**Содержание**

[Общая постановка задачи 3](#_Toc53928)

[Требования: 4](#_Toc53929)

[Спецификация 6](#_Toc53930)

[Тест план с результатами тестов: 8](#_Toc53931)

[Диаграмма классов 16](#_Toc53932)

[Заключение 18](#_Toc53933)

[Список литературы 19](#_Toc53934)

[Приложение 1 20](#_Toc53935)

[Приложение 2 38](#_Toc53936)

# Общая постановка задачи

1. Разработать класс «Банковская карта», содержащий информацию:
   * фамилия, имя, отчество,
   * название банка,
   * номер телефона клиента,  номер карты,  баланс.
2. Для разрабатываемого класса необходимо выполнить:
   * 1. Описание полей класса.
     2. Конструктор без параметров, конструктор с параметрами, конструкторы копирования, деструктор.
     3. Перегрузку одного из бинарных операторов как метод класса.
     4. Перегрузку одного из бинарных операторов через дружественную функцию.
     5. Перегрузку одного из операторов отношения как метод класса. и через дружественную функцию.
     6. Перегрузку одного из операторов отношения через дружественную функцию.
     7. Перегрузку оператора присваивания.
     8. Перегрузку префиксного инкремента как метод класса.
     9. Перегрузку постфиксного инкремента через дружественную функцию. 10) Перегрузку операторов << и >> через дружественные функции.
3. Создать массив из объектов разработанного класса. Ввести в массив данные из текстового файла. Имя файла со входными данными должны вводиться из потока cin.
4. Определить самую большую потраченную сумму. Для сравнения использовать перегруженный оператор отношения, поиск элемента выполнить в отдельной процедуре.
5. Выполнить сортировку массива методом простых вставок по балансу. Для сортировки написать и использовать шаблон функции. Результат вывести в файл в виде таблицы, сделать заголовки колонок таблицы.
6. Создать список названий банков, выдавших карты. Результат вывести в файл в виде таблицы, сделать заголовки колонок таблицы
7. Предусмотреть обработку и инициализацию исключительных ситуаций, связанных, например, с проверкой значения полей перед инициализацией и присваиванием
8. Программа должна быть написана в соответствии со стандартом программирования: C++ Programming Style Guidelines (http://geosoft.no/development/cppstyle.html).

## Требования:

Функциональные требования:

1. Требования к файлу:
   * Путь к файлу должен быть указаны верно
   * Файл не должен быть пустым
2. Требования к полям списка: • ФИО должны быть написаны только латиницей и начинаться с большой буквы.
   * Между ФИО должны стоять пробелы.
   * После Отчества ставим пробел и точку с запятой
   * Название банка должны быть взяты в кавычки
   * Название банка должно быть написано только латиницей и начинаться с большой буквы
   * Номер телефона должен начинаться с ‘+’
   * Номер телефона должен состоять только из цифр
   * Номер телефона не может содержать больше 11 цифр
   * Банковская карта должна содержать только 16 цифр
   * После каждых 4-х цифр номера банковской карты ставим пробел
   * Баланс должен быть задан только целым числом
3. Все поля, кроме последнего, должны быть отделены пробелом и точкой с запятой и пробелом( ; )

Нефункциональные требования:

1. Используется технология ООП 2. Класс bankСard должен содержать:

* Конструктор без параметров, конструктор с параметрами, конструкторы копирования, деструктор;
* Перегрузку одного из бинарных операторов как метод класса; • Перегрузку одного из бинарных операторов через дружественную функцию;
* Перегрузку одного из операторов отношения как метод класса и через дружественную функцию;
* Перегрузку одного из операторов отношения через дружественную функцию;
* Перегрузку оператора присваивания;
* Перегрузку префиксного инкремента как метод класса;
* Перегрузку постфиксного инкремента через дружественную функцию;
* Перегрузку операторов « и » через дружественные функции;

1. Данные вводятся из текстового файла.
2. Для сортировки и поиска клиента с максимальным балансом используется перегруженный оператор отношения, поиск этого клиента выполняется в отдельной процедуре.
3. Для сортировки используется функция, которая выполняет сортировку по возрастанию по одному из полей класса(балансу).
4. Результат выводится в файл в виде таблице
5. В файл выводятся названия всех банков
6. Для сортировки данных используется метод простых вставок по балансу
7. Для обработки ошибок создается собственный класс
8. Программа должна быть написана в соответствии со стандартом программирования: C++ Programming Style Guidelines

## Спецификация

|  |  |  |
| --- | --- | --- |
| **№ спецификации** | **Входные данные** | **Результат** |
| 1 | Неверно указан путь к  файлу или файл  отсутствует | Сообщение: «Файл не найден» |
| 2 | Поля не ограниченны точкой с запятой и пробелом | Сообщение: «Не верный разделитель» |
| 3.1 | Фамилия задана не с большой буквы | Сообщение: «Фамилия не с заглавной буквы в » №строки « строке» |
| 3.2 | Фамилия задана неверно(присутствует  заглавная буква или посторонний элемент) | Сообщение : «В фамилии посторонний элемент или  заглавная буква в» №строки  «строке» |
| 3.3 | Имя задано не с большой буквы | Сообщение: «Имя написано не с большой буквы в » №строки  « строке» |
| 3.4 | Имя задано неверно(присутствует  заглавная буква или посторонний элемент) | Сообщение : «В имени посторонний элемент или  заглавная буква в» №строки  «строке» |
| 3.5 | Отчество задано не с большой буквы | Сообщение: «Отчество написано не с большой буквы  » №строки « строке» |
| 3.6 | Отчество задано неверно(присутствует  заглавная буква или посторонний элемент) | Сообщение : «В отчестве посторонний элемент  или  заглавная буква в» №строки  «строке» |
| 4 | Название банка не взято в кавычки | Сообщение: «Название банка не взято в кавычки в » №строки « строке» |
| 4.1 | Название банка написано не с большой буквы | Сообщение: «Название банка написано не с большой буквы в |

