Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
(ФГАОУ ВО «ЮФУ»)

ИНДИВИДУАЛЬНАЯ РАБОТА №2
по дисциплине «Алгоритмизация и программирование»

«Объектно-ориентированное программирование на языке C++»

Выполнил
студент КТбо1-4 _____ А. А. Воронов

Принял
ст. пр. каф. САПР, к.т.н. _____ В. И. Данильченко

Таганрог 2024

1. **Цель индивидуальной работы:**

   Для заданной предметной области разработать:

   1) UML-диаграмму классов;

   2) программный код заданных классов на C++ в среде MS Visual Studio;

   3) программный код методов модификации объектов с контролем всех входных параметров;

   4) программный код метода соответственно таблице индивидуальных заданий;

   5) программу тестирования разработанных классов.

2. **Задача:** написать программу, в предметной области торговли авиабилетами, в которой созданы сущности клиента, заказа билета и дополнительных услуг. Должны быть реализованы методы контроля данных (вывод, добавление, удаление). Также нужно реализовать метод, который будет выводить список клиентов, по заданным услуге и интервалу времени. Создать код для тестирования программы.

3. **Ход работы:** я создал проект консольного приложения в Microsoft Visual Studio 2022. В нем реализовал несколько файлов: основной, с точкой входа main() — S2_IndTask.cpp, и файлы с прототипами и реализацией трех классов: Client.cpp, Client.h — класс, описывающий клиента, Ticket.cpp, Ticket.h — класс, описывающий билет, Addit.cpp, Addit.h — класс, описывающий дополнительные услуги.

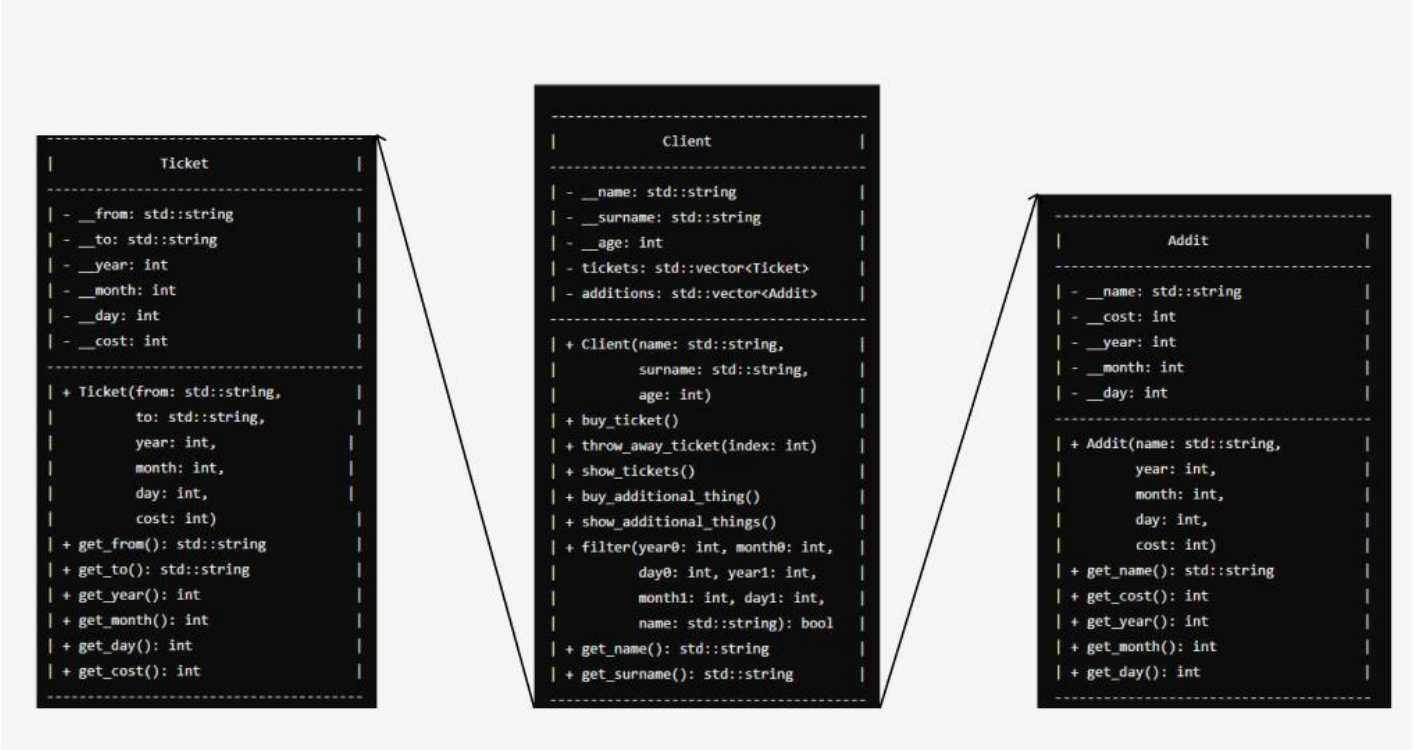
   Класс Client: имеет методы buy_ticket() — покупка билета, создание объекта класса Ticket и добавление в соответствующий контейнер vector, throw_away_ticket() — удаление билет по индексу, show_tickets() — вывод информации о всех билетах, buy_additional_thing() — покупка дополнительной услуги, создание объекта класса Addit и добавление в соответствующий контейнер vector, show_additional_things() — вывод информации о всех дополнительных услугах, filter() — фильтр по интервалу

