

## 1.Opis projektu

Projekt “SOR” to konsolowy program do obsługi wirtualnego SOR-u. Umożliwia użytkownikowi wejście w rolę pracownika SORu. Użytkownik dostaje kontrolę nad swoimi danymi, bezpieczną możliwość ich edycji; bez powodowania kolizji w działaniu innych funkcji. Może również symulować “obsługę wirtualnego SORu”.

---

## 2. Baza danych

Bazą danych w projekcie są pliki wejściowe. W celu poprawnego uruchomienia programu przekazane zostały trzy pliki wejściowe.

1. Plik z tygodniowym grafikiem pracowników
2. Plik z hasłami pracowników do systemu
3. Plik z asortymentem

Ad.1 W pierwszym wierszu prawidłowego pliku (1) zawiera się informacja o dacie pierwszego dnia tygodnia (dzień i miesiąc).

początek tygodnia: 21.06

Wiersze zawierające grafik powinny wyglądać następująco:

```
Imię Nazwisko stanowisko  grafik_poniedziałek ... grafik_niedziela
```

Gdzie grafik na dzień składa się z godziny rozpoczęcia pracy oraz godziny zakończenia pracy, oddzielonych przerwą. Poprawny format godziny to GG:MM.

Wolne powinno zostać oznaczone przez “--:--”, zarówno dla godziny rozpoczęcia pracy, jak i zakończenia. Wiersz musi zawierać grafik na wszystkie dni tygodnia

np.:

---

Andrzej Falkowicz	lekarz	00:00 09:00	00:00 09:15	00:00 12:00	--:-- --:--	00:00 10:00	00:00 09:00	00:00 08:00
-------------------	--------	-------------	-------------	-------------	-------------	-------------	-------------	-------------

Ad.2 Poprawny plik z hasłami w każdym swoim wierszu zawiera ID (numer w bazie danych z pracownikami; jest generowany automatycznie) pracownika oraz jego hasło. Przed hasłami dla lekarzy powinien się znaleźć etykieta “Lekarze:”, a przed hasłami dla pielęgniarek etykieta “Pielęgniarki:”.

np:

```
24 Jestem!Lekarzem23??
```

Plik może zawierać puste wiersze i białe znaki.

Jeżeli dla któregoś z lekarzy hasło nie będzie znajdowało się w pliku, program wygeneruje je automatycznie.

Ad.3 Poprawny plik z asortymentem zawiera w pierwszych dwóch liniach kolejno: informację o liczbie miejsc w poczekalni, oraz liczbie miejsc na sali SORu (liczbie łóżek), oddzielonych przerwą. Wiersze te powinny zawierać słowo opisujące do czego ilość się odnosi.

np:

```
1.| Poczekalnia: 12
```

```
2.| Miejsca: 34
```

Pozostałe wiersze powinny zawierać nazwę części asortymentu, przerwę oraz liczbę danej części, wyrażonej naturalną liczbą

np

```
3.| bandaż 234
```

---

### 3. Interfejs

Wynikiem pomyślnego przekazania prawidłowych plików jest wyświetlenie menu głównego programu:

1. Jestem lekarzem
2. Jestem pielęgniarką/rzem

- Ad.1 Po wybraniu tej opcji, użytkownik loguje się do systemu podając zgodne ID oraz hasło, które są porównywane z danymi w bazie danych. Prawidłowe zalogowanie umożliwia użytkownikowi wybranie jednej z czterech opcji.



---

“Wróć do menu głównego” godzina ustawiana jest automatycznie na 23:59 danego dnia.

→1. Po wybraniu tej opcji wyświetlane należy podać godzinę, imię i nazwisko poszkodowanego, stan poszkodowanego (od 1 do 3) oraz krótką notatkę. Po wprowadzeniu tych informacji sprawdzane jest czy jest możliwość przyjęcia poszkodowanego (czy jest wystarczająca liczba personelu by móc się nim zająć, asortymentu oraz czy jest miejsce). Jeżeli nie można poszkodowanego przyjąć wyświetlana jest informacja za ile będzie taka możliwość oraz opcja oddelegowania pacjenta do innego szpitala. Jeżeli przyjęcie jest możliwe, można wybrać konkretnego pracownika (jeżeli jest dostępny) lub go nie wybierać; wtedy program domyślnie przydziela mu dostępny personel.

→2. Jest to opcja wyświetlająca “stan” SOR-u. Stan jest określany na podstawie aktualnej liczbie wolnych miejsc na sali SORu oraz poczekalni, ilości dostępnych lekarzy i pielęgniarek oraz poziomu zaopatrzenia.

Stan jest w skali od 1-4 (“prawidłowy”, “poprawny”, “nieprawidłowy”, “krytyczny”). Po wybraniu tej opcji wyświetlany jest ów stan, z dokładniejszymi parametrami, biorącymi udział w wyznaczaniu stanu. Wyświetlane jest proponowane rozwiązanie by polepszyć stan SOR-u (np. gdy brakuje personelu proponowane jest dodanie lekarza/pielęgniarki).

→3. Jest to opcja umożliwiająca dodania “zewnętrznego pracownika” (czyli spoza bazy danych) np z innego oddziału. Ta funkcja może być przydatna, gdy stan na SOR-ze jest “krytyczny” i użytkownik zechce powołać innego pracownika do pomocy. Po wybraniu tej opcji należy podać godzinę, imię i nazwisko pracownika, jego stanowisko (lekarz/pielęgniarka) oraz godziny pracy w aktualnym dniu.

→4. Jest to opcja uzupełnienia asortymentu. Potrzebna gdy poziom asortymentu jest zbyt niski. Należy podać godzinę oraz o ile procent chcemy podnieść poziom zaopatrzenia.

→5. Po podaniu godziny, generowany jest raport zawierający aktualną godzinę, stan, listę pacjentów z aktualnego dnia oraz szczegółową listę zaopatrzenia.

```
*RAPORT21.0603.12 — Notatnik

Plik  Edycja  Format  Widok  Pomoc

STAN: POPRAWNY

LICZBA WOLNYCH LOZEK: 9   LICZBA ZAJETYCH MIEJSC: 1
LICZBA WOLNYCH MIEJSC W POCZEKALNI: 5
PROCENT WSZYSTKICH DOSTEPNYCH PRACOWNIKOW: 15%
LICZBA DOSTEPNYCH LEKARZY: 2   LICZBA DOSTEPNYCH PIELEGNIAREK: 1
PROCENT ZAOPATRZENIA: 85.1007

Pacjenci z dzisiaj:
1. Adrianna Krysiak Opuscil szpital stan poszkodowania: 1 notatka: skrecona kostka
2. Adam Sandler Opuscil szpital stan poszkodowania: 3 notatka: wylew
3. Magda Tkacz Opuscil szpital stan poszkodowania: 2 notatka: porod
4. Rafal Loska Na sali stan poszkodowania: 3 notatka: potrzenie na jezdni
Poziom asortymentu:
    bandaz      ilosc: 170 procent normy: 85
    paracetamol ilosc: 510 procent normy: 85
    ibuprofen   ilosc: 154 procent normy: 85.5556
    glukometr   ilosc: 85 procent normy: 85
    plaster      ilosc: 1064 procent normy: 85.12
    spirytus    ilosc: 384 procent normy: 85.3333
    wenflon     ilosc: 850 procent normy: 85
    igla        ilosc: 850 procent normy: 85
    kroplowka   ilosc: 425 procent normy: 85
    rekawiczki  ilosc: 425 procent normy: 85
    maseczka    ilosc: 639 procent normy: 85.2
    dren        ilosc: 170 procent normy: 85
```

Obraz. Przykładowy raport wygenerowany przez program.

→6. Wybranie tej opcji jest jednoznaczne z zakończeniem dnia (godzina zostaje ustawiona na 23:59). Użytkownik jest kierowany do menu głównego.

Ad. 4

Dostawca może wyświetlić aktualny stan asortymentu, oraz uzupełnić dany asortyment poprzez wpisywanie ilości uzupełnianego asortymentu.

Ad. 5

Wybranie tej opcji skutkuje zamknięciem programu.

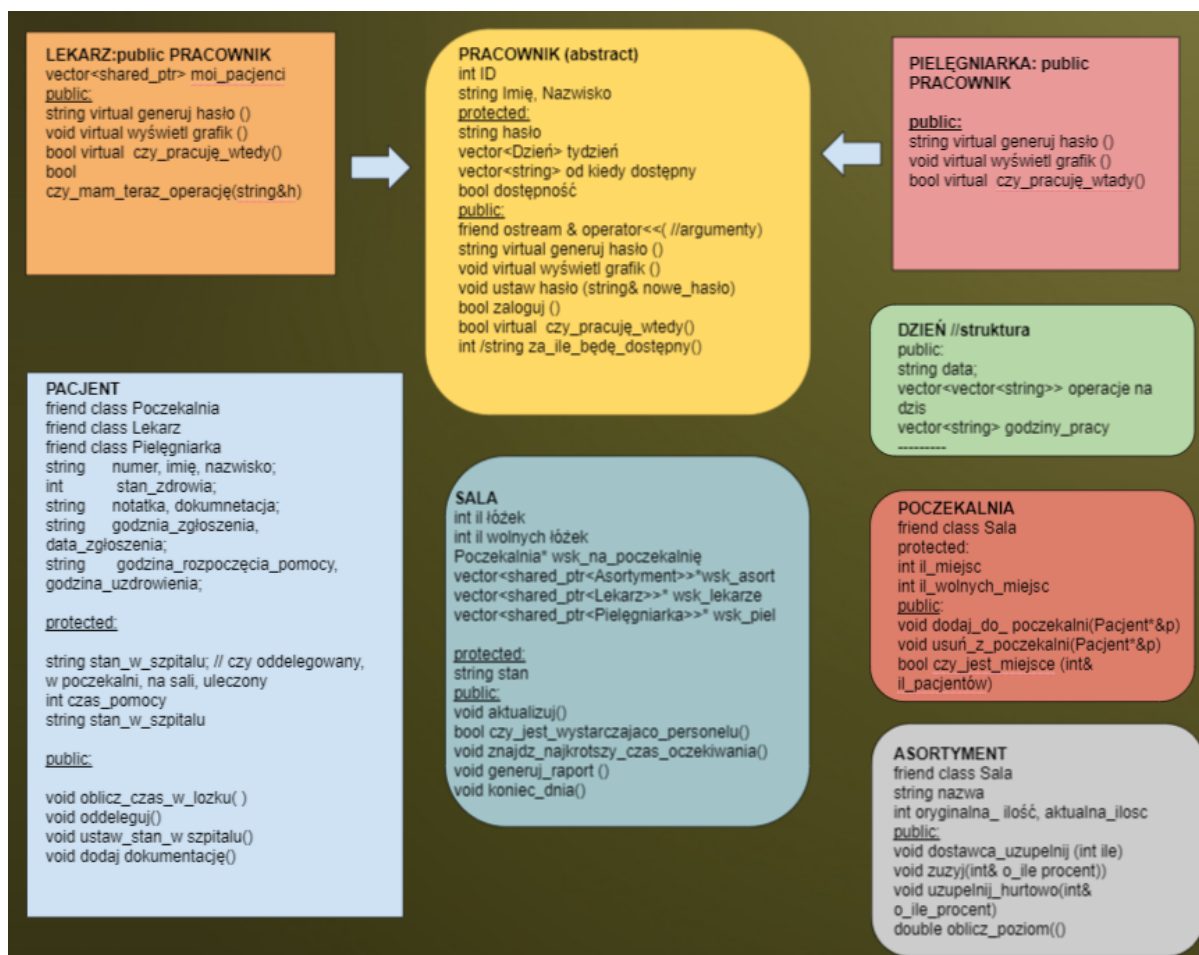
## 4. Specyfikacja wewnętrzna

Program został zrealizowany zgodnie z paradygmatem strukturalnym.

W programie rozdzielono interfejs (komunikacje z użytkownikiem) od logiki programu. Każdy obiekt w programie służy jako model abstrakcyjnego "wykonawcy", który może wykonywać pracę, opisywać i zmieniać swój stan, oraz komunikować się z innymi obiektami w programie.

### 4.1 Diagram

Klasa pracownik to klasa abstrakcyjna. "Dzień" to struktura. W diagramie są przedstawione najważniejsze metody oraz atrybuty; konstruktory i destruktory zostały pominięte.

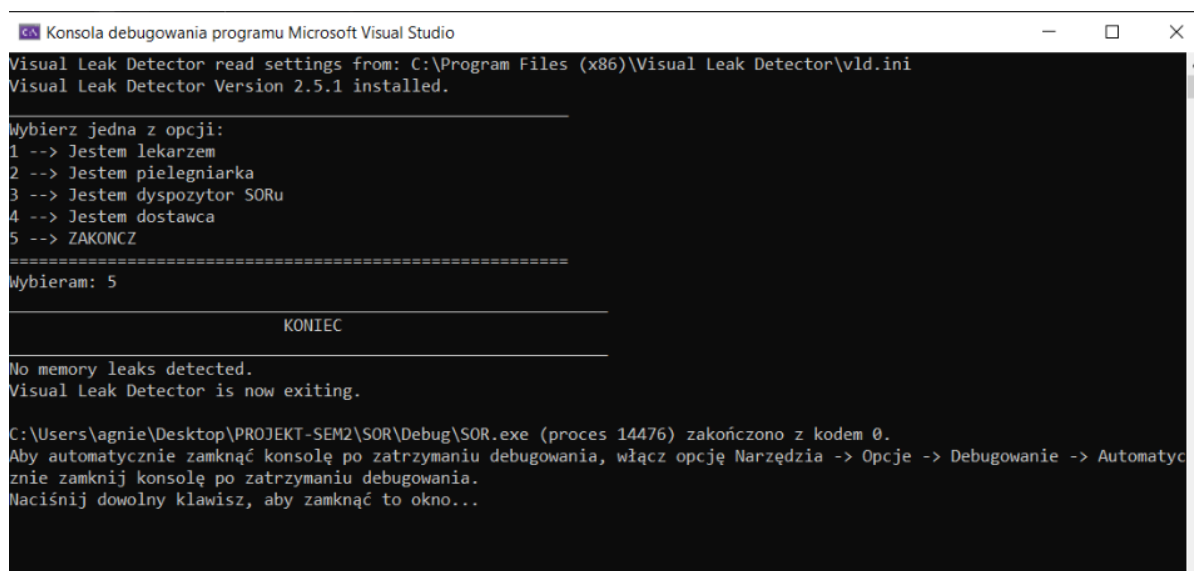


### 4.2 Szczegółowy opis typów i funkcji

Szczegółowy opis typów i funkcji zawarty jest w załączniku.

