МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

“Петрозаводский государственный университет”

Физико-технический институт

Командный проект

Дисциплина: «Технология программирования»

Тема: **«**Магазин наручных часов**»**

Выполнили студенты:

3 курса, группы 21318

Петкун Ю.А.

Коханович А.С.

Бровко Д.Р.

Научный руководитель:

Кандидат физико-математических наук

Бульба А.В

Петрозаводск 2020

Цель работы:   
Написать программу для ведению учета продаж, а также распечатыванию отчетов о финансовой деятельности магазина продажи наручных часов.

Кратко о программной реализации:   
Cреда разработки, язык, перечисление всех созданных вами заголовочных и единиц компиляции. Кратко поясняется назначение каждого из перечисленных файлов.

Программа реализовывалась на языке C++ в среде разработке Visual Studio 13. Нами были созданы заголовочные файлы и единицы компиляции:  
1. Product.h - Заголовочный файл товаров

2. Income.h - Заголовочный файл доходов

3. Expenses.h - Заголовочный файл расходов

4. function.h - Заголовочный файл функций

5. Struct.h - Заголовочный файл структур

6. Product.cpp - Исполняемый файл продуктов

7. Income.cpp - Исполняемый файл доходов

8. Expenses.cpp - Исполняемый файл расходов

9. Function.cpp - Исполняемый файл функций

10. main.cpp - Главный исполняемый файл

Пошаговое описание процесса разработки:

1. Краткое описание сюжета.

К нам пришел заказчик по имени Иванов Иван Иванович — мелкий собственник, в его собственности находится небольшое здание в городе “N”, в котором расположен магазин наручных часов, где он самостоятельно занимается их продажей. Он хочет, чтобы мы написали программу, которая упростила бы его нелегкий труд по ведению учета продаж, а также распечатыванию отчетов о своей финансовой деятельности, с учетом, что у него слабый компьютер.

2. Приблизительный вид таблиц.

Рукописные формы

На данный момент Иван Иванович записывает всю информацию о своем деле вручную в гроссбух. Он показывает нам, какие формы используются для ведения дел:

* Данные о товарах;
* доходы от продаж;
* расходы;
* годовой отчет.

В данных о товарах содержатся id, наименования часов, их количество, вид и стоимость. Таким образом, данные о товарах состоят из пяти строк.

Доходы от продаж. Здесь хранятся записи о продажах часов. В этой таблице

содержится 12 столбцов (по числу месяцев) и одна строка общей суммы продаж за месяц. Всякий раз, получая деньги от покупателя, Иванов прибавляет уплаченную сумму в соответствующую ячейку таблицы, которая показана на рис. 1.

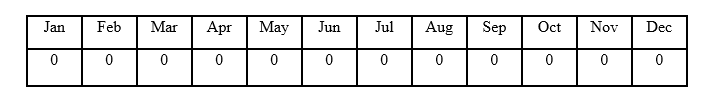


рис.1 Таблица доходов

В таблице Расходы записаны исходящие платежи. Она содержит такие столбцы как: месяц, получатель (компания или человек, на чье имя осуществляется платеж) и сумма платежа. Кроме того, есть столбец, в который Иванов вносит виды или категории платежей: ЖКХ, ремонт, налог на недвижимость, страховка, реклама, ремонт, закупка.  
Таблица расходов показана на рис. 2.

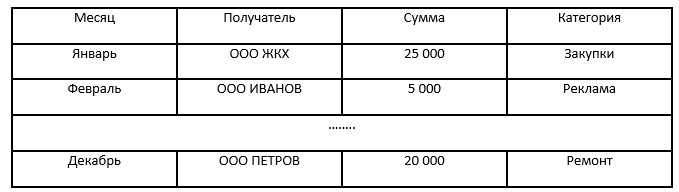


Рис.2 Таблица расходов

В годовом отчете (рис. 3) используется информация из таблицы доходов, таблицы расходов и таблицы данных о товарах для подсчета сумм, пришедших за год от клиентов и уплаченных в процессе ведения бизнеса, а также остатков товаров. Суммируются все прибыли от всех покупателей за все месяца. Также суммируются все расходы.