|  |  |  |
| --- | --- | --- |
|  |  | » №строки « строке» |
| 4.2 | В названии банка присутствует посторонний элемент | Сообщение: «В названии банка присутствует посторонний элемент в» №строки « строке» |
| 5 | Номер телефона  начинается не с ‘+’ | Сообщение: «Номер телефона начинается  не с ‘+’ в »  №строки « строке» |
| 5.1 | В номере телефона есть не цифра | Сообщение: «В номере телефона есть не цифра в »  №строки « строке» |
| 5.2 | Номер телефона состоит больше чем из 11 цифр | Сообщение: «Номер телефона слишком длинный в » №строки  « строке» |
| 6 | В первых 4-ч цифрах номера карты  присутствует не число | Сообщение: «В первых 4-ч цифрах номера карты не число в » №строки « строке» |
| 6.1 | Во вторых 4-ч цифрах номера карты  присутствует не число | Сообщение: «Во вторых 4-ч цифрах номера карты не число в » №строки « строке» |
| 6.2 | В третьих 4-ч цифрах номера карты  присутствует не число | Сообщение: «В третьих 4-ч цифрах номера карты не число в » №строки « строке» |
| 6.3 | В четвертых 4-ч цифрах номера карты  присутствует не число | Сообщение: «В четвертых 4-ч цифрах номера карты не число в » №строки « строке» |
| 7 | Баланс задан неверно или не целым числом | Сообщение: «Баланс задан не целым  положительным »  №строки « строке» |
| 8 | Все поля заданы верно | Выводится список, с максимальным балансом |
| 8.1 | Все поля заданы верно | Выводится список, отсортированный по  балансу |
| 8.2 | Все поля заданы верно | Выводится список банков, выдавших карту (вывод в файл) |

**Все ошибки обрабатываются в классе BNF**

## Тест план с результатами тестов:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№**  **теста** | **№**  **спец ифик ации** | **Входные данные** | **Ожидаемый**  **результат** | **Подтвер ждение** |
| 1 | 1 | ifstream cardsFile(“Output.txt”); | Сообщение: «Файл не найден» | + |
| 2 | 1 | Zugen Furname Atronymic ; "Bank"  +11111111111 ;  2112 2222 2122 2222 ; 2  Zuge Furname Yronymic ; "BankSber" ;  +22222222222 ; 2222 2122 2222 2222 ;  2 | Сообщение: «Не верный разделитель» | + |
| 2 | 2 | Zuge Furname Yronymic Ж "Bank" ;  +22222222222 ; 2222 2122 2222 2222 ;  2 | Сообщение: «Не верный разделитель» | + |
| 2 | 3 | Zuge Furname Yronymic;"Bank" ;  +22222222222 ; 2222 2122 2222 2222 ;  2 | Сообщение: «Не верный разделитель» | + |
| 3.1 | 1 | zuge Furname Yronymic ; "Bank" ; +22222222222 ; 2222 2122 2222 2222 ;  2 | Сообщение: «Фамилия написана не с большой буквы в » №строки « строке» | + |
| 3.2 | 1 | ZgHGe Furname Yronymic ; "Bank" ;  +22222222222 ; 2222 2122 2222 2222 ;  2 | Сообщение : «В фамилии посторонний элемент или заглавная буква в» №строки  «строке» | + |
| 3.3 | 1 | Zuge furname Yronymic ; "Bank" ;  +22222222222 ; 2222 2122 2222 2222 ;  2 | Сообщение: «Имя написано не с большой буквы в »  №строки « строке» | + |
| 3.4 | 1 | Zuge Fu3rname Yronymic ; "Bank" ;  +22222222222 ; 2222 2122 2222 2222 ;  2 | Сообщение : «В имени посторонний элемент или заглавная буква в» №строки  «строке» | + |