времени и названию услуги, get_name() и get_surname() — геттеры приватных атрибутов __name и __surname.

Класс Ticket: имеет методы get_from(), get_to(), get_year(), get_month(), get_day(), get_cost() — геттеры всех одноименных приватных атрибутов класса.

Класс Addit: имеет методы get_name(), get_year(), get_month(), get_day(), get_cost() — геттеры всех одноименных приватных атрибутов класса.

4. UML-диаграмма классов:



5. Программный код всех классов:

a. Client

Client.h

```
#pragma once

#include <iostream>
#include <string>
#include <vector>

#include "Ticket.h"
```

```
#include "Addit.h"



class Client
{
public:
	Client(std::string name, std::string surname, int age);

	void buy_ticket(); //done
	void throw_away_ticket(int index); //done
	void show_tickets(); //done
	void buy_additional_thing();
	void show_additional_things();

	bool filter(int year0, int month0, int day0, int year1, int month1, int day1,
std::string name);

	std::string get_name() { return __name; }
	std::string get_surname() { return __surname; }

private:
	std::string __name, __surname;
	int __age;
	std::vector<Ticket> tickets;
	std::vector<Addit> additions;
};
```

Client.cpp

```
#include "Client.h"


Client::Client(std::string name, std::string surname, int age)
{
	__name = name;
	__surname = surname;
	__age = age;
}

void Client::buy_ticket()
{
	std::string from, to;
	int year, month, day, cost;
	std::cout << "Enter ticket info: from, to, year, month, day, cost: ";
	std::cin >> from >> to >> year >> month >> day >> cost;
	Ticket tckt(from, to, year, month, day, cost);
	tickets.push_back(tckt);
}

void Client::throw_away_ticket(int index)
{
	tickets.erase(tickets.begin() + index);;
}

void Client::show_tickets()
{
	for (int i = 0; i < tickets.size(); i++)
	{
		std::cout
			<< "TICKET " << i << "\n"
			<< "From: " << tickets[i].get_from() << "\n"
			<< "To: " << tickets[i].get_to() << "\n"
			<< "Year: " << tickets[i].get_year() << "\n"
```

```
                    << "Month: " << tickets[i].get_month() << "\n"
                    << "Day: " << tickets[i].get_day() << "\n"
                    << "Cost: " << tickets[i].get_cost() << "\n\n";
        }
}

void Client::buy_additional_thing()
{
        std::string name;
        int year, month, day, cost;
        std::cout << "Enter thing info: name, year, month, day, cost: ";
        std::cin >> name >> year >> month >> day >> cost;
        Addit add(name, year, month, day, cost);
        additions.push_back(add);
}

void Client::show_additional_things()
{
        for (int i = 0; i < additions.size(); i++)
        {
                std::cout
                    << "ADDITIONAL THING " << i << "\n"
                    << "Name: " << additions[i].get_name() << "\n"
                    << "Year: " << additions[i].get_year() << "\n"
                    << "Month: " << additions[i].get_month() << "\n"
                    << "Day: " << additions[i].get_day() << "\n"
                    << "Cost: " << additions[i].get_cost() << "\n\n";
        }
}

bool Client::filter(int year0, int month0, int day0, int year1, int month1, int
day1, std::string name)
{
        int time0 = year0 * 365 + month0 * 31 + day0;
        int time1 = year1 * 365 + month1 * 31 + day1;

        for (int i = 0; i < additions.size(); i++)
        {
                int timex = additions[i].get_year() * 360 + additions[i].get_month() *
31 + additions[i].get_day();
                if ((time0 < timex < time1) and name == additions[i].get_name()) return
true;

        }

}
```

