Федеральное государственное бюджетное образовательное учреждение высшего образования

"ПЕТРОЗАВОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”

Физико-технический институт

Кафедра информационно-измерительных систем и физической электроники

**Отчет по лабораторной работе**

ТЕМА: «Командное задание по курсу Технология программирования»

Выполнили: студенты 3 курса

физико-технического института, гр.21316,

Поутанен Степан Андреевич

Смирнов Николай Михайлович

Тришкин Дмитрий Сергеевич

Преподаватель:

Бульба Артём Владимирович

Петрозаводск

2021

Мы разработали программное обеспечение для Ювелирного магазина.

Кратко о программной реализации: Программа была разработана в среде Visual Studio на языке программирования C++.

Список всех используемых заголовочных и исполнительных файлов:

**AnnualReport.h** // Заголовочный файл класса, отвечающего за вывод годового отчёта

**Expense.h** // Заголовочный файл класса, отвечающего за описание информации о расходах

**ExpenseInputScreen.h** // Заголовочный файл класса, отвечающего за добавление статьи расходов

**ExpenseRecord.h** // Заголовочный файл класса, отвечающего за вывод списка расходов

**Product.h** // Заголовочный файл класса, отвечающего за хранение списка товаров

**ProductInputScreen.h** // Заголовочный файл класса, отвечающего за добавление товара

**ProductList.h** // Заголовочный файл класса, отвечающего за вывод списка товаров на экран

**Sell.h** // Заголовочный файл класса, отвечающего за вывод списка продаж

**SellInputScreen.h** // Заголовочный файл класса, отвечающего за добавление продажи товара

**SellProduct.h** // Заголовочный файл класса, отвечающего за хранение списка продаж

**UserInterface.h** // Заголовочный файл класса, отвечающего за вывод и работу пользовательского интерфейса для взаимодействия с программой

**AnnualReport.cpp** // Заголовочный файл класса, отвечающего за вывод годового отчёта

**ExpenseInputScreen.cpp** // Исполнительный файл класса, отвечающего за добавление статьи расходов

**ExpenseRecord.cpp** // Исполнительный файл класса, отвечающего за вывод списка расходов

**Main.cpp** // Исполнительный файл, отвечающий за вывод консоли для взаимодействия

**Product.cpp** // Исполнительный файл класса, отвечающего за хранение списка товаров

**ProductInputScreen.cpp** // Исполнительный файл класса, отвечающего за добавление товара

**ProductList.cpp** // Исполнительный файл класса, отвечающего за вывод списка товаров на экран

**Sell.cpp** // Исполнительный файл класса, отвечающего за вывод списка продаж

**SellInputScreen.cpp** // Исполнительный файл класса, отвечающего за добавление продажи товара

**SellProduct.cpp** // Исполнительный файл класса, отвечающего за хранение списка продаж

**UserInterface.cpp** // Заголовочный файл класса, отвечающего за вывод и работу пользовательского интерфейса для взаимодействия с программой

1. В нашу небольшую студию разработки обратился владелец ювелирного салона X со следующим заказом: разработать программу для контроля работы ювелирного магазина. Программа должна обладать следующим функционалом:

1. Вносить продукцию в базу (занесение на склад)
2. Вносить имеющуюся продукцию в статус проданного (продажа)
3. Вносить расходы салона (покупка украшений у поставщика, оплата ком.услуг)
4. Вывод имеющейся на складе продукции, её характеристик и количества
5. Вывод отчёта по продажам салона
6. Вывод расходов салона
7. Вывод годового общего отчёта с указанием текущего баланса салона

2. Описание имеющихся у заказчика материалов (данных) с которыми ему приходится работать. Это могут быть таблицы с записями. Приблизительный вид таких таблиц, их назначение и смысл содержимого нужно обязательно привести в отчете. При необходимости рукописно

Таблица 1. Учёт имеющихся товаров.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Название | Тип | Материал | Артикул | Размер | Количество | Цена |
| Кольцо с Фианитом | Кольцо | Красное золото | 43057 | 17 | 161 | 2326₽ |
| Браслет дорожки | Браслет | Красное золото | 43000 | 17 | 58 | 7171₽ |
| Подвеска “Афродита” | Подвеска | Красное золото | 43005 | 0 | 92 | 3059₽ |

Таблица 2. Учёт расходов салона

|  |  |  |  |
| --- | --- | --- | --- |
| День | Месяц | Получатель | Сумма |
| 22 | 1 | ПетрГазСтрой | 15456 |
| 31 | 12 | ЛюксСтрой | 39998 |
| 26 | 11 | ФНС | 86922 |

Таблица 3. Годовой отчёт

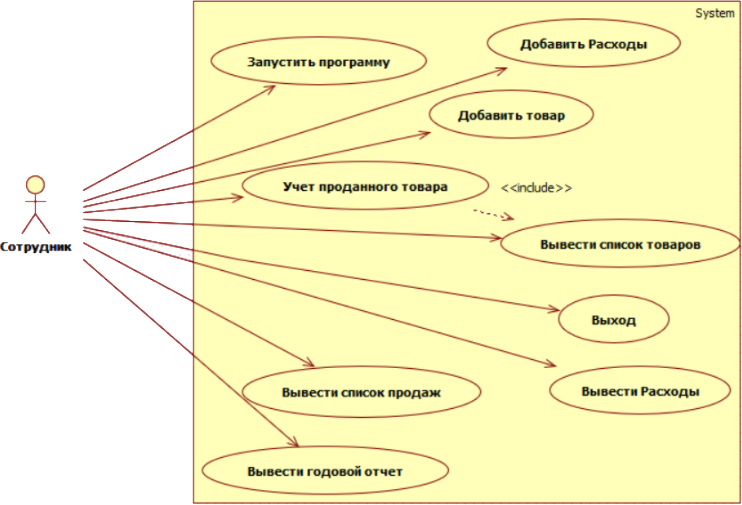
|  |  |
| --- | --- |
| Тип учитываемого: | Сумма: |
| Доходы с продаж: | 525100 |
| Расходы | 15000 |
|  | 45000 |
| Итого расходов: | 60000 |
| Баланс | 465100 |

Таблица 4. Учёт продаж

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Артикул | Январь | Февраль | Март | Апрель | Май | Июнь | Июль | Август | Сентябрь | Октябрь | Ноябрь | Декабрь |
| 43057 | **37216** | **13956** | **18608** | **27912** | **37216** | **51172** | **13956** | **20934** | **55824** | **72106** | **39542** | **16282** |
| 43000 | **157762** | **193617** | **365721** | **595193** | **164933** | **179275** | **86052** | **437431** | **186446** | **258156** | **437431** | **365721** |
| 43005 | **107065** | **198835** | **97888** | **156009** | **217189** | **110124** | **39767** | **217189** | **110124** | **186599** | **128478** | **174363** |

3.Действующий субъект - сотрудник. Сотрудник напрямую взаимодействует с системой и ведет учет товаров, их продажу, расходы на поставщиков и ЖКХ, и в конечном итоге может вывести годовой отчет, включающий в себя все расходы и доходы и финансовое состояние фирмы.

4.Список вариантов использования:



5.Документация к вариантам использования

1)Запустить программу

Выполняется запуск программы с последующим выводом на экран пользовательского интерфейса, в котором сотрудник может совершать необходимые ему действия.

2)Добавить Расходы

Выполняется вывод на экран интерфейса отвечающего за внесение статьи расходов, в котором пользователь может вести дату, название фирмы получателя и сумму.

3)Добавить товар

Выполняется вывод на экран интерфейса отвечающего за внесение нового товара в базу, в котором сотрудника просят внести следующую информацию о товаре: Название, тип, материал, артикул, размер и количество

4)Учет проданного товара

Выполняется вывод на экран интерфейса отвечающего за внесение продажи товара в базу, сотрудника просят ввести артикул проданного товара, количество проданных единиц и дату сделки.