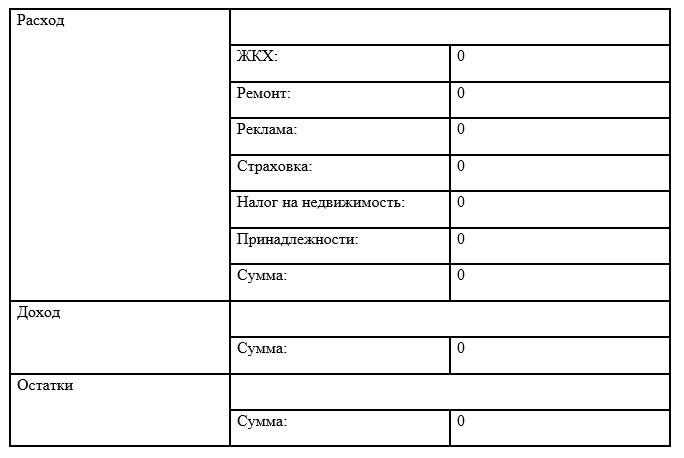


Рис.3 Годовой отчет

Годовой отчет Иван Иванович составляет только в конце года, когда все доходы и расходы декабря уже известны. Программа будет выводить частичный годовой отчет в любое время дня и ночи за время, прошедшее с начала года.

Собственно говоря, основные задачи программы можно определить

как ввод данных, вывод данных, редактирование данных, а также вывод отчетов.

3. Действующие субъекты (actors) и их функциональные обязанности.

В нашей системе И. И. Иванов является заказчиком и конечным пользователем системы.

Все варианты использования отображены на диаграмме. Владелецу магазина потребуется выполнять следующие действия:

* начать работу с программой;
* добавить товар;
* закупить товар;
* ввести сумму оплаты в таблицу доходов;
* ввести значения в таблицу расходов;
* вывести список товаров;
* вывести таблицу доходов от продаж;
* вывести таблицу расходов;
* вывести годовой отчет.

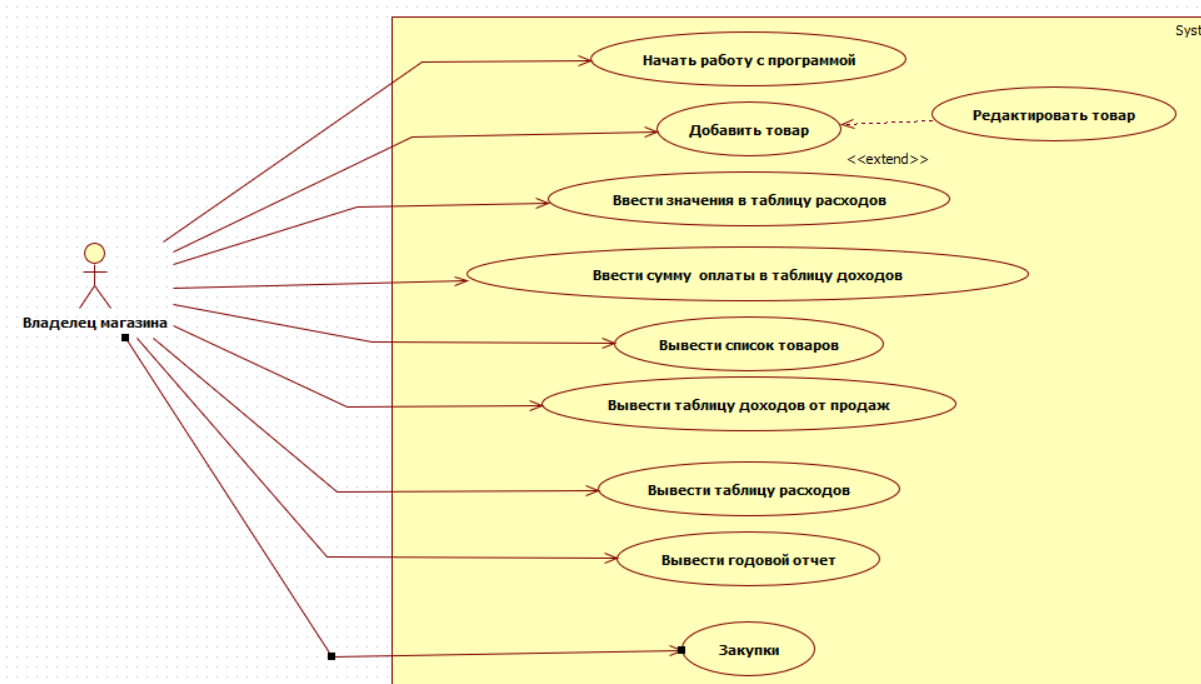
4. Список вариантов использования.  