## 5. Testowanie

Program został przetestowany na różnych danych wprowadzanych przez użytkownika na zbiorach poprawnych i typowych, na zbiorach poprawnych, ale nietypowych (np gdy użytkownik proszony o podanie dwóch godzin podaje je w niepoprawnej kolejności) wreszcie na zbiorach niepoprawnych (np gdy użytkownik proszony o wpisanie liczby wpisuje wyraz). Podawanie innych niż poprawnych i typowych nie powoduje błędu, wyświetlana jest informacja jakie dane powinien użytkownik powinien podać i jest możliwość ponownego wpisania już poprawnych danych przez użytkownika. Program został przetestowany pod kątem wycieków pamięci przez Visual Leak Detector.



```
Konsola debugowania programu Microsoft Visual Studio
Visual Leak Detector read settings from: C:\Program Files (x86)\Visual Leak Detector\vld.ini
Visual Leak Detector Version 2.5.1 installed.

Wybierz jedna z opcji:
1 --> Jestem lekarzem
2 --> Jestem pielęgniarka
3 --> Jestem dyspozytor SORu
4 --> Jestem dostawca
5 --> ZAKONCZ
=====
Wybieram: 5

KONIEC

No memory leaks detected.
Visual Leak Detector is now exiting.

C:\Users\agnie\Desktop\PROJEKT-SEM2\SOR\Debug\SOR.exe (proces 14476) zakończono z kodem 0.
Aby automatycznie zamknąć konsolę po zatrzymaniu debugowania, włącz opcję Narzędzia -> Opcje -> Debugowanie -> Automatycznie zamknij konsolę po zatrzymaniu debugowania.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

## 6. Wnioski

Program SOR nie jest skomplikowanym programem, chociaż wymaga dobrego przemyślenia logistyki programu. Najbardziej wymagające okazało się zachowanie poprawności czasowej, co nie pozwala na modyfikacje danych “cofając się w czasie”. Trudności sprawiało znalezienie rozwiązania problemu poczekalni i sprawdzanie godzin pracy pracowników.



Dodatek

Szczegółowy opis typów  
i funkcji

SOR

Generated by Doxygen 1.8.20



# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Asortyment . . . . .	??
dzien . . . . .	??
Pacjent . . . . .	??
Poczekalnia . . . . .	??
Pracownik . . . . .	??
Lekarz . . . . .	??
Pielegniarka . . . . .	??
Sala . . . . .	??



## Chapter 2

# Class Index

### 2.1 Class List

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

Asortyment	??
dzien	??
Lekarz	??
Pacjent	??
Pielegniarka	??
Poczekalnia	??
Pracownik	??
Sala	??



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

<a href="#">Funkcje.cpp</a>	.....	??
<a href="#">Funkcje.h</a>	.....	??
<a href="#">Pacjent.cpp</a>	.....	??
<a href="#">Pacjent.h</a>	.....	??
<a href="#">Pracownicy.cpp</a>	.....	??
<a href="#">Pracownicy.h</a>	.....	??
<a href="#">resource.h</a>	.....	??
<a href="#">Sala.cpp</a>	.....	??
<a href="#">Sala.h</a>	.....	??
<a href="#">SOR.cpp</a>	.....	??





## Chapter 4

# Class Documentation

### 4.1 Asortyment Class Reference

```
#include <Sala.h>
```

#### Public Member Functions

- [Asortyment](#) (string &\_nazwa, int \_ile)
- [~Asortyment](#) ()
- double [oblicz\\_poziom\\_proc](#) ()  
*Zwraca obliczona wartosc procentowego aktualnego zaopatrzenia w stosunku do oryginalnej ilosci.*
- void [dostawca\\_uzupelnij](#) (int &ile)  
*Podnosi aktualna ilosc asortymentu o przekazany parametr ile.*
- void [uzupelnij\\_hurtowo](#) (int &proc)  
*Podnosi aktualna ilosc asortymentu procentowo o parametr proc w stosunku do oryginalnej ilosci.*
- void [zuzyj](#) (int &o\_ile\_proc)  
*Obniza aktualna ilosc asortymentu procentowo o parametr proc w stosunku do oryginalnej ilosci.*

#### Protected Attributes

- string [nazwa](#)  
*nazwa asortymentu, odczytana z pliku*
- int [oryginalna\\_ilosc](#)  
*liczba asortymentu odczytana z pliku*
- int [aktualna\\_ilosc](#)  
*aktualna ilosc asortymentu*

#### Friends

- class [Sala](#)
- ostream & [operator<<](#) (ostream &s, [Asortyment](#) &asortyment)

## 4.1.1 Constructor & Destructor Documentation

### 4.1.1.1 Asortyment()

```
Asortyment::Asortyment (
    string & _nazwa,
    int _ile )
```

### 4.1.1.2 ~Asortyment()

```
Asortyment::~~Asortyment ( )
```

## 4.1.2 Member Function Documentation

### 4.1.2.1 dostawca\_uzupelnij()

```
void Asortyment::dostawca_uzupelnij (
    int & ile )
```

Podnosi aktualna ilosc asortymentu o przekazany parametr ile.

#### Parameters

<i>ile-</i>	ilosc o jaka dostawac uzupelnia dany element asortymentu
-------------	--

### 4.1.2.2 oblicz\_poziom\_proc()

```
double Asortyment::oblicz_poziom_proc ( )
```

Zwraca obliczona wartosc procentowego aktualnego zaopatrzenia w stosunku do oryginalnej ilosci.

#### Returns

procent zapatrzienia

#### 4.1.2.3 uzupełnij\_hurtowo()

```
void Asortyment::uzupełnij_hurtowo (
    int & proc )
```

Podnosi aktualna ilosc asortymentu procentowo o parametr *proc* w stosunku do oryginalnej ilosci.

##### Parameters

<i>proc</i> -	liczba rowna procentowi o jaki uzytkownik chce podniesc poziom zaopatrzenia
---------------	---

#### 4.1.2.4 zuzyj()

```
void Asortyment::zuzyj (
    int & o_ile_proc )
```

Obniza aktualna ilosc asortymentu procentowo o parametr *proc* w stosunku do oryginalnej ilosci.

##### Parameters

<i>proc</i> -	liczba rowna procentowi asortymentu jaki zuzywa pacjent
---------------	---

### 4.1.3 Friends And Related Function Documentation

#### 4.1.3.1 operator<<

```
ostream& operator<< (
    ostream & s,
    Asortyment & asortyment ) [friend]
```

#### 4.1.3.2 Sala

```
friend class Sala [friend]
```

### 4.1.4 Member Data Documentation

#### 4.1.4.1 aktualna\_ilosc

```
int Asortyment::aktualna_ilosc [protected]
```

aktualna ilosc asortymentu

#### 4.1.4.2 nazwa

```
string Asortyment::nazwa [protected]
```

nazwa asortymentu, odczytana z pliku

#### 4.1.4.3 oryginalna\_ilosc

```
int Asortyment::oryginalna_ilosc [protected]
```

liczba asortymentu odczytana z pliku

The documentation for this class was generated from the following files:

- [Sala.h](#)
- [Sala.cpp](#)

## 4.2 dzien Struct Reference

```
#include <Pracownicy.h>
```

### Public Member Functions

- [dzien](#) (string &\_data, vector< string > \_godziny\_pracy)
- [dzien](#) (string &\_data)
- [~dzien](#) ()

### Public Attributes

- string [data](#)
- vector< vector< string > > [operacje\\_na\\_dzis](#)  
*Kontener składający się z nazwy, godziny rozpoczęcia operacji i zakończenia operacji.*
- vector< string > [godziny\\_pracy](#)  
*Kontener składający się z godziny rozpoczęcia pracy i zakończenia pracy.*

## 4.2.1 Constructor & Destructor Documentation

### 4.2.1.1 dzien() [1/2]

```
dzien::dzien (
    string & _data,
    vector< string > _godziny_pracy )
```

### 4.2.1.2 dzien() [2/2]

```
dzien::dzien (
    string & _data )
```

### 4.2.1.3 ~dzien()

```
dzien::~~dzien ( )
```

## 4.2.2 Member Data Documentation

### 4.2.2.1 data

```
string dzien::data
```

### 4.2.2.2 godziny\_pracy

```
vector<string> dzien::godziny_pracy
```

Kontener składający się z godziny rozpoczęcia pracy i zakończenia pracy.

### 4.2.2.3 operacje\_na\_dzis

```
vector<vector<string > > dzien::operacje_na_dzis
```

Kontener składający się z nazwy, godziny rozpoczęcia operacji i zakończenia operacji.

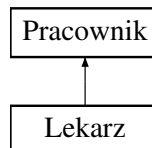
The documentation for this struct was generated from the following files:

- [Pracownicy.h](#)
- [Pracownicy.cpp](#)

## 4.3 Lekarz Class Reference

```
#include <Pracownicy.h>
```

Inheritance diagram for Lekarz:



### Public Member Functions

- [Lekarz](#) ()
- [Lekarz](#) (int &\_ID, const string &\_imie, const string &\_nazwisko, vector< [dzien](#) > &\_tydzien)
- [~Lekarz](#) ()
- virtual string [generuj\\_haslo](#) ()  
*Generuje domyslnie haslo dla obiektu typu [Lekarz](#) Wywolywana przez konstruktor [Lekarz::Lekarz\(\)](#).*
- virtual void [wyswietl\\_grafik](#) ()  
*Wyswietla grafik Wyswietla godziny pracy lekarza oraz zaplanowane operacja (nazwe operacji, godzinie rozpoczecia i zakonczenia operacji)*
- virtual void [zmien\\_grafik](#) (string &aktualna\_data)  
*Wprowadza zmiany w godzinach pracy [Lekarz](#) proszony jest o wprowadzenie daty dnia, na ktory zmienia godziny pracy, oraz nowe godziny pracy.*
- void [dodaj\\_operacje](#) (string &data, string &h)  
*Dodaje do konteneru z operacjami operacje Mozliwe jest dodanie operacji dzien po podanej dacie; nie w ten sam dzien.*
- bool [czy\\_mam\\_operacje](#) (string &data, string &h)  
*Funkcja sprawdza czy o podanej godzinie w podany dzien jest zaplanowana operacja.*
- virtual void [aktualizacja](#) (string &data, string &h, string &h\_uzdrowienia)  
*Aktualizuje [Lekarz::dostepnosc](#) oraz [Lekarz::godz\\_od\\_ktorej\\_dostepny](#).*
- vector< int > [zwroc\\_moich\\_pacjentow](#) ()

### Friends

- class [Sala](#)

## Additional Inherited Members

### 4.3.1 Constructor & Destructor Documentation

#### 4.3.1.1 Lekarz() [1/2]

```
Lekarz::Lekarz ( )
```

#### 4.3.1.2 Lekarz() [2/2]

```
Lekarz::Lekarz (
    int & _ID,
    const string & _imie,
    const string & _nazwisko,
    vector< dzien > & _tydzien )
```

#### 4.3.1.3 ~Lekarz()

```
Lekarz::~~Lekarz ( )
```

### 4.3.2 Member Function Documentation

#### 4.3.2.1 aktualizacja()

```
void Lekarz::aktualizacja (
    string & data,
    string & h,
    string & h_uzdrowienia ) [virtual]
```

Aktualizuje [Lekarz::dostepnosc](#) oraz [Lekarz::godz\\_od\\_ktorej\\_dostepny](#).

Jezeli lekarz w podanych godzinach (zarowno godziny wyzdrowienia i przyjecia) pracuje, nie ma zaplanowanej operacji i nie leczy innego pacjenta to jego [Lekarz::dostepnosc](#) jest ustawiana na 1.

#### Parameters

<i>data-</i>	data przyjecia pacjenta
<i>h-</i>	godzina o ktorej pacjent wchodzi na SOR
<i>h_uzdrowienia-</i>	godzina o ktorej pacjent wyzdrowieje



Implements [Pracownik](#).

#### 4.3.2.2 czy\_mam\_operacje()

```
bool Lekarz::czy_mam_operacje (
    string & data,
    string & h )
```

Funkcja sprawdza czy o podanej godzinie w podany dzien jest zaplanowana operacja.

##### Parameters

<i>data-</i>	data sprawdzanego dnia
<i>h-sprawdzana</i>	godzina

##### Returns

true jezeli o podanej godzinie w podany dzien jest zaplanowana operacja

#### 4.3.2.3 dodaj\_operacje()

```
void Lekarz::dodaj_operacje (
    string & data,
    string & h )
```

Dodaje do konteneru z operacjami operacje Mozliwe jest dodanie operacji dzien po podanej dacie; nie w ten sam dzien.

Nie jest mozliwe nachodzenie sie dwoch operacji na siebie.

##### Parameters

<i>data-</i>	data wprowadzanych zmian
<i>h-</i>	godzina wprowadzanych zmian

#### 4.3.2.4 generuj\_haslo()

```
string Lekarz::generuj_haslo ( ) [virtual]
```

Generuje domyslne haslo dla obiektu typu [Lekarz](#) Wywoływana przez konstruktor [Lekarz::Lekarz\(\)](#).

Gdy w pliku "Hasla.txt" nie znajduje sie ID danego lekarza haslo wygenerowane przez ta funkcje jest haslem lekarza. Haslo sklada sie ze slowa "Lekarz" i ID lekarza.

#### Returns

wygenerowane haslo

Implements [Pracownik](#).

#### 4.3.2.5 wyswietl\_grafik()

```
void Lekarz::wyswietl_grafik ( ) [virtual]
```

Wyswietla grafik Wyswietla godziny pracy lekarza oraz zaplanowane operacja (nazwe operacji, godzinie rozpoczecia i zakonczenia operacji)

Implements [Pracownik](#).

#### 4.3.2.6 zmien\_grafik()

```
void Lekarz::zmien_grafik (
    string & aktualna_data ) [virtual]
```

Wprowadza zmiany w godzinach pracy [Lekarz](#) proszony jest o wprowadzenie daty dnia, na ktory zmienia godziny pracy, oraz nowe godziny pracy.

Sprawdzana jest poprawnoŹ podanej daty i godziny (format, zakres tygodnia)

Implements [Pracownik](#).

#### 4.3.2.7 zwroc\_moich\_pacjentow()

```
vector< int > Lekarz::zwroc_moich_pacjentow ( )
```

#### Returns

wektor o tych samych wartosciach co [Lekarz::nr\\_moich\\_pacjentow](#).

### 4.3.3 Friends And Related Function Documentation

#### 4.3.3.1 Sala

```
friend class Sala [friend]
```

The documentation for this class was generated from the following files:

- [Pracownicy.h](#)
- [Pracownicy.cpp](#)

## 4.4 Pacjent Class Reference

```
#include <Pacjent.h>
```

### Public Member Functions

- [Pacjent](#) (string &\_hzgl, string &\_data\_zgl, int &\_nr, string &\_imie, string &\_nazwisko, int &\_stan\_zdrowia, string &\_notatka)
- [~Pacjent](#) ()
- void [oblicz\\_zapotrzebowanie\\_na\\_personel](#) ()  
*Na podstawie wpisanych przez uzytkownika danych przy zgloszeniu ustawia wartosci 'potrzebni\_lekarze', 'potrzebne\_pielegniarki' Gdy 'stan\_zdrowia' jest rowny 1 to 'potrzebni\_lekarze'=0, 'potrzebne\_pielegniarki'=1 Gdy 'stan\_zdrowia' jest rowny 2 to 'potrzebni\_lekarze'=1, 'potrzebne\_pielegniarki'=0 Gdy 'stan\_zdrowia' jest rowny 3 to 'potrzebni\_lekarze'=1, 'potrzebne\_pielegniarki'=1.*
- void [oblicz\\_czas\\_w\\_lozku](#) ()  
*Na podstawie wpisanych przez uzytkownika danych przy zgloszeniu ustawia wartosc 'czas\_zajecia\_lozka' Gdy 'stan\_zdrowia' jest rowny 1 to wartosc ustawiana jest rowna 10 (minut).*
- void [oddeleguj](#) ()  
*Oddelegowuje pacjenta Wywoływana gdy nie ma mozliwosci przyjecia pacjenta na sor ani poczekalni, lub gdy uzytkownik wybierze opcje oddeleguj zamiast poczekalni Ustawia 'stan\_w szpitalu' na "Oddelegowany".*
- void [ustaw\\_koniec\\_pomocy](#) ()  
*Automatycznie ustawia 'godzina\_uzdrowienia' Wywoływana w konstruktorze, ustawia 'godzina\_uzdrowienia' na --, Gdy pacjent nie idzie do poczekalni; poprzez dodanie do 'godzina\_uzdrowienia' wartosci atrybutu 'czas\_zajecia\_lozka';.*
- void [ustaw\\_koniec\\_pomocy](#) (string &h)  
*Przekazana godzinie ustawia jako atrybut 'godzina\_uzdrowienia'.*
- int [get\\_stan\\_zdrowia](#) ()  
*Zwraca atrybut "stan\_zdrowia".*
- void [ustaw\\_start\\_pomocy](#) (string &h)  
*Przekazana godzinie ustawia jako atrybut 'godzina\_rozpoczecia\_pomocy'.*
- void [ustaw\\_stan\\_w\\_szpitalu](#) (string nazwa)  
*Przekazana nazwa jest ustawiana jako atrybut "stan\_w\_szpitalu" przekazywane nazwy to "Oddelegowany", "W poczekalni", "Na sali" oraz "Opuscil szpital".*
- int [jaki\\_mam\\_numer](#) ()  
*Zwraca numer pacjenta.*
- void [dodaj\\_dokumentacje](#) ()  
*Funkcja generuje dokumentacje do pliku pacjenta W pierwszej kolejnosci ustawiany jest atrybut "dokumentacja" na "Uzupełniona".*

### Public Attributes

- int [potrzebni\\_lekarze](#)  
*Liczba potrzebnych lekarzy do wyleczenia pacjenta.*
- int [potrzebne\\_pielegniarki](#)  
*Liczba pielegniarek lekarzy do wyleczenia pacjenta.*
- int [czas\\_zajecia\\_lozka](#)  
*Czas (wyrazany w minutach) zajecia lozka na SORze; czas przebywania na SORze.*

### Private Member Functions

- string friend [dodaj\\_minuty](#) (string &godz, int &minuty)

## Private Attributes

- int [numer](#)  
*Jest to numer zgłoszenia każdego pacjenta.*
- string [imie](#)  
*Podawane przy zgłoszeniu przez użytkownika.*
- string [nazwisko](#)  
*Podawane przy zgłoszeniu przez użytkownika.*
- int [stan\\_zdrowia](#)  
*Podawane przy zgłoszeniu przez użytkownika.*
- string [notatka](#)  
*Podawana przy zgłoszeniu przez użytkownika.*
- string [dokumentacja](#)  
*Oznacza stan dokumentacji pacjenta.*
- string [godzina\\_zgloszenia](#)  
*Podawana przy zgłoszeniu przez użytkownika.*
- string [data\\_zgloszenia](#)  
*Podawana przy zgłoszeniu przez użytkownika.*
- string [godzina\\_rozpoczecia\\_pomocy](#)  
*< Ustawiana za pomocą funkcji 'void [ustaw\\_start\\_pomocy\(string& h\)](#)'.*
- string [godzina\\_uzdrowienia](#)  
*Ustawiana za pomocą funkcji 'void [ustaw\\_koniec\\_pomocy\(\)](#)'.*
- string [stan\\_w\\_szpitalu](#)  
*Jest informacja gdzie aktualnie przebywa pacjent.*

## Friends

- class [Lekarz](#)
- class [Pracownik](#)
- class [Pielęgniarka](#)
- class [Sala](#)
- ostream & [operator<<](#) (ostream &s, [Pacjent](#) &pacjent)

## 4.4.1 Constructor & Destructor Documentation

### 4.4.1.1 Pacjent()

```
Pacjent::Pacjent (
    string & _hzgl,
    string & _data_zgl,
    int & _nr,
    string & _imie,
    string & _nazwisko,
    int & _stan_zdrowia,
    string & _notatka )
```

#### 4.4.1.2 ~Pacjent()

```
Pacjent::~~Pacjent ( )
```

### 4.4.2 Member Function Documentation

#### 4.4.2.1 dodaj\_dokumentacje()

```
void Pacjent::dodaj_dokumentacje ( )
```

Funkcja generuje dokumentacje do pliku pacjenta W pierwszej kolejnosci ustawiany jest atrybut "dokumentacja" na "Uzupelniona".

Nastepnie uzytkownik jest proszony o odpowiedzenie na kilka pytan; stanowią one dokumentacje. Skonczona dokumentacja jest generowana do pliku wraz z atrybutami pacjenta, takimi jak: numer, imie, nazwisko, stan\_zdrowia, notatka, godzina\_zgloszenia, data\_zgloszenia, godzina\_roz poczenia\_pomocy, godzina\_uzdrowienia. Nazwa pliku jest generowana automatycznie sklada sie z numeru pacjenta oraz jego nazwiska.

#### 4.4.2.2 dodaj\_minuty()

```
string friend Pacjent::dodaj_minuty (
    string & godz,
    int & minuty ) [private]
```

#### 4.4.2.3 get\_stan\_zdrowia()

```
int Pacjent::get_stan_zdrowia ( )
```

Zwraca atrybut "stan\_zdrowia".

##### Returns

stan\_zdrowia

#### 4.4.2.4 jaki\_mam\_numer()

```
int Pacjent::jaki_mam_numer ( )
```

Zwraca numer pacjenta.

##### Returns

numer

#### 4.4.2.5 oblicz\_czas\_w\_lozku()

```
void Pacjent::oblicz_czas_w_lozku ( )
```

Na podstawie wpisanych przez użytkownika danych przy zgłoszeniu ustawia wartosc 'czas\_zajecia\_lozka' Gdy 'stan\_zdrowia' jest rowny 1 to wartosc ustawiana jest rowna 10 (minut).

Gdy 'stan\_zdrowia' jest rowny 2 to wartosc ustawiana jest rowna 15 (minut). Gdy 'stan\_zdrowia' jest rowny 3 to wartosc ustawiana jest rowna 22 (minuty).

#### 4.4.2.6 oblicz\_zapotrzebowanie\_na\_personel()

```
void Pacjent::oblicz_zapotrzebowanie_na_personel ( )
```

Na podstawie wpisanych przez użytkownika danych przy zgłoszeniu ustawia wartosci 'potrzebni\_lekarze', 'potrzebne\_pielegniarki' Gdy 'stan\_zdrowia' jest rowny 1 to 'potrzebni\_lekarze'=0, 'potrzebne\_pielegniarki'=1 Gdy 'stan\_zdrowia' jest rowny 2 to 'potrzebni\_lekarze'=1, 'potrzebne\_pielegniarki'=0 Gdy 'stan\_zdrowia' jest rowny 3 to 'potrzebni\_lekarze'=1, 'potrzebne\_pielegniarki'=1.

#### 4.4.2.7 oddeleguj()

```
void Pacjent::oddeleguj ( )
```

Oddelegowuje pacjenta Wywoływana gdy nie ma mozliwosci przyjecia pacjenta na sor ani poczekalni, lub gdy uzytkownik wybierze opcje oddeleguj zamiast poczekalni Ustawia 'stan\_w szpitalu' na "Oddelegowany".

#### 4.4.2.8 ustaw\_koniec\_pomocy() [1/2]

```
void Pacjent::ustaw_koniec_pomocy ( )
```

Automatycznie ustawia 'godzina\_uzdrowienia' Wywoływana w konstruktorze, ustawia 'godzina\_uzdrowienia' na --, Gdy pacjent nie idzie do poczekalni; poprzez dodanie do 'godzina\_uzdrowienia' wartosci atrybutu 'czas\_zajecia\_lozka';.

#### 4.4.2.9 ustaw\_koniec\_pomocy() [2/2]

```
void Pacjent::ustaw_koniec_pomocy (
    string & h )
```

Przekazana godzinie ustawia jako atrybut 'godzina\_uzdrowienia'.

##### Parameters

<i>h</i> -	przekazana godzina
------------	--------------------

#### 4.4.2.10 ustaw\_stan\_w\_szpitalu()

```
void Pacjent::ustaw_stan_w_szpitalu (
    string nazwa )
```

Przekazana nazwa jest ustawiana jako atrybut "stan\_w\_szpitalu" przekazywane nazwy to "Oddelegowany", "W poczekalni", "Na sali" oraz "Opuscil szpital".

##### Parameters

<i>nazwa</i> -	przekazana nazwa
----------------	------------------

#### 4.4.2.11 ustaw\_start\_pomocy()

```
void Pacjent::ustaw_start_pomocy (
    string & h )
```

Przekazana godzina ustawia jako atrybut 'godzina\_roz poczeczia\_pomocy'.

##### Parameters

<i>h</i> -	przekazana godzina
------------	--------------------

### 4.4.3 Friends And Related Function Documentation

#### 4.4.3.1 Lekarz

```
friend class Lekarz [friend]
```

#### 4.4.3.2 operator<<

```
ostream& operator<< (
    ostream & s,
    Pacjent & pacjent ) [friend]
```

#### 4.4.3.3 Pielegniarka

```
friend class Pielegniarka [friend]
```

#### 4.4.3.4 Pracownik

```
friend class Pracownik [friend]
```

#### 4.4.3.5 Sala

```
friend class Sala [friend]
```

### 4.4.4 Member Data Documentation

#### 4.4.4.1 czas\_zajecia\_lozka

```
int Pacjent::czas_zajecia_lozka
```

Czas (wyrażany w minutach) zajecia lozka na SORze; czas przebywania na SORze.

#### 4.4.4.2 data\_zgloszenia

```
string Pacjent::data_zgloszenia [private]
```

Podawana przy zgłoszeniu przez użytkownika.

#### 4.4.4.3 dokumentacja

```
string Pacjent::dokumentacja [private]
```

Oznacza stan dokumentacji pacjenta.

Jezeli nie została wygenerowana dokumentacja to atrybut ten jest ustawiaiany na "Nie uzupełniono". Zmieniana przez funkcje 'void [dodaj\\_dokumentacje\(\)](#)'.



#### 4.4.4.4 godzina\_rozpoczecia\_pomocy

```
string Pacjent::godzina_rozpoczecia_pomocy [private]
```

<Ustawiana za pomoca funkcji 'void [ustaw\\_start\\_pomocy\(string& h\)](#)'.

Symbolizuje godzinę zajęcia łóżka przez pacjenta na SORze

#### 4.4.4.5 godzina\_uzdrowienia

```
string Pacjent::godzina_uzdrowienia [private]
```

Ustawiana za pomoca funkcji 'void [ustaw\\_koniec\\_pomocy\(\)](#)'.

Symbolizuje godzinę opuszczenia pacjenta sali SORu

#### 4.4.4.6 godzina\_zgloszenia

```
string Pacjent::godzina_zgloszenia [private]
```

Podawana przy zgłoszeniu przez użytkownika.

#### 4.4.4.7 imie

```
string Pacjent::imie [private]
```

Podawane przy zgłoszeniu przez użytkownika.

#### 4.4.4.8 nazwisko

```
string Pacjent::nazwisko [private]
```

Podawane przy zgłoszeniu przez użytkownika.

#### 4.4.4.9 notatka

```
string Pacjent::notatka [private]
```

Podawana przy zgłoszeniu przez użytkownika.

#### 4.4.4.10 numer

```
int Pacjent::numer [private]
```

Jest to numer zgłoszenia każdego pacjenta.

Generowany przez globalna zmienna 'licznik pacjentow'. Numery sa niepowtarzalne i zaczynaja sie od 1

#### 4.4.4.11 potrzebne\_pielegniarki

```
int Pacjent::potrzebne_pielegniarki
```

Liczba pielegniarek lekarzy do wyleczenia pacjenta.

#### 4.4.4.12 potrzebni\_lekarze

```
int Pacjent::potrzebni_lekarze
```

Liczba potrzebnych lekarzy do wyleczenia pacjenta.

#### 4.4.4.13 stan\_w\_szpitalu

```
string Pacjent::stan_w_szpitalu [private]
```

Jest informacja gdzie aktualnie przebywa pacjent.

#### 4.4.4.14 stan\_zdrowia

```
int Pacjent::stan_zdrowia [private]
```

Podawane przy zgłoszeniu przez uzytkownika.

Przyjmuje wartosc od 1 do 3. Cyfry te symbolizuja rodzaje pomocy potrzebnej pacjentowi: 1.Pomoc standardowa  
2.Pomoc pilna 3. Pomoc natychmiastowa

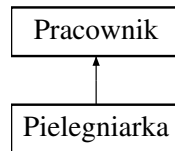
The documentation for this class was generated from the following files:

- [Pacjent.h](#)
- [Pacjent.cpp](#)

## 4.5 Pielegniarka Class Reference

```
#include <Pracownicy.h>
```

Inheritance diagram for Pielegniarka:



### Public Member Functions

- [Pielegniarka](#) ()
- [Pielegniarka](#) (int &\_ID, const string &\_imie, const string &\_nazwisko, vector< [dzien](#) > &\_tydzien)
- [~Pielegniarka](#) ()
- string [generuj\\_haslo](#) ()  
*Generuje domyslne haslo dla obiektu typu [Pielegniarka](#) Wywoływana przez konstruktor [Pielegniarka::Pielegniarka\(\)](#).*
- virtual void [wyswietl\\_grafik](#) ()  
*Wyswietla grafik Wyswietla godziny pracy pielegniarki wraz z datami.*
- virtual void [zmien\\_grafik](#) (string &aktualna\_data)  
*Wprowadza zmiany w godzinach pracy [Pielegniarka](#) proszona jest o wprowadzenie daty dnia, na ktory zmienia godziny pracy, oraz nowe godziny pracy.*
- virtual void [aktualizacja](#) (string &data, string &h, string &h\_uzdrowienia)  
*Aktualizuje [Pielegniarka::dostepnosc](#) oraz [Pielegniarka::godz\\_od\\_ktorej\\_dostepny](#).*

### Friends

- class [Sala](#)

### Additional Inherited Members

#### 4.5.1 Constructor & Destructor Documentation

##### 4.5.1.1 Pielegniarka() [1/2]

```
Pielegniarka::Pielegniarka ( )
```

#### 4.5.1.2 Pielegniarka() [2/2]

```
Pielegniarka::Pielegniarka (
    int & _ID,
    const string & _imie,
    const string & _nazwisko,
    vector< dzien > & _tydzien )
```

#### 4.5.1.3 ~Pielegniarka()

```
Pielegniarka::~~Pielegniarka ( )
```

### 4.5.2 Member Function Documentation

#### 4.5.2.1 aktualizacja()

```
void Pielegniarka::aktualizacja (
    string & data,
    string & h,
    string & h_uzdrowienia ) [virtual]
```

Aktualizuje [Pielegniarka::dostepnosc](#) oraz [Pielegniarka::godz\\_od\\_ktorej\\_dostepny](#).

Jezeli pielegniarka w podanych godzinach (zarowno godziny wyzdrowienia i przyjecia) pracuje i nie leczy innego pacjenta to jego [Pielegniarka::dostepnosc](#) jest ustawiana na 1.

##### Parameters

<i>data-</i>	data przyjecia pacjenta
<i>h-</i>	godzina o ktorej pacjent wchodzi na SOR
<i>h_uzdrowienia-</i>	godzina o ktorej pacjent wyzdrowieje

Implements [Pracownik](#).

#### 4.5.2.2 generuj\_haslo()

```
string Pielegniarka::generuj_haslo ( ) [virtual]
```

Generuje domyslne haslo dla obiektu typu [Pielegniarka](#) Wywolywana przez konstruktor [Pielegniarka::Pielegniarka\(\)](#).

Gdy w pliku "Hasla.txt" nie znajduje sie ID danego/danej pielegniarki haslo wygenerowane przez ta funkcje jest haslem pielegniarki/rza. Haslo sklada sie ze slowa "Pielegniarka" i ID pielegniarka.

**Returns**

wygenerowane haslo

Implements [Pracownik](#).

**4.5.2.3 wyswietl\_grafik()**

```
void Pielegniarka::wyswietl_grafik ( ) [virtual]
```

Wyswietla grafik Wyswietla godziny pracy pielegniarki wraz z datami.

Implements [Pracownik](#).

**4.5.2.4 zmien\_grafik()**

```
void Pielegniarka::zmien_grafik (
    string & aktualna_data ) [virtual]
```

Wprowadza zmiany w godzinach pracy [Pielegniarka](#) proszona jest o wprowadzenie daty dnia, na ktory zmienia godziny pracy, oraz nowe godziny pracy.

Sprawdzana jest poprawnoŹ podanej daty i godziny (format, zakres tygodnia)

Implements [Pracownik](#).

**4.5.3 Friends And Related Function Documentation****4.5.3.1 Sala**

