**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc153449688)

[1 ПОСТАНОВКА ЗАДАЧИ 6](#_Toc153449689)

[2 ОБЗОР ЛИТЕРАТУРЫ 7](#_Toc153449690)

[2.1 Обзор методов и алгоритмов решения поставленной задачи 8](#_Toc153449691)

[3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИОВАНИЕ. 10](#_Toc153449692)

[3.1 Структура входных и выходных данных. 10](#_Toc153449693)

[3.2 Разработка диаграммы классов. 11](#_Toc153449694)

[3.3 Описание классов. 11](#_Toc153449695)

[3.3.1 Class Item 11](#_Toc153449696)

[3.3.2 Class Electronics 12](#_Toc153449697)

[3.3.3 Class Clothing 13](#_Toc153449698)

[3.3.4 Class Computer 13](#_Toc153449699)

[3.3.5 Class Telephone 15](#_Toc153449700)

[3.3.6 Class Footwear 16](#_Toc153449701)

[3.3.7 Class Accessories 17](#_Toc153449702)

[3.3.8 Class Menu<T> 18](#_Toc153449703)

[3.3.9 MyFile<T> 19](#_Toc153449704)

[3.3.10 Class MyException 19](#_Toc153449705)

[3.3.11 Class IntException 19](#_Toc153449706)

[3.3.12 Class StringException 19](#_Toc153449707)

[3.3.13 Class DoubleException 20](#_Toc153449708)

[3.3.14 Class FileError 20](#_Toc153449709)

[3.3.15 Class StackList<T1, T2, T3, T4> 20](#_Toc153449710)

[3.3.16 Class StackListIterator< T1, T2, T3, T4> 21](#_Toc153449711)

[4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ 22](#_Toc153449712)

[4.1 Разработка алгоритмов 22](#_Toc153449713)

[5 РЕЗУЛЬТАТЫ РАБОТЫ 24](#_Toc153449714)

[ЗАКЛЮЧЕНИЕ 29](#_Toc153449715)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 30](#_Toc153449716)

[ПРИЛОЖЕНИЕ А СТРУКТУРНАЯ СХЕМА 31](#_Toc153449717)

[ПРИЛОЖЕНИЕ Б СХЕМА ЛГОРИТМА 32](#_Toc153449718)

[ПРИЛОЖЕНИЕ В ДИАГРАММА КЛАССОВ 33](#_Toc153449719)

[ПРИЛОЖЕНИЕ Г КОД ПРОГРАММЫ 34](#_Toc153449720)

[ПРИЛОЖЕНИЕ Д ВЕДОМОСТЬ ДОКУМЕНТОВ 35](#_Toc153449721)

# **ВВЕДЕНИЕ**

Первоначально разработанный Бьярне Страуструпом в 1979 году, язык C++ был задуман как преемник языка С. С — процедурный язык, где каждая функция предпринимает определенное действие. Язык C++, напротив, был задуман как объектно-ориентированный, но он реализует такие концепции, как наследование, абстракция, полиморфизм и инкапсуляция. Классы языка C++ используют свойства для содержания данных и методы для обработки этих данных. (Методы являются аналогом функций в языке С.) В результате программист больше думает о данных и о том, что с ними следует сделать. Компиляторы C++ традиционно поддержали также программы на языке С. Преимуществом этого была совместимость с устаревшим кодом, а недостатком — чрезвычайно высокая сложность компиляторов, вынужденных обеспечивать программистам эту совместимость, одновременно реализуя все новые средства, которых требует развитие языка. Язык C++ — это ключ к современному объектно-ориентированному программированию. Он создан для разработки высокопроизводительного программного обеспечения и чрезвычайно популярен среди программистов.

Современные возможности разработки прикладного программного обеспечения с использованием языка высокого уровня C++ предоставляют программистам мощные инструменты для создания удобных и эффективных приложений. В рамках данной курсовой работы была поставлена задача разработки программы, предоставляющей возможность учета личных вещей, таких техника, спортивный инвентарь и одежда.

# **1 ПОСТАНОВКА ЗАДАЧИ**

Целью работы является создание программы с интуитивно понятным пользовательским интерфейсом, оснащенным необходимыми пунктами меню, которая позволит удобно вести учет и управлять информацией о личных вещах. Для достижения этой цели будет использоваться язык программирования C++ и разработана иерархия классов с использованием наследования, включая не менее трех уровней наследования.

Задачей является написать информационную систему «Учет личных вещей». Данная сиcтема содержит программное обеспечение для персонального компьютера.

# **2 ОБЗОР ЛИТЕРАТУРЫ**

Ниже представлены основные термины в программировании, так как без понимания их невозможно будет освоить ни один из языков программирования, а также проектирование программного обеспечения:

Программирование – процесс искусства создания компьютерных программ с использованием языков программирования. Программирование рассматривается как написание кода для реализации алгоритмов на определенном языке программирования. В широком смысле программирование – это процесс создания программного обеспечения, включая разработку программ.

Программа – набор данных, предназначенных для управления компонентами системы обработки информации с целью выполнения определенного алгоритма.

Программное обеспечение – совокупность программ, необходимых для работы системы обработки информации, включая программные документы, необходимые для использования этих программ.

Разработка программного обеспечения – это процесс проектирования, написания, тестирования и поддержки компьютерных программ для решения задач множества пользователей. Она включает создание надежных и безопасных решений, которые могут справиться с неизвестными задачами и прослужить долгое время.

Программный модуль – программа или функционально завершенная часть программы, предназначенная для хранения, трансляции и объединения с другими модулями для загрузки в оперативную память.

Объектно-ориентированное программирование – методология построения программных систем в виде совокупности взаимодействующих объектов и классов объектов.

Компиляция – процесс трансляции программы с языка высокого уровня в форму, близкую к машинному языку.

Абстракция – процесс выделения существенных признаков и отвлечения от конкретных деталей предметов или явлений.

Диаграмма классов – структурная диаграмма, используемая в языке моделирования UML, которая показывает структуру иерархии классов системы, их взаимодействие, атрибуты, методы, интерфейсы и связи между ними. Цель создания диаграммы классов – визуальное представление статической структуры элементов системы (классов, типов и т.д.). Диаграмма также может содержать некоторые элементы поведения, но их динамика отображается на других типах диаграмм. Дополнительно, для облегчения восприятия, диаграмму классов можно дополнить представлением пакетов, включая вложенные структуры.

Понимание этих терминов является важным для успешного изучения языков программирования и проектирования программного обеспечения.

## **2.1 Обзор методов и алгоритмов решения поставленной задачи**

Необходимо написать программу, которая будет иметь несколько разделов с разными классами личных вещей. Каждый раздел будет иметь собственный файл для хранения информации. В программе вход с правами пользователя. Пользователь может добавлять, изменять, просматривать, искать, удалять информацию.

Для реализации данной программы буду использовать контейнер List. Так буду осуществлять связь между объектами одного класса личных вещей. Для работы с классами буду использовать шаблонный класс. Для реализации отмены действия буду реализовывать стек.

Метод добавления позволяет вводить новые объекты класса, динамически выделяя для них память. Метод просмотра будет демонстрировать список личных вещей. Метод изменения состояние позволит изменять параметры личных вещей (цена, модель, название). Можно будет удалять элементы List. Можно использовать поиск для нахождения по цене или имени.

Блок хранения данных будет состоять из 4-ёх файлов, с которыми работает программа. В файлах храниться информация об личных вещах.

# **3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИОВАНИЕ.**

В данном разделе пояснительной записки детально рассмотрим функционирование программы.

## **3.1 Структура входных и выходных данных.**

Таблица 2.1 - файл бензиновый автомобилей "Computer's.txt"

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название | Цена | Брэд | Модель | Операционная  система | Память |

Таблица 2.2 - файл телефонов "Telephones.txt"

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название | Цена | Брэнд | Модель | Сеть | Размер  экрана |

Таблица 2.3 - файл обуви "Footwears.txt"

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название | Цена | Размер | Материал | Тип | Цвет |

Таблица 2.4 – файл аксессуаров "Accessories.txt"

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название | Цена | Размер | Материал | Стиль | Брэнд |

## **3.2 Разработка диаграммы классов.**

Диаграмма классов данной работы показана в приложении В.

## **3.3 Описание классов.**

### **3.3.1 Class Item**

Программа представляет собой 2 файла, один заголовочный (Item.h), один основной (Item.cpp).

В protected секции класса используются типы данных string (название, категория) и тип данных double(стоимость), Item.h является базовым классом.

В public секции класса Item используются следующие методы:

- метод string getName() позволяет получить название личной вещи;

- метод string getCategory() позволяет получить категорию личной вещи;

- метод double getValue() позволяет получить стоимость личной вещи;

- метод void setName(string name) позволяет изменять состояние имени;

- метод void setCategory(string category) позволяет изменять состояние категории;

- метод void setValue(double value) позволяет изменять состояние стоимости;

- перегрузка оператора потокового ввода friend istream& operator>>(istream& input, Item& obj);

- перегрузка оператора потокового вывода friend ostream& operator<<(ostream& os, Item obj);

- перегрузка оператора чтения из файла friend ifstream& operator>>(ifstream& input, Item& obj)

- перегрузка оператора записи в файл friend ofstream& operator<<(ofstream& os, const Item& obj)

### **3.3.2 Class Electronics**

Программа представляет собой 2 файла, один заголовочный (Electronics.h), один основной (Electronics.cpp).

В protected секции класса используется тип данных string (брэнд, модель).

В public секции класса Electronics используются следующие методы:

- метод string getBrand() позволяет получить брэнд личной вещи;

- метод string getModel() позволяет получить модель личной вещи;

- метод void setBrand(string brand) позволяет изменять состояние брэнда;

- метод void setModel(string model) позволяет изменять состояние модели;

- перегрузка оператора потокового ввода friend istream& operator>>(istream& input, Electronics& obj);

- перегрузка оператора потокового вывода friend ostream& operator<<(ostream& os, Electronics obj);

- перегрузка оператора чтения из файла friend ifstream& operator>>(ifstream& input, Electronics& obj);

- перегрузка оператора записи в файл friend ofstream& operator<<(ofstream& os, const Electronics& obj);

### **3.3.3 Class Clothing**

Программа представляет собой 2 файла, один заголовочный (Clothing.h), один основной (Clothing.cpp).

В protected секции класса используется тип данных string (размер, материал).

В public секции класса Сlothing используются следующие методы:

- метод string getSize() позволяет получить размер личной вещи;

- метод string getMaterial() позволяет получить материал личной вещи;

- метод void setSize(string size) позволяет изменять состояние размера;

- метод void setMaterial(string material) позволяет изменять состояние материала;

- перегрузка оператора потокового ввода friend istream& operator>>(istream& input, Clothing& obj);

- перегрузка оператора потокового вывода friend ostream& operator<<(ostream& os, Clothing obj);

- перегрузка оператора чтения из файла friend ifstream& operator>>(ifstream& input, Clothing& obj);

- перегрузка оператора записи в файл friend ofstream& operator<<(ofstream& os, const Clothing& obj);

### **3.3.4 Class Computer**

Программа представляет собой 2 файла, один заголовочный (Computer.h), один основной (Computer.cpp). Сама программа использует пользовательский тип данных Computer из файла Computer.h.

В protected секции класса используются типы данных string (операционная система) и int(память).

В public секции класса Сomputer используются следующие методы:

- метод string getOperatingSystem() позволяет получить операционную систему личной вещи;

- метод int getMemory() позволяет получить информацию о памяти личной вещи;

- метод void setOperatingSystem(string operatingSystem) позволяет изменять состояние операционной системы;

- метод void setMemory(int memory) позволяет изменять состояние памяти;

- метод void Shapka() позволяет выводить шапку над характеристиками при выводе информации;

- метод void ChangeField() позволяет изменить состояния полей класса;

- метод string GetFileName() позволяет получить имя файла в котором храниться информация о классе;

- перегрузка оператора потокового ввода friend istream& operator>>(istream& input, Computer& obj);

- перегрузка оператора потокового вывода friend ostream& operator<<(ostream& os, Computer obj);

- перегрузка оператора чтения из файла friend ifstream& operator>>(ifstream& input, Computer& obj);

- перегрузка оператора записи в файл friend ofstream& operator<<(ofstream& os, const Computer& obj);

### **3.3.5 Class Telephone**

Программа представляет собой 2 файла, один заголовочный (Telephone.h), один основной (Telephone.cpp). Сама программа использует пользовательский тип данных Telephone из файла Telephone.h.

В protected секции класса используются типы данных string (мобильный оператор) и double(размер экрана).

В public секции класса Telephone используются следующие методы:

- метод string getNetwork() позволяет получить информацию о мобильном операторе личной вещи;

- метод double getScreenSize() позволяет получить информацию о размере экрана;

- метод void setNetwork(string network)позволяет изменять состояние мобильного оператора;

- метод void setScreenSize(double screenSize) позволяет изменять состояние размера экрана;

- метод void Shapka() позволяет выводить шапку над характеристиками при выводе информации;

- метод void ChangeField() позволяет изменить состояния полей класса;

- метод string GetFileName() позволяет получить имя файла в котором храниться информация о классе;

- перегрузка оператора потокового ввода friend istream& operator>>(istream& input, Telephone& obj);

- перегрузка оператора потокового вывода friend ostream& operator<<(ostream& os, Telephone obj);

- перегрузка оператора чтения из файла friend ifstream& operator>>(ifstream& input, Telephone& obj);

- перегрузка оператора записи в файл friend ofstream& operator<<(ofstream& os, const Telephone& obj);

### **3.3.6 Class Footwear**

Программа представляет собой 2 файла, один заголовочный (Footwear.h), один основной (Footwear.cpp). Сама программа использует пользовательский тип данных Footwear из файла Footwear.h.

В protected секции класса используется тип данных string (тип, цвет) .

В public секции класса Footwear используются следующие методы:

- метод string getType() позволяет получить тип личной вещи;

- метод string getColor() позволяет получить информацию о цвете личной вещи;

- метод void setType(string type)позволяет изменять состояние типа;

- метод void setColor(string color) позволяет изменять состояние цвета;

- метод void Shapka() позволяет выводить шапку над характеристиками при выводе информации;

- метод void ChangeField() позволяет изменить состояния полей класса;

- метод string GetFileName() позволяет получить имя файла в котором храниться информация о классе;

- перегрузка оператора потокового ввода friend istream& operator>>(istream& input, Footwear & obj);

- перегрузка оператора потокового вывода friend ostream& operator<<(ostream& os, Footwear obj);

- перегрузка оператора чтения из файла friend ifstream& operator>>(ifstream& input, Footwear& obj);

- перегрузка оператора записи в файл friend ofstream& operator<<(ofstream& os, const Footwear& obj);

### **3.3.7 Class Accessories**

Программа представляет собой 2 файла, один заголовочный (Accessories.h), один основной (Accessories.cpp). Сама программа использует пользовательский тип данных Accessories из файла Accessories.h.

В protected секции класса используется тип данных string (стиль, брэнд) .

В public секции класса Accessories используются следующие методы:

- метод string getStyle() позволяет получить стиль личной вещи;

- метод string getBrand() позволяет получить информацию о бренде личной вещи;

- метод void setBrand(string brand) позволяет изменять состояние бренда;

- метод void setStyle(string style) позволяет изменять состояние стиля;

- метод void Shapka() позволяет выводить шапку над характеристиками при выводе информации;

- метод void ChangeField() позволяет изменить состояния полей класса;

- метод string GetFileName() позволяет получить имя файла в котором храниться информация о классе;

- перегрузка оператора потокового ввода friend istream& operator>>(istream& input, Accessories& obj);

- перегрузка оператора потокового вывода friend ostream& operator<<(ostream& os, Accessories obj);

- перегрузка оператора чтения из файла friend ifstream& operator>>(ifstream& input, Accessories& obj);

- перегрузка оператора записи в файл friend ofstream& operator<<(ofstream& os, const Accessories& obj);

### **3.3.8 Class Menu<T>**

Программа представляет собой 2 файла, один заголовочный (Menu.h), один основной (Menu.cpp).

В private секции класса используются типы данных list<T> и MyFile<T>. В private секции класса используются следующие методы:

- метод void AddItem() добавляет объект;

- метод void DeleteItem() удаляет объект;

- метод void ChangeFields() позволяет изменят объекты ;

- метод void PrintList() выводит объект ;

- метод void ClearList() очищает list ;

- метод void PrintMenu() используется для вызова меню для работы с list;

В public секции класса используются следующие методы:

- метод void Run(bool& write\_to\_stack) используется для работы с list;

- метод void WriteFromFile() используется для записи данных из стека в list;

- метод void RewriteFile()используется для перезаписи файла;

**3.3.9 MyFile<T>**

В private секции класса используется тип данных string(название файла).

В public секции класса используются следующие методы:

- метод void write(T obj) добавляет объект в файл;

- метод void read(list<T>& MyList) извлекает объекты из файла;

- метод void DeleteFale()удаляет файл ;

### **3.3.10 Class MyException**

В private секции класса используется тип данных int (номер ошибки) и string(вид/название ошибки).

Используемый методы:

- метод string What() метод, выводящий на экран ошибку и её номер.

### **3.3.11 Class IntException**

Используемый метод:

- метод string WhatINT() метод, выводящий на экран ошибку и её номер.

### **3.3.12 Class StringException**

Используемый метод:

- метод string WhatSTRING() метод, выводящий на экран ошибку и её номер.

### **3.3.13 Class DoubleException**

Используемый метод:

- метод string WhatDOUBLE() метод, выводящий на экран ошибку и её номер.

### **3.3.14 Class** **FileError**

Используемый метод:

- метод string WhatFILE() метод, выводящий на экран ошибку и её номер:

### **3.3.15 Class StackList<T1, T2, T3, T4>**

Программа представляет собой 2 файла, один заголовочный (Stack.h), один основной (Stack.cpp).

В private секции класса используются типы данных NodeStack<T1, T2, T3, T4>\*.

В public секции класса используются следующие методы:

- метод void Push(T1 item1, T2 item2, T3 item3, T4 item4); добавляет объект;

- метод void tuple<T1, T2, T3, T4> Pop()удаляет объект;

- метод void ChangeFields() позволяет изменят объекты ;

- метод int Cout() позволяе получить количество объектов ;

- метод void Empty()очищает стек ;

- метод void StackListIterator<T1, T2, T3, T4> begin() const используется для получения ссылки на начало стека;

- метод void StackListIterator<T1, T2, T3, T4> end() const используется для получения ссылки на конец стека;

### **3.3.16 Class StackListIterator< T1, T2, T3, T4>**

В private секции класса используются типы данных NodeStack<T1, T2, T3, T4>\*.

В public секции класса используются следующие методы:

- перегрузка оператора ++ StackListIterator& operator++();

- перегрузка оператора != bool operator!=(const StackListIterator& other) const;

- перегрузка оператора tuple<T1, T2, T3, T4> operator\*() const;

# **4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ**

В данном разделе курсового проекта рассмотрены алгоритмы по шагам  
2 методов (ChangeFields() – изменение полей по индексу и SearchName(list<T>& MyList).

**4.1 Разработка алгоритмов**

Для изменения полей объекта класса используется метод ChangeFields()

1. Начало

2. Входные данные:

2.1. list<T>& MyList – лист, содержащий объекты класса T.

2.2. int num = InputINT(1, int(MyList.size())) - 1; - переменная используемая для вводаиндекса.

3. Выходные данные:

4. typename list<T>::iterator it = MyList.begin();

5. advance(it, num); - устанавливаю итератор на место изменяемого объекта

6. it->ChangeField();-вызов метода, который реализован для каждого дочернего класса.

7. Конец

Для поиска данных вызывается метод SearchName(list<T>& MyList)

1. Начало

2. Входные данные

2.1. string name = InputString(); - переменная которая будет хранить имя для поиска.

2.2. list<T>& MyList – лист, содержащий объекты класса T.

3. Выходные данные

4. Цикл for (typename list<T>::iterator it = MyList.begin(); it != MyList.end(); ++it)

5 Условие if (it->getName() == name)

6. cout << \*it - вывод найденных объектов класса.

7. Конец

# **5 РЕЗУЛЬТАТЫ РАБОТЫ**

В результате проделанной работы была создана программа, позволяющая работать с данными о личных вещах. Данная программа подойдёт для личного пользования. Так же эта программа может послужить основой для создания приложения для контролирования личных вещей.

Далее приведен пример работы программы.

После запуска программы появляется главное меню (рис 5.1). В нем пользователь получает возможность перейти к одной из 4-х категорий личных вещей или отменить последние действие.

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание

Рисунок 5.1 – Меню программы

Как показано на рисунке 5.1 пользователю предоставляется меню для выбора класса личной вещи, с которым он желает работать.

Изображение выглядит как текст, Шрифт, снимок экрана, белый

Автоматически созданное описание

Рисунок 5.2 – Меню функций доступных пользователю

Рассмотрим подробнее функционал программы, предоставленный пользователю:

1. При нажатии «1» нам будет предложено заполнить объект класса, который добавиться в лист.

Изображение выглядит как текст, Шрифт, снимок экрана, белый

Автоматически созданное описание

Рисунок 5.3 – ввод личной вещи

1. При нажатии «2» вы сможете воспользоваться удалением объекта класса из листа, выбрав его номер (рис. 5.4).

Изображение выглядит как текст, чек, Шрифт, число

Автоматически созданное описание

Рисунок 5.4 – Пример выполнения удаления

1. При нажатии «3» вы сможете вывести все объекты выбранного класса(рис.5.5.).

Изображение выглядит как текст, чек, Шрифт, число

Автоматически созданное описание

Рисунок 5.5 – Пример выполнения вывода объектов класса

1. При нажатии «4» вы сможете изменить поле выбранного объекта класса(рис.5.6). Изображение выглядит как текст, снимок экрана, число, Шрифт

   Автоматически созданное описание

Рисунок 5.6 – изменение поля выбранного объекта класса

1. При нажатии «5» вы сможете найти объект класса по минимальной цене. (рис. 5.7).

Изображение выглядит как текст, чек, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 5.7 – поиск объекта класса по минимальной цене.

1. При нажатии «6» вы сможете найти объект класса по названию. (рис. 5.8).

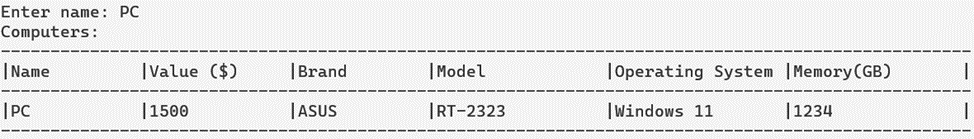


Рисунок 5.8 – поиск объекта класса по названию.

1. При нажатии «7» вы сможете отсортировать объекты класса по цене в порядке увеличения (рис. 5.9).

Изображение выглядит как текст, Шрифт, число, снимок экрана

Автоматически созданное описание

Рисунок 5.9 – сортировка объектов по цене.

1. При нажатии «8» вы очистите лист (рис. 5.10).

Изображение выглядит как текст, Шрифт, чек, снимок экрана

Автоматически созданное описание

Рисунок 5.10 – очистка листа.

1. При нажатии «9» вы вернетесь в главное меню (рис. 5.11).

Изображение выглядит как текст, Шрифт, чек, снимок экрана

Автоматически созданное описание

Рисунок 5.11 – возврат в главное меню.

В главном меню вы можете отменить последнее действие, нажав «5».

В главном меню вы можете закрыть программу, нажав «6» (рис. 5.12).

Изображение выглядит как текст, белый, чек, Шрифт

Автоматически созданное описание

Рисунок 5.12 – завершение программы.

Вся информация автоматически сохраняется в файлах соответствующие классам.

# **ЗАКЛЮЧЕНИЕ**

Хотя C++ представляет собой мультипарадигмальный язык программирования, на практике многие программисты используют его в основном как объектно-ориентированный язык, прибегая к обобщенному программированию только для дополнения ООП.

ООП рассматривается как один из инструментов разработки, доступных программисту для решения задач в коде. Однако на практике ООП часто принимается как золотой стандарт разработки программного обеспечения. Подход, основанный на объектно-ориентированном мышлении, становится отправной точкой для разработки решения. Определение объектов, их взаимодействия и архитектуры системы становятся приоритетными задачами. Переход к такому объектно-ориентированному мышлению позволяет использовать ООП не просто как инструмент, а как полноценный инструментарий для разработки программного обеспечения.

Этот подход позволяет разработчикам лучше организовывать код, повышать его модульность и повторное использование, а также облегчает сопровождение и расширение программного обеспечения. ООП предоставляет мощные концепции, такие как инкапсуляция, наследование и полиморфизм, которые способствуют созданию гибкой и расширяемой кодовой базы.

Таким образом, хотя ООП является важным инструментом разработки программного обеспечения в C++.

Программа написана в среде разработки Visual studio 2022.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1].Р. Лафоре Объектно-ориентированное программирование в С++ 4-е издание 2004г.

[2].Ю.А. Луцик Объектно-ориентированное программирование на языке С++ Минск. БГУИР. 2008г.

[3].Луцик, Ю. А. Объектно-ориентированное программирование на языке С++: учеб. пособие по курсу «Объектно-ориентированное программирование» для студ. спец. «Вычислительные машины, системы и сети» всех форм обуч. / Ю. А. Луцик, А. М. Ковальчук, И. В. Лукьянова. – Минск: БГУИР, 2003. – 203 с.:ил.

# **ПРИЛОЖЕНИЕ А**

*(обязательное)*

Cхема алгоритма

# **ПРИЛОЖЕНИЕ Б**

*(обязательное)*

Cхема алгоритма

# **ПРИЛОЖЕНИЕ В**

*(обязательное)*

Диаграмма классов

# **ПРИЛОЖЕНИЕ Г**

*(обязательное)*

Код программы

# **ПРИЛОЖЕНИЕ Д**

*(обязательное)*

Ведомость документов