**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**федеральное государственное бюджетное образовательное учреждение высшего профессионального образования**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

**Факультет Информационных систем и технологий**

**Кафедра Безопасности информационных систем**

**КУРСОВОЙ ПРОЕКТ**

по дисциплине “Технологии программирования”

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

Тема проекта: “Ремонт бытовой техники”

Автор: студент гр. ИСТ-114 \_\_\_\_\_\_\_\_\_/Константинов А.Д./

(шифр группы) (подпись) (Ф.И.О.)

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Проверил:

руководитель проекта \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_/

(должность) (подпись) (Ф.И.О.)

Санкт-Петербург

2022 г.

СОДЕРЖАНИЕ

1. Основная часть ……………...……………………………………………. 3
   1. Замысел и состав проекта ………………………………………. 3
   2. Текст программы ………………………………………………… 7
   3. Результаты работы программы …..…………………………… 15
2. Заключение………………………………………….…………………….17
3. Список использованной литературы ……………………………………18
4. Приложение 4 …………………………………………………………….18
5. **ОСНОВНАЯ ЧАСТЬ  
   1.1 ЗАМЫСЕЛ И СОСТАВ ПРОЕКТА**

Тематика курсового проекта: «Ремонт бытовой техники»: типы обслуживаемой техники, вид работ (ремонт, гарантийное обслуживание), заказы, клиенты, оплата услуг.

В данном проекте было спроектировано приложение “Сервис” с использованием различных средств языка программирования C++ и ООП подхода. Проект был реализован в виде модели MVC(Model-View-Controller, “модель-вид-контроллер”). При разработке данной модели проект был разделен на три составные части:

А) Модель, в которой описывались данные и методы для работы с данными. А также классы и объекты

B) Вид(представление). Данная часть отвечала за визуализацию информации

C) Контроллер. Данная часть отвечала за взаимодействие с пользователем

Проект представляет из себя консольное приложение с текстовым интерфейсом на русском языке, которое позволяет сдавать технику в ремонт, узнавать источник проблемы и оплачивать ремонт. Главной целью проекта было проектирование взаимодействия пользователя с электронным сервисом. Для проекта были выдвинуты данные задачи:

1. Реализовать взаимодействие пользователя с текстовым интерфейсом
2. Ввод типа ремонтируемого устройства
3. База заказов
4. База пользователей
5. Вывод информации о заказе(ремонте)
6. Поиск заказов по имени пользователя

Соответственно, главной задачей было создать подобие настоящего сервиса, с которым пользователю было бы удобно взаимодействовать. Все задачи были выполнены, приложение имеет весь заявленный функционал и возможности.

Так как при разработке программы был использован объектно-ориентированный подход программирования использовалось несколько классов:

1. Service – ключевой объект, через который происходит все взаимодействие. Является витриной виртуального-сервиса. Таблица 1
2. Client – класс, хранящий информацию о пользователе. Таблица 2
3. Order – класс, хранящий и осуществляющий основное взаимодействия вашего устройства и сервиса. Таблица 3
4. Trouble – класс, содержащий виды и типы неисправностей. Таблица 4

|  |
| --- |
| **Service** |
| * Orders:vector<Order> * Clients:vector<Client> |
| * AddNewOrder(); * AddNewClient(); * CheckClientName(string name){return number;}   + friend GetOrders();  +friend Repair(); |

Таблица 1

|  |
| --- |
| **Order** |
| -orderID:int  -clientID:int  -deviceID:int  -RepairPrice:int  -tr:Trouble  -problem:string |
| +SetOrderID(int orderId);  +SetClientID(int clientId);  +SetDeviceID(int devId);  +GetDeviceID(){return deviceID;}  +GetClientID(){return clientID;}  +GetOrderID(){return orderID;}  +GetDiagnosticsResult(){return problem;}  +GetCost(){return cost;}  +Repair(){return result;} |

Таблица 2

|  |
| --- |
| **Client** |
| -name:string  -clientID:int |
| +SetClientID(int clientID)  +SetClientName(string name)  +GetClientID(){return clientID;}  +GetClientName(){return name;} |

Таблица 3

|  |
| --- |
| **Trouble** |
| +GetTrouble(){return trouble;} |

Таблица 4

Классы являются автономными, наследования в проекте нету.

Сам проект разбит на файлы. Было решено разместить проект в 8-ми файлах, схема размещения будет представлена в приложении А

В файле “Service.h” описан класс Service и две дружественные функции GetOrders() и Repair(). Первая позволяет увидеть заказы определенного клиента, а вторая отремонтировать устройство. В файле “Service.cpp” представлена имплементация функций AddNewOrder() – создает новый заказ в списке с заказами; AddNewClient() – создает нового клиента в списке клиентов; CheckClientName() – проверяет наличие клиента в базе, и выдает индекс его записи.