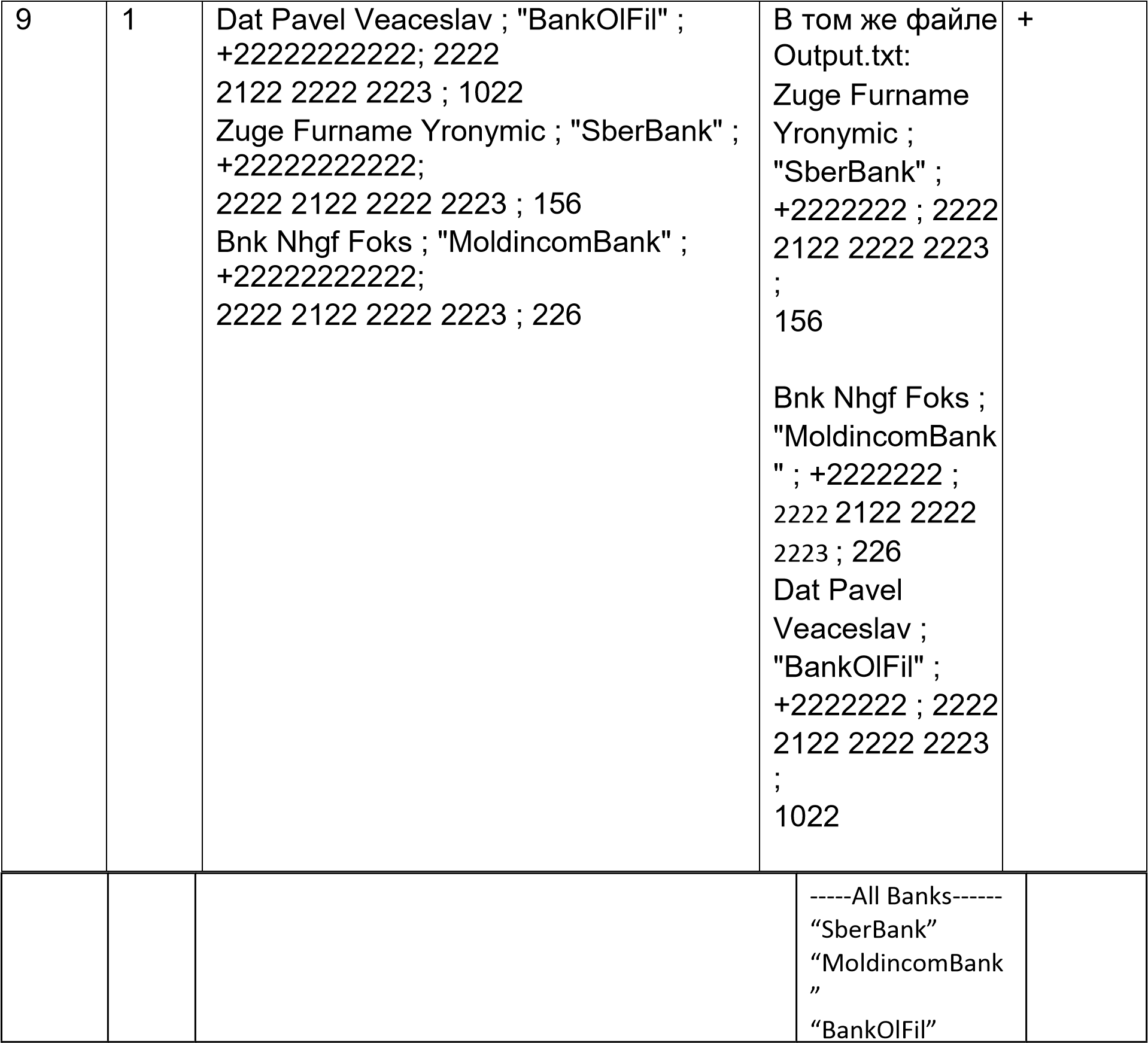
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3.5 | 1 | Zuge Furname yronymic ; "Bank" ;  +22222222222 ; 2222 2122 2222 2222 ;  2 | Сообщение: «Отчество | + |
|  |  |  | написано не с большой буквы в » №строки « строке» |  |
| 3.6 | 1 | Zugge Furname YronyДmic ; "Bank" ;  +22222222222 ; 2222 2122 2222 2222 ;  2 | Сообщение : «В отчестве посторонний элемент или заглавная буква в» №строки  «строке» | + |
| 4 | 1 | Zuge Furname Yronymic ; Bank ;  +22222222222 ; 2222 2122 2222 2222 ;  2 | Сообщение: «Название банка не взято в кавычки в » №строки « строке» | + |
| 4.1 | 1 | Zuge Furname Yronymic ; "bank" ;  +22222222222 ; 2222 2122 2222 2222 ;  2 | Сообщение: «Название банка написано не с большой буквы в »  №строки « строке» | + |
| 4.2 | 1 | Zuge Furname Yronymic ; "Bank;`@" ;  +22222222222 ; 2222 2122 2222 2222 ;  2 | Сообщение: «В названии банка присутствует посторонний элемент в » №строки « строке» | + |
| 4.2 | 2 | Zuge Furname Yronymic ; "Bank2" ;  +22222222222 ; 2222 2122 2222 2222 ;  2 | Сообщение: «В названии банка присутствует посторонний элемент в » №строки « строке» | + |
| 5 | 1 | Zuge Furname Yronymic ; "Bank" ;  222222222 ; 2222 2122 2222 2222 ; 2 | Сообщение: «Номер телефона начинается не с  ‘+’ в » №строки « строке» | + |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5.1 | 1 | Zuge Furname Yronymic ; "Bank" ;  +2f222222 ; 2222 2122 2222 2222 ; 2 | Сообщение: «В номере телефона есть не цифра в » №строки « строке» | + |
| 5.2 | 1 | Zuge Furname Yronymic ; "Bank" ;  +21254265222222 ; 2222 2122 2222 2222 ; 2 | Сообщение: «Недопустимое кол-во цифр в номере в » №строки « строке» | + |
|  |  |  |  |
| 6 | 1 | Zuge Furname Yronymic ; "Bank" ;  +22222222222 ; 2222 2g22 2222 2222 ;  2 | Сообщение: «В первых 4-ч цифрах номера карты не число в » №строки « строке» | + |
| 6.1 | 1 | Zuge Furname Yronymic ; "Bank" ;  +22222222222; 2222 2122 2g22 2222 ;  2 | Сообщение: «Во вторых 4-ч цифрах номера карты не число в » №строки « строке» | + |
| 6.2 | 1 | Zuge Furname Yronymic ; "Bank" ;  +22222222222; 2222 2122 2222 22g2 ;  2 | Сообщение: «В третьих 4-ч цифрах номера карты не число в » №строки « строке» | + |
| 6.3 | 1 | Zuge Furname Yronymic ; "Bank" ;  +22222222222; 2222 2122 2222 222h ;  2 | Сообщение: «В четвертых 4-ч цифрах номера карты не число в » №строки « строке» | + |
| 7 | 1 | Zuge Furname Yronymic ; "Bank" ;  +22222222222; 2222 2122 2222 2221 ; 564а | Сообщение: «Баланс задан не целым положительным » №строки « строке» | + |
| 7 | 2 | Zuge Furname Yronymic ; "Bank" ;  +22222222222; 2222 2122 2222 2223 ; 2.26 | Сообщение: «Баланс задан не целым положительным » №строки « строке» | + |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 7 | 3 | Zuge Furname Yronymic ; "Bank" ;  +22222222222; 2222 2122 2222 2223 ; 226 | Сообщение: «Баланс задан не целым положительным » №строки « строке» | + |
| 8 | 1 | Dat Pavel Veaceslav ; "Bank" ;  +22222222222; 2222  2122 2222 2223 ; 1022  Zuge Furname Yronymic ; "Bank" ;  +22222222222; 2222 2122 2222 2223 ;  156  Bnk Nhgf Foks ; "Bank" ;  +22222222222; 2222 2122 2222 2223 ; 226 | Сообщение: «  ==Max balance=  Dat Pavel  Veaceslav ;  "Bank" ;  +2222222 ; 2222  2122 2222 2223  ;  1022  » | + |
| 8 | 2 | Dat Pavel Veaceslav ; "Bank" ;  +22222222222; 2222  2122 2222 2223 ; 1022  Zuge Furname Yronymic ; "Bank" ;  +22222222222; 2222  2122 2222 2223 ; 1560  Bnk Nhgf Foks ; "Bank" ;  +22222222222; 2222 2122 2222 2223 ; 226 | Сообщение: «  ==Max balance=  Zuge Furname  Yronymic ;  "Bank"  ; +2222222 ;  2222  2122 2222 2223  ;  1560» | + |
| 8 | 3 | Dat Pavel Veaceslav ; "Bank" ;  +22222222222; 2222  2122 2222 2223 ; 1022  Zuge Furname Yronymic ; "Bank" ;  +22222222222; 2222  2122 2222 2223 ; 156  Bnk Nhgf Foks ; "Bank" ;  +22222222222; 2222 2122 2222 2223 ; 2266 | Сообщение: «  ==Max balance=  Bnk Nhgf Foks ;  "Bank" ;  +2222222 ; 2222  2122 2222 2223  ;  2266» | + |