```
friend class Sala [friend]
```

The documentation for this class was generated from the following files:

- [Pracownicy.h](#)
- [Pracownicy.cpp](#)

**4.6 Poczekalnia Class Reference**

```
#include <Sala.h>
```

## Public Member Functions

- [Poczekalnia](#) ()  
*Liczba wolnych miejsc w poczekalni.*
- [~Poczekalnia](#) ()
- void [ustaw\\_ile\\_miejsc](#) (int &ilosc)  
*Ustawia "il\_miejsc" i "il\_wolnych\_miejsc" na przekazana ilosc.*
- void [zajmij\\_miejsce](#) (int &ilosc)  
*Zmniejsza liczbe wolnych miejsc o podana ilosc.*
- void [zwolnij\\_miejsce](#) (int &ilosc)  
*Zwieksza liczbe wolnych miejsc o podana ilosc.*

## Protected Attributes

- int [il\\_miejsc](#)  
*Liczba miejsc w poczekalni.*
- int [il\\_wolnych\\_miejsc](#)

## Friends

- class [Sala](#)

## 4.6.1 Constructor & Destructor Documentation

### 4.6.1.1 Poczekalnia()

```
Poczekalnia::Poczekalnia ( )
```

Liczba wolnych miejsc w poczekalni.

Jest zmieniania wraz z liczba pacjentow przebywajacych w poczekalni<

### 4.6.1.2 ~Poczekalnia()

```
Poczekalnia::~~Poczekalnia ( )
```

## 4.6.2 Member Function Documentation

### 4.6.2.1 ustaw\_ile\_miejsc()

```
void Poczekalnia::ustaw_ile_miejsc (
    int & ilosc )
```

Ustawia "il\_miejsc" i "il\_wolnych\_miejsc" na przekazana ilosc.

**Parameters**

<i>ilosc-</i>	przekazana wartosc liczbowa Wywoływana tylko raz- przy odczycie pliku "Asortyment.txt".
---------------	---

**4.6.2.2 zajmij\_miejsce()**

```
void Poczekalnia::zajmij_miejsce (  
    int & ilosc )
```

Zmniejsza liczbe wolnych miejsc o podana ilosc.

**Parameters**

<i>ilosc-</i>	przekazywana liczba rowna ilosci pacjentow dochodzacych do poczekalni
---------------	---

**4.6.2.3 zwolnij\_miejsce()**

```
void Poczekalnia::zwolnij_miejsce (  
    int & ilosc )
```

Zwieksza liczbe wolnych miejsc o podana ilosc.

**Parameters**

<i>ilosc-</i>	przekazywana liczba rowna ilosci pacjentow przechodzacych do sali
---------------	---

**4.6.3 Friends And Related Function Documentation****4.6.3.1 Sala**

```
friend class Sala [friend]
```

**4.6.4 Member Data Documentation**

## 4.6.4.1 il\_miejsc

```
int Poczekalnia::il_miejsc [protected]
```

Liczba miejsc w poczekalni.

Wczytywana z pliku "Asortyment.txt". Jest stała przez cały okres trwania programu

## 4.6.4.2 il\_wolnych\_miejsc

```
int Poczekalnia::il_wolnych_miejsc [protected]
```

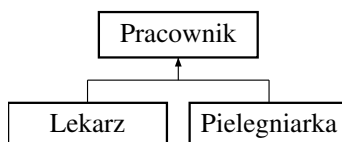
The documentation for this class was generated from the following files:

- [Sala.h](#)
- [Sala.cpp](#)

## 4.7 Pracownik Class Reference

```
#include <Pracownicy.h>
```

Inheritance diagram for Pracownik:



## Public Member Functions

- [Pracownik](#) (int &\_ID, const string &\_imie, const string &\_nazwisko, vector< [dzien](#) > &\_tydzien, bool \_↔dostepnosc)
- [Pracownik](#) (int &\_ID, const string &\_imie, const string &\_nazwisko, vector< [dzien](#) > &\_tydzien)
- [Pracownik](#) ()
- virtual [~Pracownik](#) ()
- int [zwroc\\_id](#) ()
- virtual string [generuj\\_haslo](#) ()=0  
*Metoda virtualna W zaleznosci od typu obiektu potomnego, inaczej jest generowane domysle haslo dla pracownika.*
- void [ustaw\\_haslo](#) (string &haslo)  
*Ustawia atrybut 'haslo' przekazana wartoscia parametru Wywoływana przy odczycie pliku z haslami oraz przy zmianie hasla przez uzytkownika.*
- string [zwroc\\_haslo](#) ()  
*Zwraca wartosc atrymutu 'haslo' danego pracownika Wykorzystywana do sprawdzania poprawnosci hasla- przy logowaniu przez uzytkownika.*
- void [zmien\\_haslo](#) ()  
*Funkcja do zmiany hasla Uzytkownik proszony jest o wpisanie nowego hasla.*
- bool [zaloguj](#) ()



Weryfikuje wprowadzone hasło. Użytkownik proszony jest o podanie hasła, jeżeli hasło jest poprawne zwracana jest wartość `true`, w przeciwnym wypadku `false`.

- virtual void `wyswietl_grafik` ()=0

Metoda wirtualna. W zależności od typu obiektu potomnego, w inny sposób jest wyświetlany grafik pracownika.

- virtual void `zmien_grafik` (string &aktualna\_data)=0

Metoda wirtualna. W zależności od typu obiektu potomnego, w inny sposób jest zmieniany grafik pracownika.

- virtual void `aktualizacja` (string &data, string &h, string &h\_uzdrowienia)=0

Metoda wirtualna. W zależności od typu obiektu potomnego, w inny sposób jest aktualizowany pracownik.

- string `pierwszy_dzien_tygodnia` ()

Zwraca datę pierwszego dnia tygodnia podanego w pliku "Grafik.txt".

- bool `czy_godziny_pracy` (string &data, string &h)

Funkcja sprawdza czy podana w parametrze godzina znajduje się w zakresie godzin pracy danego pracownika.

- bool `czy_dalej_bede_zajety` (string &h)

Funkcja sprawdza czy podana w parametrze godzina znajduje się w zakresie godzin pracy danego pracownika.

- void `ustaw_godzine_od_ktorej_dostepny` (string &h)

Ustawia atrybut `Pracownik::godz_od_ktorej_dostepny` wartością parametru przekazanego.

## Protected Attributes

- string `haslo`
- vector< `dzien` > `tydzien`
- bool `dostepnosc`
- string `godz_od_ktorej_dostepny`
- vector< int > `nr_moich_pacjentow`

## Private Member Functions

- int friend `roznica_godzin` (string &godz\_podana, string &godz)

## Private Attributes

- const int `ID`  
Indywidualny numer każdego pracownika.
- const string `imie`
- const string `nazwisko`

## Friends

- class `Pacjent`
- class `Sala`
- ostream & `operator<<` (ostream &s, `Pracownik` &pracownik)  
Wypisuje imię, nazwisko oraz ID pracownika.

## 4.7.1 Constructor & Destructor Documentation

#### 4.7.1.1 Pracownik() [1/3]

```
Pracownik::Pracownik (
    int & _ID,
    const string & _imie,
    const string & _nazwisko,
    vector< dzien > & _tydzien,
    bool _dostepnosc )
```

#### 4.7.1.2 Pracownik() [2/3]

```
Pracownik::Pracownik (
    int & _ID,
    const string & _imie,
    const string & _nazwisko,
    vector< dzien > & _tydzien )
```

#### 4.7.1.3 Pracownik() [3/3]

```
Pracownik::Pracownik ( )
```

#### 4.7.1.4 ~Pracownik()

```
Pracownik::~~Pracownik ( ) [virtual]
```

### 4.7.2 Member Function Documentation

#### 4.7.2.1 aktualizacja()

```
virtual void Pracownik::aktualizacja (
    string & data,
    string & h,
    string & h_uzdrowienia ) [pure virtual]
```

Metoda wirtualna W zaleznosci od typu obiektu potomnego, w inny sposob jest aktualizowany pracownik.

Implemented in [Pielegniarka](#), and [Lekarz](#).

#### 4.7.2.2 czy\_dalej\_bede\_zajety()

```
bool Pracownik::czy_dalej_bede_zajety (
    string & h )
```

Funkcja sprawdza czy podana w parametrze godzina znajduje sie w zakresie godzin pracy danego pracownika.

##### Returns

h- sprawdzana godzina

#### 4.7.2.3 czy\_godziny\_pracy()

```
bool Pracownik::czy_godziny_pracy (
    string & data,
    string & h )
```

Funkcja sprawdza czy podana w parametrze godzina znajduje sie w zakresie godzin pracy danego pracownika.

##### Parameters

<i>data</i> -	dzien, w ktorym sprawdzana jest godzina
---------------	---

##### Returns

h- sprawdzana godzina

#### 4.7.2.4 generuj\_haslo()

```
virtual string Pracownik::generuj_haslo ( ) [pure virtual]
```

Metoda wirtualna W zaleznosci od typu obiektu potomnego, inaczej jest generowane domysle haslo dla pracownika.

Implemented in [Pielegniarka](#), and [Lekarz](#).

#### 4.7.2.5 pierwszy\_dzien\_tygodnia()

```
string Pracownik::pierwszy_dzien_tygodnia ( )
```

Zwraca date pierwszego dnia tygodnia podanego w pliku "Grafik.txt".

##### Returns

data

#### 4.7.2.6 roznica\_godzin()

```
int friend Pracownik::roznica_godzin (
    string & godz_podana,
    string & godz ) [private]
```

#### 4.7.2.7 ustaw\_godzone\_od\_ktorej\_dostepny()

```
void Pracownik::ustaw_godzone_od_ktorej_dostepny (
    string & h )
```

Ustawia atrybut [Pracownik::godz\\_od\\_ktorej\\_dostepny](#) wartoscia parametru przekazanego.

##### Parameters

<i>h-godzina</i>	od ktorej bedzie pracownik dostepny
------------------	-------------------------------------

#### 4.7.2.8 ustaw\_haslo()

```
void Pracownik::ustaw_haslo (
    string & haslo )
```

Ustawia atrybut 'haslo' przekazana wartoscia parametru Wywoływana przy odczycie pliku z haslami oraz przy zmianie hasla przez uzytkownika.

##### Parameters

<i>haslo-</i>	haslo do ustawienia
---------------	---------------------

#### 4.7.2.9 wyswietl\_grafik()

```
virtual void Pracownik::wyswietl_grafik ( ) [pure virtual]
```

Metoda wirtualna W zaleznosci od typu obiektu potomnego, w inny sposob jest wyswietlany grafik pracownika.

Implemented in [Pielegniarka](#), and [Lekarz](#).

#### 4.7.2.10 zaloguj()

```
bool Pracownik::zaloguj ( )
```

Weryfikuje wprowadzone hasło Użytkownik proszony jest o podanie hasła, jeżeli hasło jest poprawne zwracana jest wartość true, w przeciwnym wypadku false.

##### Returns

powodzenie przy logowaniu

#### 4.7.2.11 zmien\_grafik()

```
virtual void Pracownik::zmien_grafik (
    string & aktualna_data ) [pure virtual]
```

Metoda wirtualna W zależności od typu obiektu potomnego, w inny sposób jest zmieniany grafik pracownika.

Implemented in [Pielęgniarka](#), and [Lekarz](#).

#### 4.7.2.12 zmien\_haslo()

```
void Pracownik::zmien_haslo ( )
```

Funkcja do zmiany hasła Użytkownik proszony jest o wpisanie nowego hasła.

Wprowadzone hasło jest ustawione jako nowa wartość atrybutu 'hasło'

#### 4.7.2.13 zwroc\_haslo()

```
string Pracownik::zwroc_haslo ( )
```

Zwraca wartość atrybutu 'hasło' danego pracownika Wykorzystywana do sprawdzania poprawności hasła- przy logowaniu przez użytkownika.

##### Returns

wartość atrybutu 'hasło'

#### 4.7.2.14 zwroc\_id()

```
int Pracownik::zwroc_id ( )
```

##### Returns

Wartość atrybutu 'ID'

## 4.7.3 Friends And Related Function Documentation

### 4.7.3.1 operator<<

```
ostream& operator<< (  
    ostream & s,  
    Pracownik & pracownik ) [friend]
```

Wypisuje imie, nazwisko oraz ID pracownika.

### 4.7.3.2 Pacjent

```
friend class Pacjent [friend]
```

### 4.7.3.3 Sala

```
friend class Sala [friend]
```

## 4.7.4 Member Data Documentation

### 4.7.4.1 dostepnosc

```
bool Pracownik::dostepnosc [protected]
```

### 4.7.4.2 godz\_od\_ktorej\_dostepny

```
string Pracownik::godz_od_ktorej_dostepny [protected]
```

### 4.7.4.3 haslo

```
string Pracownik::haslo [protected]
```

#### 4.7.4.4 ID

```
const int Pracownik::ID [private]
```

Indywidualny numer kazdego pracownika.

#### 4.7.4.5 imie

```
const string Pracownik::imie [private]
```

#### 4.7.4.6 nazwisko

```
const string Pracownik::nazwisko [private]
```

#### 4.7.4.7 nr\_moich\_pacjentow

```
vector<int> Pracownik::nr_moich_pacjentow [protected]
```

#### 4.7.4.8 tydzien

```
vector<dzien> Pracownik::tydzien [protected]
```

The documentation for this class was generated from the following files:

- [Pracownicy.h](#)
- [Pracownicy.cpp](#)

## 4.8 Sala Class Reference

