|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.01 **Прикладная информатика**

**Отчет**

**по лабораторной работе № 3**

**Дисциплина:** Разработка приложений на языке C#

Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** \_\_\_\_\_\_\_\_\_\_\_\_\_

(Подпись, дата) (И.О. Фамилия)

Студент гр. ИУ6-72Б **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** И.С. Марчук

(Подпись, дата) (И.О. Фамилия)

Москва, 2022

**Цель работы:** в этой лабораторной работе обучающиеся закрепляют знания по C#, создавая консольное приложение. Программа представляет собой автоматизированную систему учета банковских сведений.

**Задание:**

Создать консольное приложение.

Программа представляет собой автоматизированную систему учета банковских сведений.

На каждого клиента банка хранятся следующие сведения:

− Ф.И.О.;

− Возраст;

− Место работы;

− Номера счетов.

На каждом счете хранится информация о текущем балансе и история прихода, расхода. Для каждого клиента может быть создано неограниченное количество счетов. С каждым счетом можно производить следующие действия: открытие, закрытие, вклад денег, снятие денег, просмотр баланса, просмотр истории.

Вся информация должна хранится в массивах. Рекомендуется объекты клиента и счета реализовать в виде классов. Баланс счета организовать в виде свойства только для чтения.

Код программы:

using System;

using System.Linq;

using System.Xml.Linq;

namespace Bank

{

// классы данных

class Client

{

public string name;

public string surname;

public string last\_name;

public int age;

public string work;

public int[] number\_account;

public bool first;

public Client(string name\_, string surname\_, string patronymic\_, int age\_, string work\_, int[] number\_account\_, bool first\_)

{

name = name\_;

surname = surname\_;

last\_name = patronymic\_;

age = age\_;

work = work\_;

number\_account = number\_account\_;

first = first\_;

}

}

class Bill

{

public int number;

public string[] history\_coming;

public string[] history\_expenses;

public string condition;

public bool work;

public int coming;

public int expenses;

private int balance;

public string[] history;

public Bill(int number\_, string[] history\_coming\_, string[] history\_expenses\_, string condition\_, bool work\_, int coming\_, int expenses\_, int balance\_, string[] history\_)

{

number = number\_;

history\_coming = history\_coming\_;

history\_expenses = history\_expenses\_;

condition = condition\_;

work = work\_;

coming = coming\_;

expenses = expenses\_;

balance = balance\_;

history = history\_;

}

public int get\_balance()

{

return balance;

}

public void get\_money(int money)

{

balance -= money;

}

public void put\_money(int money)

{

balance += money;

}

}

class Program

{

static Bill[] bill = new Bill[100];

static Client[] clients = new Client[0];

static int number\_of\_chet = 0;

static void Main(string[] args)

{

while (true)

{

Console.WriteLine("---------------------------------------------------------");

Console.WriteLine("---------------------------------------------------------");

Console.WriteLine("---------------------------------------------------------");

Console.WriteLine("Созадть нового клиента? Введите 'y' или 'n'");

string a = Console.ReadLine();

//int kol = 0;

if (a == "y" || a == "у")

{

Array.Resize<Client>(ref clients, clients.Length + 1);

clients[clients.Length - 1] = createClient();

Console.WriteLine();

}

else

{

Console.WriteLine("Начинаю работу с существующими клиентами:");

work\_with\_client();

Console.WriteLine();

}

}

}

static Client createClient()

{

Console.Write("Введите имя: ");

string name = Console.ReadLine();

Console.Write("Введите Фамилию: ");

string surname = Console.ReadLine();

Console.Write("Введите Отчество: ");

string patronymic = Console.ReadLine();

Console.Write("Возраст: ");

int age;

while(!int.TryParse(Console.ReadLine(), out age))

{

Console.WriteLine("Введен не корректный возраст!");

}

Console.Write("Место работы: ");

string work = Console.ReadLine();

int[] mas = { 0 };

bool first = true;

return new Client(name, surname, patronymic, age, work, mas, first);