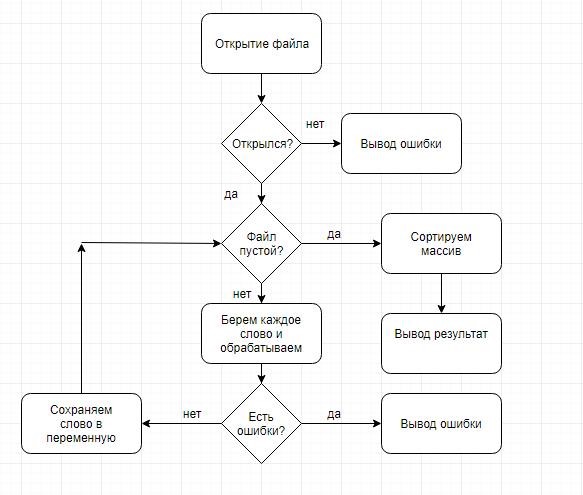
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 8.1 | 1 | Dat Pavel Veaceslav ; "Bank" ;  +22222222222; 2222  2122 2222 2223 ; 1022  Zuge Furname Yronymic ; "Bank" ;  +22222222222; 2222  2122 2222 2223 ; 156  Bnk Nhgf Foks ; "Bank" ;  +22222222222; 2222 2122 2222 2223 ; 2266 | В файле  Output.txt:  Zuge Furname  Yronymic ;  "Bank"  ; +2222222 ;  2222  2122 2222 2223  ;  156    Dat Pavel  Veaceslav ;  "Bank" ;  +2222222 ; 2222  2122 2222 2223  ;  1022    Bnk Nhgf Foks ;  "Bank" ;  +2222222 ; 2222  2122 2222 2223  ;  2266 | + |
| 8.1 | 2 | Dat Pavel Veaceslav ; "Bank" ;  +22222222222; 2222  2122 2222 2223 ; 1022  Zuge Furname Yronymic ; "Bank" ;  +22222222222; 2222  2122 2222 2223 ; 1560  Bnk Nhgf Foks ; "Bank" ;  +22222222222; 2222 2122 2222 2223 ; 226 | В файле  Output.txt:  Bnk Nhgf Foks ;  "Bank" ;  +2222222 ; 2222  2122 2222 2223  ;  226    Dat Pavel  Veaceslav ;  "Bank" ;  +2222222 ; 2222 | + |
|  |  |  | 2122 2222 2223  ;  1022    Zuge Furname  Yronymic ;  "Bank"  ; +2222222 ;  2222 |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | 2122 2222 2223  ;  1560 |  |
| 8.1 | 3 | Dat Pavel Veaceslav ; "Bank" ;  +22222222222; 2222  2122 2222 2223 ; 1022  Zuge Furname Yronymic ; "Bank" ;  +22222222222; 2222  2122 2222 2223 ; 156  Bnk Nhgf Foks ; "Bank" ;  +22222222222; 2222 2122 2222 2223 ; 226 | В файле  Output.txt:  Zuge Furname  Yronymic ;  "Bank"  ; +2222222 ;  2222  2122 2222 2223  ;  156    Bnk Nhgf Foks ;  "Bank" ;  +2222222 ; 2222  2122 2222 2223  ;  226  Dat Pavel  Veaceslav ;  "Bank" ;  +2222222 ; 2222  2122 2222 2223  ;  1022 | + |