Рис. 4 Список вариантов использования

5. Тексты сценария к вариантам использования.

“Начать программу”

Когда запускается программа, на экран должно выводиться меню, из которого пользователь может выбрать нужное действие. Это может называться экраном пользовательского интерфейса.

“Добавить товар”

На экране должно отобразиться сообщение, в котором программа просит пользователя ввести название товара, вид, количество и его цену. Эта информация должна заноситься в таблицу.

“Ввести сумму оплаты в таблицу доходов”

Экран ввода оплаты содержит сообщение, из которого пользователь узнает, что ему необходимо ввести имя товара, месяц оплаты, количество. Сумма денег рассчитывается автоматически.

Программа просматривает таблицу доходов от продаж находит соответствующую

запись в таблице и указанная сумма прибавляется в столбец того месяца, в котором производилась покупка.

“Ввести значение в таблицу расходов”

Экран ввода расхода должен содержать приглашение пользователю на ввод имени

получатель (компания или человек, на чье имя осуществляется платеж или если это убыток то прочерк ) и сумма платежа, месяц и категорию. Затем программа создает новую строку, содержащую эту информацию, и вставляет ее в таблицу расходов.

“Вывести список товаров”

Программа выводит на экран список товаров, каждая строка списка состоит из четырех полей: наименование товара, вид, количество и цена.

“Вывести таблицу доходов”

Каждая строка таблицы, которую выводит программа, состоит из названия товара и

значения суммы продаж в месяц.

“Вывести таблицу расходов”

Каждая строка таблицы, которую выводит программа, состоит из значений месяца, получателя, суммы и категории платежа.

“Вывести годовой отчет”

Программа выводит годовой отчет, состоящий из: суммарного дохода за прошедший год, списка всех расходов с указанием бюджетной категории, общей суммы расходов, результирующего годового баланса.

“Закупки”

Экран ввода закупок содержит сообщение, из которого пользователь узнает, что ему необходимо ввести поставщика, имя товара, вид, месяц оплаты, количество и цену.  
Программа заносит данные в таблицу расходов с пометкой закупка, а также если это новый товар, то он добавляется в список товаров.

