**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ…………………………………………………………..…………………………..5

1 ПОСТАНОВКА ЗАДАЧИ…………………………………………………………..6

2 СТРУКТУРА ВХОДНЫХ И ВЫХОДНЫХ ДАННЫХ……………….……………….…....7

3 ОПИСАНИЕ КЛАССОВ ……………………….……………………………………………...8

4 РАЗРАБОТКА АЛГОРИТМОВ…...…………………….…………………………………...19

4.1 Схемы алгоритмов……………………………………….……………………………..…...19

4.2 Алгоритмы по шагам………………………………………………………..……19

5 КОД ПРОГРАММЫ ………………………………………………………………...………..20

6 ДИАГРАММА КЛАССОВ …………………………………………………………...……...20

7 РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ …...………………………..……………………20

8 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ…………...………………………………….………...20

ЗАКЛЮЧЕНИЕ………………...……………...…………………………………….…..……...21

СПИСОК ЛИТЕРАТУРЫ………………………...….……………………….……...22

ПРИЛОЖЕНИЕ A (структурная схема)………………………….………………….…….......23

ПРИЛОЖЕНИЕ Б (схема алгоритма)…………………….…………………………...…….....24

ПРИЛОЖЕНИЕ В (схема алгоритма)……………..…………………..…..….…...…..25

ПРИЛОЖЕНИЕ Г (код программы)………………..……………..…………...….……...26

ПРИЛОЖЕНИЕ Д (диаграмма классов)……………….……….…………...….……...27

ПРИЛОЖЕНИЕ Е (скриншоты работы программы)…………….…………...….……...28

**ВВЕДЕНИЕ**

C++ является компилируемым строго типизированным языком программирования общего назначения. С++ поддерживает различные парадигмы программирования: процедурную, обобщенную и функциональную. Несмотря на то, что первый выпуск С++ для коммерческого использования состоялся в 1985 году, он остается актуальным и востребованным по сей день. Данный язык программирования унаследовал синтаксис от языка программирования С. Целью создания языка С++ было дополнение языка С возможностями, удобными для разработки ПО в промышленных масштабах с сохранением гибкости, скорости и портабельности С.

С++ имеет три основных концепции объектно-ориентированного программирования: инкапсуляция, наследование и полиморфизм. Инкапсуляция – это размещение в одном компоненте данных и методов, которые с ним работают. Также это означает скрытие внутренней реализации от других компонентов. Наследование – это концепция ООП, согласно которой абстрактный тип данных может наследовать данные и функциональность некоторого существующего типа, способствуя повторному использованию компонентов программного обеспечения. Полиморфизм дает возможность объектам с одинаковой спецификацией иметь различную реализацию.

В современном мире С++ широко используется для разработки компьютерных игр, прикладных программ, операционных систем, приложений для встраиваемых систем, драйверов устройств. Исходя из вышеупомянутого, можно сделать вывод, что С++ является идеальным языком программирования для написания данной курсовой работы.

**1 ПОСТАНОВКА ЗАДАЧИ**

Программа должна иметь удобный пользовательский интерфейс с необходимыми пунктами меню. В программе должна быть предусмотрена возможность создания информации в виде файлов, связанных определенным образом. Программа позволяет организовать открытие нового счета, аннулирование счета, перевод денег с одного счета на другой, снятие денег со счета, депозиты и займ под проценты. Банковский счет связывается с именем вкладчика. При реализации операции редактирования, добавления, удаления информации, необходимо предусмотреть операцию отмены последних действий.

Разработать иерархию классов с использованием наследования. Разработать и использовать в программе классы контейнеров и итераторов (свои и STL). Производить обработку исключительных ситуаций.

Для реализации программы используется объектно-ориентированный язык программирования C++, среда разработки Microsoft Visual Studio 2013. Ограничений на использование других операционных систем нет. Приложение написано на ОС Windows 10.