## Диаграмма классов

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | |  | | --- | | **myException** | | -error\_: string  -line\_ : int | | +myException(error : string, line : int)  +getError() | |  | |  | | --- | | **BNF** | | +isName(string , int) : bool  + isBankName(string, int) : bool  + isBankNumber(string, int) : bool  +isPhoneNumber(string, int) : bool  +isBalance(string, int) : bool  +isSeparator(string, int) : bool | | |
| |  | | --- | | string; iostream | | |
|  | |  | | --- | | **bankCard** | | -clientName : string  -bankName : string  -phoneNumber : string  -cardNumber : string  -balanceClient : string | | +bankCard()  +bankCard(c1 : const bankcard& )  +bankCard(clientName\_ : string, bankName\_ : string, phoneNumber\_ : string, cardNumber\_ : string, balanceClient\_ : int) | | +setClientName(text : string, index : int) : void  +setBankName(text : string, index : int) : void  +setPhoneNumber(text : string, index : int) : void  +setCardNumber(text : string, index : int) : void  +setBalanceClient(text : string, index : int) : void  +setSeparator(text : string, index : int) : void    +setClientName(text : string, index : int) : string  +setBankName(text : string, index : int) : string  +setPhoneNumber(text : string, index : int) : string  +setCardNumber(text : string, index : int) : string  +setBalanceClient(text : string, index : int) : string  +setSeparator(text : string, index : int) : string | | | | |



# Заключение

На общей постановке задачи был разработан класс bankСard, в котором хранятся данные об клиенте банковской карты. Были разработаны перегруженные операции данного класса(дружественные и методы класса). Изучен алгоритм сортировки простыми вставками, после чего в конечный файл выводится таблица вида :

ФИО ; «Название банка» ; Номер телефона клиента ; Номер карты ; Баланс

Разработана функция поиска клиента с максимальным балансом и выводит его на экран. Разработана функция вывода в тот же конечный файл название всех банков, используемых в исходном файле. Вывод банков в файл имеет вид: =====All banks===

«Bank1»

«Bank2»

«Bank3»

# Список литературы

1. Павловская Т.А. С/С++. Программирование на языке высокого уровня.- СПб.:Питер, 2002 – 464с.

1. Прата, Стивен. Язык программирования С++. Лекции и упражнения, 6-е изд.: Пер.с англ. – М.: ООО “И.Д. Вильямс”, 2013.- 1248 с.

1. [Галина Ваныкина,](http://www.intuit.ru/studies/courses/648/504/info) [Татьяна Сундукова.](http://www.intuit.ru/intuituser/demands/author/403) [Структуры и алгоритмы компьютерной обработки данных](http://www.intuit.ru/studies/courses/648/504/info) – URL: <http://www.intuit.ru/studies/courses/648/504/info>

1. [Бьерн Страуструп.](http://www.intuit.ru/intuituser/demands/author/400) Язык программирования C++ для профессионалов.– URL: <http://www.intuit.ru/studies/courses/98/98/info>

1. Р. Лафоре. Объектно-ориентированное программирование изд.4, СПб.Питер, 2004

# Приложение 1

#include <iostream>

#include <string>

#include <vector>

#include <windows.h>

#include <stdlib.h>

#include <fstream>

#include <iomanip>

#include <cmath>

#include <ctype.h>

#include <stdio.h>

#include <sstream>

#include <exception>

using namespace std;

//class for Errors class myException

{ private:

string error\_; int line\_; public:

myException(string error, int line):error\_(error), line\_(line){}

string getError(void)

{

cout<<error\_ << " в "<< line\_+1 <<" строке "<<endl;

}

};

class BNF

{

public:

bool isName(string , int); bool isBankName(string, int); bool isBankNumber(string, int); bool isPhoneNumber(string, int); bool isBalance(string, int); bool isSeparator(string, int);

};

//the main class class bankCard : public BNF

{

private:

string clientName\_; string bankName\_; string phoneNumber\_; string cardNumber\_; int balanceClient\_; public:

bankCard();

bankCard(const bankCard &c1); bankCard(string clientName, string bankName, string phoneNumber, string cardNumber, int balanceClient);

~bankCard();

//Setters

void setClientName(string text, int index);

void setBankName(string text, int index); void setPhoneNumber(string text, int index); void setCardNumber(string text, int index);

void setBalanceClient(string text, int index); void setSeparator(string text, int index);