6. Диаграммы действий

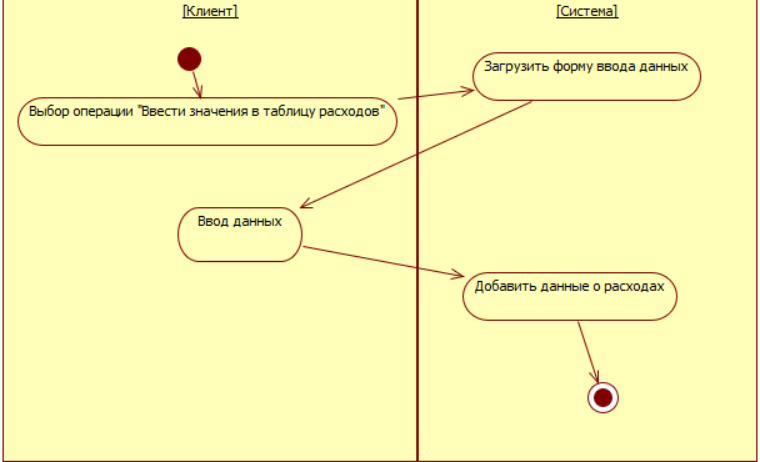
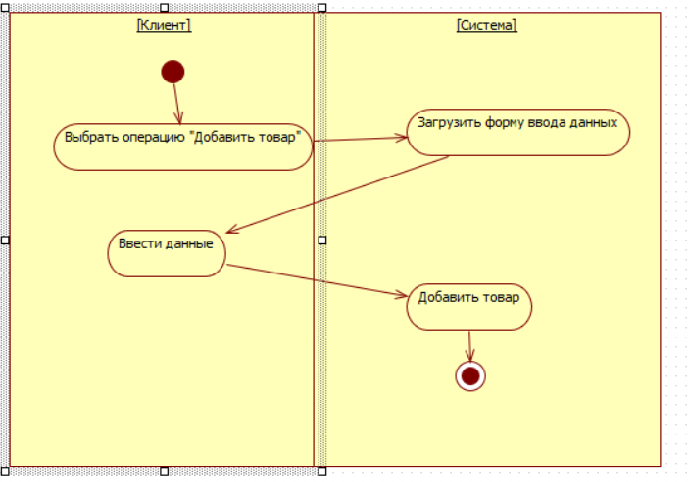


Рис.5 Диаграмма деятельности для операции “Ввести данные в таблицу расходов”

  
Рис.6 Диаграмма деятельности для операции “Добавить товар”

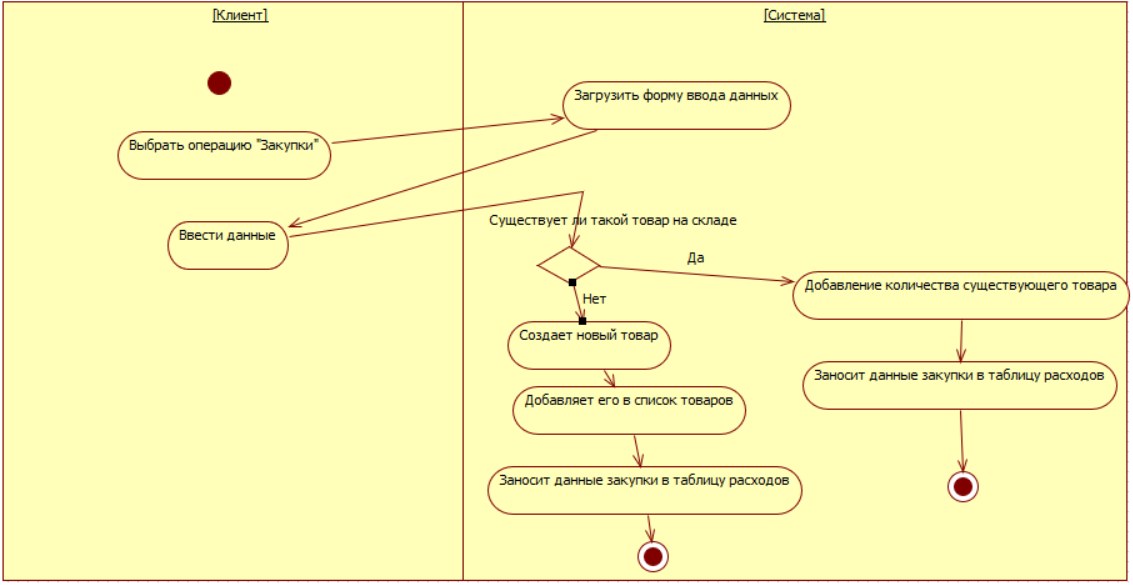


Рис.7 Диаграмма деятельности для операции “Закупки”

7. Предварительный список существительных, которые, возможно, станут классами программы.

