Міністерство освіти і науки України

Національний технічний університет України „КПІ”

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки
інформації та управління

ЗВІТ

до лабораторної роботи № 6

з дисципліни ООП

Виконав студент *ІП-61 Кушка Михайло Олександрович*

(№ групи, прізвище, ім’я, по батькові)

Прийняв *Головченко М.М.*

(посада, прізвище, ім’я, по батькові)

Київ 2017

ЗМІСТ

1. МЕТА РОБОТИ

Мета роботи - вивчити особливості шаблонів функцій та шаблонів класів в C++. Освоїти принципи роботи шаблонів класів для роботи з базовими типами і користувацькими типами.

2. ПОСТАНОВКА ЗАДАЧІ

Стоматологічна клініка. Пацієнт звертається в стоматологічну клініку зі скаргою. Консультант опитує пацієнта, фіксує його скарги, проводить огляд, після чого призначає необхідні процедури, лікування і вартість.

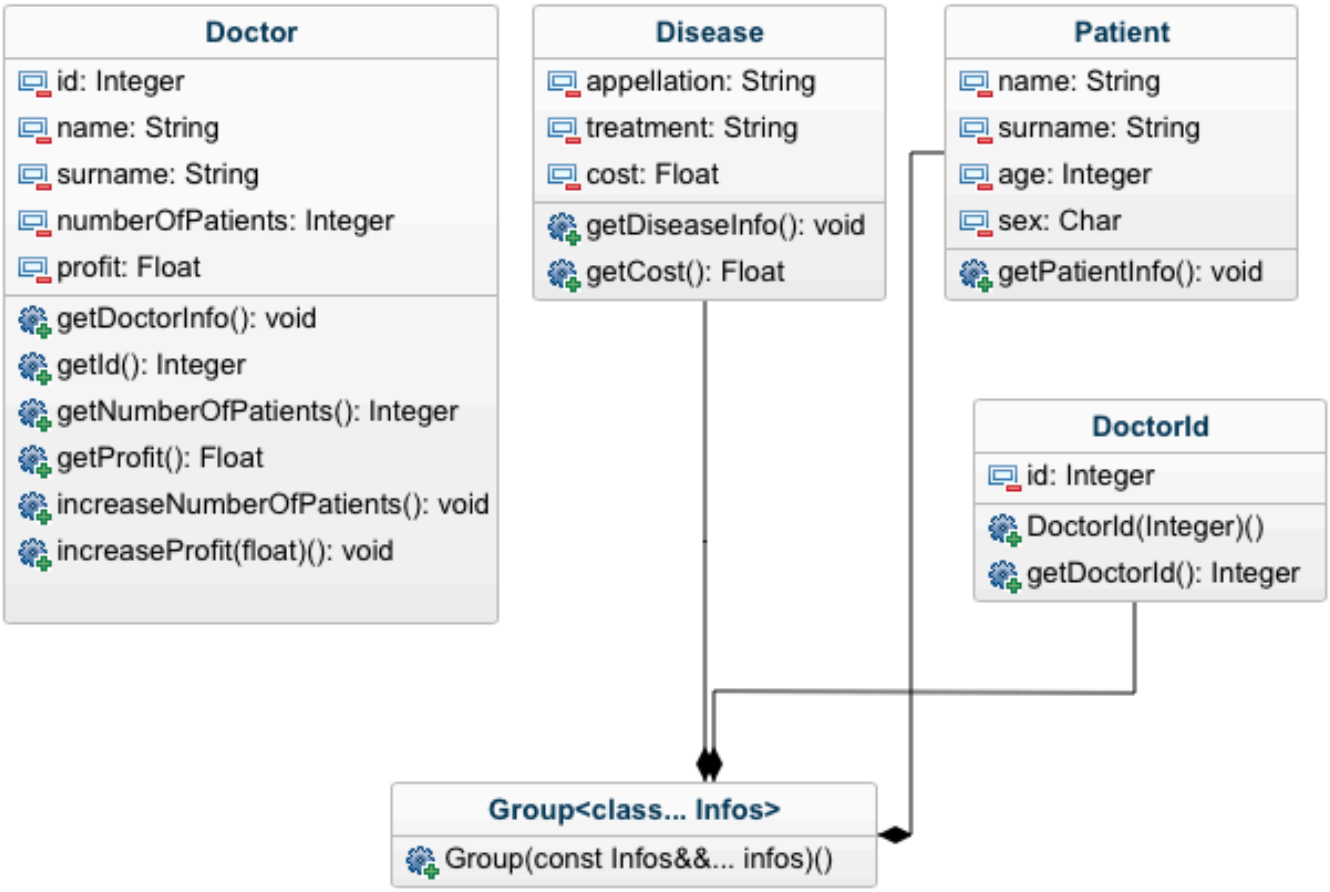
Після консультації пацієнт направляється до одного з лікарів (виходячи з захворювань пацієнта), де проходить необхідний курс лікування.

Сформувати колекцію даних з інформацією про пацієнтів їх захворюваннях і лікуючих лікарів.

3. АНАЛІТИЧНІ ВИКЛАДКИ

Перевантажені функції зазвичай використовуються для виконання подібних операцій над різними типами даних. Якщо операції ідентичні для кожного типу, це можливо виконати більш компактно і зручно, використовуючи шаблони функцій. Достатньо написати одне єдине визначення шаблону функції, а C ++ автоматично генерує різні функції для обробки кожного типу. Всі визначення шаблонів функцій починаються з ключового слова template, за яким слідує список формальних типів параметрів функції, укладений в кутові дужки. Кожен формальний тип параметра передує ключовим словом typename або class. Формальні типи параметрів - це вбудовані типи (в разі використання typename) або вбудовані типи і типи, визначені користувачем (у разі використання class). Вони використовуються для завдання типів аргументів функції і для оголошення змінних всередині тіла опису функції. Після шаблони слід звичайне опис функції.

4. UML-ДІАГРАМА КЛАСІВ



5. ВИХІДНИЙ КОД ПРОГРАМИ

prototypes.cpp

```
//
//  prototypes.cpp
//  Lab6
//
//  Created by Kushka Misha on 12/6/17.
//  Copyright © 2017 Kushka Misha. All rights reserved.
//

#include "prototypes.hpp"

// display all data from Patient class
void Patient::getPatientInfo() {
    cout << "========== Patient ==========" << endl
         << "============================" << endl
         << "Name: " << name << endl
         << "Surname: " << surname << endl
         << "Age: " << age << endl
         << "Sex: " << sex << endl << endl;
}

// display all data from Disease class
void Disease::getDiseaseInfo() {
    cout << "========== Disease ==========" << endl
         << "============================" << endl
         << "Appellation: " << appellation << endl
         << "Treatment: " << treatment << endl
         << "Cost: " << cost << endl << endl;
}

// get cost from Disease class
float Disease::getCost() {
    return cost;
}

// display all data from Doctor class
void Doctor::getDoctorInfo() {
    cout << "========== Doctor ===========" << endl
         << "============================" << endl
         << "Name: " << name << endl
         << "Surname: " << surname << endl
         << "Number of patients: " << numberOfPatients << endl
    << "Profit: " << profit << endl << endl;
}

// get doctor's id
int Doctor::getId() {
    return id;
}

// get Doctor's number of patients
int Doctor::getNumberOfPatients() {
    return numberOfPatients;
}

// get Doctor's profit
```

```
float Doctor::getProfit() {
    return profit;
}

// increase Doctor number of patients by 1
void Doctor::increaseNumberOfPatients() {
    numberOfPatients += 1;
}

// Increase Doctor profit by value
void Doctor::increaseProfit(float additionalProfit) {
    profit += additionalProfit;
}
```

prototypes.hpp

```
//
//  prototypes.hpp
//  Lab6
//
//  Created by Kushka Misha on 12/6/17.
//  Copyright © 2017 Kushka Misha. All rights reserved.
//

#ifndef prototypes_hpp
#define prototypes_hpp

#include "stdafx.hpp"

/**
 * Patient class
 */
class Patient {
    string name;
    string surname;
    int age;
    char sex;
public:
    Patient(string _name, string _surname, int _age, char _sex) :
        name(_name), surname(_surname),age(_age), sex(_sex) {};

    void getPatientInfo();
};

/**
 * Disease class
 */
class Disease {
    string appellation;
    string treatment;
    float cost;
public:
    Disease(string _appellation, string _treatment, float _cost) :
        appellation(_appellation), treatment(_treatment), cost(_cost) {};

    void getDiseaseInfo();
    float getCost();
};

/**
 * Doctor class
```

```
 */
class Doctor {
    int id;
    string name;
    string surname;
    int numberOfPatients;
    float profit;
public:
    Doctor(int _id, string _name, string _surname, int _numberOfPatients=0, float
_profit=0) :
    id(_id), name(_name), surname(_surname),numberOfPatients(_numberOfPatients),
profit(_profit) {};

    void getDoctorInfo();
    int getId();
    int getNumberOfPatients();
    float getProfit();

    void increaseNumberOfPatients();
    void increaseProfit(float);
};

/**
 * DoctorId class
 */
class DoctorId {
    int id;
public:
    DoctorId(int _id) : id(_id) {};

    int getDoctorId() {
        return id;
    }
};

/**
 * Variational template class
 */
template<class... Infos>
class Group : public Infos...
{
public:
    Group(const Infos&&... infos) : Infos(infos)... {};
};

#endif /* prototypes_hpp */
```

stdafx.hpp

```
//
//  stdafx.hpp
//  Lab6
//
//  Created by Kushka Misha on 12/6/17.
//  Copyright © 2017 Kushka Misha. All rights reserved.
//

#ifndef stdafx_hpp
#define stdafx_hpp

#include <iostream>
#include <string>
```

```
#include <vector>

using namespace std;

#endif /* stdafx_hpp */
```

main.cpp

```
//
//  main.cpp
//  Lab6
//
//  Created by Kushka Misha on 12/5/17.
//  Copyright © 2017 Kushka Misha. All rights reserved.
//

#include "prototypes.hpp"

int main() {

    Doctor doctors[] = {Doctor(0, "Nicola", "Tesla"), Doctor(1, "Franz", "Kafka")};

vector<Group<Patient, Disease, DoctorId>> arr;
    string cont = "y";

    string name, surname;
    int age;
    char sex;
    int id;

    string disease, treatment;
    float cost;

    while(cont == "y") {
        cout << endl << "Enter patient's:" << endl;
        cout << "\tName\n\t> ";
        cin >> name;
        cout << "Surname\n\t> ";
        cin >> surname;
        cout << "Age\n\t> ";
        cin >> age;
        cout << "Sex\n\t> ";
        cin >> sex;

        cout << "Disease\n\t> ";
        cin >> disease;
        cout << "Treatment\n\t> ";
        cin >> treatment;
        cout << "Cost\n\t> ";
        cin >> cost;

        cout << "Enter doctor id\n> ";
        cin >> id;

        arr.push_back(Group<Patient, Disease, DoctorId> ({name, surname, age, sex},
{disease, treatment, cost}, {id}));

        cout << endl << endl << "Continue? (y / n)\n> ";
        cin >> cont;
    }

    // Calculate profit and number of patients of every doctor
```

```
    int n = arr.size();
    int doctor_id = 0;

    for (int i = 0; i < n; i++) {
        cost = arr[i].getCost();
        doctor_id = arr[i].getDoctorId();

        doctors[doctor_id].increaseNumberOfPatients();
        doctors[doctor_id].increaseProfit(cost);
        doctors[doctor_id].getDoctorInfo();
    }

    return 0;
}
```

6. ПРИКЛАДИ РОБОТИ ПРОГРАМИ

```
    ----------------------
    | Kushka Misha, IP-61 |
    | Level: 3            |
    | Variant: 15         |
    ----------------------

=== First vector ===

Enter x
> t
Error! Please enter number, not string.
Enter x
> 5.4
Enter y
> 7
Enter z
> 9
=== Second vector ===

Enter x
> 0
Enter y
> 3.5
Enter z
> -6.7

A = (5.4, 7, 9)
B = (0, 3.5, -6.7)

c = a + b
C = (5.4, 10.5, 2.3)

c = a - b
C = (5.4, 3.5, 15.7)

c = a * b
C = (-78.4, 36.18, 18.9)

c = a * 3
C = (16.2, 21, 27)

c = -a
C = (-5.4, -7, -9)

isCollinear = a == b
isCollinear == 0

isCollinear = a == a*2
isCollinear == 1



Continue? (y / n)
>n
Program ended with exit code: 0
```

7. ВИСНОВКИ

У даній лабораторній роботі я використав обробку виключень для запобігання виникнення помилок у роботі програми при введенні некоректних даних. Також це дозволило запобігти аварійному “вильоту” програми.