**РЕФЕРАТ**

Курсовая работа содержит: 31 страницу, 6 рисунков, 2 приложения, 4 источника литературы.

Цель данной курсовой работы – создание программы, которая дает возможность вести учет абонентов и их счетов в фирме по предоставлению интернета и кабельного телевидения.

**Ключевые слова:**

С++, файл, прикладные программы, функция, сортировка, интерфейс, поиск, алгоритм, цикл, обработка действия, меню.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc517529217)

[1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 5](#_Toc517529218)

[1.1 Задание и предметная область 5](#_Toc517529219)

[1.2 Эскизный проект 5](#_Toc517529220)

[1.3 Алгоритмы программы 7](#_Toc517529221)

[2. ПРАКТИЧЕСКОЕ РЕШЕНИЕ 10](#_Toc517529222)

[2.1 Функции программы 10](#_Toc517529223)

[2.2 Клиентская база данных 12](#_Toc517529224)

[2.3 Интерфейс программы и руководство пользователя. 14](#_Toc517529225)

[ЗАКЛЮЧЕНИЕ 16](#_Toc517529226)

[Список использованных источников 17](#_Toc517529227)

[ПРИЛОЖЕНИЯ 18](#_Toc517529228)

[Приложение А 18](#_Toc517529229)

[Приложение Б 28](#_Toc517529230)

# ВВЕДЕНИЕ

В наше время компьютеры вошли в нашу жизнь очень сильно и сейчас без их применения в различных сферах жизнедеятельности человека уже не обойтись. Компьютер заменяет нам архив для документов, учебник, развлекательный центр, видеосалон и музыкальный центр, также многими деловыми людьми он используется как справочник, база данных. Так же и в армии, командирам надо запомнить очень много информации об их подчиненных, и во многих случаях ее применения ее нужно очень быстро найти, и чтобы сделать это дело легче, можно собрать все данные на компьютере.

Данная курсовая работа посвящена созданию программы для фирмы предоставляющей услуги по кабельному телевидению и интернету.

Программа курсовой работы построена на языке программирования С++. В процессе разработки использовался пакет Microsoft Visual Studio 2017 Community.

Студия разработчика компании Microsoft (Microsoft Developer Studio) - это интегрированная среда для разработки, позволяет функционировать различным средам разработки, одна из которых Visual C++.

Microsoft Visual C++ является интегрированной средой разработки, в которой вы можете легко создавать, открывать, просматривать, редактировать, сохранять, компилировать и отлаживать все свои приложения, написанные на С или C++. Преимуществом этой среды является относительная простота и легкость в изучении. Применение C++ как инструментального языка позволяет получить быстрые и компактные программы.

C++ – универсальный язык программирования, он задуман так, чтобы сделать программирование более приятным для серьезного программиста. Ключевым понятием C++ является класс. Класс – это тип, определяемый пользователем. Классы обеспечивают скрытие данных, гарантированную инициализацию данных, неявное преобразование типов для типов, определенных пользователем, динамическое задание типа, контролируемое пользователем управление памятью и механизмы перегрузки операций. C++ предоставляет гораздо лучше, чем в C, средства выражения модульной программы и проверки типов. В C++ сохранены возможности языка C по работе с основными объектами аппаратного обеспечения (биты, байты, слова, адреса и т.п.). Это позволяет весьма эффективно реализовывать типы, определяемые пользователем. C++ и его стандартные библиотеки спроектированы так, чтобы обеспечивать переносимость.

Одним словом, С++ – это универсальный объектно-ориентированный язык программирования, который представляет гибкие и эффективные средства определения новых типов, используя определения новых типов, точно соответствуют концепциям приложения, программист может разделять программу. Поэтому именно его было решено использовать для написания программного продукта курсовой работы.

# 1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

## 1.1 Задание и предметная область

В наше время компьютер помогает человеку в нахождении разнообразной информации при расчетах, служит справочником либо журналом. Программа данной курсовой работы является именно таким журналом по ведению клиентов фирмы, которая занимается предоставлением услуг интернета и кабельного телевидения.

Задачей данной курсовой работы было создание программы, которая будет вести учет всех клиентов и их счетов. Производить поиск и добавление новых клиентов, а также вести учет и прием оплат за услуги. Если разбить задачу на пункты, то оно будет заключаться в следующем:

1. Разработка интерфейса будущей программы;

2. Разработка базы данных для хранения информации (в файле);

3. Разработка алгоритма взаимодействия программы с базой данных;

4. Написание функций по написанным алгоритмам.

Одним из основных требований к программе есть простой и интуитивно понятный интерфейс, чтобы с программой мог работать даже ребенок.

## 1.2 Эскизный проект

Программа работает в консольном режиме, что уменьшает размер исполняемого файла и дает возможность запускать его на любых версиях Windows без установки дополнительных библиотек. Как было отмечено в предыдущем разделе, программа должна быть проста в использовании и интуитивно понятна пользователю, поэтому меню программы:

1. Основное меню, из которого можно выбрать дальнейшие действия:

- обслуживание посетителей;

- работа с клиентской базой;

- завершение работы программы.

1. Под меню «Обслуживание посетителей» имеет следующие пункты:

- пополнить счёт абонента;

- добавить нового абонента;

- заморозить аккаунт абонента;

- выход в основное меню.

1. Под меню «Работа с клиентской базой» имеет следующие пункты:

- посмотреть абонентскую базу;

- найти абонента;

- удалить абонента;

- выход в основное меню.

Для написания программы был выбран Microsoft Visual Studio 2017 Community. Microsoft Visual Studio 2017 Community является интегрированной средой разработки, в которой можно легко создавать, открывать, просматривать, редактировать, сохранять, компилировать и отлаживать программы, написанные на различных языках программирования включая С++. Подавляющее большинство программных продуктов мирового класса создана с использованием среды разработки Microsoft Visual Studio.

Среда разработки Microsoft Visual Studio 2017 Community изображена на рисунке 1.1.

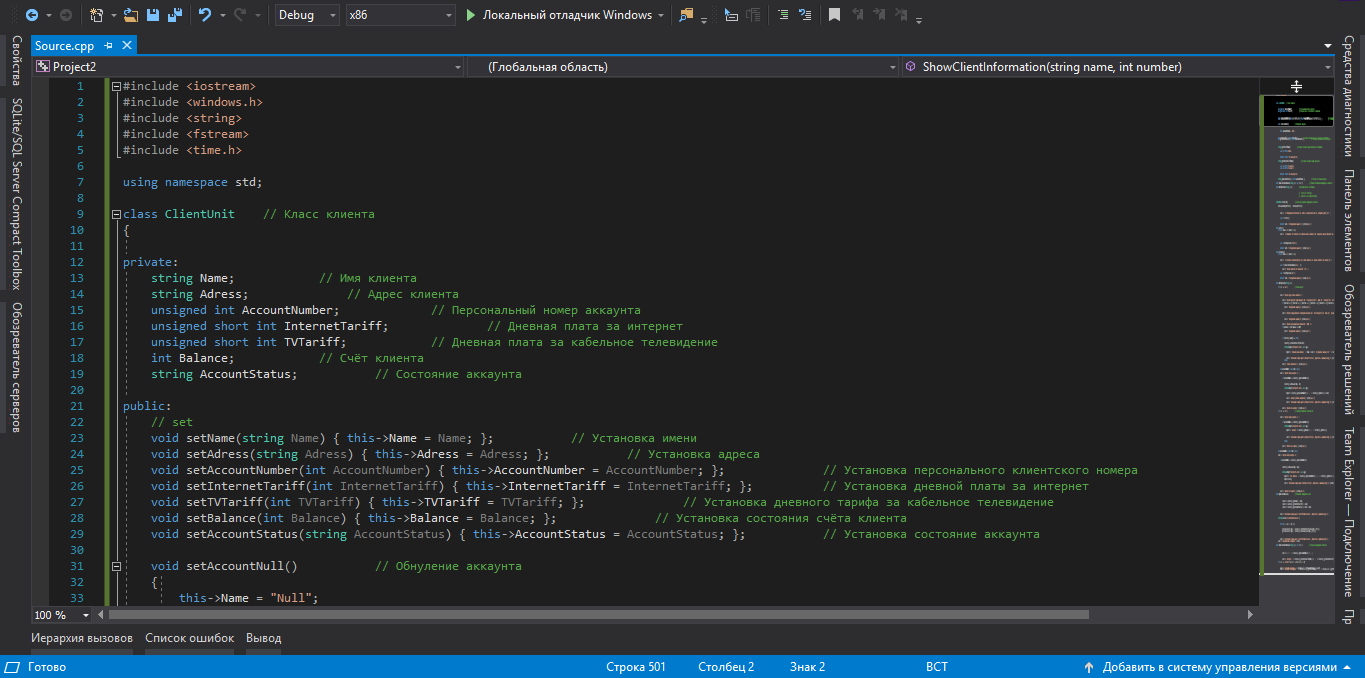


Рис 1.1 Среда разработки Microsoft Visual Studio 2017 Community.

## 1.3 Алгоритмы программы

Для выполнения поставленных задач необходимо описать функции, которые реализуют требования, описанные в эскизном проекте, то есть алгоритмы заполнения информации о студентах, группах, приказов, заполнение оценок. Сначала дадим определение алгоритма вообще.

Алгоритм - это конечный набор правил, который определяет последовательность операций для решения конкретного множества задач и обладает шестью важными чертами: конечность, определенность, введение, заключение, понятность, массовость.

Он содержат следующий ряд общих требований:

1. Концовка - при корректно заданных исходных данных алгоритм должен завершать работу и выдавать результат за конечное число шагов.

2. Определенность - в каждый момент времени следующий шаг работы однозначно определяется состоянием системы. Таким образом, алгоритм выдает один и тот же результат для одних и тех же исходных данных. С другой стороны, существуют вероятностные алгоритмы, в которых следующий шаг работы зависит от текущего состояния системы и случайного числа, генерируется.

3. Введение - алгоритм должен иметь некоторое число входных данных.

4. Заключение - завершение алгоритма определенными результатами.

5. Ясность - алгоритм для исполнителя должен включать только те команды, которые ему (исполнителю) доступны, которые входят в его систему команд.

6. Массовость - алгоритм должен быть применим к различным наборов исходных данных.

*Алгоритм отображения меню и под меню:*

1. Вывести в консоль основное меню;

2. Запросить ввод номера пункта меню;

3. Проверить на наличие данного пункта меню;

4. Если выбранный пункт меню существует, требуется выполнить алгоритм, который подвязан к данному пункту меню;

5. Если выбранного пункта меню нет, вывести на экран соответствующее сообщение и запросить повторные ввод номера пункта меню;

*Алгоритм добавления нового абонента:*

1. Запросить ввод имени нового абонента;

2. Запросить ввод адреса нового абонента;

3. Выбрать желаемый интернет тариф;

4. Проверить на корректность выбранного тарифа, если введенная информация не корректная, вывести соответствующее сообщение и запросить повторный выбор тарифа.

5. Выбрать тариф кабельного телевидения;

6. Проверить на корректность выбранного тарифа, если введенная информация не корректная, вывести соответствующее сообщение и запросить повторный выбор тарифа.

7. Запросить ввод начального баланса нового абонента;

8. Сохранить введенные данные в базу.

*Алгоритм пополнения счёта абонента:*

1. Запросить ввод номера абонента;

2. Проверить есть ли данный номер абонента в базе, если данного абонента нет, вывести на экран соответствующее сообщение и вернуть в меню;

3. Запросить сумму пополнения;

4. Сохранить введенную информацию в базу.

5. Вернуться в меню.

*Алгоритм заморозки счёта абонента:*

1. Запросить ввод номера абонента;

2. Если введенный номер абонента не найден в базе, вывести на экран соответствующее сообщение и вернуть в меню;

3. Если введенный номер абонента найден в базе, произвести заморозку счёта;

4. Сохранить введенную информацию в базу.

5. Вернуться в меню.

*Алгоритм поиска абонента:*

1. Запросить ввод лицевого счёта абонента;

2. Если введенный лицевой счет абонента не найден в базе, вывести на экран соответствующее сообщение и вернуть в меню;

3. Если введенный лицевой счет есть в базе, вывести в консоль полную информацию об абоненте.

*Алгоритм удаления абонента:*

1. Запросить ввод номера абонента;

2. Если введенный номер абонента не найден в базе, вывести на экран соответствующее сообщение и вернуть в последнее открытое меню;

3. Если введенный номер абонента есть в базе, произвести удаление всех его записей из базы;

4. Сохранить обновленную информацию в базе.

5. Вернуться в меню.

# 2. ПРАКТИЧЕСКОЕ РЕШЕНИЕ

## 2.1 Функции программы

Для написания программы необходимо знание языка программирования С++. Для этого достаточно написать функции по описанным в предыдущем разделе алгоритмах, также надо использовать функции по чтению и записи данных в файл.

Полный код программы представлен в приложении А, ниже рассмотрим основные функции программы.

Функция отображения информации об абонентах:

void ShowClientInformation(string name, int number) // Показать информацию о абонентах

{

system("cls");

if (name == "all" && number == -1)

{

for (int i = 0; i < 20; i++)

{

cout << i << ". " << Clients[i].getAccountNumber() << ", ";

cout << Clients[i].getName() << endl;

cout << Clients[i].getAdress() << endl;

cout << "Баланс: " << Clients[i].getBalance() << " руб \n";

cout << "Состояние аккаунта: " << Clients[i].getAccountStatus() << endl;

cout << "Кабельное телевидение: " << Clients[i].getTVTariffName() << ", " << Clients[i].getTVTariff() << " руб/день \n";

cout << "Интернет: " << Clients[i].getInternetTariffName() << ", " << Clients[i].getInternetTariff() << " руб/день \n\n";

}

}

if (name == "current" && number >= 0 && number < 20)

{

cout << Clients[number].getAccountNumber() << ", ";

cout << Clients[number].getName() << endl;

cout << Clients[number].getAdress() << endl;

cout << "Баланс: " << Clients[number].getBalance() << " руб \n";

cout << "Состояние аккаунта: " << Clients[number].getAccountStatus() << endl;

cout << "Кабельное телевидение: " << Clients[number].getTVTariffName() << ", " << Clients[number].getTVTariff() << " руб/день \n";

cout << "Интернет: " << Clients[number].getInternetTariffName() << ", " << Clients[number].getInternetTariff() << " руб/день \n";

}

system("pause");

}

Функция инициализации клиентской базы при запуске программы:

void Initialization()

{

ifstream Client("ClientInformation.base");

if (Client.is\_open())

{

string tmp;

for (int i = 0; i < 20; i++)

{

getline(Client, tmp); Clients[i].setName(tmp);

getline(Client, tmp); Clients[i].setAdress(tmp);

getline(Client, tmp); Clients[i].setAccountNumber(atoi(tmp.c\_str()));

getline(Client, tmp); Clients[i].setInternetTariff(atoi(tmp.c\_str()));

getline(Client, tmp); Clients[i].setTVTariff(atoi(tmp.c\_str()));

getline(Client, tmp); Clients[i].setBalance(atoi(tmp.c\_str()));

getline(Client, tmp); Clients[i].setAccountStatus(tmp);

getline(Client, tmp);

}

Client.close();

}

else

cout << "Невозможно открыть файл \"ClientInformation.base\", обратитесь к администратору \n";

cout << "Инициализация завершена" << endl;

system("pause");

}

Функция обновления информация об абоненте:

void UpdateInformation() // Обновление информации в базе

{

ofstream Client("ClientInformation.base");

if (Client.is\_open())

{

for (int i = 0; i < 20; i++)

{

Client << Clients[i].getName() << endl;

Client << Clients[i].getAdress() << endl;

Client << Clients[i].getAccountNumber() << endl;

Client << Clients[i].getInternetTariff() << endl;

Client << Clients[i].getTVTariff() << endl;

Client << Clients[i].getBalance() << endl;

Client << Clients[i].getAccountStatus() << endl << endl;

}

Client.close();

}

else

cout << "Невозможно открыть файл \"ClientInformation.base\", обратитесь к администратору \n";

}

## 2.2 Клиентская база данных

В роли базы данных был использован текстовый документ, в котором хранится полная информация обо всех абонентах фирмы.

Для удобной работы с клиентской базой, был создан класс клиента, в котором описаны все поля, требуемые для заполнения:

class ClientUnit // Класс клиента

{

private:

string Name; // Имя клиента

string Adress; // Адрес клиента

unsigned int AccountNumber; // Персональный номер аккаунта

unsigned short int InternetTariff; // Дневная плата за интернет

unsigned short int TVTariff; // Дневная плата за кабельное телевидение

int Balance; // Счёт клиента

string AccountStatus; // Состояние аккаунта

public:

// set

void setName(string Name) { this->Name = Name; }; // Установка имени

void setAdress(string Adress) { this->Adress = Adress; }; // Установка адреса

void setAccountNumber(int AccountNumber) { this->AccountNumber = AccountNumber; }; // Установка персонального клиентского номера

void setInternetTariff(int InternetTariff) { this->InternetTariff = InternetTariff; }; // Установка дневной платы за интернет

void setTVTariff(int TVTariff) { this->TVTariff = TVTariff; }; // Установка дневного тарифа за кабельное телевидение

void setBalance(int Balance) { this->Balance = Balance; }; // Установка состояния счёта клиента

void setAccountStatus(string AccountStatus) { this->AccountStatus = AccountStatus; }; // Установка состояние аккаунта

void setAccountNull() // Обнуление аккаунта

{

this->Name = "Null";

this->Adress = "Null";

this->InternetTariff = 0;

this->TVTariff = 0;

this->Balance = 0;

this->AccountStatus = "Null";

};

// get

string getName() { return Name; }; // Возврат имени

string getAdress() { return Adress; }; // Возврать адреса

int getAccountNumber() { return AccountNumber; }; // Возврат номера аккаунта

int getTVTariff() { return TVTariff; }; // Возврат дневной платы за кабельное телевидение

int getInternetTariff() { return InternetTariff; }; // Возврать дневной платы за интернет

bool Empty() // Если экземпляр пуст - вернёт true, иначе - false

{

if (this->Name == "Null")

return true;

else

return false;

};

string getTVTariffName() // Возврат названия тарифа кабельного телевидения

{

switch (TVTariff)

{

case 1: return "Мини";

break;

case 2: return "Средний";

break;

case 3: return "Мега";

break;

default: return "Не используется";

}

};

string getInternetTariffName() // Возврат названия тарифа интернета

{

switch (InternetTariff)

{

case 3: return "Мини";

break;

case 5: return "Безлим 10";

break;

case 6: return "Безлим 35";

break;

case 8: return "Безлим 50";

break;

case 10: return "Безлим 100";

break;

default: return "Не используется";

}

};

int getBalance() { return Balance; }; // Возврат баланса аккаунта

string getAccountStatus() { return AccountStatus; }; // Возврат состояния аккаунта

};

Файл клиентской базы данных изображен на рисунке 2.1.

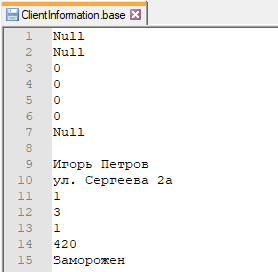


Рис 2.1. Внешний вид файла с клиентской базой

## 2.3 Интерфейс программы и руководство пользователя.

Последний раздел курсовой работы посвящен внешнему виду полученной программы. В результате работы была получена программа с удобным интерфейсом, которая позволяет вести учет абонентов и их счетов.

После запуска программы будет произведена инициализацию клиентской базы данных. После успешной инициализации требуется нажать любую клавишу и откроется доступ к основному меню.

В случае провала инициализации, требуется обратить к администратору данной программы.

Для добавления нового пользователя, требуется выбрать пункт меню "1. Обслуживание посетителей" -> "2. Добавить нового абонента", впоследствии чего, будет предоставлена возможность для добавления информации о новом абоненте. После заполнения информации об абоненте, требуется выбрать интернет тариф, выбрать тариф кабельного телевидения и ввести начальную сумму пополнения счёта. Как только будет введена вся информация, программа автоматически произведет сохранение в базу.

Что бы пополнить счет абонента, требуется выбрать пункт меню "1. Обслуживание посетителей" -> "1. Пополнить счёт абонента", затем ввести номер счёта абонента, после чего ввести сумму пополнения.

Для заморозки абонентского счёта, нужно выбрать "1. Обслуживание посетителей" -> "2. Заморозить аккаунт абонента", затем ввести номер счёта абонента и нажать клавишу ENTER, после успешной операции, абонентский счёт будет заморожен.

Что бы просмотреть клиентскую базу, нужно произвести следующие действия: выбрать пункт меню "2. Работа с клиентской базой" -> "1. Посмотреть абонентскую базу", после выбора данного пункта меню, в консоле будет отображена полная информация о всех абонентах, которые есть в базе.

Для поиска конкретного абонента, требуется выбрать пункт меню "2. Работа с клиентской базой" -> "2. Найти абонента", затем ввести номер счёта абонента и нажать клавишу ENTER, после успешной операции, в консоле будет отображена полная информация о выбранном абоненте.

Чтобы удалить абонента, требуется выбрать пункт меню "2. Работа с клиентской базой" -> "3. Удалить абонента", затем ввести номер счёта абонента и нажать клавишу ENTER, после успешной операции, в консоле будет отображена сообщение об успешном удалении выбранного абонента.

Внешний вид программы представлен в приложении В.

# ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы была получена программа, которая позволяет хранить информацию об абонентах и их счетах, вести контроль счетов, пополнять, замораживать либо удалять счета.

Программа имеет удобный интерфейс, и это позволяет пользоваться ею людям без высокого уровня знаний работы с компьютером, нетребовательна к системным ресурсам.

Во время выполнения курсовой работы были изучены функции работы с файлами, способы создания классов, методы использования различных операторов, циклов, принципы работы с консольным приложением.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Microsoft Corporation. - MSDN Library 2001
2. Рихтер Дж. Windows для профессионалов: создание эффективных Win32-приложений с учетом специфики 64-разрядной версии Windows/Пер, с англ. — 4-е изд. — СПб: Питер; М.: Издательско- торговый дом «Русская Редакция», 2004. — 749 с.: ил.
3. Круглински Д., Уингоу С., Шеферд Дж. Программирование на Microsoft Visual C++ 6.0 для профессионалов/Пер, с англ. -СПб: Питер; М.: Издательско-торговый дом «Русская Редакция», 2004. — 861 с.: ил.
4. Вигерс Карл Разработка требований к программному обеспечению/Пер, с англ. — М.: Издательско-торговый дом «Русская Редакция», 2004. —576с.: ил.

# ПРИЛОЖЕНИЯ

## Приложение А

#include <iostream>

#include <windows.h>

#include <string>

#include <fstream>

#include <time.h>

using namespace std;

class ClientUnit // Класс клиента

{

private:

string Name; // Имя клиента

string Adress; // Адрес клиента

unsigned int AccountNumber; // Персональный номер аккаунта

unsigned short int InternetTariff; // Дневная плата за интернет

unsigned short int TVTariff; // Дневная плата за кабельное телевидение

int Balance; // Счёт клиента

string AccountStatus; // Состояние аккаунта

public:

// set

void setName(string Name) { this->Name = Name; }; // Установка имени

void setAdress(string Adress) { this->Adress = Adress; }; // Установка адреса

void setAccountNumber(int AccountNumber) { this->AccountNumber = AccountNumber; }; // Установка персонального клиентского номера

void setInternetTariff(int InternetTariff) { this->InternetTariff = InternetTariff; }; // Установка дневной платы за интернет

void setTVTariff(int TVTariff) { this->TVTariff = TVTariff; }; // Установка дневного тарифа за кабельное телевидение

void setBalance(int Balance) { this->Balance = Balance; }; // Установка состояния счёта клиента

void setAccountStatus(string AccountStatus) { this->AccountStatus = AccountStatus; }; // Установка состояние аккаунта

void setAccountNull() // Обнуление аккаунта

{

this->Name = "Null";

this->Adress = "Null";

this->InternetTariff = 0;

this->TVTariff = 0;

this->Balance = 0;

this->AccountStatus = "Null";

};

// get

string getName() { return Name; }; // Возврат имени

string getAdress() { return Adress; }; // Возврать адреса

int getAccountNumber() { return AccountNumber; }; // Возврат номера аккаунта

int getTVTariff() { return TVTariff; }; // Возврат дневной платы за кабельное телевидение

int getInternetTariff() { return InternetTariff; }; // Возврать дневной платы за интернет

bool Empty() // Если экземпляр пуст - вернёт true, иначе - false

{

if (this->Name == "Null")

return true;

else

return false;

};

string getTVTariffName() // Возврат названия тарифа кабельного телевидения

{

switch (TVTariff)

{

case 1: return "Мини";

break;

case 2: return "Средний";

break;

case 3: return "Мега";

break;

default: return "Не используется";

}

};

string getInternetTariffName() // Возврат названия тарифа интернета

{

switch (InternetTariff)

{

case 3: return "Мини";

break;

case 5: return "Безлим 10";

break;

case 6: return "Безлим 35";

break;

case 8: return "Безлим 50";

break;

case 10: return "Безлим 100";

break;

default: return "Не используется";

}

};

int getBalance() { return Balance; }; // Возврат баланса аккаунта

string getAccountStatus() { return AccountStatus; }; // Возврат состояния аккаунта

};

void Initialization(); // Функция начальной инициализации

void ShowClientInformation(string name, int number); // Функция отображения информации о клиентах

void Service(); // Функция обслуживания клиентов

void ClientBase(); // Функция работы с клиентской базой

void UpdateInformation(); // Функция обновления информации в базе

void Refill(); // Функция пополнения счёт

void UnitOperation(string name); // Функций работы с абоненрами

//string getDate()

//{

// string temp;

// time\_t rawtime;

// struct tm \* timeinfo;

//

// time(&rawtime);

// timeinfo = localtime(&rawtime);

// temp = asctime(timeinfo);

// return temp;

//}

ClientUnit Clients[20]; // Массив для хранения информации о клиентах

void main()

{

SetConsoleOutputCP(1251); SetConsoleCP(1251);

Initialization();

for (int Start = 0; Start != 3;)

{

system("cls");

cout << "1. Обслуживание посетителей \n2. Работа с клиентской базой \n3. Завершить работу \n>> ";

cin >> Start;

switch (Start)

{

case 1: Service();

break;

case 2: ClientBase();

break;

case 3: UpdateInformation(); system("pause");

break;

default: cout << "Некорректный вариант \n"; system("pause");

}

}

}

void Service()

{

for (int Choose = 0; Choose != 4;)

{

system("cls");

cout << "1. Пополнить счёт абонента \n2. Добавить нового абонента \n3. Заморозить аккаунт абонента \n4. Выход \n>> ";

cin >> Choose;

switch (Choose)

{

case 1: Refill();

break;

case 2: UnitOperation("new");

break;

case 3: UnitOperation("frozen");

break;

case 4: // Выход

UpdateInformation();

break;

default: cout << "Некорректный вариант\n"; system("pause");

}

}

}

void ClientBase()

{

for (int Choose = 0; Choose != 4;)

{

system("cls");

unsigned short int number;

cout << "1. Постотреть абонентскую базу \n2. Найти абонента \n3. Удалить абонента \n4. Выход \n>> ";

cin >> Choose;

switch (Choose)

{

case 1: ShowClientInformation("all", -1);

break;

case 2: system("cls");

cout << "Введите лицевой счёт абонента(0 - 19) >> ";

cin >> number;

ShowClientInformation("current", number);

break;

case 3: UnitOperation("del");

break;

case 4: // Выход

UpdateInformation();

break;

default: cout << "Некорректный вариант\n"; system("pause");

}

}

}

void UnitOperation(string name)

{

system("cls");

if (name == "new") // Новый аккаунт

{

string Name, Adress;

int Internet, TV, Balance;

cout << "Введите имя нового абонента >>";

cin.get();

getline(cin, Name);

cout << "Введите адрес нового абонента >>";

getline(cin, Adress);

for (;;)

{

cout << "Введите интернет тариф абонента \n0 - Не используется\n 3 - Мини \n5 - Безлим 10 \n6 - Безлим 35 \n8 - Безлим 50 \n10 - Безлим 100 \n>>";

cin >> Internet;

if (Internet == 0 || Internet == 3 || Internet == 5 || Internet == 6 || Internet == 8 || Internet == 10)

break;

else

cout << "Некорректный вариант\n"; system("pause");

}

for (;;)

{

cout << "Введите тариф кабельного телевидения абонента \n0 - Не используется \n1 - Мини \n2 - Средний \n3 - Мега \n>>";

cin >> TV;

if (TV == 0 || TV == 1 || TV == 2 || TV == 3)

break;

else

cout << "Некорректный вариант\n"; system("pause");

}

for (;;)

{

cout << "Введите начальный баланс абонента(0 - 5000) >>";

cin >> Balance;

if (Balance >= 0 && Balance <= 5000)

break;

else

cout << "Некорректный вариант\n"; system("pause");

}

bool flag = false;

for (int i = 0; i < 20; i++)

{

if (Clients[i].Empty() == true)

{

flag = true;

Clients[i].setName(Name);

Clients[i].setAdress(Adress);

Clients[i].setInternetTariff(Internet);

Clients[i].setTVTariff(TV);

Clients[i].setBalance(Balance);

ofstream Report("AllReports.total", ios::app);

if (Report.is\_open())

{

// Report << getDate() << endl;

Report << "Добавлен новый абонент, " << Name << endl << "Ему присвоен лицевой счёт: " << Clients[i].getAccountNumber() << endl << endl;

Report.close();

}

else

cout << "Невозможно открыть файл \"AllReports.total\", обратитесь к администратору \n"; system("pause");

break;

}

}

if (flag = false)

cout << "Cписок переполнен \n"; system("pause");

}

if (name == "frozen") // Замораживаю аккаунт

{

int AccountNumber; bool Find = false;

system("cls");

cout << "Введите номер аккаунта >> ";

cin >> AccountNumber;

for (int i = 0; i < 20; i++)

{

if (AccountNumber == Clients[i].getAccountNumber())

{

int tmp = Clients[i].getBalance();

Clients[i].setAccountStatus("Заморожен");

Clients[i].setBalance(tmp - 20);

Find = true;

UpdateInformation();

ofstream Report("AllReports.total", ios::app);

if (Report.is\_open())

{

// Report << getDate() << endl;

Report << Clients[i].getAccountNumber() << ", " << Clients[i].getAdress() << endl;

Report << "Аккаунт заморожен" << endl << endl;

Report.close();

cout << "Аккаунт успешно заморожен\n"; system("pause");

}

else

cout << "Невозможно открыть файл \"AllReports.total\", обратитесь к администратору \n"; system("pause");

break;

}

}

if (Find == false)

cout << "Абонент не найден\n"; system("pause");

}

if (name == "del") // Удаление/отключение пользователя

{

int AccountNumber; bool Find = false;

system("cls");

cout << "Введите номер аккаунта >> ";

cin >> AccountNumber;

for (int i = 0; i < 20; i++)

{

if (AccountNumber == Clients[i].getAccountNumber())

{

Find = true;

ofstream Report("AllReports.total", ios::app);

if (Report.is\_open())

{

// Report << getDate() << endl;

Report << "Аккаунт" << Clients[i].getName() << ", " << Clients[i].getAdress();

Report << " удалён из базы" << endl << endl;

Report.close();

}

else

cout << "Невозможно открыть файл \"AllReports.total\", обратитесь к администратору \n"; system("pause");

Clients[i].setAccountNull();

break;

}

}

if (Find == false)

cout << "Абонент не найден\n"; system("pause");

}

}

void Refill()

{

int AccountNumber; bool Find = false;

system("cls");

cout << "Введите номер аккаунта >> ";

cin >> AccountNumber;

for (int i = 0; i < 20; i++)

{

if (AccountNumber == Clients[i].getAccountNumber())

{

int tmp = Clients[i].getBalance(), sum = 0;

cout << "Введите сумму >> ";

cin >> sum;

Clients[i].setBalance(tmp + sum);

Find = true;

UpdateInformation();

ofstream Report("AllReports.total", ios::app);

if (Report.is\_open())

{

// Report << getDate() << endl;

Report << "Счёт абонента " << Clients[i].getAccountNumber() << ", " << Clients[i].getAdress() << endl;

Report << "Пополнен на " << sum << "руб" << endl << endl;

Report.close();

}

else

cout << "Невозможно открыть файл \"AllReports.total\", обратитесь к администратору \n"; system("pause");

break;

}

}

if (Find == false)

cout << "Абонент не найден\n"; system("pause");

}

void UpdateInformation() // Обновление информации в базе

{

ofstream Client("ClientInformation.base");

if (Client.is\_open())

{

for (int i = 0; i < 20; i++)

{

Client << Clients[i].getName() << endl;

Client << Clients[i].getAdress() << endl;

Client << Clients[i].getAccountNumber() << endl;

Client << Clients[i].getInternetTariff() << endl;

Client << Clients[i].getTVTariff() << endl;

Client << Clients[i].getBalance() << endl;

Client << Clients[i].getAccountStatus() << endl << endl;

}

Client.close();

}

else

cout << "Невозможно открыть файл \"ClientInformation.base\", обратитесь к администратору \n";

}

void Initialization()

{

ifstream Client("ClientInformation.base");

if (Client.is\_open())

{

string tmp;

for (int i = 0; i < 20; i++)

{

getline(Client, tmp); Clients[i].setName(tmp);

getline(Client, tmp); Clients[i].setAdress(tmp);

getline(Client, tmp); Clients[i].setAccountNumber(atoi(tmp.c\_str()));

getline(Client, tmp); Clients[i].setInternetTariff(atoi(tmp.c\_str()));

getline(Client, tmp); Clients[i].setTVTariff(atoi(tmp.c\_str()));

getline(Client, tmp); Clients[i].setBalance(atoi(tmp.c\_str()));

getline(Client, tmp); Clients[i].setAccountStatus(tmp);

getline(Client, tmp);

}

Client.close();

}

else

cout << "Невозможно открыть файл \"ClientInformation.base\", обратитесь к администратору \n";

cout << "Инициализация завершена" << endl;

system("pause");

}

void ShowClientInformation(string name, int number) // Показать информацию о абонентах

{

system("cls");

if (name == "all" && number == -1)

{

for (int i = 0; i < 20; i++)

{

cout << i << ". " << Clients[i].getAccountNumber() << ", ";

cout << Clients[i].getName() << endl;

cout << Clients[i].getAdress() << endl;

cout << "Баланс: " << Clients[i].getBalance() << " руб \n";

cout << "Состояние аккаунта: " << Clients[i].getAccountStatus() << endl;

cout << "Кабельное телевидение: " << Clients[i].getTVTariffName() << ", " << Clients[i].getTVTariff() << " руб/день \n";

cout << "Интернет: " << Clients[i].getInternetTariffName() << ", " << Clients[i].getInternetTariff() << " руб/день \n\n";

}

}

if (name == "current" && number >= 0 && number < 20)

{

cout << Clients[number].getAccountNumber() << ", ";

cout << Clients[number].getName() << endl;

cout << Clients[number].getAdress() << endl;

cout << "Баланс: " << Clients[number].getBalance() << " руб \n";

cout << "Состояние аккаунта: " << Clients[number].getAccountStatus() << endl;

cout << "Кабельное телевидение: " << Clients[number].getTVTariffName() << ", " << Clients[number].getTVTariff() << " руб/день \n";

cout << "Интернет: " << Clients[number].getInternetTariffName() << ", " << Clients[number].getInternetTariff() << " руб/день \n";

}

system("pause");

}

## Приложение Б

# 

Рис. 1. Основное меню программы.

# 

Рис. 2. Под меню «1. Обслуживание посетителей».

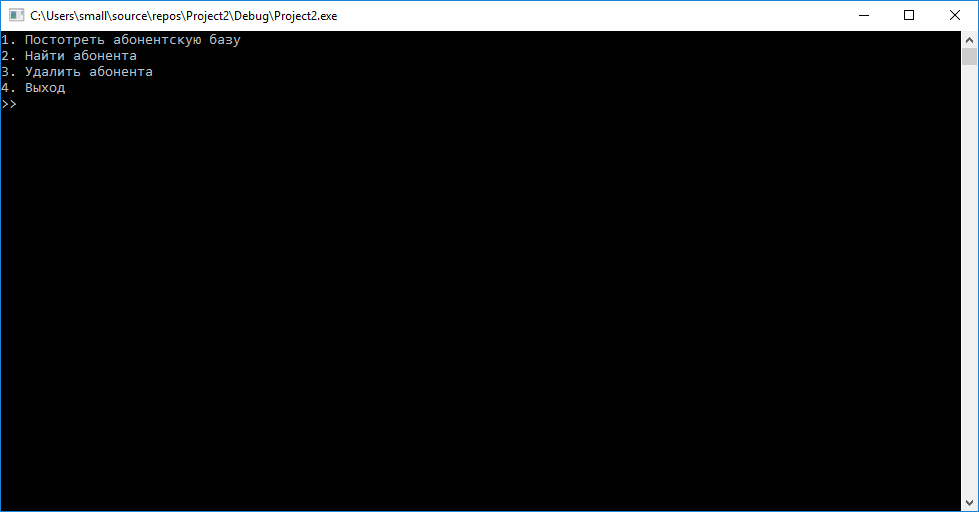


Рис 3. Под меню «2. Работа с клиентской базой»

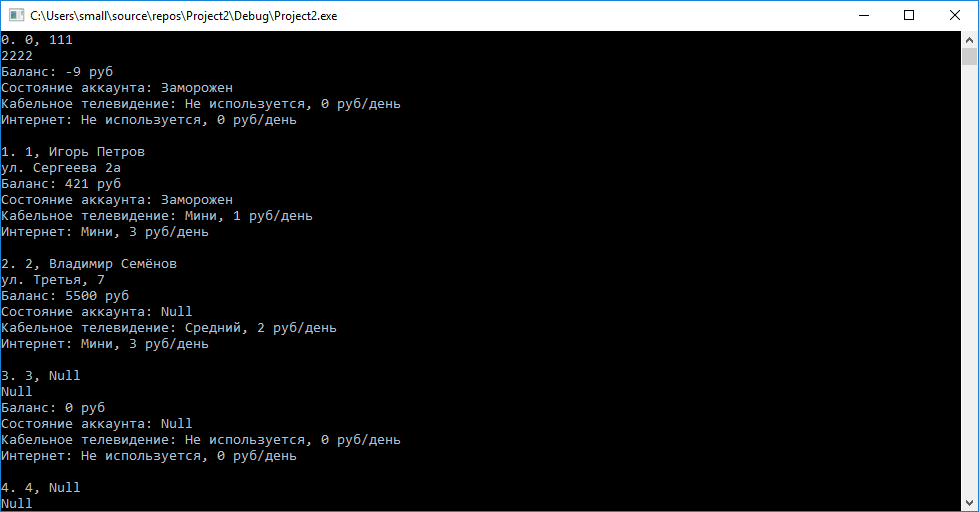


Рис 4. Отображения полной клиентской базы