b. Ticket

Ticket.h

```
#pragma once
#include<iostream>
#include <string>



class Ticket
{
public:
```

```
	Ticket(std::string from, std::string to, int year, int month, int day, int
cost);

	std::string get_from() { return __from; }
	std::string get_to() { return __to; }
	int get_year() { return __year; }
	int get_month() { return __month; }
	int get_day() { return __day; }
	int get_cost() { return __cost; }

private:
	std::string __from, __to;
	int __year, __month, __day, __cost;
};
```

Ticket.cpp

```
#include "Ticket.h"

Ticket::Ticket(std::string from, std::string to, int year, int month, int day, int
cost)
{
	__from = from;
	__to = to;
	__year = year;
	__month = month;
	__day = day;
	__cost = cost;
}
```

c. Addit

Addit.h

```
#pragma once
#include <string>

class Addit
{
public:
	Addit(std::string name, int year, int month, int day, int cost);

	std::string get_name() { return __name; }
	int get_cost() { return __cost; }
	int get_year() { return __year; }
	int get_month() { return __month; }
	int get_day() { return __day; }

private:
	std::string __name;
	int __cost, __year, __month, __day;
};
```

Addit.cpp

```
#include "Addit.h"
```

```
Addit::Addit(std::string name, int year, int month, int day, int cost)
{
	__name = name;
	__cost = cost;
}
```

6. Программный код методов модификации:

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

std::vector <Client> DB;

void add_client()
{
    std::cout << "Inter Client Info: Name, Surname, Age: ";
    std::string name, surname;
    int age;
    std::cin >> name >> surname >> age;
    Client clnt(name, surname, age);
    DB.push_back(clnt);
}

int main()
{


    while(true)
    {
        std::cout
            << "0. Exit\n"
            << "1. Add Client\n"
            << "2. Buy Ticket\n"
            << "3. Throw Away Ticket\n"
            << "4. Show Tickets\n"
            << "5. Buy Additional Thing\n"
            << "6. Show Additional Things\n"
            << "7. Filter By Time And Thing Name\n"
            << "Enter: ";



        int idx;
        std::cin >> idx;
        switch (idx)
        {
            case(0):
                break;

            case(1):

                add_client();
                std::cout << "Success!\n\n";
                break;