//Getters

string getClientName(void)

{

return clientName\_;

}

string getBankName(void)

{

return bankName\_;

}

string getPhoneNumber(void)

{

return phoneNumber\_;

}

string getCardNumber(void)

{

return cardNumber\_;

}

int getBalanceClient(void)

{

return balanceClient\_;

}

string getSeparator(void)

{

return " ; ";

}

friend bool operator==(bankCard c1, bankCard c2); friend bool operator>(bankCard c1, bankCard c2); bool operator<(bankCard c1); bankCard operator++(); friend int operator++(bankCard, int); bankCard operator=(bankCard& c1); friend ostream& operator<<(ostream &out, const bankCard &card); friend istream& operator>> (istream &s, bankCard &c1); bool operator!=(bankCard& c1); bankCard operator+=(const bankCard& c2);

};

//Проверяем задано ли имя правильно(проверяем, что ФИО написаны с большой буквы, что в них не содержатся посторонние символы и задано только латиницей) bool BNF::isName(string s, int index)

{

int pos = index;

//Проверим фамилию на заглавную букву

if(s[0] != toupper(s[0]))

throw myException("Фамилия не с заглавной буквы ", pos);

//ищем первый пробел и проверяем фамилию на ошибки int i1; for(int i = 1; i<s.length(); i++)

{

if(s[i] == ' ')

{

i1 = i; break;

}

if(!(s[i]>='a' && s[i]<='z'))

throw myException("В фамилии постороний элемент или заглавная буква ", pos);

}

//Проверим, что имя написано с большой буквы if(s[i1+1] != toupper(s[i1+1]))

throw myException("Имя написано не с большой буквы ", pos);

//Найдем второй пробел и проверим на ошибки имя int i2; int sp = i1+2; for(sp; sp<s.length(); sp++)

{

if(s[sp] == ' ')

{

i2 = sp; break;

}

if(!(s[sp]>='a' && s[sp]<='z'))

throw myException("В имени посторонний элемен или заглавная

буква ", pos);

}

//Проверим отчество на заглавную букву if(s[i2+1] != toupper(s[i2+1]))

throw myException("Отчество написано не с большой буквы ", pos);

//Проверим отчество на ошибки int spc = i2+2; for(spc; spc<s.length(); spc++)

{

if(!(s[spc] >= 'a' && s[spc] <= 'z'))

throw myException("В отчестве посторонний элемент или заглавная буква ", pos);

}

return true;

}

//Проверяем задано ли название банка верно(в одно слово, взято в ковычки, и не встречаются посторонние символы, кроме "-") bool BNF::isBankName(string s, int pos)

{

int n = s.length();

if(s[0] != '\"' || s[n-1]!='\"')

throw myException("Название банка не взято в кавычки ", pos); if(s[1] != toupper(s[1]))

throw myException("Название банка написано не с большой буквы ", pos);

for(int i = 1; i<s.length()-1; i++)

{

if(!(s[i]>='a' && s[i]<='z' || s[i]) == '-')

throw myException("В названии банка присутсует русский \nсимвол или посторонний элемент ", pos);

}

return true;

}

//Проверим номер телефона(вид +12314567890, кол-во цифр не должно превышать 11 элементов) bool BNF::isPhoneNumber(string s, int pos)

{

if(s[0]!='+')

throw myException("Номер телефона начинается не с '+' ", pos); for(int i = 1; i<s.length(); i++)

{

if(!isdigit(s[i])) throw myException("В номере телефона есть не цифра ", pos);

}

if(s.length() != 12)

throw myException("Недопустимое кол-во цифр в номере ", pos);

return true;

}

//Проверяем номер карты(вид 1111 1111 1111 1111) bool BNF::isBankNumber(string s, int pos)

{

for(int i = 0; i<4; i++) if(!isdigit(s[i]))

throw myException("В первых 4-х цифрах номера карты не число

", pos);

for(int i = 5; i<9; i++)

if(!isdigit(s[i]))

throw myException("Во вторых 4-х цифрах номера карты не число ", pos);

for(int i = 10; i<14; i++) if(!isdigit(s[i]))

throw myException("В третьих 4-х цифрах номера карты не число ", pos);

for(int i = 15; i<19; i++) if(!isdigit(s[i]))

throw myException("В последних 4-х цифрах номера карты не число ", pos);

return true;

}

//Проверяем баланс(вид 123) bool BNF::isBalance(string s, int pos)

{

for(int i = 0; i<s.length(); i++)

{

if(!isdigit(s[i]))

throw myException("Баланс задан не целым положительным числом ", pos);

}

return true;

}

//Проверяем разделительный знак(;) bool BNF::isSeparator(string s, int pos)

{

if(s[0] != ';')

throw myException("Не верный разделитель ", pos);

return true;

}

//Constructors and destructors bankCard::bankCard(void){} bankCard::bankCard(string clientName, string bankName, string phoneNumber, string cardNumber, int balanceClient):

clientName\_(clientName), bankName\_(bankName), phoneNumber\_(phoneNumber), cardNumber\_(cardNumber), balanceClient\_(balanceClient)

{}

bankCard::bankCard(const bankCard &c1):clientName\_(c1.clientName\_), bankName\_(c1.bankName\_), phoneNumber\_(c1.phoneNumber\_), cardNumber\_(c1.cardNumber\_), balanceClient\_(c1.balanceClient\_)

{}

bankCard::~bankCard()