1. Интерфейс.

2. Товар.

3. Экран ввода товаров.

4. Наименование товара.

5. Цена товара.

6. Количество товара.

7. Список товаров.

8. Суммарные расходы по категориям.

9. Экран ввода оплаты товара.

10. Месяц.

11. Сумма проданных товаров.

12. Таблица доходов.

13. Суммарные продажи.

14. Расход.

15. Экран ввода расходов.

16. Получатель.

17. Размер платежа.

18. Годовой отчет.

19. Категория.

20. Строка в таблице расходов.

21. Таблица расходов.

22. Вид.

23. Отчеты.

24. Доход.

8. Необходимы список существительных.

1. Расходы.

2. Товар.

3. Доходы.

9. По каждому классу составьте список атрибутов с пояснением предназначения

каждого. Список атрибутов приложите к отчету.

Список атрибутов для класса Товары:  
1. Указатель на структуру данных о товаре.

2. Id.

Список атрибутов для структуры данных о товаре:  
1. Id  
2. Количество товара

3. Цена товара

4. Наименование товара  
5. Вид товара

Список атрибутов для класса Доходы:  
1. Указатель на структуру данных о доходе  
2. Id

Список атрибутов для структуры данных о доходе:  
1-12. Суммы по месяцам

Список атрибутов для класса Расходы:  
1. Указатель на структуру данных о расходах

Список атрибутов для структуры данных о расходах:

1. Дата(Месяц)

2. Получатель  
3. Сумма  
4. Категория

10. Сообщения, которыми будут обмениваться классы.  
Методы класса товары:

1. Копирование объектов товаров - В уровне доступа public есть указатель на структуру объектов товары, который позже очистится. Нужен для копирования объектов в динамический массив структуры “товары”.

2. Рекопирование объектов товары - Нужна для копирования в основное поле объектов структуры, которая находится в уровне доступа private, измененных данных о товарах. Очистка временного указателя на структуру объектов.

3. Добавить объект структуры - выделение памяти под новый объект структуры.

4. Добавление структуры для временного указателя на объекты товары - выделение памяти под новый объект структуры.

5. Добавить продукт - Создается объект структуры с данными о товаре.

6. Вывод - выводит список и данные о товарах.

7. Редактировать - Изменить данные о товарах.

8. Остатки - Нужен для подсчета остатков, данные о которых вносятся в годовой отчет.

Методы класса доходы

1. Добавить доход - Создание объекта структуры и добавление данных в таблицу доходов.

2. Добавить объект структуры - нужен для выделения памяти под новый объект структуры.

3. Вывод - Вывод таблицы доходов.

4. Сумма - Нужна для вычисления общей суммы доходов, которая будет использоваться в годовом отчете.

Методы класса расходы  
1. Добавить объект структуры - Нужен для выделения памяти под новый объект структуры.

2. Добавить расход - Создание объекта структуры расход и добавление данных о нем в таблицу расходы.

3. Вывод - вывод таблицы расходов.

4. Закупки - Добавление новых товаров, а также добавление данных о расходе в таблицу расходов.

5. Годовой отчет - вывод таблицы годовой отчет.  
6. Данные отчета - формирует данные для годового отчета.