            case(2):
                int id; //index in vector
                std::cout << "Enter Client ID: ";
                std::cin >> id;
                DB[id].buy_ticket();
                std::cout << "Success!\n\n";
```

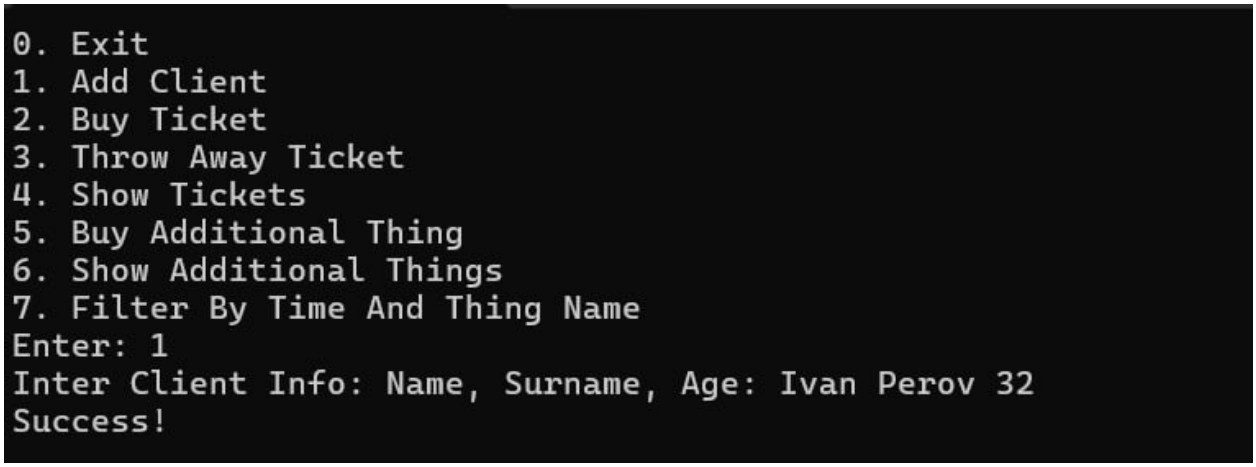
```
                break;

            case(3):
                id = 0;
                int indx; //index in vector
                std::cout << "Enter Client ID: ";
                std::cin >> id;
                std::cout << "Enter Ticket Index: ";
                std::cin >> indx;
                DB[id].throw_away_ticket(indx);
                std::cout << "Success!\n\n";
                break;

            case(4):
                id = 0;
                std::cout << "Enter Client ID: ";
                std::cin >> id;
                std::cout << "------\n";
                DB[id].show_tickets();
                std::cout << "------\n";

            case(5):
                id = 0; //index in vector
                std::cout << "Enter Client ID: ";
                std::cin >> id;
                DB[id].buy_additional_thing();
                std::cout << "Success!\n\n";
                break;

            case(6):
                id = 0;
                std::cout << "Enter Client ID: ";
                std::cin >> id;
                std::cout << "------\n";
                DB[id].show_additional_things();
                std::cout << "------\n";
                break;

            case(7):
                int y0, m0, d0, y1, m1, d1;
                std::string name;
                std::cout << "Enter Name of a Thing: ";
                std::cin >> name;
                std::cout << "Enter start Y, M, D: ";
                std::cin >> y0 >> m0 >> d0;
                std::cout << "Enter finish Y, M, D: ";
                std::cin >> y1 >> m1 >> d1;

                std::cout << "\n------\n";
                for (int i = 0; i < DB.size(); i++)
                {
                    if (DB[i].filter(y0, m0, d0, y1, m1, d1, name) == true)
                    {
                        std::cout
                            << "CLIENT " << i << "\n"
                            << "Name: " << DB[i].get_name() << "\n"
                            << "Surname: " << DB[i].get_surname() << "\n\n";
                    }

                }
                std::cout << "------\n\n";

        }
    }
}
```

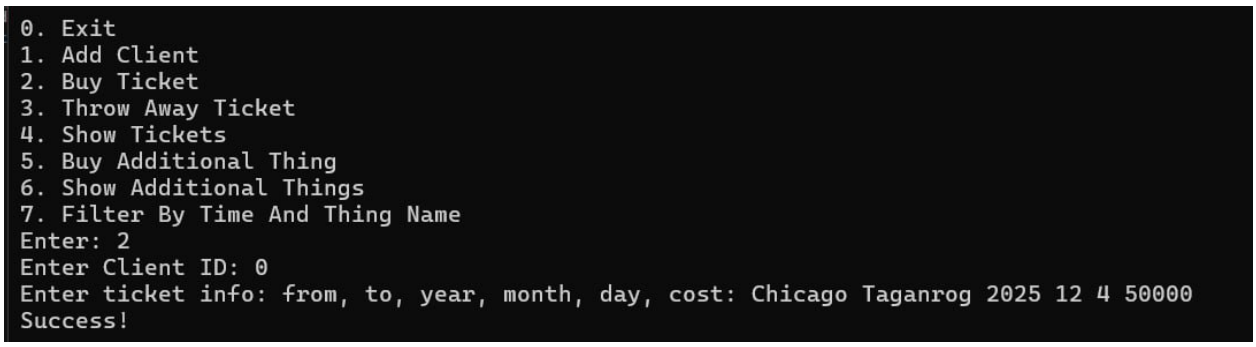
7. Программа тестирования:

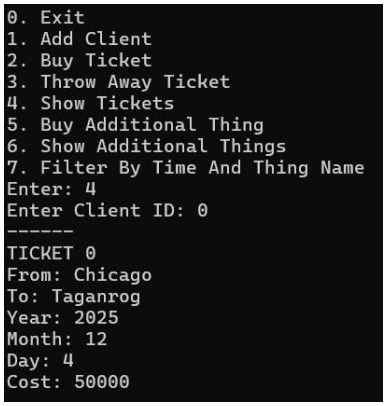
```
#include <gtest/gtest.h>
#include <iostream>
#include <sstream>
#include "Client.h"

class MockInput {
public:
    MockInput(const std::string& input) : mock_input(input), original_input(nullptr)
{
        original_input = &std::cin;
        std::cin.rdbuf(mock_input.rdbuf());
    }
    ~MockInput() {
        std::cin.rdbuf(original_input);
    }
private:
    std::istringstream mock_input;
    std::istream* original_input;
};
class ClientTest : public ::testing::Test {
protected:
    void SetUp() override {

        DB.clear();
        Client clnt("Ivan", "Petrov", 30);
        DB.push_back(clnt);
    }

    void TearDown() override {

        DB.clear();
    }
};
TEST_F(ClientTest, BuyTicketTest) {
    {
        MockInput mock_input("CityA CityB 2024 6 13 50\n");
        std::cin.ignore();
        DB[0].buy_ticket();
    }
    ASSERT_EQ(DB[0].show_tickets().size(), 1);
}
TEST_F(ClientTest, ThrowAwayTicketTest) {
    DB[0].buy_ticket();
    {
        MockInput mock_input("0\n");
        std::cin.ignore();
        DB[0].throw_away_ticket(0);
    }
    ASSERT_EQ(DB[0].show_tickets().size(), 0);
}
TEST_F(ClientTest, BuyAdditionalThingTest) {
    {
        MockInput mock_input("TestThing 2024 6 13 100\n");
        std::cin.ignore();
        DB[0].buy_additional_thing();
    }
    ASSERT_EQ(DB[0].show_additional_things().size(), 1);
```

```
}
TEST_F(ClientTest, FilterTest) {
    DB[0].buy_additional_thing();
    {
        MockInput mock_input("TestThing 2024 6 13 2024 12 31\n");
        std::cin.ignore();
        ASSERT_TRUE(DB[0].filter(2024, 6, 13, 2024, 12, 31, "TestThing"));
    }
}
int main(int argc, char** argv) {
    ::testing::InitGoogleTest(&argc, argv);
    return RUN_ALL_TESTS();
}
```

8. Вывод основной программы:







9. **Вывод:** были выполнены все цели и задачи. Реализованы 3 сущности в предметной области торговли авиабилетами, описывающие клиента, билет, дополнительную услугу. Реализован фильтр по заданной услуге и периоду времени. Программа протестирована при помощи google test в visual studio.

10. **Источники:**

http://cppstudio.com/post/439/

https://metanit.com/cpp/tutorial/3.8.php

https://metanit.com/cpp/tutorial/7.2.php

https://metanit.com/cpp/tutorial/2.17.php

https://habr.com/ru/articles/667880/