{}

//Setters void bankCard::setClientName(string text, int index)

{

if(isName(text,index))

clientName\_ = text;

}

void bankCard::setBankName(string text, int index)

{

if(isBankName(text, index))

bankName\_ = text;

}

void bankCard::setPhoneNumber(string text, int index)

{

if(isPhoneNumber(text, index)) phoneNumber\_ = text;

}

void bankCard::setCardNumber(string text, int index)

{

if(isBankNumber(text, index)) cardNumber\_ = text;

}

void bankCard::setBalanceClient(string text, int index)

{

if(isBalance(text, index))

{

//преобразуем строку в int

istringstream iss(text, istringstream::in);

int value;

iss>>value; balanceClient\_ = value;

}

}

void bankCard::setSeparator(string text, int index)

{

isSeparator(text, index);

}

//Ovvrides bool operator==(bankCard c1, bankCard c2) {

if(c1.balanceClient\_ == c2.balanceClient\_ && c1.clientName\_ == c2.clientName\_

&&

c1.bankName\_ == c2.bankName\_ && c1.phoneNumber\_ == c2.phoneNumber\_ && c1.cardNumber\_ == c2.cardNumber\_)

return true;

return false;

}

bool operator>(bankCard c1, bankCard c2)

{

return (c1.balanceClient\_ >= c2.balanceClient\_);

}

ostream& operator<<(ostream &out, const bankCard &card)

{

out<<card.clientName\_<<" ; "<<card.bankName\_<<" ;

"<<card.phoneNumber\_<<" ; "<<card.cardNumber\_<<" ; "<<card.balanceClient\_<<endl; return out;

}

bankCard bankCard::operator++()

{

balanceClient\_ += 1; return \*this;

}

bool bankCard::operator!=(bankCard& c1)

{

if(this->clientName\_ != c1.clientName\_, this->bankName\_ != c1.bankName\_, this->phoneNumber\_ != c1.phoneNumber\_, this->cardNumber\_ != c1.cardNumber\_,

this->balanceClient\_ != c1.balanceClient\_)

return true;

else

return false;

}

bankCard bankCard::operator=(bankCard& c1)

{

if(\*this != c1) return bankCard(this->clientName\_ = c1.clientName\_, this->bankName\_ = c1.bankName\_, this->phoneNumber\_ = c1.phoneNumber\_, this->cardNumber\_ = c1.cardNumber\_,

this->balanceClient\_ = c1.balanceClient\_); else

return \*this;

}

int operator++(bankCard c1, int)

{

c1.balanceClient\_+=1; return c1.balanceClient\_;

}

istream& operator>> (istream &s, bankCard &c1)

{

cout<<"Enter the name of client: "; cin>>c1.clientName\_; cout<<"\nEnter name of the bank: "; cin>>c1.bankName\_;

cout<<"\nEnter telefon number: "; cin>>c1.phoneNumber\_; cout<<"\nEnter card number: "; cin>>c1.cardNumber\_; cout<<"\nEnter balance: "; cin>>c1.balanceClient\_;

return s;

}

bool bankCard::operator<(bankCard c1)

{

return (this->balanceClient\_ < c1.balanceClient\_);

}

bankCard bankCard::operator+=(const bankCard& c2)

{

this->balanceClient\_ += c2.balanceClient\_; return \*this;

}

////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////

const int MAX\_SIZE = 100;

//Поиск максимального баланса

void putAllBanks(string output[], bankCard arr[], int &index)

{

int pos = 1; bool marker;

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

{

marker = true;

for(int j = 0; j<pos; j++)

{

if(arr[i].getBankName() == output[j])

marker = false;

}

if(marker)

{

output[pos-1] = arr[i].getBankName();

pos++;

}

}

index = pos;

}

//Распределение в порядке возрастания баланса template<typename T> void insertionSort(bankCard arr[], T n)

{

T i, j;

bankCard key;

for (i = 1; i < n; i++)

{

key = arr[i]; j = i - 1;

while (j >= 0 && arr[j] > key)

{

arr[j + 1] = arr[j]; j = j-1;

}

arr[j + 1] = key;

}

}

//Вывод максимального на экран и сортировка массива void coutMaxBalance(bankCard allData[], int &index)

{

cout<<"=============Max balance============="<<endl;

insertionSort(allData, index); cout<<allData[index-1].getClientName()<<" ; "; cout<<allData[index-1].getBankName()<<" ; ";

cout<<allData[index-1].getPhoneNumber()<<" ; "; cout<<allData[index-1].getCardNumber()<<" ; "; cout<<allData[index-1].getBalanceClient()<<endl;

}

string separator()

{

return " ; ";

}

int main()

{

try{

setlocale(LC\_ALL, "RUS"); string allBanks[MAX\_SIZE]; bankCard allData[MAX\_SIZE]; int index = 0; string fileDir;

ifstream cardsFile; //Вход ofstream responseFile; //Вывод

//Чтение из файла

cardsFile.open("List.txt"); if(!cardsFile.is\_open())

throw "Файл не найден";

responseFile.open("Output.txt");

string name, surname, patronymic, bank,

phoneNumber,cardNumber,razd1,razd2,razd3,razd4,card1,card2,card3,card4; string balanceClient; while(index < MAX\_SIZE &&

cardsFile>>name>>surname>>patronymic>>razd1>>bank>>razd2>>phoneNumber>>r azd3>>card1>>card2>>card3>>card4>>razd4>>balanceClient)