5)Вывести список товаров

Программа выводит на экран список товаров, которые числятся активными в базе, отображая их Название, тип, материал, артикул, размер и текущее количество единиц на складе.

6)Вывести Расходы

Программа выполняет вывод на экран месяца и дня, в который был осуществлён расход, наименование организации, которая получила выплату и сумма выплаты.

7)Вывести список продаж

Выполняется вывод на экран артикулов проданных товаров и сумм продаж по месяцам

8)Вывести годовой отчет

Выполняется вывод на экран суммы всего проданного за год, далее список расходов, их общая сумма и баланс, рассчитываемый как разность доходов и расходов.

9)Выход

Программа завершает работу.

6. Диаграммы деятельности:

Диаграмма деятельности варианта использования “Учет проданного товара”

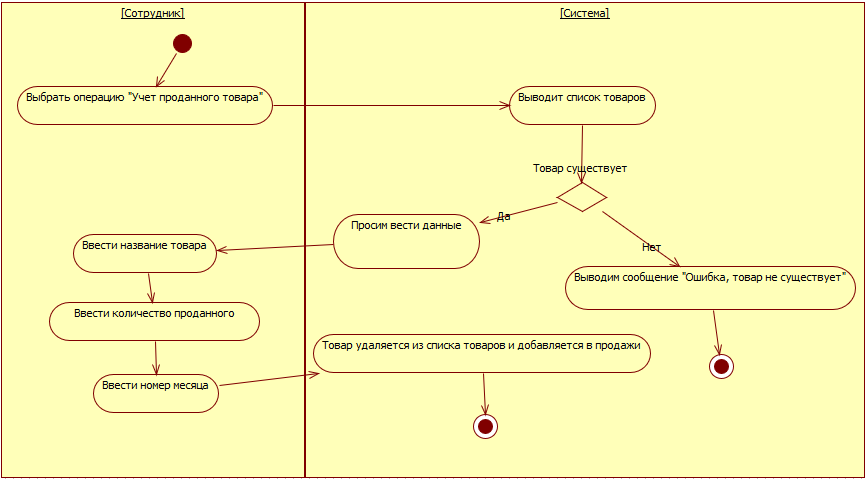
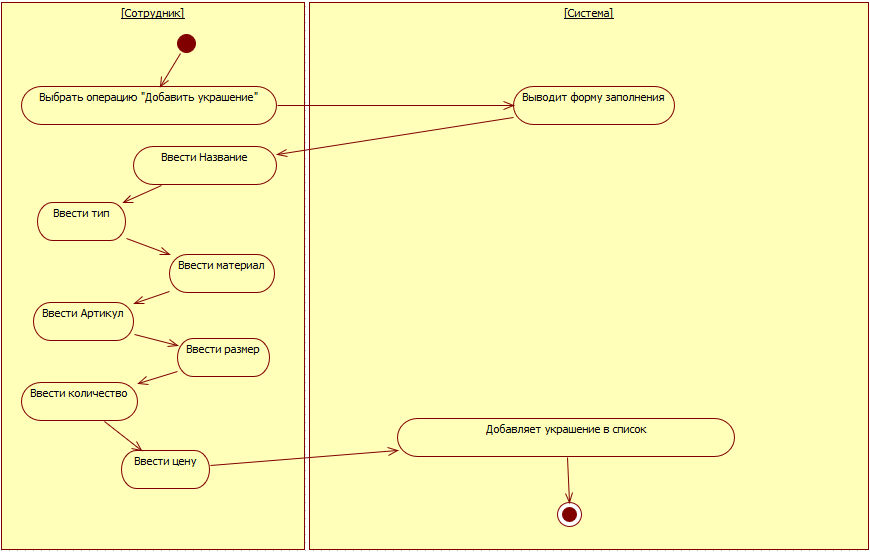


Диаграмма деятельности варианта использования “Добавить украшение”



7.Список существительных

1. Украшение
2. Список украшений
3. Товар
4. Название
5. Тип
6. Материал
7. Продукт
8. Артикул
9. Размер
10. Количество
11. Цена
12. Номер месяца
13. Количество проданного
14. Продажи
15. Список продуктов
16. Годовой отчёт
17. Список продаж
18. Список расходов
19. Расходы
20. Экран интерфейса пользователя

8.Список классов:

1. Продукт
2. Список продуктов
3. Экран ввода продукта
4. Экран пользовательского интерфейса
5. Продажи
6. Список продаж
7. Экран ввода продаж
8. Годовой отчёт
9. Расходы
10. Список расходов
11. Экран списка расходов

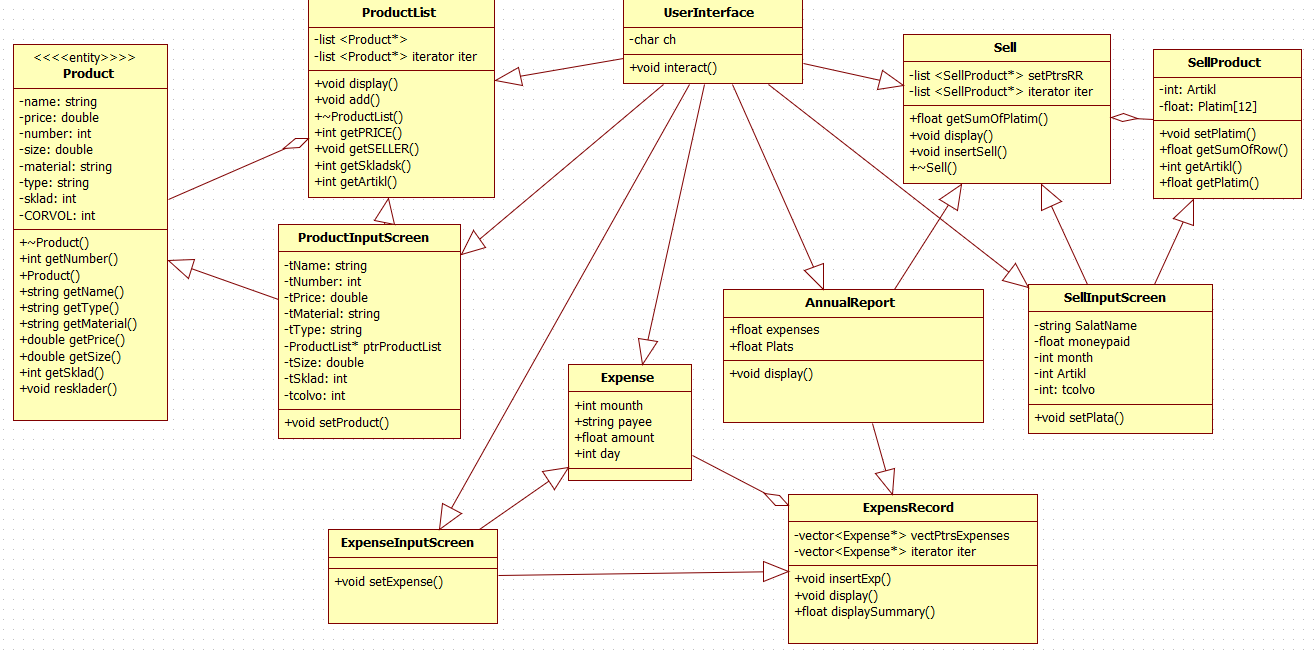
9.Атрибуты классов:

1. Продукт
   1. name- наименование товара
   2. price-стоимость товара
   3. number-артикул
   4. size-размер
   5. material-материал товара
   6. type-тип товара
   7. sklad-количество товара на складе
   8. CORVOL-количество проданных единиц
2. Список продуктов
   1. ListProduct-Список хранящий информацию о товарах
   2. IterIterator- итератор списка ListProduct
3. Экран ввода продукта
   1. tName- наименование товара
   2. tNumber-артикул
   3. tPrice-стоимость товара
   4. tMaterial-материал товара
   5. tType-тип товара
   6. tSize-размер
   7. tSklad-количество товара на складе
   8. tcolvo-количество проданных единиц
4. Экран пользовательского интерфейса
   1. ch-отвечает за нажатую клавишу
5. Продажи
   1. ListSellProduct-список хранящий информацию о продажах
   2. iterIterator-итератор списка ListSellProduct
6. Список продаж
   1. Artikl-Артикул проданного товара
   2. Platim[12]-суммы продаж по месяцам
7. Экран ввода продаж
   1. SalatName-Название проданного товара
   2. moneypaid-сумма сделки
   3. month-Месяц продажи
   4. Artikl-Артикул проданного товара
   5. tcolvo-Количество проданных единиц
8. Годовой отчёт
   1. expenses-сумма доходов
   2. Plats-сумма расходов
9. Расходы
   1. month-месяц оплаты
   2. day-день оплаты
   3. payee-фирма получатель
   4. amount-сумма оплаты
10. Список расходов
    1. Vector Expense-вектор, хранящий информацию о расходах
    2. IterIterator-итератор вектора Vector Expense
11. Экран ввода расходов
    1. -

10.Методы классов:

1. Продукт
   1. метод getNumber- возвращает артикул товара
   2. метод getName- возвращает название товара
   3. метод getType- возвращает тип товара
   4. метод getMaterial- возвращает материал товара
   5. метод getPrice-возвращает цену товара
   6. метод getSize-возвращает размер товара
   7. метод getSklad-возвращает количество товара на складе
   8. метод resklader-изменяет количество товара на складе
2. Список продуктов
   1. метод display-выводит список товаров
   2. метод add-добавляет новый товар в список
   3. метод getPRICE-возвращает цену товара по названию
   4. метод getSELLER-изменяет количество товара на складе при продаже
   5. метод getSklaksk-возвращает количество товаров на складе по названию
   6. метод getArtikl- возвращает артикул по названию товара
3. Экран ввода продукта
   1. метод setProduct-добавить данные о новом товаре
4. Экран пользовательского интерфейса
   1. метод interact-запуск интерактивного меню программы
5. Продажи
   1. метод getSumOfPlatim-подсчёт общей суммы сделок
   2. метод display-вывод таблицы продаж
   3. метод insertSell-добавить сделку таблицу
6. Список продаж
   1. метод setPlatim-привязывает сделку к месяцу
   2. метод getSumOfRow-вычисляет сумму сделок за 12 месяцев
   3. метод getArtikl-возвращает артикул
   4. метод getPlatim-возвращает сумму сделок за месяц
7. Экран ввода продаж
   1. метод setPlata-добавляет сумму сделки в нужный месяц
8. Годовой отчёт
   1. метод display-выводит на экран годовой отчёт
9. Расходы
   1. -
10. Список расходов
    1. метод insertExp-вносит статью расходов в таблицу
    2. метод display-выводит на экран расходы
    3. метод displaySummary-рассчитывает расходы для годового отчёта
11. Экран ввода расходов
    1. метод setExpense- добавляет статью расходов

11. Диаграмма классов



12. Диаграммы последовательности

Диаграмма последовательности варианта использования “Добавить товар”

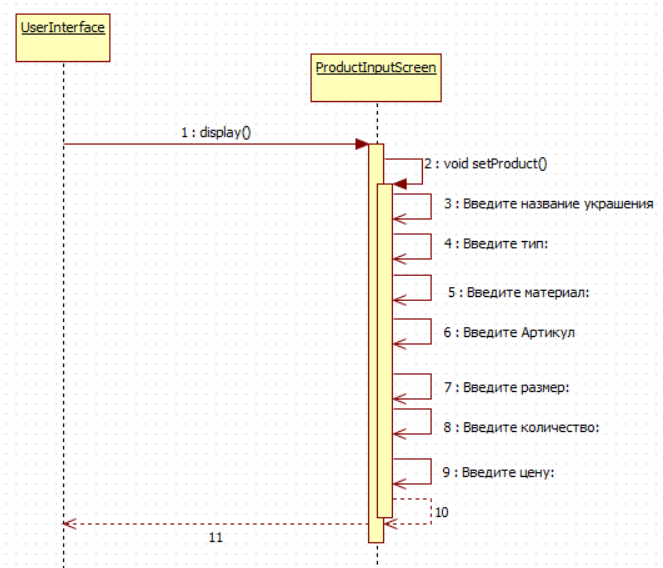


Диаграмма последовательности варианта использования “Добавить расходы”

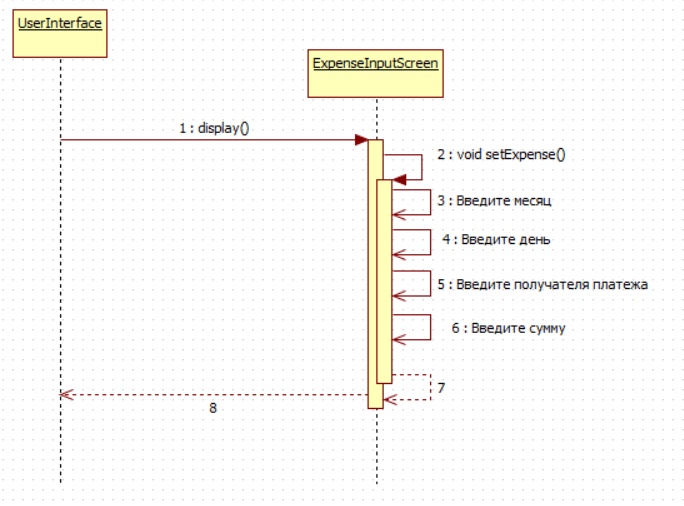
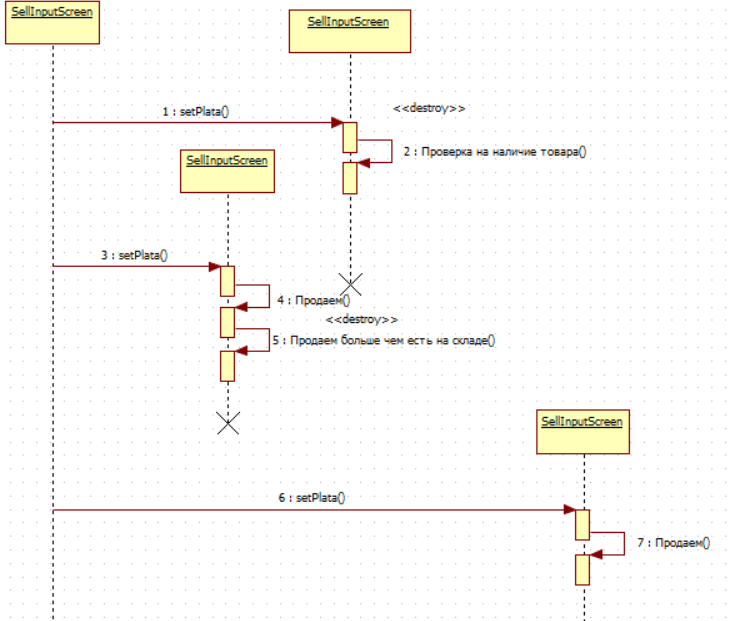


Диаграмма последовательности варианта использования “Учет проданного товара”



13. Код заголовочных файлов

**AnnualReport.h**

#include "ExpenseRecord.h"

#include "Sell.h"

class AnnualReport

{

private:

Sell\* ptrRR; // записи доходов

ExpenseRecord\* ptrER; // записи расходов

float expenses, Plats; // суммы доходов и расходов

public:

AnnualReport(Sell\*, ExpenseRecord\*);

void display(); // отображение годового отчета

};

**Expense.h**

#include <iostream>

#include <vector>

#include <list>

#include <string>

#include <iomanip>

#include <Windows.h>

#include <numeric> //для accumulate()

using namespace std;

class Expense

{

public:

int month, day; // месяц и день уплаты расходов

string payee; // кому платим

float amount; // сколько платим

Expense()

{ }

Expense(int m, int d, string p, float a) :

month(m), day(d), payee(p), amount(a)

{

/\* тут пусто! \*/

}

};

**ExpenseInputScreen.h**

#include "Expense.h"

#include "ExpenseRecord.h"

#include <iostream>

#include <vector>

#include <list>

#include <string>

#include <iomanip>

#include <Windows.h>

#include <numeric>

using namespace std;

class ExpenseInputScreen

{

private:

ExpenseRecord\* ptrExpenseRecord; // запись о расходах

public:

ExpenseInputScreen(ExpenseRecord\*);

void setExpense();

};

**ExpenseRecord.h**

#include "Expense.h"

#include <iostream>

#include <vector>

#include <iomanip>

#include <Windows.h>

using namespace std;

class ExpenseRecord

{

private:

vector<Expense\*> vectPtrsExpenses; //вектор указателей на расходы

vector<Expense\*>::iterator iter;

public:

~ExpenseRecord();

void insertExp(Expense\*);

void display();

float displaySummary(); // нужно для годового отчета

};

**Product.h**

#ifndef Product\_h

#define Product\_h

#include <iostream>

#include <vector>

#include <list>

#include <string>

#include <iomanip>

#include <Windows.h>

#include <numeric>

using namespace std;

void getaLine(string& inStr); // получение строки текста

char getaChar();

class Product

{

private:

string name, type, material;

int number, sklad, CORVOL;

double price, size;

public:

Product(string n, string n1, string n2, double aNo1, int aNo, double aNo2, int aNo3);

~Product();

int getNumber(); //возвращает кол-во на складе

string getName();

string getType();

int getSklad();

void resklader(int);

string getMaterial();

double getPrice();

double getSize();

};

#endif Product\_h

**ProductInputScreen.h**

#include "ProductList.h"

#include "Product.h"

class ProductInputScreen

{

private:

ProductList\* ptrProductList;

string tName;

string tType;

string tMaterial;

double tPrice, tSize;

int tNumber, tSklad, tcolvo;

public:

ProductInputScreen(ProductList\* ptrTL) : ptrProductList(ptrTL)

{

/\* тут пусто \*/

}

void setProduct(); // добавить данные о товаре

};

**ProductList.h**

#include "Product.h"

#ifndef ProductList\_h

#define ProductList\_h

class ProductList

{

private:

// установить указатели на украшение

list <Product\*> setPtrsProd;

list <Product\*>::iterator iter; //итератор

public:

~ProductList(); // деструктор (удаление украшений)

void add(Product\*); // добавить украшение в список

int getPRICE(string); // возвращает цену

void display(); // вывод списка украшений

int getArtikl(string); // возвращает артикул

void getSELLER(string, int);

int getSkladsk(string);

};

#endif ProductList\_h

**Sell.h**

#include "SellProduct.h"

#include <iostream>

#include <vector>

#include <list>

#include <string>

#include <iomanip>

#include <Windows.h>

#include <numeric>

using namespace std;

//////////////////// класс Sell ///////////////////////

//класс Sell. Он хранит непосредственно записи об продажах.

//С ним будет взаимодействовать экран добавления продажи украшений.

class Sell

{

private:

list <SellProduct\*> setPtrsRR; //

list <SellProduct\*>::iterator iter;

public:

~Sell();

void insertSell(int, int, float); // добавить сделку

void display(); // вывести строки по сделкам

float getSumOfPlatim(); // подсчёт общей суммы сделок

};

**SellInputScreen.h**

#include "Sell.h"

#include "ProductList.h"

#include "SellProduct.h"

using namespace std;

////////////////////класс SellInputScreen //////////////////

class SellInputScreen

{

private:

ProductList\* ptrProductList; // список товаров

Sell\* ptrSell; // список сделок

string SalatName; // Название товара

float moneypaid; // Сумма сделки

int month, tcolvo; // за месяц

int Artikl; // артикул

public:

SellInputScreen(ProductList\* ptrTL, Sell\* ptrRR) : ptrProductList(ptrTL),

ptrSell(ptrRR)

{

/\*тут пусто\*/

}

void setPlata(); // добавить сумму сделки в нужном месяце

};

**SellProduct.h**

#include <iostream>

#include <iomanip>

#include <Windows.h>

#include <numeric> //для accumulate()

using namespace std;

class SellProduct

{

private:

int Artikl; // артикул

float Platim[12]; // месяцы

public:

SellProduct(int); // конструктор с одним параметром

void setPlatim(int, float); // добавить сумму сделки в нужном месяце

float getSumOfRow();

int getArtikl(); //запрос артикула

float getPlatim(int); //Запрос суммы сделки за месяц

};

**UserInterface.h**

#include "ProductInputScreen.h"

#include "ExpenseInputScreen.h"

#include "SellInputScreen.h"

#include "ProductList.h"

#include "Sell.h"

#include "AnnualReport.h"

class UserInterface

{

private:

ProductList\* ptrProductList;

ProductInputScreen\* ptrProductInputScreen;

Sell\* ptrSell;

SellInputScreen\* ptrSellInputScreen;

ExpenseRecord\* ptrExpenseRecord;

ExpenseInputScreen\* ptrExpenseInputScreen;

AnnualReport\* ptrAnnualReport;

char ch;

public:

UserInterface();

~UserInterface();

void interact();

};

14. Код исполняемых файлов

**AnnualReport.cpp**

#include "AnnualReport.h"

////////////////методы класса AnnualReport/////////////////

//Конструктор

AnnualReport::AnnualReport(Sell\* pRR, ExpenseRecord\* pER) : ptrRR(pRR), ptrER(pER)

{ /\* пусто \*/

}

//---------------------------------------------------------

void AnnualReport::display()

{

cout << "Годовой отчет\n--------------\n" << endl;

cout << "Доходы\n" << endl;

cout << "\tПродажи:\t\t";

Plats = ptrRR->getSumOfPlatim();

cout << Plats << endl;

cout << "Расходы\n" << endl;

expenses = ptrER->displaySummary();

cout << "Расходы всего:\t\t";

cout << expenses << endl;

// вычисляем прибыльность

cout << "\nБаланс:\t\t\t" << (Plats - expenses) << endl;

}

**ExpenseInputScreen.cpp**

#include "ExpenseInputScreen.h"

#include <iostream>

using namespace std;

//////////////методы класса ExpenseInputScreen/////////////

void getaLine(string& inStr); // получение строки текста

char getaChar();

// конструктор

ExpenseInputScreen::ExpenseInputScreen(ExpenseRecord\* per) : ptrExpenseRecord(per)

{

/\*пусто\*/

}

//------------------------------------------------------

void ExpenseInputScreen::setExpense()

{

int month, day;

string category, payee;

float amount;

cout << "Введите месяц (1-12): ";

cin >> month;

cin.ignore(80, '\n');

cout << "Введите день (1-31): ";

cin >> day;

cin.ignore(80, '\n');

cout << "Введите получателя платежа (ООО 'РосКосмСтрой'): ";

getaLine(payee);

cout << "Введите сумму (14500(целое число)): ";

cin >> amount;

cin.ignore(80, '\n');

// создаем новый расход

Expense\* ptrExpense = new Expense(month, day, payee, amount);

// вставляем расход в список всех расходов

ptrExpenseRecord->insertExp(ptrExpense);

}

**ExpenseRecord.cpp**

#include "ExpenseRecord.h"

#include <iostream>

using namespace std;

ExpenseRecord::~ExpenseRecord() // деструктор

{ // удалить объекты expense

// удалить указатели на вектор

while (!vectPtrsExpenses.empty())

{

iter = vectPtrsExpenses.begin();

delete\* iter;

vectPtrsExpenses.erase(iter);

}

}

//--------------------------------------------------------

void ExpenseRecord::insertExp(Expense\* ptrExp)

{

vectPtrsExpenses.push\_back(ptrExp);

}

//---------------------------------------------------------

void ExpenseRecord::display() // распечатываем все расходы

{

cout << setw(5) << left << "День"

<< setw(10) << "Месяц"

<< setw(25) << "Получатель"

<< setw(15) << "Сумма"

<< endl;

if (vectPtrsExpenses.size() == 0) // В контейнере нет расходов

cout << "\*\*\*Расходов нет\*\*\*\n" << endl;

else

{

iter = vectPtrsExpenses.begin();

while (iter != vectPtrsExpenses.end())

{ // распечатываем все расходы

cout << setw(5) << left << (\*iter)->day

<< setw(10) << (\*iter)->month

<< setw(25) << (\*iter)->payee

<< setw(15) << (\*iter)->amount

<< endl;

iter++;

}

cout << endl;

}

}

//--------------------------------------------------------

// используется при составлении годового отчета

float ExpenseRecord::displaySummary()

{

float totalExpenses = 0; // Сумма по всем категориям расходов

if (vectPtrsExpenses.size() == 0)

{

cout << "\tВсе категории\t0\n";

return 0;

}

iter = vectPtrsExpenses.begin();

while (iter != vectPtrsExpenses.end())

{

//выводим на экран категории расходов

cout << '\t' << '\t' << ((\*iter)->amount) << endl;

totalExpenses += (\*iter)->amount; //подсчитываем все расходы

iter++;

}

return totalExpenses;

}

**Main.cpp**

#include "UserInterface.h"

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

UserInterface theUserInterface;

theUserInterface.interact();

return 0;

}

**Product.cpp**

#include <iostream>

#include "Product.h"

void getaLine(string& inStr) // получение строки текста

{

char temp[21];

cin.get(temp, 20, '\n');

cin.ignore(20, '\n'); //число пропускаемых символов и символ разделения

inStr = temp;

}

//---------------------------------------------------------

char getaChar() // получение символа

{

char ch = cin.get();

cin.ignore(80, '\n'); //число пропускаемых символов и символ разделения

return ch;

}

//---------------------------------------------------------

////////////////методы класса Product///////////////////////

//в конструкторе задаём параметры украшений

Product::Product(string n, string n1, string n2, double aNo1, int aNo, double aNo2, int aNo3) : name(n), type(n1), material(n2), size(aNo1), number(aNo), price(aNo2), sklad(aNo3)

{

/\* тут пусто \*/

}

//---------------------------------------------------------

Product::~Product() // деструктор

{

/\* тут тоже пусто \*/

}

//---------------------------------------------------------

int Product::getNumber() //геттер возвращает артикул

{

return number;

}

void Product::resklader(int CORVOL) //функция уменьшения кол-ва на складе

{

sklad = sklad - CORVOL;

}

int Product::getSklad() //геттер возвращает колличество на складе

{

return sklad;

}

double Product::getSize() //геттер возвращает размер

{

return size;

}

double Product::getPrice() //геттер возвращает цену

{

return price;

}

//--------------------------------------------------------

string Product::getName() //геттер возвращает название

{

return name;

}

string Product::getType() //геттер возвращает тип

{

return type;

}

string Product::getMaterial() //геттер возвращает материал

{

return material;

}

**ProductInputScreen.cpp**

#include <iostream>

#include "ProductInputScreen.h"

///////////метод класса ProducttInputScreen//////////////////

void ProductInputScreen::setProduct() // добавить данные об украшении

{

cout << "Введите название украшения: " << endl;

getaLine(tName);

cout << "Введите тип: " << endl;

getaLine(tType);

cout << "Введите материал: " << endl;

getaLine(tMaterial);

cout << "Введите Артикул(не более 9 цифр): " << endl;

cin >> tNumber;

cout << "Введите размер: " << endl;

cin >> tSize;

cout << "Введите количество: " << endl;

cin >> tSklad;

cout << "Введите цену: " << endl;

cin >> tPrice;

cin.ignore(80, '\n');

Product\* ptrProduct = new Product(tName, tType, tMaterial, tSize, tNumber, tPrice, tSklad); // создать украшение

ptrProductList->add(ptrProduct); // занести в список украшений

}

**ProductList.cpp**

#include <iostream>

#include "ProductList.h"

////////////////методы класса ProductList///////////////////

ProductList::~ProductList() // деструктор

{

while (!setPtrsProd.empty()) // удаление всех украшений,

{ // удаление указателей из контейнера

iter = setPtrsProd.begin();

delete\* iter;

setPtrsProd.erase(iter);

}

}

//---------------------------------------------------------

void ProductList::add(Product\* ptrT)

{

setPtrsProd.push\_back(ptrT); // вставка нового украшения в список

}

//---------------------------------------------------------

int ProductList::getArtikl(string tName) // получить артикул украшения по названию

{

int Artikl;

iter = setPtrsProd.begin();

while (iter != setPtrsProd.end())

{ //поиск украшения в списке

Artikl = (\*iter)->getNumber();

if (tName == ((\*iter)->getName())) // ищем украшение по имени и возвращаем его артикул

{

return Artikl;

}

iter++;

}

return -1;

}

//--------------------------------------------------------

//---------------------------------------------------------

int ProductList::getSkladsk(string tName) // получить кол-во на складе по названию украшения

{

int Skladsk;

iter = setPtrsProd.begin();

while (iter != setPtrsProd.end())

{ //поиск украшения в списке

Skladsk = (\*iter)->getSklad();

if (tName == ((\*iter)->getName())) // ищем украшение по имени и возвращаем его кол-во на складе

{

return Skladsk;

}

iter++;

}

return -1; // если нет - возвращаем -1

}

//--------------------------------------------------------

//---------------------------------------------------------

int ProductList::getPRICE(string tName) // получить цену по названию украшения

{

int PRICE;

iter = setPtrsProd.begin();

while (iter != setPtrsProd.end())

{ //поиск украшения в списке

PRICE = (\*iter)->getPrice();

if (tName == ((\*iter)->getName())) // ищем украшение по имени и возвращаем его цену

{

return PRICE;

}

iter++;

}

return -1; // если нет - возвращаем -1

}

//--------------------------------------------------------

//---------------------------------------------------------

void ProductList::getSELLER(string tName, int tcolvo) // изменить кол-во на складе при продаже

{

int itog;

iter = setPtrsProd.begin();

while (iter != setPtrsProd.end())

{ //поиск украшения в списке

itog = (\*iter)->getPrice();

if (tName == ((\*iter)->getName())) // находим нужное украшение и после проверки уменьшаем кол-во на складе

{ if ((\*iter)->getSklad() >= tcolvo)

{

(\*iter)->resklader(tcolvo);

}

}

iter++;

}

}

//--------------------------------------------------------

void ProductList::display() // вывод списка украшений

{

cout << setw(22) << left << "Название украшения"

<< setw(8) << "Тип"

<< setw(15) << "Материал"

<< setw(15) << "Размер"

<< setw(22) << "Артикул"

<< setw(22) << "Кол-во на складе"

<< setw(22) << "Цена"

<< endl;

if (setPtrsProd.empty()) { // если список украшений пуст

cout << "\*\*\*Нет украшений\*\*\*\n" << endl; // выводим запись, что он пуст)

}

else

{

iter = setPtrsProd.begin();

while (iter != setPtrsProd.end()) // распечатываем все украшения

{

cout << setw(22) << left << (\*iter)->getName()

<< setw(8) << (\*iter)->getType()

<< setw(15) << (\*iter)->getMaterial()

<< setw(15) << (\*iter)->getSize()

<< setw(22) << (\*iter)->getNumber()

<< setw(22) << (\*iter)->getSklad()

<< setw(22) << (\*iter)->getPrice()

<< endl;

\*iter++;

}

}

}

**Sell.cpp**

#include "Sell.h"

#include <iostream>

using namespace std;

/////////////////методы класса Sell//////////////////

Sell::~Sell() // деструктор

{ // удалить строки с платежами,

// удалить указатели из множества.

while (!setPtrsRR.empty())

{

iter = setPtrsRR.begin();

delete\* iter;

setPtrsRR.erase(iter);

}

}

//---------------------------------------------------------

void Sell::insertSell(int Artikl, int month, float amount)

{

iter = setPtrsRR.begin(); // Инициализация итератора

while (iter != setPtrsRR.end()) // условие выхода

{ // если текущий объект совпадает с созданным для поиска,

if (Artikl == (\*iter)->getArtikl())

{ // заносим сделку в список

(\*iter)->setPlatim(month, amount);

return;

}

else

iter++;

} // если не нашли строку - создаем новую

SellProduct\* ptrRow = new SellProduct(Artikl);

ptrRow->setPlatim(month, amount); // заносим в нее платеж

setPtrsRR.push\_back(ptrRow); // заносим строку в вектор

}

//---------------------------------------------------------

void Sell::display() // отобразить все строки

{

cout << setw(22) << left << "Артикул"

<< setw(8) << "Янв"

<< setw(8) << "Фев"

<< setw(8) << "Мар"

<< setw(8) << "Апр"

<< setw(8) << "Май"

<< setw(8) << "Июн"

<< setw(8) << "Июл"

<< setw(8) << "Авг"

<< setw(8) << "Сен"

<< setw(8) << "Окт"

<< setw(8) << "Ноя"

<< setw(8) << "Дек"

<< endl;

if (setPtrsRR.empty())

cout << "\*\*\*Нет платежей!\*\*\*\n" << endl;

else

{

iter = setPtrsRR.begin(); //итератор на список с указателями на объекты SellProduct

while (iter != setPtrsRR.end())

{

cout << setw(22) << (\*iter)->getArtikl() << '\t'; // вывести артикул продукта

for (int j = 0; j < 12; j++) // вывести 12 месяцев и продажи в них

{

if (((\*iter)->getPlatim(j)) == 0)

cout << setw(8) << " 0 ";

else

cout << setw(8) << (\*iter)->getPlatim(j) << " ";

}

cout << endl;

iter++;

}

cout << endl;

cout << endl;

}

}

//---------------------------------------------------------

float Sell::getSumOfPlatim() // сумма всех сделок

{

float sumRents = 0.0;

iter = setPtrsRR.begin();

while (iter != setPtrsRR.end())

{ // плюсуем суммы всех сделок

sumRents += (\*iter)->getSumOfRow();

iter++;

}

return sumRents;

}

**SellInputScreen.cpp**

/////////////////методы класса SellInputScreen/////////////

#include "SellInputScreen.h"

void SellInputScreen::setPlata()

{

int skaldsk;

ptrProductList->display();

cout << "Введите наименование товара: ";

getaLine(SalatName);

// получить название товара по артикулу

Artikl = ptrProductList->getArtikl(SalatName);

if (Artikl > 0) // если артикул найден и такой товар существует -

{

cout << "Введите количество проданного: " << endl;

cin >> tcolvo;

cin.ignore(80, '\n');

cout << "Введите номер месяца для квартального отчёта: " << endl;

cin >> month;

cin.ignore(80, '\n');

month--; // (внутренняя нумерация 0-11)

// вставляем ренту в запись об оплате

skaldsk = ptrProductList->getSkladsk(SalatName);

if (skaldsk >= tcolvo)

{

moneypaid = ptrProductList->getPRICE(SalatName);

moneypaid = tcolvo \* moneypaid;

ptrProductList->getSELLER(SalatName, tcolvo);

ptrSell->insertRent(Artikl, month, moneypaid);

}

else

{

cout << "Ошибка, нельзя столько продать.\n" << endl;

}

}

else

cout << "Ошибка, товар не существует.\n" << endl;

}

//---------------------------------------------------------

**SellProduct.cpp**

#include "SellProduct.h"

/////////////////методы класса SellProduct/////////////////////

SellProduct::SellProduct(int an) : Artikl(an) //конструктор

{ //Алгоритм fill() помещает копию значения value (у нас это 0)

//в каждый элемент диапазона, ограниченного парой итераторов [first,last).

//Т.е. в конструкторе просто инициализируем массив значениями 0.

fill(&Platim[0], &Platim[12], 0);

}

//---------------------------------------------------------

void SellProduct::setPlatim(int m, float am) // сеттер расхода за месяц m, сумма - am

{

Platim[m] += am; // привязываем сделку к месяцу

}

//---------------------------------------------------------

int SellProduct::getArtikl() // геттер запрос артикула

{

return Artikl;

}

//---------------------------------------------------------

float SellProduct::getPlatim(int month) //Геттер запрос суммы сделок за месяц month

{

return Platim[month];

}

//---------------------------------------------------------

float SellProduct::getSumOfRow() // cумма сделок в строке

{ //По умолчанию алгоритм accumulate() суммирует элементы.

//Нужно указать точку старта, конечную точку и значение от которого начинаем прибавлять.

//Чаще всего это ноль, но может быть и результат других вычислений.

return accumulate(&Platim[0], &Platim[12], 0);

}

**UserInterface.cpp**

#include <iostream>

#include "UserInterface.h"

////////////////методы класса userInterface//////////////

UserInterface::UserInterface()

{

ptrProductList = new ProductList;

ptrSell = new Sell;

ptrExpenseRecord = new ExpenseRecord;

}

//---------------------------------------------------------

UserInterface::~UserInterface()

{

delete ptrProductList;

delete ptrSell;

delete ptrExpenseRecord;

}

//---------------------------------------------------------

void UserInterface::interact()

{

while (true)

{

cout << "Для ввода данных нажмите '1', \n"

<< "Для вывода отчета '2', \n"

<< "Для выхода '0': \n";

ch = getaChar();

if (ch == '1') // ввод данных

{

cout << "Нажмите для добавления украшения '3', \n"

<< "Для занесения продажи в базу '4', \n"

<< "Для записи расходов '5': \n";

ch = getaChar();

switch (ch)

{

//экраны ввода существуют только во время их

//использования

case '3': ptrProductInputScreen =

new ProductInputScreen(ptrProductList);

ptrProductInputScreen->setProduct();

delete ptrProductInputScreen;

break;

case '4': ptrSellInputScreen =

new SellInputScreen(ptrProductList, ptrSell);

ptrSellInputScreen->setPlata();

delete ptrSellInputScreen;

break;

case '5': ptrExpenseInputScreen =

new ExpenseInputScreen(ptrExpenseRecord);

ptrExpenseInputScreen->setExpense();

delete ptrExpenseInputScreen;

break;

default: cout << "Неизвестная функция\n";

break;

} // конец секции switch

} // конец условия if

else if (ch == '2') // вывод данных

{

cout << "Нажмите для вывода украшений '3', \n"

<< "Для вывода списка продаж '4' \n"

<< "Для вывода расходов '5', \n"

<< "Для вывода годового отчета '6': \n";

ch = getaChar();

switch (ch)

{

case '3': ptrProductList->display();

break;

case '4': ptrSell->display();

break;

case '5': ptrExpenseRecord->display();

break;

case '6':

ptrAnnualReport = new AnnualReport(ptrSell,

ptrExpenseRecord);

ptrAnnualReport->display();

delete ptrAnnualReport;

break;

default: cout << "Неизвестная функция вывода\n";

break;

} // конец switch

} // конец elseif

else if (ch == '0')

return; // выход

else

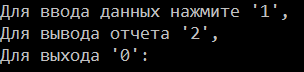
cout << "Неизвестная функция. Нажимайте только '1', '2' или '0'\n";

} // конец while

}

15.Руководство пользователя

При запуске программы пользователя встречает меню

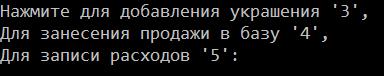


Его средствами можно:

1.Внести данные(Добавить новое украшение в базу, занести продажи или расходы)

Рассмотрим данный раздел программы подробнее:

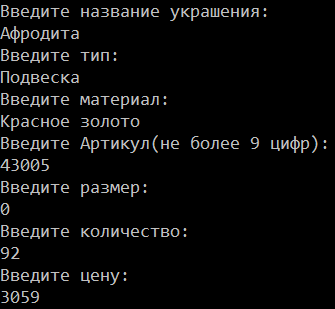
При выборе этого варианта появляется меню выбора добавления данных:



Рассмотрим детально каждый пункт меню:

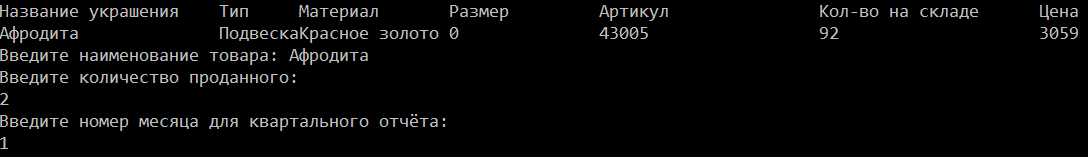
3.При выборе данного пункта пользователю будет предложено ввести: название, тип, материал, Артикул, размер, количество на складе и цену нового товара, для того чтобы занести его в базу.

Ниже представлен пример добавления в базу нового украшения:

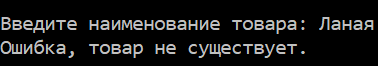


4.При выборе данного пункта пользователю будет предложено занести продажу в базу, для этого он, ориентируясь на меню товаров должен ввести: название нужного товара, количество проданных единиц и месяц в котором была совершена сделка.

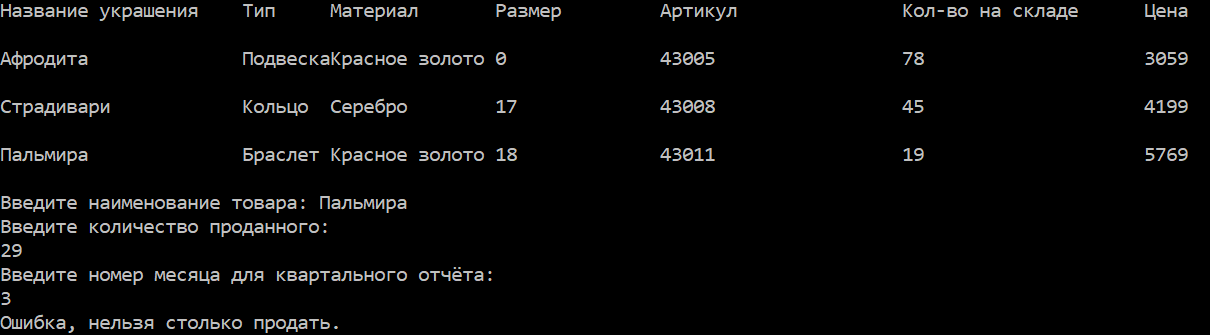
Ниже представлен пример занесения продажи в базу:



Если пользователь введет название товара, который не числится на складе программа выдаст ошибку:

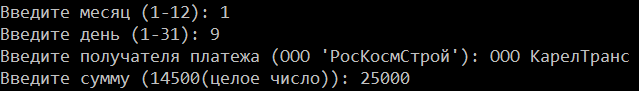


Если пользователь попробует списать больше товаров, чем есть на складе, то программа выдаст ошибку:



5.При выборе данного пункта пользователю будет предложено занести в базу статью расходов фирмы, для этого от него потребуется ввести: дату оплаты, введя месяц, а после число, название фирмы получателя и оплаченную сумму.

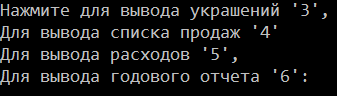
Ниже представлен пример занесения статьи расходов в базу:



2.Вывести отчеты (Вывод списка украшений, продаж, расходов и годового отчета)

Рассмотрим данный раздел программы подробнее:

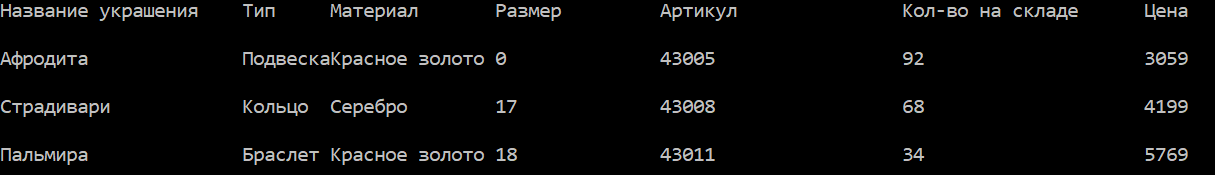
При выборе этого варианта появляется меню выбора отчётов:



Рассмотрим детально каждый пункт меню:

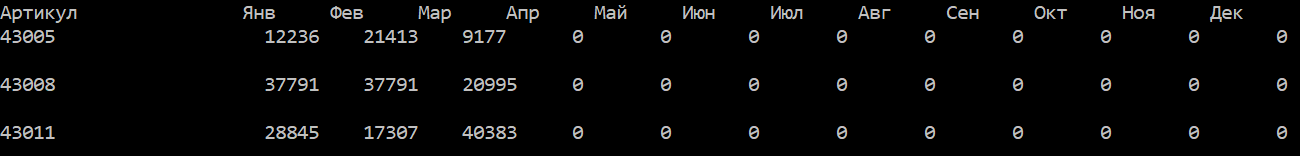
3.При выборе данного пункта пользователю на экран будет выведен список активных товаров

Ниже представлен пример такого списка:



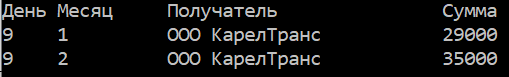
4.При выборе данного пункта пользователю на экран будет выведен список продаж

Ниже представлен пример такого списка:



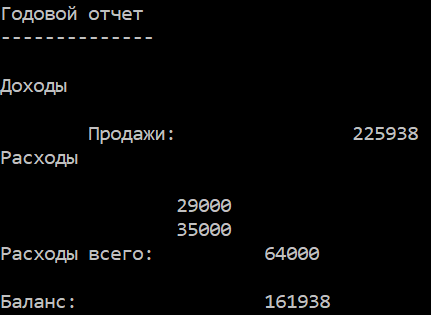
5.При выборе данного пункта пользователю на экран будет выведен список расходов

Ниже представлен пример такого списка:



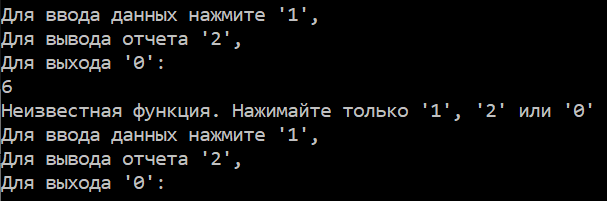
6.При выборе данного пункта пользователю на экран будет выведен годовой отчёт, который отразит насколько фирма оказалась в плюсе или минусе за прошедший год

Ниже представлен пример такого отчёта:



3.Выйти из программы

Если пользователь нажмет клавишу, которая не задействуется в данном разделе программы, то появится ошибка:

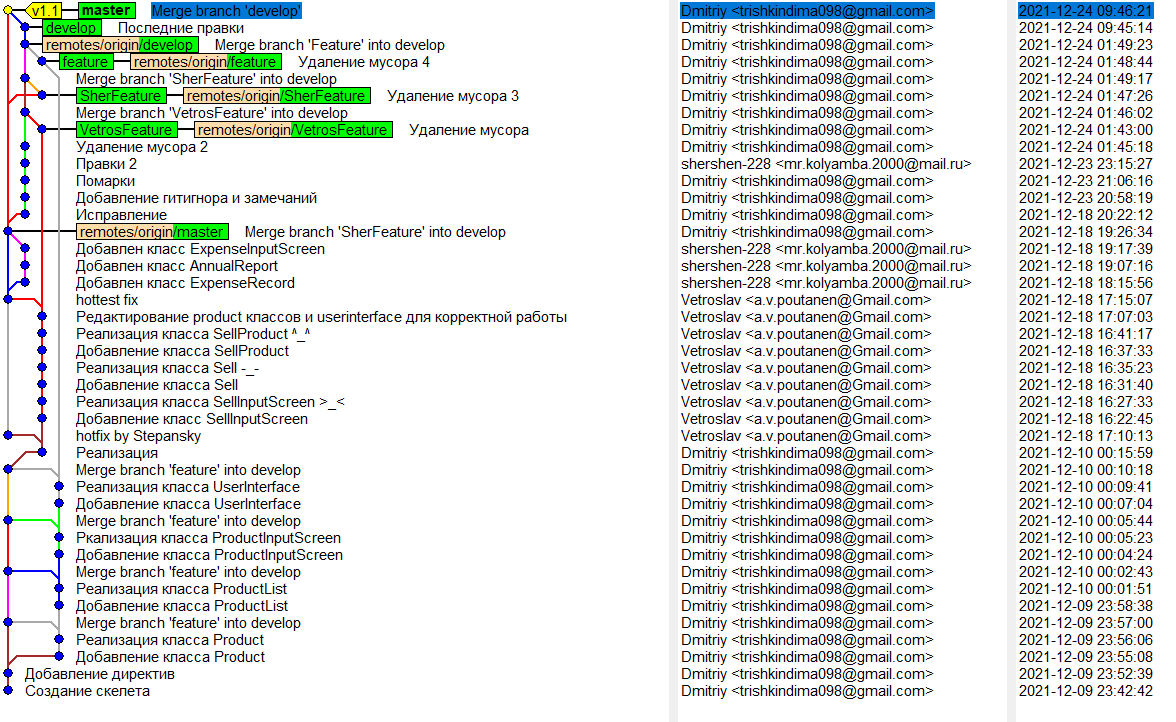


**Система контроля версий**

Репозиторий:

<https://github.com/Dm1try1/Final>

Графическая интерпретация истории через “gitk”



**Пояснения к коммитам:**

**Тришкин Д.С.**

“Создание скелета”-создал проект в VisualStudio и добавил заголовочные файлы UserInterface.h, ProductList.h, ProductInputScreen.h, Product.h, и исполнительные UserInterface.cpp, ProductList.cpp, ProductInputScreen.cpp, Product.cpp, Main.cpp.

“Добавление директив”- создал ветку develop, добавил в каждый файл директивы.

“Добавление класса Product”- создал класс, добавил поля и методы.

“Реализация класса Product”- реализовал функции.

“Слияние feature c develop” - слил ветки.

“Добавление класса ProductList”- создал класс, добавил поля и методы.

“Реализация класса ProductList”- реализовал функции.

“Слияние feature c develop”-слил ветки.

“Добавление класса ProductInputScreen”- создал класс, добавил поля и методы.

“Реализация класса ProductInputScreen”-реализовал функции.

“Слияние feature c develop”-слил ветки.

“Добавление класса UserInterface”- создал класс, добавил поля и методы.

“Реализация класса UserInterface”-реализовал функции.

“Слияние feature c develop”-слил ветки.

“Слияние develop c master”-слил ветки, добавил tag v1.0 и назвал Рабочая версия..

“Слияние SherFeature c develop”-слил ветки.

“Исправление”- исправил описание.

“Добавление гитигнора и замечаний”- добавил гитигнор и поменял комментарии.

“Помарки”-сделал помарку для ввода(указал чтоб вводили целое число).

“Слияние develop c master”-слил ветки, добавил tag v1.1 и назвал Реализация.

“Удаление мусора(…)”- удалил ненужные файлы.

“Отчет”-добавил отчет в репозиторий.

**Поутанен Степан Андреевич и его коммиты:**

Создал ветку VetrosFeature в которой и работал

“hotfix by Stepansky”- исправление мелких недочетов для корректной работы программы

“Добавление класс SellInputScreen”-Добавлен заголовочного файла SellInputScreen.h для класса SellInputScreen в котором описал атрибуты и функции которые будет использовать использовать исполнительный файл

“Реализация класса SellInputScreen>\_<”-Создал исполнительный файл SellInputScreen.cpp в котором реализовал функции из заголовочного файла, используя описанные переменные, на выходе получив класс отвечающий за внесения продажи товаров в базу.

“Добавление класса Sell”-Добавлен заголовочного файла Sell.h для класса Sell в котором описал атрибуты и функции которые будет использовать использовать исполнительный файл

“Реализация класса Sell-\_-”-Создал исполнительный файл Sell.cpp в котором реализовал функции из заголовочного файла, используя описанные переменные, на выходе получив класс отвечающий за вывод списка продаж.

“Добавление класса SellProduct “-Добавлен заголовочного файла SellProduct.h для класса SellProduct в котором описал атрибуты и функции которые будет использовать использовать исполнительный файл

“Реализация класса SellProduct^\_^”-Создал исполнительный файл SellProduct.cpp в котором реализовал функции из заголовочного файла, используя описанные переменные, на выходе получив класс отвечающий за хранение списка продаж.

“hottestfix”-Редактирование product классов и userinterface для корректной работы программы

После этого слил ветку VetrosFeature с веткой develop

**Коммиты Смирнова Николая:**

“Добавлен класс ExpenceRecord” - создал и добавил класс ExpenceRecord и заголовочный файл Expence.h для записи расходов, добавил поля и методы

“Добавлен класс AnnualReport” - создал и добавил класс AnnualReport для годового отчёта, добавил поля и методы

“Добавлен класс ExpenseInputScreen” - создал и добавил класс ExpenseInputScreen для вывода расходов, добавил поля и методы

“Правки 2” - исправление комментариев и оптимизация переменных

**Вывод**: В ходе выполнения командного проекта нами был разработан сценарий, по которому нам необходимо разработать программу для контроля работы ювелирного магазина, составлены диаграммы вариантов использования, классов и деятельности при помощи объектно-ориентированной среды проектирования WhiteStarUML, составлен план разработки и написана программа, реализующая все фукнции, описанные в ТЗ. В ходе разработки активно использовалась система контроля версий Git, с помощью которой каждый из нас привносил изменения в программу. Программа была написана на языке программирования C++ с использованием объектно-ориентированного подхода. В качестве среды программирования была выбрана VisualStudio. В написанной нами программе отсутствуют неиспользуемые переменные, реализована очистка памяти. Наблюдаются сбои программы при вводе некорректных значений (в случае ввода букв вместо цифр). Цель командного проекта была достигнута, ПО для контроля работы ювелирного магазина разработана и работает.