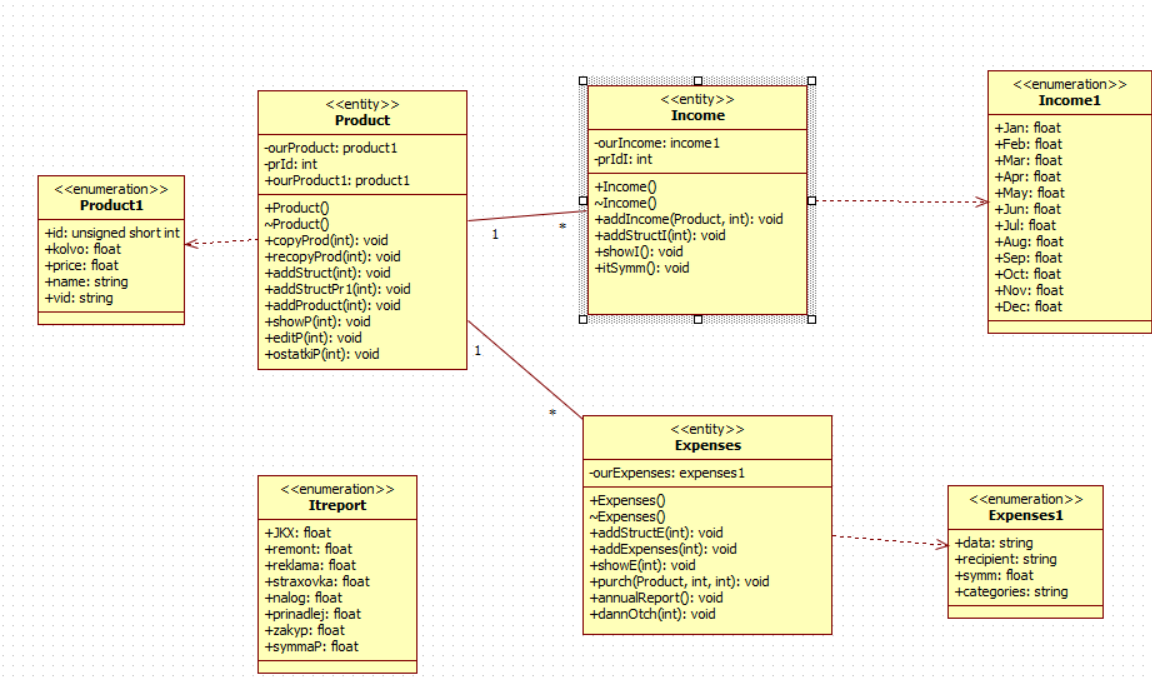
11. Диаграмма классов. 

Рис.8 Диаграмма классов.

12. Диаграммы последовательностей UML

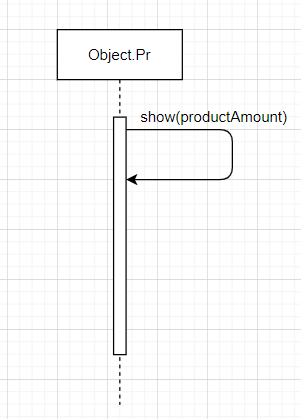


Рис.9 Диаграмма последовательностей для варианта использования “Вывод списка товаров”.

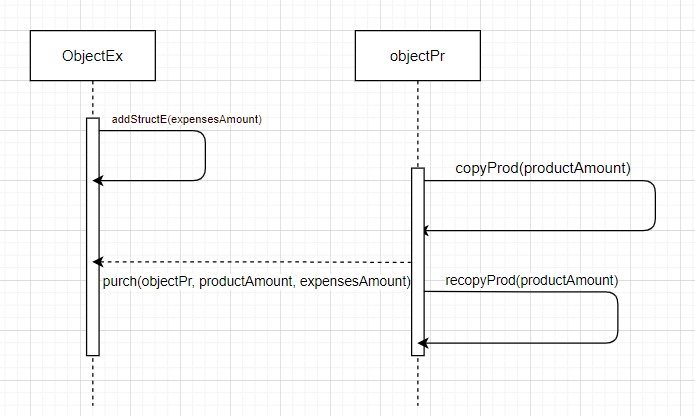


Рис.10 Диаграмма последовательностей для варианта использования “Закупки”.

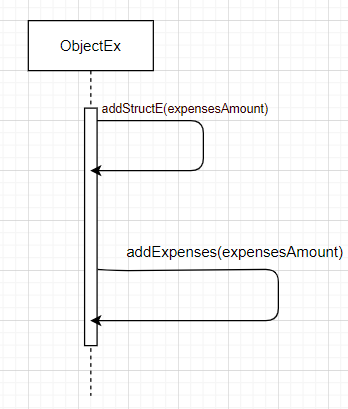


Рис.11 Диаграмма последовательностей для варианта использования “Ввести значения в таблицу расходов”.

13. Написание кода программы.

*Заголовочные файлы (.h)*

**Product.h**

#ifndef \_\_PRODUCT\_H\_\_

#define \_\_PRODUCT\_H\_\_

#include "Struct.h"

using namespace std;

class Product // Товар

{

private:

int prId; // id товара

product1\* ourProduct; // Указатель на структуру товары

public:

product1\* ourProduct1; // Временный указатель на структуру, который очистится

Product(); // Конструктор

~Product(); // Деструктор

void copyProd(int); // Копирование объектов во временный указатель

void recopyProd(int); // Присваивание измененных данных объектов структур

void addStruct(int); // Выделение памяти под объект структуры

void addStructPr1(int); // Выделение памяти под времены объект структуры

void addProduct(int); // Создание объекта товара

void showP(int); // Вывод товара

void editP(int); // Редактирование товара

void ostatkiP(int); // Сумма остатков

};

#endif

**Income.h**

#ifndef \_\_INCOME\_H\_\_

#define \_\_INCOME\_H\_\_

#include "Struct.h"

#include "Product.h"

using namespace std;

class Income // Доход

{

private:

income1\* ourIncome; // Указатель на структуру доходы

int prIdI; // id объекта дохода

public:

Income();

~Income();

void addIncome(Product&, int); // Создание объекта доходы

void addStructI(int); // Выделение памяти под объект дохода

void showI(); // Вывод таблицы доходов

void itSymm(); // Общая сумма доходов

};

#endif

**Expenses.h**

#ifndef \_\_EXPENSES\_H\_\_

#define \_\_EXPENSES\_H\_\_

#include "Struct.h"

#include "Product.h"

using namespace std;

class Expenses // Расход

{

private:

expenses1\* ourExpenses; // Указатель на структуру расходы

public:

Expenses();

~Expenses();

void addStructE(int); // Выделить память под объект структуры товара

void addExpenses(int); // Создать объект структуры товара

void showE(int); // Вывод таблицы расходы

void purch(Product&, int&, int); // Закупки

void annualReport(); // Вывод годового отчета

void dannOtch(int); // Формирование данных для годового отчета

};

#endif

**Struct.h**

#ifndef \_\_STRUCT\_H\_\_

#define \_\_STRUCT\_H\_\_

#include <string>

using namespace std;

struct itreport // Структура данных категорий расходов

{

float JKX; // ЖКХ

float remont; // Ремонт

float reklama; // Реклама

float straxovka; // Страховка

float nalog; // Налог на недвижимость

float prinadlej; // Принадлежности

float zakyp; // Закупка

float symmaP; // Общая сумма доходов

};

struct product1 // Структура данных о товарах

{

unsigned short int id;

float kolvo; // Количество товара

float price; // Цена товара

string name; // Наименование товара

string vid; // Вид товара

};

struct income1 // Структура данных о доходах

{

float Jan = 0, Feb = 0, Mar = 0, Apr = 0, May = 0, Jun = 0, Jul = 0, Aug = 0, Sep = 0, Oct = 0, Nov = 0, Dec = 0; //Сумма по месяцам

};

struct expenses1 // Структура данных о расходах

{

string data; // Месяц

string recipient; // Получатель

float symm; // Сумма расхода

string categories; // Категория расходов

};

#endif

**function.h**

#ifndef \_\_function\_H\_\_

#define \_\_function\_H\_\_

void menu();

void punkt5();

#endif

14. Исполняемые файлы (.cpp).