{

allData[index].setClientName(name + " " + surname + " " + patronymic, index);

allData[index].setSeparator(razd1, index); allData[index].setBankName(bank, index); allData[index].setSeparator(razd2, index); allData[index].setPhoneNumber(phoneNumber, index); allData[index].setSeparator(razd3, index); allData[index].setCardNumber(card1+ " " + card2 + " " + card3 + " " + card4, index);

allData[index].setSeparator(razd4, index); allData[index].setBalanceClient(balanceClient, index);

index++;

}

//Сортировка и вывод максимального элемента coutMaxBalance(allData, index);

//Вывод в файл в таблице int i = 0;

//responseFile<<" Name Client Bank Name Phone

Card Balance"<<endl;

responseFile<<setw(30)<<left<<"==Name

Client=="<<separator()<<setw(20)<<"==Bank

Name=="<<separator()<<setw(12)<<"==Phone=="<<separator()<< setw(20)<<"==Card=="<<separator()<<setw(8)<<"==Balance=="<<endl;

responseFile<<"--------------------------------------------------------------------------------------

-------------------------";

responseFile<<setw(33)<<right<<separator()<<setw(23)<<separator()<<setw(15)

<<separator()<<setw(23)<<separator()<<setw(11)<<endl;

while(i<index)

{

responseFile<<setw(30)<<left<<allData[i].getClientName()<<separator();

responseFile<<setw(20)<<allData[i].getBankName()<<separator(); responseFile<<setw(12)<<allData[i].getPhoneNumber()<<separator(); responseFile<<setw(20)<<allData[i].getCardNumber()<<separator(); responseFile<<setw(8)<<allData[i].getBalanceClient()<<endl;

i++; }

//Вывод в тот же файл названия всех банков responseFile << "\n"; responseFile << "\n"; responseFile << "--------------ALL BANKS--------------\n";

i = 0;

putAllBanks(allBanks, allData, index); while (i < index ) { responseFile << allBanks[i] << "\n";

i++;

}

return 0;

}

catch(myException& ex)

{

cerr<<ex.getError()<<endl;

}

catch(const char\* errorString)

{

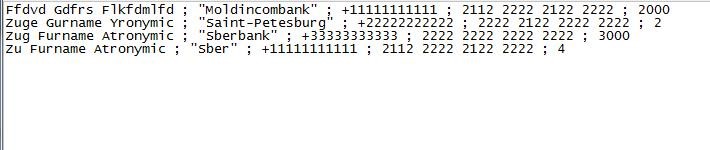
cerr<<errorString<<endl;

}

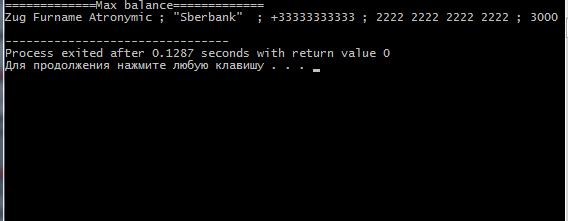
}

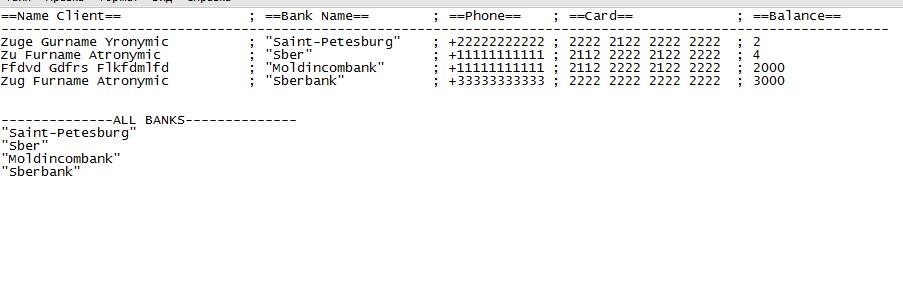
# Приложение 2

Входной файл:

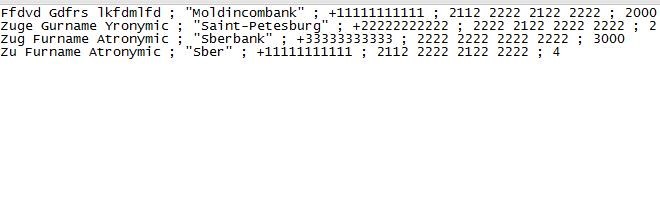


Результат:

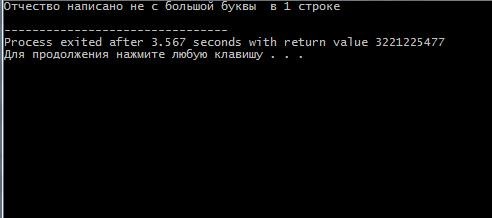




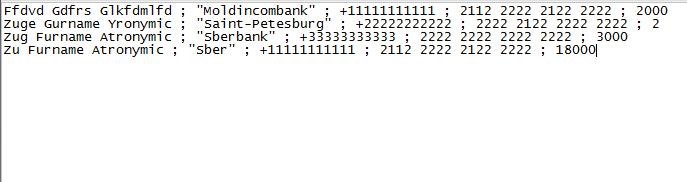
Входной файл:



Вывод:



Входной файл:



Вывод:

