Julia/Documents/PK2/e0f8bf12-gr42-repo/Projekt/rezerwacja/table.h

```
#include <string>
#include <memory>
#include "table_origin.h"
```

### Komponenty

- class **Table**

### Definicje

- #define **TABLE_H**

### Definicje typów

- typedef std::shared_ptr< **Table** > **smart**

### 5.17.1 Dokumentacja definicji

### TABLE_H

```
#define TABLE_H
```

### 5.17.2 Dokumentacja definicji typów

### smart

```
typedef std::shared_ptr< Table>  smart
```

## 5.18 table.h

**Idź do dokumentacji tego pliku.**

```
00001
00002 #pragma once
00003 #ifndef TABLE_H
00004 #define TABLE_H
00005
00006 #include <string>
00007 #include <memory>
00008 #include "table_origin.h"
00009
00011 class Table : public TableOrigin
00012 {
00013 public:
00014     std::string city;
00015     int number;
00016     int seats;
00017
00018     typedef std::shared_ptr<Table> smart;
00019
00025     Table(std::string& c, int& n, int& s, smart& next);
00026
00027     /***Konstruktor trojargumentowy
00028     @param c Miasto
00029     @param n Numer stolika
00030     @param s Liczba miejsc przy stoliku */
00031     Table(std::string& c, int& n, int& s);
00032
00034     Table();
00035
00037     void display();
00038
00039 private:
00040     smart pointer;
00041
00042     friend class List_table;
00043
00048     friend std::ostream& operator«(std::ostream& s, List_table& list);
00049
00054     friend std::ostream& operator«(std::ostream& s, Table& t);
00055 };
00056
00057 typedef std::shared_ptr<Table> smart;
00058
00059 #endif
```

## 5.19 Dokumentacja pliku C:/Users/Julia/Documents/PK2/e0f8bf12-gr42-repo/↩ Projekt/rezerwacja/table_origin.h

### Komponenty

- class **TableOrigin**

### Definicje

- #define **TABLE_ORIGIN_H**

### 5.19.1 Dokumentacja definicji

#### TABLE_ORIGIN_H

```
#define TABLE_ORIGIN_H
```

## 5.20 table_origin.h

**Idź do dokumentacji tego pliku.**

```
00001
00002 #pragma once
00003 #ifndef TABLE_ORIGIN_H
00004 #define TABLE_ORIGIN_H
00005
00007 class TableOrigin
00008 {
00009 public:
00010     virtual void display() = 0;
00011 };
00012 #endif
```