В файле “Client.h” описан класс Client и четыре функции:

А)SetClientID() – запоминает идентификатор данного клиента

Б) SetClientName() –запоминает имя клиента

В) GetClientID() – возвращает идентификатор клиента

Г) GetClientName() – возвращает имя клиента

В файле “Client.cpp” представлена имплементация функций класса Client

В файле “Order.h” описан класс Order. Его главная задача хранить все параметры ремонта, от типа устройства, сданного на обслуживание, до стоимости проведенных манипуляций. Его приватные поля (orderID, clientID, deviceID, RepairPrice, tr, problem) и множество функций, направленных на сохранение и вычисление параметров ремонта. В файле “Order.cpp” представлена имплементация функций класс Order.

В файле “Trouble.h” описан класс Trouble. Его главная задача – генерировать рандомную неисправность. В функции GetTruble() как раз и реализуется данная цель.

В файле “TermPaper\_26.cpp” происходит запуск самой программы и взаимодействие пользователя с консольным интерфейсом. Там же и реализованы дружественные функции классу Service Repair() и GetOrders(). Также с переменном service хранится экземпляр класса “Service”, через который и происходит основное взаимодействие с данными.

* 1. **ТЕКСТ ПРОГРАММЫ**

**Файл “TermPaper\_26.cpp”:**

#include <iostream>

#include <locale.h>

#include "Service.h"

using namespace std;

Service service;

void Repair()

{

service.AddNewOrder();

cout << "Как вас зовут:";

std::string name;

cin >> name;

if (service.CheckClientName(name) == -1)

{

cout << "Рады Вас приветствовать " << name << endl;

service.AddNewClient();

service.Clients[service.Clients.capacity() - 1].SetClientName(name);

}

else

{

cout << "Рады Вас снова приветствовать " << name << endl;

}

service.Orders[service.Orders.capacity() - 1].SetClientID(service.Clients[service.CheckClientName(name)].GetClientID());

cout << "Что у вас за устройство?" << endl

<< "1. Стиральная машина" << endl

<< "2. Посудомоечная машина" << endl

<< "3. Компьютер" << endl

<< "4. Телефон" << endl

<< "5. Ноутбук" << endl

<< "Так что у вас за устройство:";

int deviceID;

cin >> deviceID;

service.Orders[service.Orders.capacity() - 1].SetDeviceID(deviceID);

cout << "У вас уже прошла гарантия на устройство?" << endl

<< "1. Да" << endl

<< "2. Нет" << endl;

int typeOfRepair = 0;

cin >> typeOfRepair;

cout << "НАЧИНАЕМ ДИАГНОСТИКУ..." << endl;

cout << endl;

cout << endl;

cout << endl;

cout << endl;

cout << "В ходе диагностики была найдена неисправность:" << service.Orders[service.Orders.capacity() - 1].GetDiagnosticsResult() << endl;

if (typeOfRepair == 2)

{

cout << "Ремонт для вас будет бесплантым, поскольку гарантия еще есть" << endl;

}

else

{

cout << "Стоимость ремонта составит:" << service.Orders[service.Orders.capacity() - 1].GetCost() << endl;

}

cout << endl;

cout << endl;

cout << endl;

cout << endl;

cout << "Вы согласны на ремонт?" << endl << "1. Да" << endl << "2. Нет" << endl << "Ваше решение:";

int repairG = 0;

cin >> repairG;

if (repairG == 2)

{

cout << "Поняли Вас. Возврат в основное меню..." << endl;

return;

}

else

{

cout << "Ремонтируем...." << endl;

cout << "Результат ремонта:" << service.Orders[service.Orders.capacity() - 1].Repair() << endl;

}

cout << "Благодарим Вас " << name << " за посещение нашего сервиса! Поставьте нам 5 по курсовой работе!" << endl;

cout << endl;

cout << endl;

}

void GetOrders()

{

cout << "Введите Ваше имя: ";

std::string name;

cin >> name;

if (service.CheckClientName(name) == -1)

{

cout << "Извините, но вы у нас ничего не чинили..." << endl;

return;

}

cout << "Снова здравствуйте, " << name << "!" << endl;

cout << "Вот ваши прошлые заказы:" << endl;

int enumirator = 0;

for (size\_t i = 0; i < service.Orders.capacity(); i++)

{

if (service.Orders[i].GetClientID() == service.Clients[service.CheckClientName(name)].GetClientID())

{

Order tmp = service.Orders[i];

cout << "[" << enumirator << "]: " << endl

<< "Тип устройства:" << tmp.GetDeviceID() << endl

<< "Найденная неисправность:" << tmp.GetDiagnosticsResult() << endl

<< "Итоговая стоимость ремонта: " << tmp.GetCost() << endl

<< "Результат ремонта: " << tmp.Repair() << endl;

enumirator++;

}

}

}

int main()

{

setlocale(LC\_ALL, "Russian");

while (true)

{

cout << "Приветствуем Вас в нашем сервисе-программе!" << endl << "У нас Вы можете починить свою технику!"