}

// поиск клиента

static int find\_client()

{

int clientPos = -1;

int same\_counter = 0;

Console.Write("Поиск: Введите имя:");

string name = Console.ReadLine();

for (int i = 0; i < clients.Length; i++)

if (clients[i].name.Trim().Equals(name))

{

clientPos = i;

same\_counter++;

}

if (same\_counter == 0)

{

Console.WriteLine("Поиск: Пользователь не найден");

}

else if (same\_counter == 1)

{

Console.WriteLine("Поиск: Пользователь найден!");

}

else if (same\_counter > 1)

{

Console.WriteLine("Поиск: Найдено несколько пользователей с таким именем.");

Console.Write("Введите Фамилию: ");

string surname = Console.ReadLine();

clientPos = -1;

same\_counter = 0;

for (int i = 0; i < clients.Length; i++)

if (clients[i].name == name & clients[i].surname == surname) {

clientPos = i;

same\_counter++;

}

if (same\_counter == 0)

{

Console.WriteLine("Поиск: Пользователь не найден");

}

else if (same\_counter == 1)

{

Console.WriteLine("Поиск: Пользователь найден!");

}

else if (same\_counter > 1)

{

Console.WriteLine("Поиск: Найдено несколько пользователей с таким именем и фамилией.");

Console.Write("Введите Отчество: ");

string patronymic = Console.ReadLine();

for (int i = 0; i < clients.Length; i++)

if (clients[i].name == name & clients[i].surname == surname & clients[i].last\_name == patronymic)

{

clientPos = i;

same\_counter++;

}

if (same\_counter == 0)

{

Console.WriteLine("Поиск: Пользователь не найден");

}

else if (same\_counter == 1)

{

Console.WriteLine("Поиск: Пользователь найден!");

}

else if (same\_counter > 1)

{

Console.WriteLine("Поиск: Найдено несколько пользователей с таким именем и фамилией. ");

Console.WriteLine("Поиск: Показан последний добавленный!");

}

}

}

return clientPos;

}

static void work\_with\_client()

{

String input\_buffer;

// поиск клиента

int clientPos = find\_client();

if (clientPos != -1)

{

// Пользователь найден

do

{

Console.WriteLine("Создать новый счет? Введите 'y' или 'n'");

string a = Console.ReadLine();

Console.WriteLine(" ");

if (a == "y" || a == "у")

{

if (clients[clientPos].first == false)

{

Array.Resize<int>(ref clients[clientPos].number\_account, clients[clientPos].number\_account.Length + 1);

}

else

{

clients[clientPos].first = false;

}

int number = clients[clientPos].number\_account.Length - 1;

clients[clients.Length - 1].number\_account[number] = number\_of\_chet;

number\_of\_chet++;

string[] history\_coming = { };

string[] history\_expenses = { };

string condition = "открыт";

bool work = true;

int coming = 0;

int expenses = 0;

Console.Write("Введите баланс счета");

int balance = 0;

while (!int.TryParse(Console.ReadLine(), out balance) || balance < 0)

{

Console.WriteLine("Введено не корректное число!");

}

string[] history = { };

bill[number] = new Bill(number, history\_coming, history\_expenses, condition, work, coming, expenses, balance, history);

Console.WriteLine("Новый счет создан");

}

else

{

Console.WriteLine("Выберем один из старых счетов, с которым будем работать");

Console.Write("Счета пользователя: ");

Console.WriteLine(string.Join(",", clients[clients.Length - 1].number\_account));

Console.WriteLine("Счет, с которым будем работать: ");

int numer\_work\_chet=0;

while (!int.TryParse(Console.ReadLine(), out numer\_work\_chet) || numer\_work\_chet < 0)

{

Console.WriteLine("Введено не корректное число!");

}

if (clients[clientPos].number\_account.Contains(numer\_work\_chet)) {

Console.WriteLine();

Console.WriteLine("Операция, которая будет выполняться?");

Console.WriteLine("Введите 1, если нужно открыть/закрыть счета");

Console.WriteLine("Введите 2, если нужно просмотреть баланс");

Console.WriteLine("Введите 3, если нужно просмотреть историю");

Console.WriteLine("Введите 4, если нужно положить на счет");

Console.WriteLine("Введите 5, если нужно снять деньги");

string b = Console.ReadLine();

switch (b)

{

case "1":

Console.WriteLine("Состояние счета:");

Console.WriteLine(bill[numer\_work\_chet].condition);

if (bill[numer\_work\_chet].condition == "открыт")

{

Console.WriteLine("Закрыть счет? y/n");

a = Console.ReadLine();

if (a == "y" || a == "у")

{

bill[numer\_work\_chet].condition = "закрыт";

bill[numer\_work\_chet].work = false;

}

}

else

{

Console.WriteLine("Открыть счет? y/n");

a = Console.ReadLine();

if (a == "y" || a == "у")

{

bill[numer\_work\_chet].condition = "открыт";

bill[numer\_work\_chet].work = true;

}

}

break;

case "2":

if (bill[numer\_work\_chet].work == true)

{

Console.WriteLine("Балланс: " + bill[numer\_work\_chet].get\_balance());

}

else

{

Console.WriteLine("Счет закрыт");

}

break;

case "3":

Console.WriteLine("История работы со счетом:");

Console.WriteLine(", " + string.Join(",", bill[numer\_work\_chet].history));

break;

case "4":

if (bill[numer\_work\_chet].work == true)

{

Console.WriteLine("Сумма, которую собираетесь положить на счет:");

int money = Int32.Parse(Console.ReadLine());

bill[numer\_work\_chet].put\_money(money);

Array.Resize<string>(ref bill[numer\_work\_chet].history\_coming, bill[numer\_work\_chet].history\_coming.Length + 1);

bill[numer\_work\_chet].history\_coming[bill[numer\_work\_chet].history\_coming.Length - 1] = "Внесено: " + money;

Array.Resize<string>(ref bill[numer\_work\_chet].history, bill[numer\_work\_chet].history.Length + 1);

bill[numer\_work\_chet].history[bill[numer\_work\_chet].history.Length - 1] = "Внесено: " + money;

Console.WriteLine("Балланс: " + bill[numer\_work\_chet].get\_balance());

}

else

{

Console.WriteLine("Счет закрыт");

}

break;

case "5":

if (bill[numer\_work\_chet].work == true)

{

Console.WriteLine("Сумма, которую собираетесь снять со счета:");

int money = Int32.Parse(Console.ReadLine());

if (money < bill[numer\_work\_chet].get\_balance())

{

bill[numer\_work\_chet].get\_money(money);

Array.Resize<string>(ref bill[numer\_work\_chet].history\_expenses, bill[numer\_work\_chet].history\_expenses.Length + 1);

Array.Resize<string>(ref bill[numer\_work\_chet].history, bill[numer\_work\_chet].history.Length + 1);

bill[numer\_work\_chet].history\_expenses[bill[numer\_work\_chet].history\_expenses.Length - 1] = "Снято: " + money;

bill[numer\_work\_chet].history[bill[numer\_work\_chet].history.Length - 1] = "Снято: " + money;

Console.WriteLine("Балланс: " + bill[numer\_work\_chet].get\_balance());

}

else

{

Console.WriteLine("На счету недостаточно средств");

}

}

else

{

Console.WriteLine("Счет закрыт");

}

break;

}

}

else

Console.WriteLine("Такого счета нет!");

}

Console.WriteLine("Завершить редактирование клиента? Введите y/n");

input\_buffer = Console.ReadLine();

} while (!(String.Compare(input\_buffer, "y") == 0 || String.Compare(input\_buffer, "у") == 0));

}

}

}

}

Работа программы показана на рисунках 1 - 3.

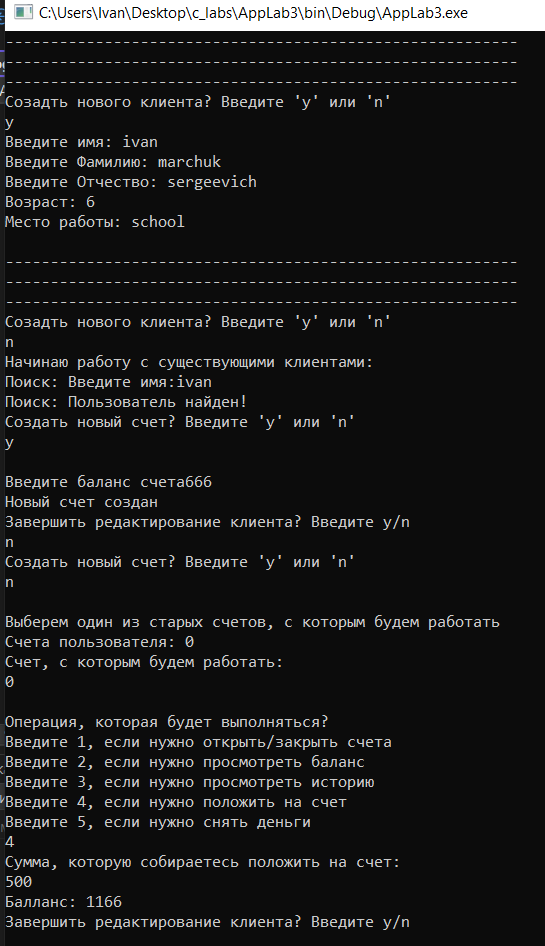


Рисунок 1 – Создание клиента, счета у него и занесение средств на счет

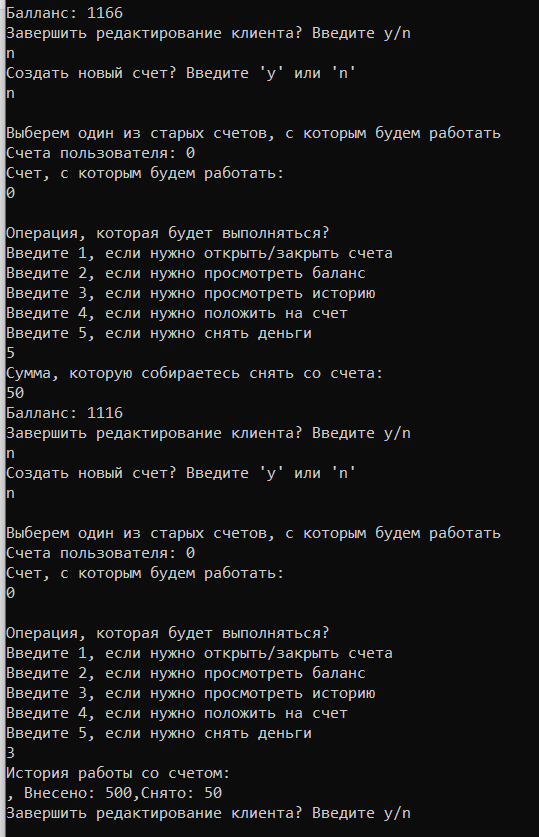


Рисунок 2 – Добвление денег на счет, снятие и просмотр истории операций

**Вывод:** В ходе лабораторной работы мною была разработана небольшая банковская система, в которой на каждого клиента банка хранятся следующие сведения: Ф.И.О., возраст, место работы, номера счетов, информация о текущем балансе и история прихода, расхода. Также для каждого клиента может быть создано неограниченное количество счетов. С каждым счетом можно производить следующие действия: открытие, закрытие, вклад денег, снятие денег, просмотр баланса, просмотр истории.