**2 СТРУКТУРА ВХОДНЫХ И ВЫХОДНЫХ ДАННЫХ**

В программе используется 3 текстовых файла:

- Файл account.txt. Хранит данные о пользователях системы (имя пользователя, баланс):

|  |
| --- |
| username 1000  -1 |

- Файл deposit.txt. Хранит данные депозита (имя пользователя, счет депозита, депозитный процент):

|  |
| --- |
| username 1000 5  -1 |

- Файл loan.txt. Хранит данные о займах (сумма взятого займа, процент займа, имя пользователя у которого был взят займ):

|  |
| --- |
| 800 7 name  -1 |

Также в ходе работы программы для хранения промежуточных данных используются STL-контейнеры и контейнер List, написанный вручную. При изменении данных в программе, происходит изменение в файле после закрытия программы.

**3 ОПИСАНИЕ КЛАССОВ**

**3.1 Класс Account – родительский класс счета пользователя**

|  |  |  |  |
| --- | --- | --- | --- |
| Protected поля класса | | | |
| Имя | Тип | | Описание |
| username | string | | Имя пользователя |
| balance | double | | Баланс пользователя |
| Public методы класса | | | |
| Имя | Принимаемые параметры | Возвращаемое значение | Описание |
| getUsername | void | const string& | Возвращает значение username |
| getBalance | void | double | Возвращает значение balance |
| setUsername | const string &tmp | void | Устанавливает значение tmp |
| setBalance | double \_balance | void | Устанавливает значение balance |
| operator >> | istream& is, Account& tmp | istream | Инициализирует объект класса Account |
| operator << | ostream& os, const Account & tmp | ostream | Выводит на экран объект класса Account |

**3.2 Класс Deposit – предназначен для работы с депозитами**

|  |  |  |  |
| --- | --- | --- | --- |
| Private поля класса | | | |
| Имя | Тип | | Описание |
| percent | int | | Процентная ставка депозита |
| Public методы класса | | | |
| Имя | Принимаемые параметры | Возвращаемое значение | Описание |
| setPercent | int percent | void | Устанавливает значение percent |
| getPercent | void | const int | Возвращает значение percent |
| updateBalance | Deposit\* deposit | void | Обновляет баланс каждые 60 секунд |
| operator >> | istream& is, Deposit& tmp | istream | Инициализирует объект класса Deposit |
| operator << | ostream& os, const Deposit & tmp | ostream | Выводит на экран объект класса Deposit |

**3.3 Класс MoneyTransfer – предназначен для перевода денег на другие счета**

|  |  |  |  |
| --- | --- | --- | --- |
| Protected поля класса | | | |
| Имя | Тип | | Описание |
| partnerUsername | string | | Имя аккаунта на который переводим деньги |
| Public методы класса | | | |
| 1 | 2 | 3 | 4 |

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Имя | Принимаемые параметры | Возвращаемое значение | Описание |
| setPartnerUsername | const string & partnerUsername | void | Устанавливает значение partnerUsername |
| getPartnerUsername | void | const string& | Возвращает значение partnerUsername |
| operator >> | istream& is,  MoneyTransfer & tmp | istream | Инициализирует объект класса MoneyTransfer |
| operator << | ostream& os, const  MoneyTransfer & tmp | ostream | Выводит на экран объект класса MoneyTransfer |
| operator == | MoneyTransfer tmp1, MoneyTransfer tmp2 | bool | Перегрузка оператора сравнения |
| operator = | const MoneyTransfer &tmp | void | Перегрузка оператора присваивания |

**3.4 Класс Loan – предназначен для работы с займами**

|  |  |  |  |
| --- | --- | --- | --- |
| Private поля класса | | | |
| Имя | Тип | | Описание |
| partnerUsername | string | | Имя аккаунта у которого берем займ |
| Public методы класса | | | |
| 1 | 2 | 3 | 4 |

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Имя | Принимаемые параметры | Возвращаемое значение | Описание |
| getPartnerUsername | void | const string | Возвращает значение partnerUsername |
| setPartnerUsername | string tmp | void | Устанавливает значение partnerUsername |
| operator >> | istream& is,  Loan& tmp | istream | Инициализирует объект класса Loan |
| operator << | ostream& os,  const Loan & tmp | ostream | Выводит на экран объект класса Loan |

**3.5 Базовый класс Exception – предназначен для обработки базовых исключений**

|  |  |  |  |
| --- | --- | --- | --- |
| Protected поля класса | | | |
| Имя | Тип | | Описание |
| error | int | | Код ошибки |
| Public методы класса | | | |
| Имя | Принимаемые параметры | Возвращаемое значение | Описание |
| ErrorText | void | virtual void | Выводит сообщение об ошибке |

**3.6 Класс InputException – предназначен для обработки исключений ввода**

|  |  |  |  |
| --- | --- | --- | --- |
| Public методы класса | | | |
| 1 | 2 | 3 | 4 |

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Имя | Принимаемые параметры | Возвращаемое значение | Описание |
| ErrorText | void | void | Выводит ошибку |

**3.7 Класс ListException – предназначен для обработки исключений контейнера**

|  |  |  |  |
| --- | --- | --- | --- |
| Public методы класса | | | |
| Имя | Принимаемые параметры | Возвращаемое значение | Описание |
| ErrorText | void | void | Выводит сообщение об ошибке в контейнере |

**3.8 Шаблонная структура Node – предназначена для работы с шаблонным классом-контейнером List**

|  |  |  |
| --- | --- | --- |
| Public поля класса | | |
| Имя | Тип | Описание |
| data | T | Хранит узел списка |
| next | Node<T>\* | Указывает на следующий элемент списка |
| prev | Node<T>\* | Указывает на предыдущий элемент списка |

**3.9 Шаблонный класс-контейнер List – предназначен для хранения данных различного типа**

|  |  |  |  |
| --- | --- | --- | --- |
| Protected поля класса | | | |
| Имя | Тип | | Описание |
| 1 | 2 | | 3 |
| 1 | 2 | | 3 |
| Iterator | friend class | | Используется для работы класса-контейнера List |
| head | Node<T>\* | | Указывает на начало списка |
| tail | Node<T>\* | | Указывает на конец списка |
| amount | long | | Количество элементов в списке |
| Public методы класса | | | |
| Имя | Принимаемые параметры | Возвращаемое значение | Описание |
| size | void | long | Возвращает размер списка |
| push\_head | T input\_object | void | Добавляет элемент в начало списка |
| push\_tail | T input\_object | void | Добавляет элемент в конец списка |
| pop\_head | void | T | Удаляет первый элемент из списка |
| pop\_tail | void | T | Удаляет последний элемент из списка |
| pop | int num | T | Удаляет текущий элемент из списка |
| 1 | 2 | 3 | 4 |

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| operator[] | long num | T& | Перегруженный оператор индексации |
| begin | void | Node<T>\* | Возвращает указатель на начало списка. Используется при работе с классом Iterator |
| end | void | Node<T>\* | Возвращает указатель на конец списка. Используется при работе с классом Iterator |
| output | void | void | Вывод данных об аккаунте из списка на экран |
| output\_loans | void | void | Вывод данных о займах из списка на экран |

**3.10 Шаблонный класс Iterator – предназначен для работы с классом-контейнером List**

|  |  |  |  |
| --- | --- | --- | --- |
| Private поля класса | | | |
| Имя | Тип | | Описание |
| current | Node<T>\* | | Указатель на текущий элемент итератора |
| Public методы класса | | | |
| Имя | Принимаемые параметры | Возвращаемое значение | Описание |
| 1 | 2 | 3 | 4 |

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| get\_node | void | Node<T>\* | Возвращает указатель на текущий элемент итератора |
| operator++ | int i | bool | Сдвигает итератор вниз на одно значение |
| operator-- | int i | bool | Сдвигает итератор вверх на одно значение |
| operator== | Iterator& iter | bool | Сравнивает  элементы двух объектов класса List<T> |
| operator != | Iterator& iter | bool | Сравнивает  элементы двух объектов класса List<T> |
| operator\*() | void | T& | Перегруженный оператор разадресации |

**3.11 Шаблонный класс File – предназначен для работы с файлами**

|  |  |  |  |
| --- | --- | --- | --- |
| Public методы класса | | | |
| Имя | Принимаемые параметры | Возвращаемое значение | Описание |
| FileOpenAccount | List<F>& tmp,  const char\* file | static void | Открывает файл “account.txt” и считывает из него информацию в список List<Account> list |
| 1 | 2 | 3 | 4 |

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| FileOpenDeposit | List<F>& tmp,  const char\* file | static void | Открывает файл “deposit.txt” и считывает из него информацию в список List<Deposit> list |
| FileOpenLoan | List<F>& tmp,  const char\* file | static void | Открывает файл “loan.txt” и считывает из него информацию в список List<Loan> list |
| FileOpenMoneyTransfer | List<F>& tmp,  const char\* file | static void | Открывает файл текущего аккаунта и считывает из него информацию в список List<MoneyTransfer> list |
| FileSaveAccount | List<F>& tmp,  const char\* file | static void | Открывает файл “account.txt” и записывает в него информацию из списка  List<Account> list |
| FileSaveDeposit | List<F>& tmp,  const char\* file | static void | Открывает файл “deposit.txt” и записывает в него информацию из списка  List<Deposit> list |
| FileSaveLoan | List<F>& tmp,  const char\* file | static void | Открывает файл “loan.txt” и записывает в него информацию из списка  List<Loan> list |
| FileSaveMoneyTransfer | List<F>& tmp,  const char\* file,  string username | static void | Открывает файл текущего аккаунта и записывает в него информацию из списка  List< MoneyTransfer > list |

**3.12 Шаблонный класс Interface – набор всех возможных манипуляций с аккаунтом пользователя**

|  |  |  |  |
| --- | --- | --- | --- |
| Public методы класса | | | |
| 1 | 2 | 3 | 4 |
| Имя | Принимаемые параметры | Возвращаемое значение | Описание |
| outputTransfer | T& MainUser | static void | Используется для вывода переводов на экран |
| showAccounts | List<Account>& list | static void | Используется для вывода всех аккаунтов на экран |
| transferTo | T& MainUser, List<T>& list | static void | Используется для операции перевода денег на другой аккаунт |
| cancelTransfer | T& MainUser, List<T>& list | static void | Используется для операции отмены перевода денег |
| getDeposit | List<T>&list, Account& MainUser | Deposit | Используется для получения доступа к депозиту |
| getLoans | List<T>&list, List<T>&loanList, Account& MainUser | static void | Используется для получения доступа к займу |
| putDeposit | Deposit& MainDeposit,  List< Deposit >& listD,  T& MainUser, List<T>& list | static void | Используется для пополнения средств на депозите |
| outDeposit | Deposit& MainDeposit,  List< Deposit >& listD,  T& MainUser, List<T>& list | static void | Используется для вывода средств с депозита |
| 1 | 2 | 3 | 4 |

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| payLoan | Account& user,  List< Account >& bankBookList,  List<Loan>& loanList,  List<Loan>& loans | static void | Используется для погашения задолженности по займу |
| deleteAccount | Account& user,  List< Account >& bankBookList,  List< Deposit >& depositList,  List<Loan>& loanList | static void | Используется для  удаления аккаунта |
| menu | Account& user,  List< Deposit >& depositList,  List<Loan>& loanList,  List< Account >& bankBookList | static void | Меню работы с  программой банковской системы |
| createLoan | Deposit& MainDeposit,  Account& user,  List< Account >& bankBookList,  List<Loan>& loanList,  List<Loan>& loans | static void | Используется для взятия займа под проценты |

**4 РАЗРАБОТКА АЛГОРИТМОВ**

**4.1 Схема Алгоритмов**

4.1.1 Алгоритм функции **push\_head(T input\_object)**.

Позволяет добавлять элементы в шаблонный класс список с головы. Представлен в Приложении А.

4.1.2 Алгоритм функции **FileSaveDeposit(List <F> &tmp, const char \*file)**. Записывает в файл данные о депозитах. Представлен в Приложении Б.

**4.2 Алгоритмы по шагам**

4.2.1 Для алгоритма по шагам рассмотрим метод

**FileSaveAccount(List <F> &tmp, const char \*file)** шаблонного класса File:

Шаг 1: Начало.

Шаг 2: Открываем текстовый файл для записи данных. Если файл

не открылся, то выводим сообщение “Невозможно открыть файл!” и переходим к шагу 5.

Шаг 3: Контейнер List записывает в файл данные.

Шаг 4: Записываем в файл “-1”, что означает условное окончание записи.

Шаг 5: Конец.

4.2.2 Для алгоритма по шагам рассмотрим метод **pop\_head()**

шаблонного класса-контейнера List:

Шаг 1: Начало.

Шаг 2: Если указатель head на начало списка List равен nullptr, то возвращается пустой шаблонный объект класса T, происходит переход к шагу 8.

Шаг 3: Создаём локальный объект data, который принимает значение

head->data и указатель node на начало списка – head;

Шаг 4: Если имеется head, то nullptr присваивается head->prev;

Шаг 5: Очищаем объём динамической памяти, на который указывает node;

Шаг 6: Уменьшаем параметр списка amount на одно значение;

Шаг 7: Возвращаем data;

Шаг 8: Конец.

**5 КОД ПРОГРАММЫ**

Предоставляется в Приложении Г.

**6 ДИАГРАММА КЛАССОВ**

Предоставляется в Приложении Д.

**7 РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ**

Предоставляются в Приложении Е.

**8 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

1. Для работы с программой необходимо запустить файл “BankSystem.exe”;

2. После запуска программы авторизуйтесь;

3. В меню программы выберите действие, которое необходимо совершить в вашем аккаунте;

4. По окончании работы нажмите 0;

5. Программа успешно сохраняет результаты работы в файлах “account.txt”, ”deposit.txt” и ”loan.txt”, расположенных в папке с программой.

**ЗАКЛЮЧЕНИЕ**

В результате выполнения курсового проекта, была разработана программа реализации банковских услуг, позволяющая пользователю вводить различную информацию, выводить ее на экран, изменять, удалять, добавлять, а также, в случае допущения ошибки пользователем, отменять последние действия. Использовалась среда разработки Visual Studio 2013 и операционная система Windows 10. К достоинствам программы можно отнести простой интерфейс, что в свою очередь обеспечивает удобство использования программы обычными пользователями. В дальнейшем планируется усовершенствование программы, а именно, усовершенствование интерфейса и добавление новых функций. После освоения в ходе учебного курса основ и принципов работы с базами данных планируется переход с системы хранения информации в текстовых файлах на хранение в базе данных, что предоставит возможность добавления информации нескольким пользователям одновременно и позволит увеличить скорость работы программы.

Системные требования:

-Операционная система Windows 98, XP, Vista, 7, 8, 10;

-Процессор Intel Pentium III 1 GHz;

-Оперативная память 512 МВ;

-Свободное место на жестком диске: 2 Мб;

# **СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

[1] Герберт, Ш. Самоучитель C++/Ш. Герберт. Санкт-Петербург, 2003г. – 678 с.

[2] Страуструп, Б. Программирование. Принципы и практика использования C++/ Б.Страуструп, Вильямс, 2018 г. – 991 с.

[3] Ю. А. Луцик. Объектно-ориентированное программирование на языке С++: учеб. пособие/ Ю. А. Луцик, В. Н. Комличенко. – Минск: БГУИР, 2008. –

266 с.

[4] Д. Харт. Системное программирование в среде Windows/Д. Харт. Издательство “Питер”, 2012 г. – 251 с.

[5] Х. М. Дейтел, П. Дж. Дейтел. Как программировать на С++: Пятое издание. Пер. с англ. – М.: ООО “Бином-Пресс”, 2008 г. – 1456 с.

**ПРИЛОЖЕНИЕ А**

*(обязательное)*

Схема структурная

**ПРИЛОЖЕНИЕ Б**

*(обязательное)*

Схема алгоритма

**ПРИЛОЖЕНИЕ В**

*(обязательное)*

Схема алгоритма

**ПРИЛОЖЕНИЕ Г**

*(обязательное)*

Код программы

**ПРИЛОЖЕНИЕ Д**

*(обязательное)*

Диаграмма классов

**ПРИЛОЖЕНИЕ E**

*(обязательное)*

Результаты работы программы

**ПРИЛОЖЕНИЕ Ж**

*(обязательное)*

Ведомость документов