<< "Вы можете:" << endl << "1. Сдать устройство на ремонт" << endl << "2. Узнать историю своих заказов" << endl << "3. Выйти из программы" << endl;

cout << "Что бы вы хотели сделать:";

int action = 0;

cin >> action;

switch (action)

{

case 1:

Repair();

break;

case 2:

GetOrders();

break;

default:

cout << "Goodbye!" << endl;

return 0;

break;

}

}

}

**Файл “Service.cpp”:**

#include "Service.h"

void Service::AddNewOrder()

{

Order newOrder;

newOrder.SetOrderID(Orders.capacity());

Orders.push\_back(newOrder);

}

void Service::AddNewClient()

{

Client newClient;

newClient.SetClientID(Clients.capacity());

Clients.push\_back(newClient);

}

int Service::CheckClientName(std::string name)

{

for (size\_t i = 0; i < Clients.capacity(); i++)

{

if (name == Clients[i].GetClientName())

{

return i;

}

}

return -1;

}

**Файл “Order.cpp”:**

#include "Order.h"

Order::Order()

{

RepairPrice = std::rand();

problem = tr.GetTrouble();

}

void Order::SetOrderID(int orderId)

{

orderID = orderId;

}

int Order::GetOrderID()

{

return orderID;

}

void Order::SetClientID(int clientId)

{

clientID = clientId;

}

int Order::GetClientID()

{

return clientID;

}

std::string Order::GetDiagnosticsResult()

{

return problem;

}

int Order::GetCost()

{

return RepairPrice;

}

std::string Order::Repair()

{

int r = std::rand();

if (r % 2 == 0)

{

return "Успешный ремонт";

}

else

{

return "Ремонт не удался(";

}

}

void Order::SetDeviceID(int devID)

{

deviceID = devID;

}

int Order::GetDeviceID()

{

return deviceID;

}

**Файл “Client.cpp”:**

#include "Client.h"

void Client::SetClientID(int clientId)

{

clientID = clientId;

}

int Client::GetClientID()

{

return clientID;

}

void Client::SetClientName(std::string clientName)

{

name = clientName;

}

std::string Client::GetClientName()

{

return name;

}

**Файл “Client.h”:**

#pragma once

#include <vector>

#include <string>

class Client

{

private:

std::string name;

int clientID;

public:

void SetClientID(int clientId);

int GetClientID();

void SetClientName(std::string clientName);

std::string GetClientName();

};

**Файл “Order.h”:**

#pragma once

#include "Trouble.h"

#include <string>

class Order

{

public:

Order();

void SetOrderID(int orderId);

int GetOrderID();

void SetClientID(int clientId);

int GetClientID();

void SetDeviceID(int devID);

int GetDeviceID();

std::string GetDiagnosticsResult();

int GetCost();

std::string Repair();

private:

int orderID;

int clientID;

int deviceID;

int RepairPrice;

Trouble tr;

std::string problem;

};

Файл “Service.h”:

#pragma once

#include <vector>

#include "Order.h"

#include "Client.h"

class Service

{

public:

friend void Repair();

friend void GetOrders();

private:

std::vector <Order> Orders;

std::vector <Client> Clients;

void AddNewOrder();

void AddNewClient();

int CheckClientName(std::string name);

};

**Файл “Trouble.h”:**

#pragma once

#include <iostream>

class Trouble

{

public:

std::string GetTrouble()

{

std::string problems[4] = { "Сгорела плата","Ошибка в работе ПО","Физическое повреждение элементов устройства","Сгорел контроллер" };

return problems[rand() % 4];

}

};

* 1. **Результаты работы программы**

В ходе работы были выполнены все полученные задачи и требования. Результаты работы программы можно увидеть на следующих рисунках.

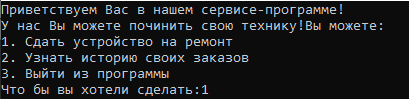


Рисунок 1 – Начальный вид пользовательского интерфейса



Рисунок 2 – Программа узнает имя пользователя(клиента)

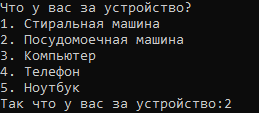


Рисунок 3 – Выбор типа устройства пользователя

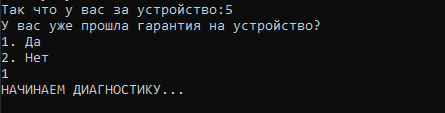


Рисунок 4 – Программа спрашивает о наличии действительной гарантии на устройство и начинает диагностику

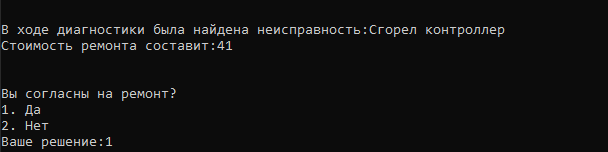


Рисунок 5 – Программа рассказывает о найденной неисправности и предлагает отремонтировать устройство.

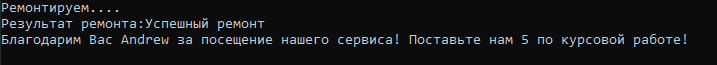


Рисунок 6 – Программа успешно ремонтирует устройства пользователя и благодарит его за посещение сервиса

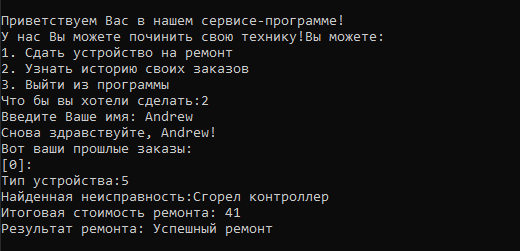


Рисунок 7 – Программа дает посмотреть историю ремонтов у пользователя.

**2 Заключение**

- Поставленные задачи были решены путем использования ООП в языке C++

- Проект включает в себя четыре класса: “Service”, ”Client”, ”Order”, ”Trouble”. Все классы обладают необходимым функционалом, а именно данные классы позволяют хранить и использовать пользовательские данные

- Для осуществления взаимодействия пользователя с программой, был создан бесконечный цикл, в котором происходит работа программы.

- В проекте использованы следующие средства языка:

А) Строковый тип данных – используется string из пространства имен std

Б) Динамические массивы(векторы) – для упрощения реализаций хранения списка клиентов и списка заказов, используется динамически наполняемый vector.

В) Разделение проекта на файлы – для упрощения работы над проектом, было решено разбить все на отдельные файлы

Г) Пользовательские типы данных(Order и Client) – так как необходимо хранить весьма специфические параметры устройств пользователей, то было решено использовать пользовательский тип данных, который облегчил задачу хранения и использования данных

- Самой сложной частью программы было объединение данных клиент-заказ. Так как заказов у одного клиента может быть много, то было решено закрепить за каждым клиентом уникальных идентификатор, при помощи которого можно найти его заказы, поскольку в параметры заказа прописывается идентификатор пользователя.

- Весь код занимает 309 строчек. Для запуска и сборки проекта использовалась IDE Microsoft Visual Studio 2019. Компилятор MSVC под платформу Win32. В проекте использованы всего две библиотеки для работы с данными: “vector” и “string”

- Данный проект можно использовать в реальной работе маленького сервиса, правда надо будет внести ряд изменений и дополнений, а именно расширить перечень хранимых данных, перенести хранение в полноценную базу данных и кардинально изменить пользовательский интерфейс – сделать его графическим, поскольку консольное взаимодействие весьма затруднительно для пользователей.

**3 Список использованной литературы:**

1. Векторы в C++: для начинающих [электронный ресурс] URL: https://codelessons.ru/cplusplus/vektory-v-c-dlya-nachinayushhix.html

Дата обращения 4.05.2022

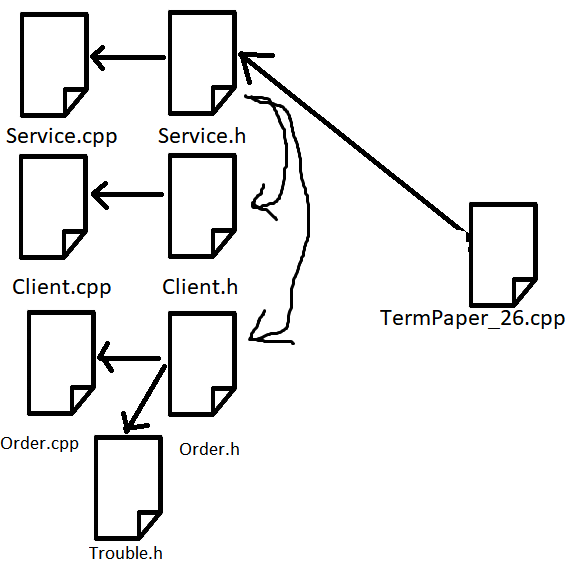
1. String class [электронный ресурс] URL: <https://www.cplusplus.com/reference/string/string/>

Дата обращения 16.05.2022

1. Поточный ввод-вывод в C++ [электронный ресурс] URL: <https://prog-cpp.ru/cpp-std/>

Дата обращения 20.05.2022

**4 Приложение А**

****