projekt plan

Generated by Doxygen 1.13.1

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 lekcja Class Reference

Klasa reprezentujaca pojedyncza lekcje.

#include <klasy.h>

Public Member Functions

- lekcja (std::string przedmiot, std::string typ, int parzystosc, sala *nr, nauczyciel *n)
- void pokaz_dane ()

  Zwraca dane lekcji do cout.
- void zwroc_dane_p (std::string &p) const

  Zwraca nazwe przedmoitu przez referencje.
- std::string getPrzedmiot () const

  Zwraca nazwe przedmoitu.
- std::string getTyp () const

  Zwraca nazwtype przedmoitu.
- int getParzystosc () const

  Zwraca parzystosc przedmoitu.
- nauczyciel * getNauczyciel () const

  Zwraca prowadzacego przedmoitu.
- sala * getSala () const

  Zwraca sale przedmoitu.

Protected Attributes

- std::string przedmiot
- std::string typ
- int parzystosc
- sala * klasa
- nauczyciel * prowadzacy

4.1.1 Detailed Description

Klasa reprezentujaca pojedyncza lekcje.

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

- Project_ppk2/klasy.h

4.2 lista< t > Class Template Reference

Klasa listy jednokierunkowej do przechowywania wskaznikow na obiekty.

#include <klasy.h>

Public Member Functions

- Node< t > * getHead () const
- void push_back (t val)

  Dodaje element na koniec listy.
- void remove (t val)

  Usuwa wszystkie elementy listy i zwalnia pamiec.

4.2.1 Detailed Description

template<typename t>
class lista< t >

Klasa listy jednokierunkowej do przechowywania wskaznikow na obiekty.

Template Parameters

t	Typ wskaznika na obiekt.

4.2.2 Member Function Documentation

4.2.2.1 push_back()

template<typename t>
void lista< t >::push_back (
t val) [inline]

Dodaje element na koniec listy.

Parameters

val	Wartosc do dodania.

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

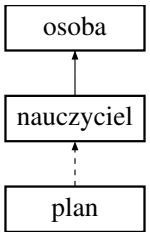
- Project_ppk2/klasy.h

4.3 nauczyciel Class Reference

Klasa reprezentujaca nauczyciela, dziedziczy po osobie.

#include <klasy.h>

Inheritance diagram for nauczyciel:



Public Member Functions

- nauczyciel (const std::string imie, const std::string nazwisko, const std::string przedmiot_↩ prowadzony)
- void pokaz_dane ()

  Zwraca dane nauczyciela do cout.
- void zwroc_dane (std::string &i, std::string &n) const

  Zwraca dane nauczyciela przez referencje.
- std::string getPrzedmiot_prowadzony () const

  Zwraca przedmiot prowadzony nauczyciela.

Public Member Functions inherited from osoba

- osoba (const std::string &imie, const std::string &nazwisko)
- void inicjaly ()

  Zwraca inicjaly osoby.
- std::string imiee () const

  Zwraca imie osoby.
- std::string nazwiskoo () const

  Zwraca nazwisko osoby.

4.3.1 Detailed Description

Klasa reprezentujaca nauczyciela, dziedziczy po osobie.

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

- Project_ppk2/klasy.h

4.4 Node< t > Struct Template Reference

Struktura wezla listy jednokierunkowej.

#include <klasy.h>

Public Member Functions

- Node (t val)

Public Attributes

- t value
- Node * next

4.4.1 Detailed Description

template<typename t>
struct Node< t >

Struktura wezla listy jednokierunkowej.

Template Parameters

t	Typ danych przechowywany w liscie.

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

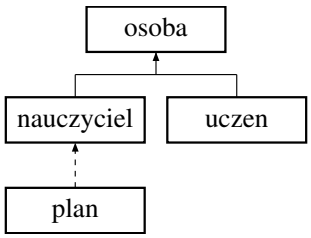
- Project_ppk2/klasy.h

4.5 osoba Class Reference

Klasa bazowa dla osob nauczycieli i uczniow.

#include <klasy.h>

Inheritance diagram for osoba:



Public Member Functions

- osoba (const std::string &imie, const std::string &nazwisko)
- void inicjaly ()

  Zwraca inicjaly osoby.
- std::string imiee () const

  Zwraca imie osoby.
- std::string nazwiskoo () const

  Zwraca nazwisko osoby.

4.5.1 Detailed Description

Klasa bazowa dla osob nauczycieli i uczniow.

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

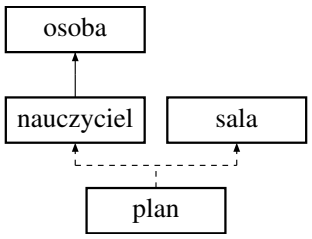
- Project_ppk2/klasy.h

4.6 plan Class Reference

Klasa odpowiedzialna za reprezentacje planu lekcji jako tabeli.

#include <klasy.h>

Inheritance diagram for plan:



Public Member Functions

- plan ()
  Inicjalizuje szkielet planu.
- void pokaz_plan ()
  Wyswietla plan.
- void czy_sa_myslniki (int &x)
  Sprawdza czy na linijce x nie ma myslnikow, by nic nie dac na szkielet planu.
- void dodaj_do_planu (int h, int m, std::string dzien, std::string przedmiot, std::string typ, int ht, int mt, lista< lekcja * > &lekcje, int nr_gr, bool laduj)
  Dodaje lekcje do planu.

Public Attributes

- std::string tabela [57][16]

Additional Inherited Members

Protected Member Functions inherited from nauczyciel

- nauczyciel (const std::string imie, const std::string nazwisko, const std::string przedmiot_prowadzony)
- void pokaz_dane ()
  Zwraca dane nauczyciela do cout.
- void zwroc_dane (std::string &i, std::string &n) const
  Zwraca dane nauczyciela przez referencje.
- std::string getPrzedmiot_prowadzony () const
  Zwraca przedmiot prowadzony nauczyciela.

Protected Member Functions inherited from osoba

- osoba (const std::string &imie, const std::string &nazwisko)
- void inicjaly ()

  Zwraca inicjaly osoby.
- std::string imiee () const

  Zwraca imie osoby.
- std::string nazwiskoo () const

  Zwraca nazwisko osoby.

Protected Member Functions inherited from sala

- sala (std::string typ, std::string numer)
- void pokaz_dane ()

  Zwraca dane sali do cout.
- void pokaz_dane_2 ()

  Zwraca dane sali do cout z endl na koncu.
- void zwroc_dane_s (std::string &s) const

  Zwraca nummer sali przez referencje.
- std::string getNumer () const

  Zwraca numer sali.
- std::string getTyp () const

  Zwraca typ sali.

4.6.1 Detailed Description

Klasa odpowiedzialna za reprezentacje planu lekcji jako tabeli.

4.6.2 Member Function Documentation

4.6.2.1 dodaj_do_planu()

```
void plan::dodaj_do_planu (
            int h,
            int m,
            std::string dzien,
            std::string przedmiot,
            std::string typ,
            int ht,
            int mt,
            lista< lekcja * > & lekcje,
            int nr_gr,
            bool laduj)   [inline]
```

Dodaje lekcje do planu.

Parameters

h	Godzina rozpoczecia (np. 8).
m	Minuta rozpoczecia (np. 15).

Parameters

dzien	Dzien tygodnia.
przedmiot	Nazwa przedmiotu.
typ	Typ zajec (CW, WYK, LAB...).
ht	Liczba godzin trwania.
mt	Liczba minut trwania.
lekcje	Lista dostepnych lekcji do wyboru.
nr_gr	Numer grupy.
laduj	czy nalezy ja ladowac do pliku zapis_planu.

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

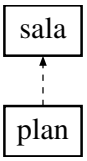
- Project_ppk2/klasy.h

4.7 sala Class Reference

Klasa reprezentujaca sale lekcyjna.

#include <klasy.h>

Inheritance diagram for sala:



Public Member Functions

- sala (std::string typ, std::string numer)
- void pokaz_dane ()
  Zwraca dane sali do cout.
- void pokaz_dane_2 ()
  Zwraca dane sali do cout z endl na koncu.
- void zwroc_dane_s (std::string &s) const
  Zwraca nummer sali przez referencje.
- std::string getNumer () const
  Zwraca numer sali.
- std::string getTyp () const
  Zwraca typ sali.

4.7.1 Detailed Description

Klasa reprezentujaca sale lekcyjna.

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

- Project_ppk2/klasy.h

4.8 struktura_planow Struct Reference

Przechowuje wszystkie plany lekcji dla grup 1 6.

#include <klasy.h>

Public Member Functions

- plan * get_plan (int numer_grupy)

Public Attributes

- plan p1
- plan p2
- plan p3
- plan p4
- plan p5
- plan p6

4.8.1 Detailed Description

Przechowuje wszystkie plany lekcji dla grup 1 6.

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

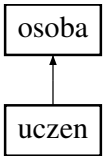
- Project_ppk2/klasy.h

4.9 uczen Class Reference

Klasa reprezentujaca ucznia, dziedziczy po osobie.

#include <klasy.h>

Inheritance diagram for uczen:



Public Member Functions

- uczen (const std::string imie, const std::string nazwisko, const std::string numer_gr)
- void pokaz_dane ()

  Zwraca dane ucznia do cout.
- void zwroc_dane (std::string &i, std::string &n) const

  Zwraca dane ucznia przez referencje.
- std::string getNumer_gr () const

  Zwraca nr_gr ucznia.

Public Member Functions inherited from osoba

- osoba (const std::string &imie, const std::string &nazwisko)
- void inicjaly ()

  Zwraca inicjaly osoby.
- std::string imiee () const

  Zwraca imie osoby.
- std::string nazwiskoo () const

  Zwraca nazwisko osoby.

4.9.1 Detailed Description

Klasa reprezentujaca ucznia, dziedziczy po osobie.

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

- Project_ppk2/klasy.h

Chapter 5

File Documentation

5.1 Project_ppk2/klasy.h File Reference

#include <iostream>
#include <string>
#include <sstream>
#include <fstream>
#include <iomanip>

Classes

- struct Node< t >
  Struktura wezla listy jednokierunkowej.
- class lista< t >
  Klasa listy jednokierunkowej do przechowywania wskaznikow na obiekty.
- class osoba
  Klasa bazowa dla osob nauczycieli i uczniow.
- class uczen
  Klasa reprezentujaca ucznia, dziedziczy po osobie.
- class nauczyciel
  Klasa reprezentujaca nauczyciela, dziedziczy po osobie.
- class sala
  Klasa reprezentujaca sale lekcyjna.
- class lekcja
  Klasa reprezentujaca pojedyncza lekcje.
- class plan
  Klasa odpowiedzialna za reprezentacje planu lekcji jako tabeli.
- struct struktura_planow
  Przechowuje wszystkie plany lekcji dla grup 1 6.

Functions

- std::ostream & operator<< (std::ostream &os, const uczen *u)
  Przeciazony operator << dla klasy uczen.
- std::ostream & operator<< (std::ostream &os, const nauczyciel *n)
  Przeciazony operator << dla klasy nauczyciel.
- std::ostream & operator<< (std::ostream &os, const sala *s)
  Przeciazony operator << dla klasy sala.
- std::ostream & operator<< (std::ostream &os, const lekcja *l)
  Przeciazony operator << dla klasy lekcja.

5.1.1 Function Documentation

5.1.1.1 operator<<() [1/4]

```
std::ostream & operator<< (
            std::ostream & os,
            const lekcja * l)
```

Przeciazony operator << dla klasy lekcja.

Parameters

os	Strumien wyjsciowy.
u	Wskaznik na lekcje.

Returns

Referencja do strumienia wyjsciowego.

5.1.1.2 operator<<() [2/4]

```
std::ostream & operator<< (
            std::ostream & os,
            const nauczyciel * n)
```

Przeciazony operator << dla klasy nauczyciel.

Parameters

os	Strumien wyjsciowy.
u	Wskaznik na nauczyciela.

Returns

Referencja do strumienia wyjsciowego.

5.1.1.3 operator<<() [3/4]

```
std::ostream & operator<< (
            std::ostream & os,
            const sala * s)
```

Przeciazony operator << dla klasy sala.

Parameters

os	Strumien wyjsciowy.
u	Wskaznik na sale.

Returns

Referencja do strumienia wyjsciowego.

5.1.1.4 operator<<() [4/4]

```
std::ostream & operator<< (
                std::ostream & os,
                const uczen * u)
```

Przeciazony operator << dla klasy uczen.

Parameters

os	Strumien wyjsciowy.
u	Wskaznik na ucznia.

Returns

Referencja do strumienia wyjsciowego.

5.2 klasy.h

Go to the documentation of this file.

```
00001
00002
00003 #ifndef KLASY_H
00004 #define KLASY_H
00005
00006 #include <iostream>
00007 #include <string>
00008 #include <sstream>
00009 #include <fstream>
00010 #include <iomanip>
00011
00016 template <typename t>
00017 struct Node {
00018     t value;
00019     Node* next;
00020
00021     Node(t val) : value(val), next(nullptr) {}
00022 };
00023
00028 template <typename t>
00029 class lista {
00030     Node<t>* head;
00031     Node<t>* tail;
00032
00033 public:
00034     lista() : head(nullptr), tail(nullptr) {}
00035
00036     Node<t>* getHead() const {
00037         return head;
00038     }
00039
00043     void push_back(t val) {
```

```
00044         Node<t>* newNode = new Node<t>(val);
00045
00046         if (tail == nullptr) {
00047             head = tail = newNode;
00048         }
00049         else {
00050             tail->next = newNode;
00051             tail = newNode;
00052         }
00053     }
00054
00056     void remove(t val) {
00057         if (head == nullptr) return;
00058
00059         if (head->value == val) {
00060             Node<t>* temp = head;
00061             head = head->next;
00062             if (temp == tail) {
00063                 tail = nullptr;
00064             }
00065             delete temp;
00066             return;
00067         }
00068
00069         Node<t>* current = head;
00070         while (current->next != nullptr && current->next->value != val) {
00071             current = current->next;
00072         }
00073
00074         if (current->next != nullptr) {
00075             Node<t>* temp = current->next;
00076             current->next = temp->next;
00077             if (temp == tail) {
00078                 tail = current;
00079             }
00080             delete temp;
00081         }
00082     }
00083
00084     ~lista() {
00085         Node<t>* current = head;
00086         while (current != nullptr) {
00087             Node<t>* next = current->next;
00088             delete current;
00089             current = next;
00090         }
00091     }
00092 };
00093
00094
00098 class osoba {
00099     std::string imie, nazwisko;
00100 public:
00101     osoba() {}
00102     osoba(const std::string& imie, const std::string& nazwisko) : imie(imie), nazwisko(nazwisko){
00103
00104     }
00105
00107     void inicjaly() {
00108         std::cout « std::endl;
00109         std::cout « "Imie: " « imie « std::endl « "Nazwisko: " « nazwisko « std::endl;
00110     }
00111
00113     std::string imiee() const{
00114         return imie;
00115     }
00117     std::string nazwiskoo() const{
00118         return nazwisko;
00119     }
00120
00121     ~osoba() {};
00122 };
00123
00127 class uczen : public osoba{
00128     std::string numer_gr;
00129 public:
00130     uczen(const std::string imie, const std::string nazwisko, const std::string numer_gr) : osoba(imie, nazwisko),
      numer_gr(numer_gr) {
00131
00132     }
00133
00135     void pokaz_dane() {
00136         osoba::inicjaly();
00137         std::cout « "Grupa dziekanska: "«numer_gr « std::endl;
00138     }
00140     void zwroc_dane(std::string& i, std::string& n) const{
00141         i = osoba::imiee();
```

```
00142         n = osoba::nazwiskoo();
00143     }
00145     std::string getNumer_gr() const {
00146         return numer_gr;
00147     }
00148
00149     ~uczen() {};
00150 };
00151
00155 class nauczyciel : public osoba {
00156     std::string przedmiot_prowadzony;
00157 public:
00158     nauczyciel() {}
00159     nauczyciel(const std::string imie, const std::string nazwisko, const std::string przedmiot_prowadzony) : osoba(imie,
      nazwisko), przedmiot_prowadzony(przedmiot_prowadzony){
00160
00161     }
00163     void pokaz_dane() {
00164         osoba::inicjaly();
00165         std::cout « "Przedmiot prowadzony: " « przedmiot_prowadzony « std::endl;
00166     }
00168     void zwroc_dane(std::string& i, std::string& n) const {
00169         i = osoba::imiee();
00170         n = osoba::nazwiskoo();
00171     }
00173     std::string getPrzedmiot_prowadzony() const {
00174         return przedmiot_prowadzony;
00175     }
00176
00177     ~nauczyciel() {};
00178 };
00179
00183 class sala {
00184     std::string typ; //wykladowa, laboratoiryjna, komputerowa, ogolna...
00185     std::string numer;
00186 public:
00187     sala() {}
00188     sala(std::string typ, std::string numer) : typ(typ), numer(numer) {
00189
00190     }
00191
00193     void pokaz_dane() {
00194         std::cout « "Numer sali: " « numer « std::endl;
00195         std::cout « "Typ sali: " « typ « std::endl;
00196     }
00198     void pokaz_dane_2() {
00199         std::cout « "Numer sali: " « numer « std::endl;
00200         std::cout « "Typ sali: " « typ « std::endl;
00201         std::cout « std::endl;
00202     }
00204     void zwroc_dane_s(std::string& s) const {
00205         s = numer;
00206     }
00208     std::string getNumer() const {
00209         return numer;
00210     }
00212     std::string getTyp() const {
00213         return typ;
00214     }
00215
00216     ~sala() {};
00217 };
00218
00222 class lekcja {
00223 protected:
00224     std::string przedmiot, typ;
00225     int parzystosc; //0 nie wazne, 1 parzysty dzien, 2 nieparzysty
00226     sala* klasa;
00227     nauczyciel* prowadzacy;
00228 public:
00229     lekcja() {}
00230     lekcja(std::string przedmiot, std::string typ, int parzystosc, sala* nr, nauczyciel* n) : przedmiot(przedmiot), typ(typ),
      parzystosc(parzystosc), prowadzacy(n), klasa(nr) {
00231
00232     }
00233
00235     void pokaz_dane() {
00236         std::cout « "----------------" « std::endl;
00237         std::cout « przedmiot « ":" « std::endl « "Typ: " « typ « std::endl;
00238         if (parzystosc == 0) {
00239             std::cout « "Co tydzien" « std::endl;
00240         }
00241         else if (parzystosc == 1) {
00242             std::cout « "Tylko parzyste tygodnie" « std::endl;
00243         }
00244         else if (parzystosc == 2) {
00245             std::cout « "Tylko nieparzyste tygodnie" « std::endl;
```

```
00246         }
00247         std::cout « "Prowadzacy: ";
00248         prowadzacy->pokaz_dane();
00249         klasa->pokaz_dane();
00250         std::cout « "-----------------" « std::endl;
00251     }
00253     void zwroc_dane_p(std::string& p) const {
00254         p = przedmiot;
00255     }
00257     std::string getPrzedmiot() const {
00258         return przedmiot;
00259     }
00261     std::string getTyp() const {
00262         return typ;
00263     }
00265     int getParzystosc() const {
00266         return parzystosc;
00267     }
00269     nauczyciel* getNauczyciel() const {
00270         return prowadzacy;
00271     }
00273     sala* getSala() const {
00274         return klasa;
00275     }
00276
00277     ~lekcja() {};
00278 };
00279
00283 class plan : protected nauczyciel, protected sala {
00284     int w = 57;
00285     int k = 16;
00286 public:
00287     std::string tabela[57][16];
00288
00290     plan() { //tworzy szkielet planu
00291         tabela[0][0] = "Godz.";
00292         tabela[0][2] = "      Poniedzialek";
00293         tabela[0][4] = "          Wtorek";
00294         tabela[0][6] = "          Sroda";
00295         tabela[0][8] = "        Czwartek";
00296         tabela[0][10] = "         Piatek";
00297         tabela[0][12] = "          Sobota";
00298         tabela[0][14] = "        Niedziela";
00299
00300         //myslniki poziome
00301         std::stringstream myslniki, myslinik0;
00302         myslniki « std::setw(25) « std::setfill('-') « "-"; //to dla wierszy
00303         myslinik0 « std::setw(6) « std::setfill('-') « "-"; //to dla godziny
00304
00305         for (int j = 1; j < w; j += 5) {
00306             tabela[j][0] = myslinik0.str();
00307             for (int i = 1; i < k; i++) {
00308                 if (i % 2 == 1) {
00309
00310                 }
00311                 else {
00312                     tabela[j][i] = myslniki.str();
00313                 }
00314             }
00315         }
00316         // myslniki pionowe
00317         for (int j = 1; j < k; j += 2) { //to dla |
00318             for (int i = 0; i < w; i++) {
00319                 if (i == 1) {
00320                     tabela[1][j] = "-";
00321                 }
00322                 else {
00323                     tabela[i][j] = "|";
00324                 }
00325             }
00326         }
00327
00328         //zajmuje sie godzinami
00329         int h = 8;
00330         int m = 00;
00331         for (int i = 2; i <= 55; i++) {
00332             if (m == 60) { // jak jest 60 minuta to zwieksza godzine o 1
00333                 h++;
00334                 m = 00;
00335                 i++;
00336             }
00337
00338             std::stringstream oss;
00339             oss « std::setw(2) « std::setfill('0') « h « ":" « std::setw(2) « std::setfill('0') « m;
00340             if (m == 00) {
00341                 tabela[i][0] = oss.str();
00342             }
```

```
00343
00344             else { //tu sa minuty(co 15) mozna zakomentowac
00345                 tabela[i][0] = oss.str();
00346             }
00347             m += 15;
00348         }
00349     }
00350
00352     void pokaz_plan() {
00353         std::cout « "Plan lekcji:" « std::endl « std::endl;
00354         for (int x = 0; x < w; x++) {
00355             for (int y = 0; y < k; y++) {
00356                 if (y == 0) {
00357                     std::cout « std::setw(6) « std::left « tabela[x][y]; //wyswietla godziny
00358                 }
00359                 else if (y % 2 == 1) { //wyswietla |
00360                     std::cout « tabela[x][y];
00361                     if (y == 15) {
00362                         std::cout « std::endl;
00363                     }
00364                 }
00365                 else {
00366                     std::cout « std::setw(25) « std::left « tabela[x][y]; //wyswietla komorki ogolnie
00367                 }
00368             }
00369         }
00370     }
00371
00373     void czy_sa_myslniki(int& x) {
00374         if ((x - 1) % 5 == 0) {
00375             x++;
00376         }
00377     }
00378
00391     void dodaj_do_planu(int h, int m, std::string dzien, std::string przedmiot, std::string typ, int ht, int mt,
    lista<lekcja*>& lekcje, int nr_gr, bool laduj) {
00392         int x, y, i_minuty;
00393         std::stringstream fale;
00394         std::string nr_sali, imie, nazwisko;
00395
00396         fale « std::setw(25) « std::setfill('~') « "~";
00397
00398         if (dzien == "poniedzialek" || dzien == "pon") {
00399             y = 2;
00400         }
00401         else if (dzien == "wtorek" || dzien == "wt") {
00402             y = 4;
00403         }
00404         else if (dzien == "sroda" || dzien == "sr") {
00405             y = 6;
00406         }
00407         else if (dzien == "czwartek" || dzien == "czw") {
00408             y = 8;
00409         }
00410         else if (dzien == "piatek" || dzien == "pt") {
00411             y = 10;
00412         }
00413         else if (dzien == "sobota" || dzien == "sob") {
00414             y = 12;
00415         }
00416         else if (dzien == "niedziela" || dzien == "nd") {
00417             y = 14;
00418         }
00419         else {
00420             std::cout « "Podano zly dzien" « std::endl;
00421             std::exit(1);
00422         }
00423
00424         if (h < 8 || h > 18 || (ht == 0 && mt != 45)) {
00425             std::cout « "Podano zla godzine" « std::endl;
00426             std::exit(1);
00427         }
00428
00429         if (m == 00) {
00430             i_minuty = 0;
00431         }
00432         else if (m == 15) {
00433             i_minuty = 1;
00434         }
00435         else if (m == 30) {
00436             i_minuty = 2;
00437         }
00438         else if (m == 45) {
00439             i_minuty = 3;
00440         }
00441         else {
00442             std::cout « "Podano zla minute" « std::endl;
```

```
00443           std::exit(1);
00444       }
00445
00446       lekcja* zgodna = nullptr;
00447       for (Node<lekcja*>* curr = lekcje.getHead(); curr != nullptr; curr = curr->next) {
00448           if (curr->value->getPrzedmiot() == przedmiot && curr->value->getTyp() == typ) {
00449               zgodna = curr->value;
00450               break;
00451           }
00452       }
00453       if (zgodna == nullptr) {
00454           std::cout « "Nie znaleziono lekcji o nazwie: " « przedmiot « std::endl;
00455           return;
00456       }
00457       nauczyciel* prow = zgodna->getNauczyciel();
00458       sala* sala_ss = zgodna->getSala();
00459       prow->zwroc_dane(imie, nazwisko);
00460       sala_ss->zwroc_dane_s(nr_sali);
00461
00462       x = 2 + 5 * (-8 + h) + i_minuty; //gorna granica lekcji
00463       tabela[x][y] = fale.str();
00464       x++;
00465
00466       //nazwa lekcji
00467       int rozmiar_p = przedmiot.size();
00468       czy_sa_myslniki(x);
00469       if (rozmiar_p <= 25) {
00470           std::stringstream nazwa_lekcji;
00471           nazwa_lekcji « std::setw(25) « std::left « przedmiot;
00472           tabela[x][y] = nazwa_lekcji.str();
00473           x++;
00474           x++;
00475       }
00476       else {
00477           std::string pierwsze_25 = przedmiot.substr(0, 25);
00478           std::string drugie_25 = przedmiot.substr(25, std::min(25, rozmiar_p - 25));
00479
00480           std::stringstream linia1, linia2;
00481           linia1 « std::setw(25) « pierwsze_25;
00482           linia2 « std::setw(25) « std::left « drugie_25;
00483
00484           tabela[x][y] = linia1.str();
00485           x++;
00486           czy_sa_myslniki(x);
00487           tabela[x][y] = linia2.str();
00488           x++;
00489       }
00490
00491       //typ lekcji, sala
00492       czy_sa_myslniki(x);
00493       std::stringstream typ_ss;
00494       typ_ss « typ « "      Sala: " « nr_sali « std::setw(12);
00495       tabela[x][y] = typ_ss.str();
00496       x++;
00497
00498       //nauczyciel
00499       czy_sa_myslniki(x);
00500       std::stringstream imie_ss, nazwisko_ss;
00501       imie_ss « imie « std::setw(25);
00502       tabela[x][y] = imie_ss.str();
00503       x++;
00504       czy_sa_myslniki(x);
00505       nazwisko_ss « nazwisko « std::setw(25);
00506       tabela[x][y] = nazwisko_ss.str();
00507
00508       x = 2 + 5 * (-8 + h) + i_minuty;
00509
00510       //dolna granica lekcji
00511       int ile = ht * 4 + mt / 15;
00512       int x_specjal = x;
00513       int przesuniecie = 0;
00514
00515       for (int i = 0; i < ile; ++i) { // to robi ze prawidolo sie wyswietla linia konca lekcji
00516           int test = x_specjal + i + przesuniecie;
00517           if ((test - 1) % 5 == 0) {
00518               przesuniecie++;
00519           }
00520       }
00521
00522       x_specjal = x + ile + przesuniecie;
00523       czy_sa_myslniki(x_specjal);
00524       tabela[x_specjal][y] = fale.str();
00525
00526       if (laduj == false) {
00527           std::ofstream zapis_planu("zapis_planu.txt", std::ios::app); //dopisuje na koniec
00528           if (zapis_planu) {
00529               zapis_planu « "Dzien: " « dzien « std::endl;
```

```
00530                 zapis_planu « "Godzina rozpoczecia: " « h « " " « m « std::endl;
00531                 zapis_planu « "Trwa: " « ht « " " « mt « std::endl;
00532                 zapis_planu « "Przedmiot: " « przedmiot « std::endl;
00533                 zapis_planu « "Typ: " « typ « std::endl;
00534                 zapis_planu « "Nr grupy: " « nr_gr « std::endl;
00535                 zapis_planu « "--------------------------" « std::endl;
00536             }
00537         }
00538     }
00539
00540     ~plan() {};
00541 };
00542
00543 std::ostream& operator«(std::ostream& os, const uczen* u);
00544 std::ostream& operator«(std::ostream& os, const nauczyciel* n);
00545 std::ostream& operator«(std::ostream& os, const sala* s);
00546 std::ostream& operator«(std::ostream& os, const lekcja* l);
00547
00551 struct struktura_planow {
00552     plan p1, p2, p3, p4, p5, p6;
00553
00554     plan* get_plan(int numer_grupy) {
00555         switch (numer_grupy) {
00556         case 1: return &p1;
00557         case 2: return &p2;
00558         case 3: return &p3;
00559         case 4: return &p4;
00560         case 5: return &p5;
00561         case 6: return &p6;
00562         default: return nullptr;
00563         }
00564     }
00565 };
00566
00567 #endif
```

5.3 Project_ppk2/lista.cpp File Reference

#include <iostream>
#include <algorithm>
#include <cctype>
#include <fstream>
#include "klasy.h"

Functions

- void inicjalizuj (lista< nauczyciel * > &nauczyciele, lista< uczen * > &uczniowie, lista< lekcja * > &lekcje, lista< sala * > &sale, struktura_planow &plany)

  Wczytuje nauczycieli, uczniow, lekcje, sale i plan.
- void usun_listy (lista< nauczyciel * > &nauczyciele, lista< uczen * > &uczniowie, lista< lekcja * > &lekcje, lista< sala * > &sale)

  Usuwa dynamicznie zaalokowane dane z list.
- std::ostream & operator<< (std::ostream &os, const uczen *u)

  Przeciazony operator << dla klasy uczen.
- std::ostream & operator<< (std::ostream &os, const nauczyciel *n)

  Przeciazony operator << dla klasy nauczyciel.
- std::ostream & operator<< (std::ostream &os, const sala *s)

  Przeciazony operator << dla klasy sala.
- std::ostream & operator<< (std::ostream &os, const lekcja *l)

  Przeciazony operator << dla klasy lekcja.

5.3.1 Function Documentation

5.3.1.1 inicjalizuj()

```
void inicjalizuj (
        lista< nauczyciel * > & nauczyciele,
        lista< uczen * > & uczniowie,
        lista< lekcja * > & lekcje,
        lista< sala * > & sale,
        struktura_planow & plany)
```

Wczytuje nauczycieli, uczniow, lekcje, sale i plan.

Inicjalizuje dane do list

Parameters

nauczyciele	Do tej listy sa wczytywani nauczyciele
uczniowie	Do tej listy sa wczytywani uczniowie
lekcje	Do tej listy sa wczytywane lekcje
sale	Do tej listy sa wczytywane sale

Author

Dyba

5.3.1.2 operator<<() [1/4]

```
std::ostream & operator<< (
        std::ostream & os,
        const lekcja * l)
```

Przeciazony operator << dla klasy lekcja.

Parameters

os	Strumien wyjsciowy.
u	Wskaznik na lekcje.

Returns

Referencja do strumienia wyjsciowego.

5.3.1.3 operator<<() [2/4]

```
std::ostream & operator<< (
        std::ostream & os,
        const nauczyciel * n)
```

Przeciazony operator << dla klasy nauczyciel.

Parameters

os	Strumien wyjsciowy.
u	Wskaznik na nauczyciela.

Returns

Referencja do strumienia wyjsciowego.

5.3.1.4 operator<<() [3/4]

```
std::ostream & operator<< (
        std::ostream & os,
        const sala * s)
```

Przeciazony operator << dla klasy sala.