```
#include <Sala.h>
```

## Public Member Functions

- [Sala](#) ()
- [~Sala](#) ()
- void [ustaw\\_il\\_lozek](#) (int &\_il)
 

*Ustawia wartosc atrybutu 'il\_lozek'.*
- void [przekaz\\_wskazniki](#) ([Poczekalnia](#) \*\_na\_poczekalnie, vector< shared\_ptr< [Lekarz](#) >> \*\_na\_lekarzy, vector< shared\_ptr< [Pielegniarka](#) >> \*\_na\_pielegniarki, vector< shared\_ptr< [Asortyment](#) >> \*\_na\_asortyment)
 

*Ustawia wskazniki klasy [Sala](#) na obiekty na ktore wskazuja przekazane wskazniki.*
- void [przekaz\\_wskaznik\\_pacjenci](#) (vector< shared\_ptr< [Pacjent](#) >> \*\_wsk\_pacjenci\_dzis)
 

*Ustawia wskaznik klasy [Sala](#) 'wsk\_pacjenci\_dzis' na obiekt na ktory wskazuje przekazany wskaznik Na kazdy dzien tygodnia jest tworzony jeden taki wskaznik, jest on przekazywany tylko raz.*
- void [oblicz\\_calosciowy\\_poziom\\_asortymentu](#) ()
 

*Oblicza srednia wartosc procentowa zaopatrzenia sali Wartosc wyliczona moze byc wieksza od 100.*
- void [oblicz\\_poziom\\_personelu](#) ()
 

*Oblicza procent dostepnych pracownikow ze wszystkich pracownikow.*
- string [wylicz\\_stan](#) ()
 

*Wylicza aktualny stan SORu.*
- void [ustaw\\_stan](#) ()
 

*Ustawia atrybut 'stan' jako wartosc zwracana przez wywolana wewnatrz funkcji funkcje 'wylicz\_stan()'.*
- void [uruchom\\_sor](#) (string &godz, string &data)
 

*Funkcja ustawia dostepnosc personelu w podanej dacie i godzinie.*
- bool [czy\\_trzeba\\_oddelegowac](#) ()
 

*Funkcja sprawdza za pomoca 'wsk\_na\_poczekalnie' czy jest miejsce w poczekalni.*
- void [aktualizuj](#) (string &data, string &h, string &h\_uzdrowienia)
 

*Aktualizuje dane pracownikow dla danego pacjenta.*
- bool [czy\\_jest\\_wystarczajaco\\_personelu](#) (string &data, string &h\_zgloszenia, string &h\_uzdrownienia, int &il\_lek, int &il\_piel)
 

*Sprawdza czy jest wystarczajaca liczba personelu dla pacjenta by go przyjac na sale SORu.*
- void [wybierz\\_konkretnego\\_prowadzacego](#) (int stan\_pacjenta, string &h\_zgloszenia, string &godzina\_uzdrowienia, int nr\_pacjenta)
 

*Umozliwia wybranie konkretnego pracownika.*
- void [wybierz\\_dowolny\\_komplet\\_pracownikow](#) (int stan\_pacjenta, string &h\_zgloszenia, string &godzina\_uzdrowienia, int nr\_pacjenta)
 

*Losowo wybiera komplet pracownikow dla pacjenta Pracownikowi przypisuje sie danego pacjenta poprzez dodanie jego numeru do 'vector<int> nr\_moich\_pacjentow', oraz odpowiednie zmienienie godziny od ktorej dany pracownik bedzie znow dostepny (rownej godzinie o ktorej pacjent opusci SOR).*
- void [zajmij\\_lozko](#) (int il\_lozek)
 

*Funkcja zmniejsza parametr wolnych lozek i zwieksza parametr zajetych lozek o ta sama ilosc.*
- void [zwolnij\\_lozko](#) (int il\_lozek)
 

*Funkcja zmniejsza parametr zajetych lozek i zwieksza parametr wolnych lozek.*
- void [zuzyj\\_asortyment](#) (int il\_lozek, int stan\_pacjenta)
 

*Zmniejsza o liczbe procent atrybut 'poziom\_asort' Jezeli stan pacjenta jest rowny 1, to 'poziom\_asort' jest zmniejszany o 2 procent Jezeli stan pacjenta jest rowny 2, to 'poziom\_asort' jest zmniejszany o 3 procent Jezeli stan pacjenta jest rowny 3, to 'poziom\_asort' jest zmniejszany o 5 procent Kazdy element w wektorze wskazywanym przez 'wskaznik\_na\_asortyment' jest rowniez zmniejszany o ten sam procent.*
- vector< string > [znajdz\\_najkrotszy\\_czas\\_oczekiwania](#) (string &data, string &h\_zgloszenia, int stan\_pacjenta, int nr\_pacjenta)
 

*Funkcja szuka najkrotszy czas oczekiwania w poczekalni.*
- void [zajmij\\_miejsce\\_w\\_poczekalni](#) (int il)
 

*Przez 'wsk\_na\_poczekalnie' wywolwana jest funkcja [Poczekalnia::zajmij\\_miejsce\(il\)](#)*
- void [zwolnij\\_miejsce\\_w\\_poczekalni](#) (int il)



- Przez 'wsk\_na\_poczekalnie' wywoływana jest funkcja `Poczekalnia::zwolnij_miejsce(il)`
- void `sprawdz_pacj_na_sali_i_w_poczekalni` (string &h)
  - Przez 'wsk\_pacjenci\_dzis' sprawdzany i aktualizowany jest stan pacjentów przebywających na sali i poczekalni. Wywoływana przed każdym zgłoszeniem czy wyświetlaniem stanu SORu.
- int `odczytaj_ile_miejsc_wolnych_w_poczekalni` ()
  - Przez 'wsk\_na\_poczekalnie' sprawdzany jest atrybut 'il\_wolnych\_miejsc'.
- void `czy_ktos_nie_skonczyl_pracy` (string &data, string &h)
  - Funkcja sprawdza czy ktorys z pracowników nie skonczyl pracy. Wywoływana przed każdym zgłoszeniem czy wyświetlaniem stanu SORu.
- void `rada` ()
  - Wyswietla rade dla dyspozytora, która poprawi stan SORu.
- void `koniec_dnia` ()
  - Funkcja wywoływana po wybraniu przez użytkownika 6 opcji w menu dyspozytora.
- void `uzupelnienie_hurtowe_asortymentu` (int &proc)
  - Podnosi procentowy poziom ogólnego zaopatrzenia Soru. Za pomocą wskaźnika 'wskaznik\_na\_asortyment' dla każdego elementu wywoływana jest funkcja `Asortyment::uzupelnij_hurtowo(proc)`.
- vector< `dzien` > `przygotuj_grafik` ()
  - szukuje i zwraca szablon grafiku dla nowego pracownika
- void `generuj_raport` (const string &nazwa\_raportu)
  - Generuje raport SORu do pliku tekstowego. W raporcie jest wyświetlany stan Soru, lista pacjentów przebywających na sorze oraz szczegółowa lista zawierająca każdy element zaopatrzenia.

## Protected Attributes

- string `stan`
  - nazwa stanu
- double `poziom_asort`
  - procentowy poziom ogólnego zaopatrzenia Soru
- int `proc_dost_personelu`
  - procent dostępnych pracowników
- int `il_dost_lekarzy`
  - liczba dostępnych lekarzy
- int `il_dost_piel`
  - liczba dostępnych pielęgniarek

## Private Attributes

- int `il_lozek`
  - Liczba łóżek odczytana z pliku.
- int `il_wolnych_lozek`
  - liczba aktualnie wolnych łóżek
- int `il_zajetych_lozek`
  - liczba zajętych łóżek
- `Poczekalnia * wsk_na_poczekalnie`
  - wskaznik na obiekt typu `Poczekalnia`
- vector< shared\_ptr< `Lekarz` > > \* `wskaznik_na_lekarzy`
  - wskaznik na bazę lekarzy
- vector< shared\_ptr< `Pielęgniarka` > > \* `wskaznik_na_pielęgniarki`
  - wskaznik na bazę pielęgniarek
- vector< shared\_ptr< `Asortyment` > > \* `wskaznik_na_asortyment`
  - wskaznik na bazę asortymentu
- vector< shared\_ptr< `Pacjent` > > \* `wsk_pacjenci_dzis`
  - wskaznik na kontener z pacjentami danego dnia

## Friends

- ostream & [operator<<](#) (ostream &s, [Sala](#) &sala)

## 4.8.1 Constructor & Destructor Documentation

### 4.8.1.1 Sala()

```
Sala::Sala ( )
```

### 4.8.1.2 ~Sala()

```
Sala::~~Sala ( )
```

## 4.8.2 Member Function Documentation

### 4.8.2.1 aktualizuj()

```
void Sala::aktualizuj (
    string & data,
    string & h,
    string & h_uzdrowienia )
```

Aktualizuje dane pracownikow dla danego pacjenta.

Dla kazdego pracownika wskazywanego w obiekcie wskazywanym przez 'wskaznik\_na\_lekarzy' oraz 'wskaznik\_na\_pielegniarki' jest wywoływana ich metoda wirtualna 'aktualizacja(string& data, string& h, string& h\_uzdrowienia)'

#### Parameters

<i>data</i> -	data podana przez uzytkownika po wybraniu 3 opcji w menu glownym
<i>h</i> -	jest to godzina zgloszenia pacjenta.
<i>h_uzdrowienia</i> -	jest to godzina o ktorej pacjent by opuscil sor gdyby mogl odrazu wejsc na SOR po zgloszeniu

### 4.8.2.2 czy\_jest\_wystarczajaco\_personelu()

```
bool Sala::czy_jest_wystarczajaco_personelu (
    string & data,
```

```

string & h_zgloszenia,
string & h_uzdrownienia,
int & il_lek,
int & il_piel )

```

Sprawdza czy jest wystarczająca liczba personelu dla pacjenta by go przyjąć na sale SORu.

Dla każdego pracownika wskazywanego w obiekcie wskazywanym przez 'wskaznik\_na\_lekarzy' oraz 'wskaznik\_na\_pielegniarki' są sprawdzane godziny pracy godziny zgłoszenia i godziny uzdrowienia pacjenta. Dla lekarzy dodatkowo sprawdzane są godziny operacji w grafiku. Funkcja zapobiega sytuacji, by jakkolwiek z pracowników musiał pracować poza godzinami pracy.

#### Parameters

<i>data-</i>	data podana przez użytkownika po wybraniu 3 opcji w menu głównym
<i>h-</i>	jest to godzina zgłoszenia pacjenta.
<i>h_uzdrownienia-</i>	jest to godzina o której pacjent by opuścił sor gdyby mógł od razu wejść na SOR po zgłoszeniu
<i>il_lek-</i>	liczba potrzebnych lekarzy
<i>il_piel-</i>	liczba potrzebnych pielęgniarek

#### Returns

true jeżeli jest wystarczająca liczba personelu

#### 4.8.2.3 czy\_ktos\_nie\_skonczy\_l\_pracy()

```

void Sala::czy_ktos_nie_skonczy_l_pracy (
    string & data,
    string & h )

```

Funkcja sprawdza czy ktoryś z pracowników nie skończył pracy Wywoływana przed każdym zgłoszeniem czy wyświetlaniem stanu SORu.

Za pomocą wskaźników 'wskaznik\_na\_lekarzy' oraz 'wskaznik\_na\_pielegniarki' porównywane są godziny pracy pracowników w danym dniu z parametrem h. Jeżeli pracownik skończył już pracę jego dostępność ustawiana jest na 0, a godzina od której jest dostępny na –:-

#### Parameters

<i>data-data</i>	podana przez użytkownika po wybraniu 3 opcji w menu głównym
<i>h-</i>	godzina następnego zgłoszenia wprowadzonego przez użytkownika

#### 4.8.2.4 czy\_trzeba\_oddelegowac()

```

bool Sala::czy_trzeba_oddelegowac ( )

```

Funkcja sprawdza za pomocą 'wsk\_na\_poczekalni' czy jest miejsce w poczekalni.

**Returns**

true jezeli jest miejsce w poczekalni

**4.8.2.5 generuj\_raport()**

```
void Sala::generuj_raport (
    const string & nazwa_raportu )
```

Generuje raport SORu do pliku tekstowego W raporcie jest wyswietlany stan Soru, lista pacjentow przebywajacych na sorze oraz szczegolowa lista zawierajaca kazdy element zaopatrzenia.

**Parameters**

<i>nazwa_raportu-</i>	nazwa pliku, do ktorego zostanie wygenerowany raport
-----------------------	--

**4.8.2.6 koniec\_dnia()**

```
void Sala::koniec_dnia ( )
```

Funkcja wywoływana po wybraniu przez uzytkownika 6 opcji w menu dyspozytora.

Wszystkim pacjenci przebywajacy w szpitalu opuszczaj szpital; zuzywajac przy tym asortyment. Zwalniane sa lozka na sali oraz miejsca w poczekalni

**4.8.2.7 oblicz\_calosciowy\_poziom\_asortymentu()**

```
void Sala::oblicz_calosciowy_poziom_asortymentu ( )
```

Oblicza srednia wartosc procentowa zaopatrzenia sali Wartosc wyliczona moze byc wieksza od 100.

**4.8.2.8 oblicz\_poziom\_personelu()**

```
void Sala::oblicz_poziom_personelu ( )
```

Oblicza procent dostepnych pracownikow ze wszystkich pracownikow.

#### 4.8.2.9 odczytaj\_ile\_miejsc\_wolnych\_w\_poczekalni()

```
int Sala::odczytaj_ile_miejsc_wolnych_w_poczekalni ( )
```

Przez 'wsk\_na\_poczekalnie' sprawdzany jest atrybut 'il\_wolnych\_miejsc'.

##### Returns

wartosc rowna ilosci wolnych miwjsc w poczekalni

#### 4.8.2.10 przekaz\_wskaznik\_pacjenci()

```
void Sala::przekaz_wskaznik_pacjenci (
    vector< shared_ptr< Pacjent >> * _wsk_pacjenci_dzis )
```

Ustawia wskaznik klasy [Sala](#) 'wsk\_pacjenci\_dzis' na obiekt na ktory wskazuje przekazany wskaznik Na kazdy dzien tygodnia jest tworzony jeden taki wskaznik, jest on przekazywany tylko raz.

##### Parameters

<code>_wsk_pacjenci_dzis-</code>	wskaznik na wektor inteligentnych wskaznikow na obiekty typu <a href="#">Pacjent</a>
----------------------------------	--

#### 4.8.2.11 przekaz\_wskazniki()

```
void Sala::przekaz_wskazniki (
    Poczekalnia * _na_poczekalnie,
    vector< shared_ptr< Lekarz >> * _na_lekarzy,
    vector< shared_ptr< Pielegniarka >> * _na_pielegniarki,
    vector< shared_ptr< Asortyment >> * _na_asortyment )
```

Ustawia wskazniki klasy [Sala](#) na obiekty na ktore wskazuja przekazane wskazniki.

##### Parameters

<code>_na_poczekalnie-</code>	wskaznik na obiekt typu poczekalnia
<code>_na_lekarzy-</code>	wskaznik na wektor inteligentnych wskaznikow na obiekty typu <a href="#">Lekarz</a>
<code>_na_pielegniarki-</code>	wskaznik na wektor inteligentnych wskaznikow na obiekty typu <a href="#">Pielegniarka</a>
<code>_na_asortyment-</code>	wskaznik na wektor inteligentnych wskaznikow na obiekty typu <a href="#">Asortyment</a>

#### 4.8.2.12 przygotuj\_grafik()

```
vector< dzien > Sala::przygotuj_grafik ( )
```

szykuje i zwraca szablon grafiku dla nowego pracownika

### Returns

szablon grafiku

#### 4.8.2.13 rada()

```
void Sala::rada ( )
```

Wyswietla rade dla dyspozytora, ktora poprawi stan SORu.

#### 4.8.2.14 sprawdz\_pacj\_na\_sali\_i\_w\_poczekalni()

```
void Sala::sprawdz_pacj_na_sali_i_w_poczekalni (
    string & h )
```

Przez 'wsk\_pacjenci\_dzis' sprawdzany i aktualizowany jest stan pacjentow przebywajacych na sali i poczekalni. Wywoływana przed kazdym zgłoszeniem czy wyświetlaniem stanu SORu.

Dla pacjentow ktorych 'stan\_w\_szpitalu' to "Na sali", sprawdzane jest czy parametr h, jest wiekszy od godziny o ktorej powinien opuscic SOR, jezeli tak, to jego 'stan\_w\_szpitalu' ustawiany jest na 'Opuscil szpital'. Dla pacjentow ktorych 'stan\_w\_szpitalu' to "W poczekalni", sprawdzane jest czy parametr h, jest wiekszy od godziny o ktorej powinien opuscic SOR, jezeli tak, to jego 'stan\_w\_szpitalu' ustawiany jest na 'Opuscil szpital'. Jezeli jego godzina wejscia na SOR jest mniejsza od parametru h i godzina opuszczenia Soru jest wieksza od h, to 'stan\_w\_szpitalu' ustawiany jest na 'Na sali'.

### Parameters

<i>h</i> -	godzina zgłoszenia, przed ktorym sprawdzane sa sala i poczekalnia
------------	---

#### 4.8.2.15 uruchom\_sor()

```
void Sala::uruchom_sor (
    string & _godz,
    string & _data )
```

Funkcja ustawia dostepnosc personelu w podanej dacie i godzinie.

Sprawdzone sa godziny pracy pracownika, czy nie ma zaplanowanych juz na ta godzine w grafiku wydarzen (jezeli [Pracownik](#) jest obiektem typu [Lekarz](#)). Jezeli podana godzina jest w obrebach godzin pracy pracownika, a lekarz nie ma zaplanowanych operacji na ta godzine, to dostepnosc tego pracownika jest ustawiana na 1. Ustawiany jest tez stan SORu poprzez wywołanie funkcji '[ustaw\\_stan\(\)](#)'. Funkcja ta zostaje wywołana gdy uzytkownik w menu glownym wybierze opcje 'Jestem dyspozytorem SORu' i wpisze prawidlowa date i godzine.

## Parameters

<code>_data-</code>	podana przez uzytkownika data
<code>_godz-</code>	podana przez uzytkownika godzina

**4.8.2.16 ustaw\_il\_lozek()**

```
void Sala::ustaw_il_lozek (
    int & _il )
```

Ustawia wartosc atrybutu 'il\_lozek'.

Jest wywoływana przy odczycie pliku "Asortyment.txt".

## Parameters

<code>↔</code>	liczba lozek
<code>_↔</code>	
<code>il-</code>	

**4.8.2.17 ustaw\_stan()**

```
void Sala::ustaw_stan ( )
```

Ustawia atrybut 'stan' jako wartosc zwracana przez wywołana wewnatrz funkcji funkcje ['wylicz\\_stan\(\)'](#).

**4.8.2.18 uzupełnienie\_hurtowe\_asortymentu()**

```
void Sala::uzupelnienie_hurtowe_asortymentu (
    int & proc )
```

Podnosi procentowy poziom ogolnego zaopatrzenia Soru Za pomoca wskaznika 'wskaznik\_na\_asortyment' dla kazdego elementu wywoływana jest funkcja Asortyment::uzupelnij\_hurtowo(proc);.

## Parameters

<code>proc-</code>	liczba rowna procentowej mianie w zaopatrzeniu
--------------------	--

**4.8.2.19 wybierz\_dowolny\_komplet\_pracownikow()**

```
void Sala::wybierz_dowolny_komplet_pracownikow (
    int stan_pacjenta,
    string & h_zgloszenia,
    string & godzina_wyzdrowienia,
    int nr_pacjenta )
```

Losowo wybiera komplet pracownikow dla pacjenta Pracownikowi przypisuje sie danego pacjenta poprzez dodanie jego numeru do 'vector <int> nr\_moich\_pacjentow', oraz odpowiednie zmienienie godziny od ktorej dany pracownik bedzie znow dostepny (rownej godzinie o ktorej pacjent opusci SOR).

Zmieniana jest tez dostepnosc pracownika na 0.

**Parameters**

<i>stan_pacjenta-</i>	stan pacjenta
<i>h_zgloszenia-</i>	godzina zgloszenia pacjenta
<i>godzina_wyzdrowienia-</i>	godzina o ktorej pacjent opusci sale SORu
<i>nr_pacjenta-</i>	numer pacjenta

**4.8.2.20 wybierz\_konkretnego\_prowadzacego()**

```
void Sala::wybierz_konkretnego_prowadzacego (
    int stan_pacjenta,
    string & h_zgloszenia,
    string & godzina_wyzdrowienia,
    int nr_pacjenta )
```

Umozliwia wybranie konkretnego pracownika.

Uzytkownikowi wyswietla sie lista dostepnych pracownikow prowadzacych wraz z ich numerami ID. Dla pacjentow potrzebujacych 'pomoc standardowa' sa wyswietlane aktualnie dostepne pielegniarki, w innym przypadku sa wyswietlani lekarze. Uzytkownik wprowadza ID wybranego pracownika. Nastepnie pracownikowi przypisuje sie danego pacjenta przez dodanie jego numeru do 'vector <int> nr\_moich\_pacjentow', oraz odpowiednie zmienienie godziny od ktorej dany pracownik bedzie znow dostepny (rownej godzinie o ktorej pacjent opusci SOR). Jezeli 'stan\_pacjenta' byl rowny 3, to pielegniarka jest wybierana losowo. Zmieniana jest tez dostepnosc pracownika na 0.

**Parameters**

<i>stan_pacjenta-</i>	stan pacjenta
<i>h_zgloszenia-</i>	godzina zgloszenia pacjenta
<i>godzina_wyzdrowienia-</i>	godzina o ktorej pacjent opusci sale SORu
<i>nr_pacjenta-</i>	numer pacjenta

**4.8.2.21 wylicz\_stan()**

```
string Sala::wylicz_stan ( )
```



Wylicza aktualny stan SORu.

Mozliwe stany to: 'PRAWIDLOWY','POPRAWNY','NIEPRAWIDLOWY','KRYTYCZNY'. Do obliczenia stanu brane sa pod uwage poziom zaopatrzenia, liczbe dostepnych lekarzy i pielegniarek oraz liczbe wolnych miejsc w poczekalni i lozek na sali.

#### Returns

Nazwa stanu

#### 4.8.2.22 zajmij\_lozko()

```
void Sala::zajmij_lozko (
    int il_lozek )
```

Funkcja zmniejsza parametr wolnych lozek i zwiększa parametr zajętych lozek o ta sama ilość.

#### Parameters

<i>il_lozek</i> -	liczba zajmowanych lozek
-------------------	--------------------------

#### 4.8.2.23 zajmij\_miejsce\_w\_poczekalni()

```
void Sala::zajmij_miejsce_w_poczekalni (
    int il )
```

Przez 'wsk\_na\_poczekalnie' wywoływana jest funkcja Poczekalnia::zajmij\_miejsce(il)

#### Parameters

<i>il</i> -	liczba zajmowanych miejsc
-------------	---------------------------

#### 4.8.2.24 znajdz\_najkrotszy\_czas\_oczekiwnania()

```
vector< string > Sala::znajdz_najkrotszy_czas_oczekiwnania (
    string & data,
    string & h_zgloszenia,
    int stan_pacjenta,
    int nr_pacjenta )
```

Funkcja szuka najkrotszy czas oczekiwania w poczekalni.

Za pomoca wskaźników 'wskaźnik\_na\_lekarzy' oraz 'wskaźnik\_na\_pielęgniarki' wyszukiwana jest najmniejsza różnica między godziną zgłoszenia pacjenta (*h\_zgloszenia*), a godziną od której pracownik jest dostępny. Funkcja wyświetla informacje o długości oczekiwania w poczekalni. Jeżeli stan poszkodowanego jest równy 3 to wybierany jest dłuższy czas oczekiwania między czasem oczekiwania na lekarza i pielęgniarkę. Następnie pyta użytkownika czy woli oddelegować pacjenta czy posłać go do poczekalni. Jeżeli użytkownik wybierze oddelegowanie, zostaje wywołana funkcja [Pacjent::oddeleguj\(\)](#). W przeciwnym razie pacjent zostaje przypisany do pracownika/-ów o najkrótszym czasie oczekiwania. Pracownikowi przypisuje się danego pacjenta poprzez dodanie jego numeru do 'vector <int> nr\_moich\_pacjentow', oraz odpowiednie zmniejszenie godziny od której dany pracownik będzie znowu dostępny (równej godzinie o której pacjent opuści SOR; która jest równa sumie godzinie zgłoszenia, czasu oczekiwania oraz czasu potrzebnego do wyleczenia). Zmieniana jest też dostępność pracownika na 0.

#### Parameters

<i>data-</i>	data podana przez użytkownika po wybraniu 3 opcji w menu głównym
<i>h_zgloszenia-</i>	godzina zgłoszenia pacjenta
<i>stan_pacjenta-</i>	stan pacjenta
<i>nr_pacjenta-</i>	numer pacjenta

#### Returns

wektor z dwoma stringami reprezentującymi godziny, jeden z godziną o której pacjent wejdzie na SOR, druga, z godziną o której z niego wyjdzie

#### 4.8.2.25 zuzyj\_asortyment()

```
void Sala::zuzyj_asortyment (
    int il_lozek,
    int stan_pacjenta )
```

Zmniejsza o liczbę procent atrybut 'poziom\_asort'. Jeżeli stan pacjenta jest równy 1, to 'poziom\_asort' jest zmniejszany o 2 procent. Jeżeli stan pacjenta jest równy 2, to 'poziom\_asort' jest zmniejszany o 3 procent. Jeżeli stan pacjenta jest równy 3, to 'poziom\_asort' jest zmniejszany o 5 procent. Każdy element w wektorze wskazywanym przez 'wskaźnik\_na\_asortyment' jest również zmniejszany o ten sam procent.

Jeżeli *il\_lozek* jest różna od 1 to procent o który zmniejszone są dane jest mnożony przez *il\_lozek*.

#### Parameters

<i>il_lozek-</i>	liczba pacjentów
<i>stan_pacjenta-</i>	stan pacjenta

#### 4.8.2.26 zwolnij\_lozko()

```
void Sala::zwolnij_lozko (
    int il_lozek )
```

Funkcja zmniejsza parametr zajętych łóżek i zwiększa parametr wolnych łóżek.

**Parameters**

<i>il_lozek-</i>	liczba zwalnianych lozek
------------------	--------------------------

**4.8.2.27 zwolnij\_miejsce\_w\_poczekalni()**

```
void Sala::zwolnij_miejsce_w_poczekalni (
    int il )
```

Przez 'wsk\_na\_poczekalnie' wywoływana jest funkcja Poczeklania::zwolnij\_miejsce(il)

**Parameters**

<i>il-</i>	liczba zwalnianych miejsc
------------	---------------------------

**4.8.3 Friends And Related Function Documentation****4.8.3.1 operator<<**

```
ostream& operator<< (
    ostream & s,
    Sala & sala ) [friend]
```

**4.8.4 Member Data Documentation****4.8.4.1 il\_dost\_lekarzy**

```
int Sala::il_dost_lekarzy [protected]
```

liczba dostępnych lekarzy

**4.8.4.2 il\_dost\_piel**

```
int Sala::il_dost_piel [protected]
```

liczba dostępnych pielęgniarek

#### 4.8.4.3 il\_lozek

```
int Sala::il_lozek [private]
```

Liczba lozek odczytana z pliku.

#### 4.8.4.4 il\_wolnych\_lozek

```
int Sala::il_wolnych_lozek [private]
```

liczba aktualnie wolnych lozek

#### 4.8.4.5 il\_zajetych\_lozek

```
int Sala::il_zajetych_lozek [private]
```

liczba zajetych lozek

#### 4.8.4.6 poziom\_asort

```
double Sala::poziom_asort [protected]
```

procentowy poziom ogolnego zaopatrzenia Soru

#### 4.8.4.7 proc\_dost\_personelu

```
int Sala::proc_dost_personelu [protected]
```

procent dostępnych pracowników

#### 4.8.4.8 stan

```
string Sala::stan [protected]
```

nazwa stanu

#### 4.8.4.9 wsk\_na\_poczekalnie

```
Poczekalnia* Sala::wsk_na_poczekalnie [private]
```

wskaznik na obiekt typu [Poczekalnia](#)

#### 4.8.4.10 wsk\_pacjenci\_dzis

```
vector<shared_ptr<Pacjent> >* Sala::wsk_pacjenci_dzis [private]
```

wskaznik na kontener z pacjentami danego dnia

#### 4.8.4.11 wskaznik\_na\_asortyment

```
vector< shared_ptr<Asortyment> >* Sala::wskaznik_na_asortyment [private]
```

wskaznik na baze asortymentu

#### 4.8.4.12 wskaznik\_na\_lekarzy

```
vector< shared_ptr<Lekarz> >* Sala::wskaznik_na_lekarzy [private]
```

wskaznik na baze lekarzy

#### 4.8.4.13 wskaznik\_na\_pielegniarki

```
vector< shared_ptr<Pielegniarka> >* Sala::wskaznik_na_pielegniarki [private]
```

wskaznik na baze pielegniarek

The documentation for this class was generated from the following files:

- [Sala.h](#)
- [Sala.cpp](#)

## Chapter 5

# File Documentation

### 5.1 Asortyment.txt File Reference

### 5.2 Funkcje.cpp File Reference

```
#include "Funkcje.h"
#include "Pracownicy.h"
#include "Sala.h"
#include "Pacjent.h"
```

#### Functions

- void `odczytaj_plik_grafik` (string &nazwa\_plik\_personel, vector< shared\_ptr< [Lekarz](#) >> &\_lekarze, vector< shared\_ptr< [Pielegniarka](#) >> &\_pielegniarki)  
*Odczytuje plik z grafikiem Ustala date pierwszego dnia tygodnia Funkcja odczytuje dane kazdego pracownika.*
- void `odczytaj_plik_asortyment` (string &nazwa\_pliku\_asortyment, vector< shared\_ptr< [Asortyment](#) >> &\_asortyment, [Sala](#) \*&\_sala, [Poczekalnia](#) \*&\_poczekalnia)  
*Odczytuje plik z asortymentem Liczba podana po "Poczekalnia:" jest ustawiana jako liczba miejsc w poczekalni, a liczba po "Miejsca:" jest ustawiana jako liczba lozek na sali Soru.*
- void `odczytaj_hasla` (string &nazwa\_pliku\_hasla, vector< shared\_ptr< [Lekarz](#) >> \*&wsk\_lekarze, vector< shared\_ptr< [Pielegniarka](#) >> \*&wsk\_pielegniarki)  
*Odczytuje plik z haslami do logowania dla pracownikow Kazdy wers zawiera ID pracownika i haslo.*
- bool `sprawdz_format_godziny` (string &h)  
*Sprawdza format godziny W pierwszej kolejnosci sprawdzana jest dlugosc przekazanego lancucha znakow.*
- bool `sprawdz_format_daty` (string &data)  
*Sprawdza format daty W pierwszej kolejnosci sprawdzana jest dlugosc przekazanego lancucha znakow.*
- int `menu_glowne` ()  
*Wyswietla opcje do wybrania dla uzytkownika oraz zwraca wybrana opcje.*
- int `menu_lek` ()  
*Wyswietla opcje do wybrania dla uzytkownika oraz zwraca wybrana opcje.*
- int `menu_grafik` ()  
*Wyswietla opcje do wybrania dla uzytkownika oraz zwraca wybrana opcje.*
- int `menu_grafik_piel` ()  
*Wyswietla opcje do wybrania dla uzytkownika oraz zwraca wybrana opcje.*

- int `menu_dyspozytor` ()  
*Wyswietla opcje do wybrania dla uzytkownika oraz zwraca wybrana opcje.*
- int `menu_dostawca` ()  
*Wyswietla opcje do wybrania dla uzytkownika oraz zwraca wybrana opcje.*
- int `menu_zdarzenie` ()  
*Wyswietla opcje do wybrania dla uzytkownika oraz zwraca wybrana opcje.*
- bool `czyint` (string &str)  
*Funkcja sprawdza czy przekazany lancuch znakow jest liczba Jezeli kazdy znak w parametrze jest cyfra, to lancuch znakow jest liczba.*
- bool `poprawnosc_godziny` (string &ostatnia\_h, string &ostatnia\_dat, string &podana\_h, string &podana\_dat, string &poczatek\_tygodnia)  
*Sprawdza czy podana godzina i data jest pozniejsza od ostatnio wprowadzonej godziny i daty do programu Najpierw funkcja sprawdza dlugosc parametrow 'podana\_h' i 'podana\_dat' powinny miec dlugosc rowna 5.*
- string `inkrementuj_dzien` (string &dz)  
*Na podstawie daty pierwszego dnia w tygodniu inkrementuje dni Wykorzystywana do odczytywania pliku "Grafik.txt".*
- string `dodaj_minuty` (string &godz, int &minuty)  
*Dodaje podana liczbe minut do przekazanej godziny.*
- int `roznica_godzin` (string &godz\_podana, string &godz)  
*Oblicza roznice (w minutach) miedzy dwiema godzinami.*

## 5.2.1 Function Documentation

### 5.2.1.1 czyint()

```
bool czyint (
    string & str )
```

Funkcja sprawdza czy przekazany lancuch znakow jest liczba Jezeli kazdy znak w parametrze jest cyfra, to lancuch znakow jest liczba.

#### Parameters

<i>str-</i>	sprawdzany lancuch znakow
-------------	---------------------------

#### Returns

true jezeli paramtetr jest liczba

### 5.2.1.2 dodaj\_minuty()

```
string dodaj_minuty (
    string & godz,
    int & minuty )
```

Dodaje podana liczbe minut do przekazanej godziny.

## Parameters

<i>godz-</i>	godzina
<i>minuty-</i>	liczba minut do dodania

## Returns

zmieniona godzine

### 5.2.1.3 inkrementuj\_dzien()

```
string inkrementuj_dzien (  
    string & dz )
```

Na podstawie daty pierwszego dnia w tygodniu inkrementuje dni Wykorzystywana do odczytywania pliku "Grafik.txt".

Sprawdza czy występuje inkrementacja miesięcy oraz ile dni ma miesiąc w dacie. Na tej podstawie poprawnie (zgodnie z kalendarzem) ustala datę następnego dnia

## Parameters

<i>dz-</i>	dzien względem którego funkcja ustala datę następnego dnia
------------	--

## Returns

data następnego dnia

### 5.2.1.4 menu\_dostawca()

```
int menu_dostawca ( )
```

Wyświetla opcje do wybrania dla użytkownika oraz zwraca wybraną opcję.

Wyświetlane są opcje menu dostawcy. Następnie użytkownik proszony jest o wpisanie wybranej opcji (cyfra od 1 do 3). Jeżeli wpisana odpowiedź nie jest liczbą lub nie jest na liście opcji to zwracana jest wartość 0.

## Returns

Cyfra wybranej opcji lub 0



#### 5.2.1.5 menu\_dyspozytor()

```
int menu_dyspozytor ( )
```

Wyswietla opcje do wybrania dla uzytkownika oraz zwraca wybrana opcje.

Wyswietlane sa opcje menu dyspozytora. Nastepnie uzytkownik proszony jest o wpisanie wybranej opcji (cyfra od 1 do 6). Jezeli wpisana odpowiedz nie jest liczba lub nie jest na liscie opcji to zwracana jest wartosc 0.

##### Returns

Cyfra wybranej opcji lub 0

#### 5.2.1.6 menu\_glowne()

```
int menu_glowne ( )
```

Wyswietla opcje do wybrania dla uzytkownika oraz zwraca wybrana opcje.

Wyswietlane sa opcje menu glownego. Nastepnie uzytkownik proszony jest o wpisanie wybranej opcji (cyfra od 1 do 5). Jezeli wpisana odpowiedz nie jest liczba lub nie jest na liscie opcji to zwracana jest wartosc 0.

##### Returns

Cyfra wybranej opcji lub 0

#### 5.2.1.7 menu\_grafik()

```
int menu_grafik ( )
```

Wyswietla opcje do wybrania dla uzytkownika oraz zwraca wybrana opcje.

Wyswietlane sa opcje menu opcji grafik z menu lekarza. Nastepnie uzytkownik proszony jest o wpisanie wybranej opcji (cyfra od 1 do 3). Jezeli wpisana odpowiedz nie jest liczba lub nie jest na liscie opcji to zwracana jest wartosc 0.

##### Returns

Cyfra wybranej opcji lub 0

#### 5.2.1.8 menu\_grafik\_piel()

```
int menu_grafik_piel ( )
```

Wyswietla opcje do wybrania dla uzytkownika oraz zwraca wybrana opcje.

Wyswietlane sa opcje menu grafik dla pielegniarki. Nastepnie uzytkownik proszony jest o wpisanie wybranej opcji (cyfra od 1 do 3). Jezeli wpisana odpowiedz nie jest liczba lub nie jest na liscie opcji to zwracana jest wartosc 0.

##### Returns

Cyfra wybranej opcji lub 0

#### 5.2.1.9 menu\_lek()

```
int menu_lek ( )
```

Wyswietla opcje do wybrania dla uzytkownika oraz zwraca wybrana opcje.

Wyswietlane sa opcje menu lekarza. Nastepnie uzytkownik proszony jest o wpisanie wybranej opcji (cyfra od 1 do 4). Jezeli wpisana odpowiedz nie jest liczba lub nie jest na liscie opcji to zwracana jest wartosc 0.

##### Returns

Cyfra wybranej opcji lub 0

#### 5.2.1.10 menu\_zdarzenie()

```
int menu_zdarzenie ( )
```

Wyswietla opcje do wybrania dla uzytkownika oraz zwraca wybrana opcje.

Wyswietlane sa opcje po wybraniu opcji 1 w menu dyspozytora. Nastepnie uzytkownik proszony jest o wpisanie wybranej opcji (cyfra od 1 do 2). Jezeli wpisana odpowiedz nie jest liczba lub nie jest na liscie opcji to zwracana jest wartosc 0.

##### Returns

Cyfra wybranej opcji lub 0

#### 5.2.1.11 odczytaj\_hasla()

```
void odczytaj_hasla (
    string & nazwa_pliku_hasla,
    vector< shared_ptr< Lekarz >> *& wsk_lekarze,
    vector< shared_ptr< Pielegniarka >> *& wsk_pielegniarki )
```

Odczytuje plik z haslami do logowania dla pracownikow. Kazdy wers zawiera ID pracownika i haslo.

Wartosci oddzielone przerwa. Nastepnie z pomoca wskaznikow na kontener do przechowywania inteligentnych wskaznikow na obiekty typu [Lekarz](#) i [Pielegniarka](#) zmieniane sa domyslne hasla dla pracownikow, ktorych ID jest w pliku.

## Parameters

<i>nazwa_pliku_hasla-</i>	nazwa pliku zawierającego hasła pracowników
<i>wsk_lekarze-wskaznik</i>	na kontener do przechowywania inteligentnych wskazników na obiekty typu <a href="#">Lekarz</a>
<i>wsk_pielegniarki-wskaznik</i>	na kontener do przechowywania inteligentnych wskazników na obiekty typu <a href="#">Pielegniarka</a>

## 5.2.1.12 odczytaj\_plik\_asortyment()

```
void odczytaj_plik_asortyment (
    string & nazwa_pliku_asortyment,
    vector< shared_ptr< Asortyment >> & _asortyment,
    Sala *& _sala,
    Poczekalnia *& _poczekalnia )
```

Odczytuje plik z asortymentem Liczba podana po "Poczekalnia:" jest ustawiana jako liczba miejsc w poczekalni, a liczba po "Miejsca:" jest ustawiana jako liczba łóżek na sali Soru.

Każdy wers w pliku zawiera najpierw nazwę asortymentu oraz oryginalną ilość. Wartości oddzielone są przeciskiem. Każdy element asortymentu jest dodawany do kontenera z asortymentem.

## Parameters

<i>nazwa_pliku_asortyment-</i>	nazwa pliku zawierającego poprawny plik z asortymentem
<i>_asortyment-</i>	pusty kontener do przechowywania inteligentnych wskazników na obiekty typu <a href="#">Asortyment</a>
<i>_sala-</i>	wskaznik na obiekt typu <a href="#">Sala</a>
<i>_poczekalnia-wskaznik</i>	na obiekt typu Poczekalnia

## 5.2.1.13 odczytaj\_plik\_grafik()

```
void odczytaj_plik_grafik (
    string & nazwa_plik_personel,
    vector< shared_ptr< Lekarz >> & _lekarze,
    vector< shared_ptr< Pielegniarka >> & _pielegniarki )
```

Odczytuje plik z grafikiem Ustala datę pierwszego dnia tygodnia Funkcja odczytuje dane każdego pracownika.

Jego imię, nazwisko oraz godziny pracy w każdym dniu tygodnia. Na podstawie etykiety z tytułem pracownika (lekarz/ pielęgniarka) decyduje jakiego typu ma utworzyć dynamiczny obiekt. Utworzone obiekty dodaje do odpowiednich kontenerów.

## Parameters

<i>nazwa_plik_personel-</i>	nazwa pliku zawierającego prawidłowy plik z grafikiem
<i>_lekarze-</i>	pusty kontener do przechowywania inteligentnych wskazników na obiekty typu <a href="#">Lekarz</a>
<i>_pielegniarki-</i>	pusty kontener do przechowywania inteligentnych wskazników na obiekty typu <a href="#">Pielegniarka</a>

#### 5.2.1.14 poprawnosc\_godziny()

```
bool poprawnosc_godziny (
    string & ostatnia_h,
    string & ostatnia_dat,
    string & podana_h,
    string & podana_dat,
    string & poczatek_tygodnia )
```

Sprawdza czy podana godzina i data jest pozniejsza od ostatnio wprowadzonej godziny i daty do programu Najpierw funkcja sprawdza dlugosc parametrow 'podana\_h' i 'podana\_dat' powinny miec dlugosc rowna 5.

Nastepnie sprawdzany jest format godziny (powinen byc: GG:MM) i daty (powinen byc DD.MM). Kolejnym krokiem jest sprawdzenie ile miesiac z 'poczatek\_tygodnia' ma dni, czy nie jest to miesiac luty czy grudzien. Ustalane jest czy w ciagu tygodnia nastepuje inkrementacja miesiaca. Sprawdzane jest czy podana\_dat znajduje sie w zakresie tygodnia. Na koniec ustalane jest czy ostatnia data jest mniejsza lub rowna podanej dacie i czy ostatnia godzina jest mniejsza od podanej godziny. Jezeli tak jest to zwracana jest wartosc true

##### Parameters

<i>ostatnia_h-</i>	ostatnia wprowadzona godzina zmian w programie
<i>ostatnia_dat-</i>	ostatnia wprowadzona data zmian w programie
<i>podana_h-</i>	wprowadzona godzina przez uzytkownika
<i>podana_dat-</i>	wprowadzona data przez uzytkownika
<i>poczatek_tygodnia-</i>	podana w pliku "Grafik.txt" data pierwszego dnia w tygodniu

##### Returns

true jezeli podana godzina i data jest poprawna

#### 5.2.1.15 roznica\_godzin()

```
int roznica_godzin (
    string & godz_podana,
    string & godzif )
```

Oblicza roznice (w minutach) miedzy dwiema godzinami.

Wykorzystywana do obliczenia najkrotszego czasu oczekiwania w poczekalni. Jezeli godz\_podana jest wieksza badz rowna niz godzif to zwraca jest wartosc 0. W przeciwnym wypadku zwraca jest roznica minut miedzy godzif i godz\_podana

##### Parameters

<i>godz_podana-</i>	odjemna (godzina przyjecia pacjenta)
<i>godzif-</i>	godzina wzgledem ktorej sprawdzana jest roznica (godzina od ktorej pracownik jest dostepny)

**Returns**

liczba minut

**5.2.1.16 sprawdz\_format\_daty()**

```
bool sprawdz_format_daty (  
    string & data )
```

Sprawdza format daty W pierwszej kolejnosci sprawdzana jest dlugosc przekazanego lancucha znakow.

Jezeli jest inna niz 5 zwracana jest wartosc false. Nastpnie sprawdzany jest format daty, ktory powinien byc DD.MM . Pierwsze dwa miejsca sa dla liczby reprezentujacej dzien, ostatnie dwa dla miesiaca, pomiedzy nimi powinna znajdowac sie kropka.

**Parameters**

<i>data-</i>	sprawdzany lancuch znakow
--------------	---------------------------

**Returns**

true jezeli podany lancuch spelnia warrunki daty

**5.2.1.17 sprawdz\_format\_godziny()**

```
bool sprawdz_format_godziny (  
    string & h )
```

Sprawdza format godziny W pierwszej kolejnosci sprawdzana jest dlugosc przekazanego lancucha znakow.

Jezeli jest inna niz 5 zwracana jest wartosc false. Nastpnie sprawdzany jest format godziny, ktory powinien byc GG:MM . Pierwsze dwa miejsca sa dla liczby reprezentujacej godzinie, ostatnie dwa dla minut, pomiedzy nimi powinien znajdowac sie dwukropek.

**Parameters**

<i>h-</i>	sprawdzany lancuch znakow
-----------	---------------------------

**Returns**

true jezeli podany lancuch spelnia warrunki godziny

**5.3 Funkcje.h File Reference**

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

```
#include <memory>
#include <iostream>
#include <fstream>
#include <sstream>
#include "Pracownicy.h"
#include "Sala.h"
#include <iomanip>
#include <algorithm>
```

## Macros

- `#define FUNKCJE_H`

## Functions

- `int menu_glowne ()`  
*Wyswietla opcje do wybrania dla uzytkownika oraz zwraca wybrana opcje.*
- `int menu_lek ()`  
*Wyswietla opcje do wybrania dla uzytkownika oraz zwraca wybrana opcje.*
- `int menu_grafik ()`  
*Wyswietla opcje do wybrania dla uzytkownika oraz zwraca wybrana opcje.*
- `int menu_grafik_piel ()`  
*Wyswietla opcje do wybrania dla uzytkownika oraz zwraca wybrana opcje.*
- `int menu_dyspozytor ()`  
*Wyswietla opcje do wybrania dla uzytkownika oraz zwraca wybrana opcje.*
- `int menu_dostawca ()`  
*Wyswietla opcje do wybrania dla uzytkownika oraz zwraca wybrana opcje.*
- `int menu_zdarzenie ()`  
*Wyswietla opcje do wybrania dla uzytkownika oraz zwraca wybrana opcje.*
- `bool czyint (string &str)`  
*Funkcja sprawdza czy przekazany lancuch znakow jest liczba Jezeli kazdy znak w parametrze jest cyfra, to lancuch znakow jest liczba.*
- `bool poprawnosc_godziny (string &ostatnia_h, string &ostatnia_dat, string &podana_h, string &podana_dat, string &poczatek_tygodnia)`  
*Sprawdza czy podana godzina i data jest pozniejsza od ostatnio wprowadzonej godziny i daty do programu Najpierw funkcja sprawdza dlugosc parametrow 'podana\_h' i 'podana\_dat' powinny miec dlugosc rowna 5.*
- `string inkrementuj_dzien (string &dz)`  
*Na podstawie daty pierwszego dnia w tygodniu inkrementuje dni Wykorzystywana do odczytywania pliku "Grafik.txt".*
- `void odczytaj_plik_grafik (string &nazwa_plik_personel, vector< shared_ptr< Lekarz >> &_lekarze, vector< shared_ptr< Pielegniarka >> &_pielegniarki)`  
*Odczytuje plik z grafikiem Ustala date pierwszego dnia tygodnia Funkcja odczytuje dane kazdego pracownika.*
- `void odczytaj_plik_asortyment (string &nazwa_pliku_asortyment, vector< shared_ptr< Asortyment >> &_asortyment, Sala *&_sala, Poczekalnia *&_poczekalnia)`  
*Odczytuje plik z asortymentem Liczba podana po "Poczekalnia:" jest ustawiana jako liczba miejsc w poczekalni, a liczba po "Miejsca:" jest ustawiana jako liczba lozek na sali Soru.*
- `void odczytaj_hasla (string &nazwa_pliku_hasla, vector< shared_ptr< Lekarz >> *&wsk_lekarze, vector< shared_ptr< Pielegniarka >> *&wsk_pielegniarki)`  
*Odczytuje plik z haslami do logowania dla pracownikow Kazdy wers zawiera ID pracownika i haslo.*
- `bool sprawdz_format_godziny (string &h)`  
*Sprawdza format godziny W pierwszej kolejnosci sprawdzana jest dlugosc przekazanego lancucha znakow.*
- `bool sprawdz_format_daty (string &data)`

*Sprawdza format daty. W pierwszej kolejności sprawdzana jest długość przekazanego łańcucha znaków.*

- string `dodaj_minuty` (string &godz, int &minuty)  
*Dodaje podaną liczbę minut do przekazanej godziny.*
- int `roznica_godzin` (string &godz\_podana, string &godzif)  
*Oblicza różnicę (w minutach) między dwiema godzinami.*

## 5.3.1 Macro Definition Documentation

### 5.3.1.1 FUNKCJE\_H

```
#define FUNKCJE_H
```

## 5.3.2 Function Documentation

### 5.3.2.1 czyint()

```
bool czyint (
    string & str )
```

Funkcja sprawdza czy przekazany łańcuch znaków jest liczbą. Jeżeli każdy znak w parametrze jest cyfrą, to łańcuch znaków jest liczbą.

#### Parameters

<code>str</code>	sprawdzany łańcuch znaków
------------------	---------------------------

#### Returns

true jeżeli parametr jest liczbą

### 5.3.2.2 dodaj\_minuty()

```
string dodaj_minuty (
    string & godz,
    int & minuty )
```

Dodaje podaną liczbę minut do przekazanej godziny.

## Parameters

<i>godz-</i>	godzina
<i>minuty-</i>	liczba minut do dodania

## Returns

zmieniona godzine

### 5.3.2.3 inkrementuj\_dzien()

```
string inkrementuj_dzien (
    string & dz )
```

Na podstawie daty pierwszego dnia w tygodniu inkrementuje dni Wykorzystywana do odczytywania pliku "Grafik.txt".

Sprawdza czy występuje inkrementacja miesięcy oraz ile dni ma miesiąc w dacie. Na tej podstawie poprawnie (zgodnie z kalendarzem) ustala datę następnego dnia

## Parameters

<i>dz-</i>	dzień względem którego funkcja ustala datę następnego dnia
------------	--

## Returns

data następnego dnia

### 5.3.2.4 menu\_dostawca()

```
int menu_dostawca ( )
```

Wyświetla opcje do wybrania dla użytkownika oraz zwraca wybraną opcję.

Wyświetlane są opcje menu dostawcy. Następnie użytkownik proszony jest o wpisanie wybranej opcji (cyfra od 1 do 3). Jeżeli wpisana odpowiedź nie jest liczbą lub nie jest na liście opcji to zwracana jest wartość 0.

## Returns

Cyfra wybranej opcji lub 0



#### 5.3.2.5 menu\_dyspozytor()

```
int menu_dyspozytor ( )
```

Wyswietla opcje do wybrania dla uzytkownika oraz zwraca wybrana opcje.

Wyswietlane sa opcje menu dyspozytora. Nastepnie uzytkownik proszony jest o wpisanie wybranej opcji (cyfra od 1 do 6). Jezeli wpisana odpowiedz nie jest liczba lub nie jest na liscie opcji to zwracana jest wartosc 0.

##### Returns

Cyfra wybranej opcji lub 0

#### 5.3.2.6 menu\_glowne()

```
int menu_glowne ( )
```

Wyswietla opcje do wybrania dla uzytkownika oraz zwraca wybrana opcje.

Wyswietlane sa opcje menu glownego. Nastepnie uzytkownik proszony jest o wpisanie wybranej opcji (cyfra od 1 do 5). Jezeli wpisana odpowiedz nie jest liczba lub nie jest na liscie opcji to zwracana jest wartosc 0.

##### Returns

Cyfra wybranej opcji lub 0

#### 5.3.2.7 menu\_grafik()

```
int menu_grafik ( )
```

Wyswietla opcje do wybrania dla uzytkownika oraz zwraca wybrana opcje.

Wyswietlane sa opcje menu opcji grafik z menu lekarza. Nastepnie uzytkownik proszony jest o wpisanie wybranej opcji (cyfra od 1 do 3). Jezeli wpisana odpowiedz nie jest liczba lub nie jest na liscie opcji to zwracana jest wartosc 0.

##### Returns

Cyfra wybranej opcji lub 0

#### 5.3.2.8 menu\_grafik\_piel()

```
int menu_grafik_piel ( )
```

Wyswietla opcje do wybrania dla uzytkownika oraz zwraca wybrana opcje.

Wyswietlane sa opcje menu grafik dla pielegniarki. Nastepnie uzytkownik proszony jest o wpisanie wybranej opcji (cyfra od 1 do 3). Jezeli wpisana odpowiedz nie jest liczba lub nie jest na liscie opcji to zwracana jest wartosc 0.

##### Returns

Cyfra wybranej opcji lub 0

#### 5.3.2.9 menu\_lek()

```
int menu_lek ( )
```

Wyswietla opcje do wybrania dla uzytkownika oraz zwraca wybrana opcje.

Wyswietlane sa opcje menu lekarza. Nastepnie uzytkownik proszony jest o wpisanie wybranej opcji (cyfra od 1 do 4). Jezeli wpisana odpowiedz nie jest liczba lub nie jest na liscie opcji to zwracana jest wartosc 0.

##### Returns

Cyfra wybranej opcji lub 0

#### 5.3.2.10 menu\_zdarzenie()

```
int menu_zdarzenie ( )
```

Wyswietla opcje do wybrania dla uzytkownika oraz zwraca wybrana opcje.

Wyswietlane sa opcje po wybraniu opcji 1 w menu dyspozytora. Nastepnie uzytkownik proszony jest o wpisanie wybranej opcji (cyfra od 1 do 2). Jezeli wpisana odpowiedz nie jest liczba lub nie jest na liscie opcji to zwracana jest wartosc 0.

##### Returns

Cyfra wybranej opcji lub 0

#### 5.3.2.11 odczytaj\_hasla()

```
void odczytaj_hasla (
    string & nazwa_pliku_hasla,
    vector< shared_ptr< Lekarz >> *& wsk_lekarze,
    vector< shared_ptr< Pielegniarka >> *& wsk_pielegniarki )
```

Odczytuje plik z haslami do logowania dla pracownikow Kazdy wers zawiera ID pracownika i haslo.

Wartosci oddzielone przerwa. Nastepnie z pomoca wskaznikow na kontener do przechowywania inteligentnych wskaznikow na obiekty typu [Lekarz](#) i [Pielegniarka](#) zmieniane sa domyslne hasla dla pracownikow, ktorych ID jest w pliku.

## Parameters

<i>nazwa_pliku_hasla-</i>	nazwa pliku zawierającego hasła pracowników
<i>wsk_lekarze-wskaznik</i>	na kontener do przechowywania inteligentnych wskazników na obiekty typu <a href="#">Lekarz</a>
<i>wsk_pielegniarki-wskaznik</i>	na kontener do przechowywania inteligentnych wskazników na obiekty typu <a href="#">Pielegniarka</a>

## 5.3.2.12 odczytaj\_plik\_asortyment()

```
void odczytaj_plik_asortyment (
    string & nazwa_pliku_asortyment,
    vector< shared_ptr< Asortyment >> & _asortyment,
    Sala *& _sala,
    Poczekalnia *& _poczekalnia )
```

Odczytuje plik z asortymentem Liczba podana po "Poczekalnia:" jest ustawiana jako liczba miejsc w poczekalni, a liczba po "Miejsca:" jest ustawiana jako liczba łóżek na sali Soru.

Każdy wers w pliku zawiera najpierw nazwę asortymentu oraz oryginalną ilość. Wartości oddzielone są przeciskiem. Każdy element asortymentu jest dodawany do kontenera z asortymentem.

## Parameters

<i>nazwa_pliku_asortyment-</i>	nazwa pliku zawierającego poprawny plik z asortymentem
<i>_asortyment-</i>	pusty kontener do przechowywania inteligentnych wskazników na obiekty typu <a href="#">Asortyment</a>
<i>_sala-</i>	wskaznik na obiekt typu <a href="#">Sala</a>
<i>_poczekalnia-wskaznik</i>	na obiekt typu Poczekalnia

## 5.3.2.13 odczytaj\_plik\_grafik()

```
void odczytaj_plik_grafik (
    string & nazwa_plik_personel,
    vector< shared_ptr< Lekarz >> & _lekarze,
    vector< shared_ptr< Pielegniarka >> & _pielegniarki )
```

Odczytuje plik z grafikiem Ustala datę pierwszego dnia tygodnia Funkcja odczytuje dane każdego pracownika.

Jego imię, nazwisko oraz godziny pracy w każdym dniu tygodnia. Na podstawie etykiety z tytułem pracownika (lekarz/ pielęgniarka) decyduje jakiego typu ma utworzyć dynamiczny obiekt. Utworzone obiekty dodaje do odpowiednich kontenerów.

## Parameters

<i>nazwa_plik_personel-</i>	nazwa pliku zawierającego prawidłowy plik z grafikiem
<i>_lekarze-</i>	pusty kontener do przechowywania inteligentnych wskazników na obiekty typu <a href="#">Lekarz</a>
<i>_pielegniarki-</i>	pusty kontener do przechowywania inteligentnych wskazników na obiekty typu <a href="#">Pielegniarka</a>

#### 5.3.2.14 poprawnosc\_godziny()

```
bool poprawnosc_godziny (
    string & ostatnia_h,
    string & ostatnia_dat,
    string & podana_h,
    string & podana_dat,
    string & poczatek_tygodnia )
```

Sprawdza czy podana godzina i data jest pozniejsza od ostatnio wprowadzonej godziny i daty do programu Najpierw funkcja sprawdza dlugosc parametrow 'podana\_h' i 'podana\_dat' powinny miec dlugosc rowna 5.

Nastepnie sprawdzany jest format godziny (powinen byc: GG:MM) i daty (powinen byc DD.MM). Kolejnym krokiem jest sprawdzenie ile miesiac z 'poczatek\_tygodnia' ma dni, czy nie jest to miesiac luty czy grudzien. Ustalane jest czy w ciagu tygodnia nastepuje inkrementacja miesiaca. Sprawdzane jest czy podana\_dat znajduje sie w zakresie tygodnia. Na koniec ustalane jest czy ostatnia data jest mniejsza lub rowna podanej dacie i czy ostatnia godzina jest mniejsza od podanej godziny. Jezeli tak jest to zwracana jest wartosc true

##### Parameters

<i>ostatnia_h-</i>	ostatnia wprowadzona godzina zmian w programie
<i>ostatnia_dat-</i>	ostatnia wprowadzona data zmian w programie
<i>podana_h-</i>	wprowadzona godzina przez uzytkownika
<i>podana_dat-</i>	wprowadzona data przez uzytkownika
<i>poczatek_tygodnia-</i>	podana w pliku "Grafik.txt" data pierwszego dnia w tygodniu

##### Returns

true jezeli podana godzina i data jest poprawna

#### 5.3.2.15 roznica\_godzin()

```
int roznica_godzin (
    string & godz_podana,
    string & godzif )
```

Oblicza roznice (w minutach) miedzy dwiema godzinami.

Wykorzystywana do obliczenia najkrotszego czasu oczekiwania w poczekalni. Jezeli godz\_podana jest wieksza badz rowna niz godzif to zwraca jest wartosc 0. W przeciwnym wypadku zwraca jest roznica minut miedzy godzif i godz\_podana

##### Parameters

<i>godz_podana-</i>	odjemna (godzina przyjecia pacjenta)
<i>godzif-</i>	godzina wzgledem ktorej sprawdzana jest roznica (godzina od ktorej pracownik jest dostepny)

**Returns**

liczba minut

**5.3.2.16 sprawdz\_format\_daty()**

```
bool sprawdz_format_daty (  
    string & data )
```

Sprawdza format daty W pierwszej kolejności sprawdzana jest długość przekazanego łańcucha znaków.

Jeżeli jest inna niż 5 zwracana jest wartość false. Następnie sprawdzany jest format daty, który powinien być DD.MM . Pierwsze dwa miejsca są dla liczby reprezentującej dzień, ostatnie dwa dla miesiąca, pomiędzy nimi powinna znajdować się kropka.

**Parameters**

<i>data-</i>	sprawdzany łańcuch znaków
--------------	---------------------------

**Returns**

true jeżeli podany łańcuch spełnia warunki daty

**5.3.2.17 sprawdz\_format\_godziny()**

```
bool sprawdz_format_godziny (  
    string & h )
```

Sprawdza format godziny W pierwszej kolejności sprawdzana jest długość przekazanego łańcucha znaków.

Jeżeli jest inna niż 5 zwracana jest wartość false. Następnie sprawdzany jest format godziny, który powinien być GG:MM . Pierwsze dwa miejsca są dla liczby reprezentującej godzinę, ostatnie dwa dla minut, pomiędzy nimi powinien znajdować się dwukropek.

**Parameters**

<i>h-</i>	sprawdzany łańcuch znaków
-----------	---------------------------

#### Returns

true jeżeli podany lancuch spełnia warunki godziny

## 5.4 Grafik.txt File Reference

## 5.5 Hasla.txt File Reference

## 5.6 Pacjent.cpp File Reference

```
#include "Funkcje.h"
#include "Pracownicy.h"
#include "Sala.h"
#include "Pacjent.h"
```

### Functions

- ostream & [operator<<](#) (ostream &s, [Pacjent](#) &pacjent)

### 5.6.1 Function Documentation

#### 5.6.1.1 [operator<<\(\)](#)

```
ostream& operator<< (
    ostream & s,
    Pacjent & pacjent )
```

## 5.7 Pacjent.h File Reference

```
#include <string>
#include <vector>
#include <memory>
#include <iostream>
#include <fstream>
#include <sstream>
#include <iomanip>
#include <algorithm>
```

### Classes

- class [Pacjent](#)

## Macros

- #define [PACJENT\\_H](#)

### 5.7.1 Macro Definition Documentation

#### 5.7.1.1 PACJENT\_H

```
#define PACJENT_H
```

## 5.8 Pracownicy.cpp File Reference

```
#include "Pracownicy.h"  
#include "Sala.h"  
#include "Funkcje.h"  
#include "Pacjent.h"
```

## Functions

- ostream & [operator<<](#) (ostream &s, [Pracownik](#) &pracownik)

### 5.8.1 Function Documentation

#### 5.8.1.1 operator<<()

```
ostream& operator<< (  
    ostream & s,  
    Pracownik & pracownik )
```

## 5.9 Pracownicy.h File Reference

```
#include <string>  
#include <vector>  
#include <memory>  
#include <iostream>  
#include <fstream>  
#include <sstream>  
#include <iomanip>  
#include <algorithm>
```

## Classes

- struct [dzien](#)
- class [Pracownik](#)
- class [Lekarz](#)
- class [Pielegniarka](#)

## Macros

- `#define` [PRACOWNICY\\_H](#)

## 5.9.1 Macro Definition Documentation

### 5.9.1.1 PRACOWNICY\_H

```
#define PRACOWNICY_H
```

## 5.10 resource.h File Reference

## 5.11 Sala.cpp File Reference

```
#include "Pracownicy.h"  
#include "Pacjent.h"  
#include "Sala.h"  
#include "Funkcje.h"
```

## Functions

- ostream & [operator<<](#) (ostream &s, [Asortyment](#) &asortyment)
- ostream & [operator<<](#) (ostream &s, [Sala](#) &sala)

### 5.11.1 Function Documentation

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

```
ostream& operator<< (  
    ostream & s,  
    Asortyment & asortyment )
```



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

```
ostream& operator<< (
    ostream & s,
    Sala & sala )
```

## 5.12 Sala.h File Reference

```
#include <string>
#include <vector>
#include <memory>
#include <iostream>
#include <fstream>
#include <sstream>
#include <algorithm>
```

### Classes

- class [Asortyment](#)
- class [Poczekalnia](#)
- class [Sala](#)

### Macros

- #define [SALA\\_H](#)

### 5.12.1 Macro Definition Documentation

#### 5.12.1.1 SALA\_H

```
#define SALA_H
```

## 5.13 SOR.cpp File Reference

```
#include "Pacjent.h"
#include "Pracownicy.h"
#include "Funkcje.h"
#include "Sala.h"
#include <string>
#include <vector>
#include <memory>
#include <iostream>
#include <fstream>
#include <sstream>
#include <algorithm>
```

## Functions

- `int main ()`

### 5.13.1 Function Documentation

#### 5.13.1.1 `main()`

```
int main ( )
```

to moze byc przyczyna wielu bledow