Parameters

os	Strumien wyjsciowy.
u	Wskaznik na sale.

Returns

Referencja do strumienia wyjsciowego.

5.3.1.5 operator<<() [4/4]

```
std::ostream & operator<< (
        std::ostream & os,
        const uczen * u)
```

Przeciazony operator << dla klasy uczen.

Parameters

os	Strumien wyjsciowy.
u	Wskaznik na ucznia.

Returns

Referencja do strumienia wyjsciowego.

5.3.1.6 usun_listy()

```
void usun_listy (
        lista< nauczyciel * > & nauczyciele,
        lista< uczen * > & uczniowie,
        lista< lekcja * > & lekcje,
        lista< sala * > & sale)
```

Usuwa dynamicznie zaalokowane dane z list.

Usuwa listy

Parameters

nauczyciele	Usuwa liste nauczyciele
uczniowie	Usuwa liste uczniowie
lekcje	Usuwa liste lekcje
sale	Usuwa liste sale

Author

Dyba

5.4 Project_ppk2/lista.h File Reference

#include "klasy.h"

Functions

- void inicjalizuj (lista< nauczyciel * > &nauczyciele, lista< uczen * > &uczniowie, lista< lekcja * > &lekcje, lista< sala * > &sale, struktura_planow &plany)

  Wczytuje nauczycieli, uczniow, lekcje, sale i plan.
- void usun_listy (lista< nauczyciel * > &nauczyciele, lista< uczen * > &uczniowie, lista< lekcja * > &lekcje, lista< sala * > &sale)

  Usuwa dynamicznie zaalokowane dane z list.

5.4.1 Function Documentation

5.4.1.1 inicjalizuj()

void inicjalizuj (
lista< nauczyciel * > & nauczyciele,
lista< uczen * > & uczniowie,
lista< lekcja * > & lekcje,
lista< sala * > & sale,
struktura_planow & plany)

Wczytuje nauczycieli, uczniow, lekcje, sale i plan.

Inicjalizuje dane do list

Parameters

nauczyciele	Do tej listy sa wczytywani nauczyciele
uczniowie	Do tej listy sa wczytywani uczniowie
lekcje	Do tej listy sa wczytywane lekcje
sale	Do tej listy sa wczytywane sale

Author

Dyba

5.4.1.2 usun_listy()

```
void usun_listy (
                lista< nauczyciel * > & nauczyciele,
                lista< uczen * > & uczniowie,
                lista< lekcja * > & lekcje,
                lista< sala * > & sale)
```

Usuwa dynamicznie zaalokowane dane z list.

Usuwa listy

Parameters

nauczyciele	Usuwa liste nauczyciele
uczniowie	Usuwa liste uczniowie
lekcje	Usuwa liste lekcje
sale	Usuwa liste sale

Author

Dyba

5.5 lista.h

Go to the documentation of this file.

```
00001
00002
00003 #ifndef LISTA_H
00004 #define LISTA_H
00005
00006 #include "klasy.h"
00007
00015 void inicjalizuj(lista<nauczyciel*>& nauczyciele, lista<uczen*>& uczniowie, lista<lekcja*>& lekcje, lista<sala*>& sale,
       struktura_planow& plany);
00016
00024 void usun_listy(lista<nauczyciel*>& nauczyciele, lista<uczen*>& uczniowie, lista<lekcja*>& lekcje, lista<sala*>& sale);
00025 #endif
```

5.6 Project_ppk2/main.cpp File Reference

```
#include <iostream>
#include <algorithm>
#include <cctype>
#include <fstream>
#include "klasy.h"
#include "lista.h"
```

Functions

- void alt (std::string &p)
- void f_plan (struktura_planow &plany, lista< lekcja * > &lekcje)

  Obsluguje interfejs zwiazany z planem lekcji. Umozliwia wyswietlanie i dodawanie lekcji do planu.
- void f_uczen (lista< uczen * > &uczniowie, struktura_planow &plany)

  Obsluguje interfejs zwiazany z uczniami. Umozliwia wyswietlanie danych i dodawanie uczniow.
- void f_nauczyciel (lista< nauczyciel * > &nauczyciele, struktura_planow &plany, lista< lekcja * > &lekcje)

  Obsluguje interfejs zwiazany z nauczycielami. Umozliwia wyswietlanie danych nauczycieli.
- void f_lekcji (lista< lekcja * > &lekcje)

  Obsluguje interfejs zwiazany z lekcjami. Umozliwia wyswietlanie danych lekcji.
- void f_sala (lista< sala * > &sale, lista< lekcja * > &lekcje)

  Obsluguje interfejs zwiazany z salami. Umozliwia wyswietlanie danych sal.
- int main ()

Index