**Expenses.cpp**

#include "Expenses.h"

#include <iostream>

#include <iomanip>

#include "Struct.h"

using namespace std;

itreport report; // Глобальный объект структуры

extern float insymm; // Глобальная переменная общей суммы доходов

extern float ostatki; // Глобальная переменная остатка товаров

void Expenses::dannOtch(int amountEx) // Заполнение объекта структуры данных суммы категорий расхода

{

report.JKX = 0;

report.remont = 0;

report.reklama = 0;

report.straxovka = 0;

report.nalog = 0;

report.prinadlej = 0;

report.zakyp = 0;

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

{

if (ourExpenses[i].categories == "ЖКХ")

{

report.JKX = ourExpenses[i].symm + report.JKX;

}

if (ourExpenses[i].categories == "Ремонт")

{

report.remont = ourExpenses[i].symm + report.remont;

}

if (ourExpenses[i].categories == "Реклама")

{

report.reklama = ourExpenses[i].symm + report.reklama;

}

if (ourExpenses[i].categories == "Страховка")

{

report.straxovka = ourExpenses[i].symm + report.straxovka;

}

if (ourExpenses[i].categories == "Налог на недвижимость")

{

report.nalog = ourExpenses[i].symm + report.nalog;

}

if (ourExpenses[i].categories == "Принадлежности")

{

report.prinadlej = ourExpenses[i].symm + report.prinadlej;

}

if (ourExpenses[i].categories == "Закупки")

{

report.zakyp = ourExpenses[i].symm + report.zakyp;

}

report.symmaP = report.JKX + report.remont + report.reklama + report.straxovka + report.nalog + report.prinadlej;

}

}

Expenses::Expenses() // Конструктор

{

ourExpenses = 0; // Присваивает указателю на структуру нулевое значение

}

Expenses::~Expenses() // Деструктор

{

delete[] ourExpenses;

}

void Expenses::addStructE(int amount) // Выделение памяти под объект структуры

{

if (amount == 0)

{

ourExpenses = new expenses1[amount + 1]; // выделение памяти для первой структуры

}

else

{

expenses1\* tempObjE = new expenses1[amount + 1];

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

{

tempObjE[i] = ourExpenses[i]; // копируем во временный объект

}

delete[] ourExpenses;

ourExpenses = tempObjE;

}

}

void Expenses::addExpenses(int amount) // Добавление данных в объект структуры

{

int date;

int categ;

cout << "Месяц:" << endl;

cout << "1. Январь" << endl;

cout << "2. Февраль" << endl;

cout << "3. Март" << endl;

cout << "4. Апрель" << endl;

cout << "5. Май" << endl;

cout << "6. Июнь" << endl;

cout << "7. Июль" << endl;

cout << "8. Август" << endl;

cout << "9. Сентябрь" << endl;

cout << "10. Октябрь" << endl;

cout << "11. Ноябрь" << endl;

cout << "12. Декабрь" << endl;

cout << "Выберите месяц: ";

cin >> date;

switch (date)

{

case 1:

{

ourExpenses[amount].data = "Январь";

break;

}

case 2:

{

ourExpenses[amount].data = "Февраль";

break;

}

case 3:

{

ourExpenses[amount].data = "Март";

break;

}

case 4:

{

ourExpenses[amount].data = "Апрель";

break;

}

case 5:

{

ourExpenses[amount].data = "Май";

break;

}

case 6:

{

ourExpenses[amount].data = "Июнь";

break;

}

case 7:

{

ourExpenses[amount].data = "Июль";

break;

}

case 8:

{

ourExpenses[amount].data = "Август";

break;

}

case 9:

{

ourExpenses[amount].data = "Сентябрь";

break;

}

case 10:

{

ourExpenses[amount].data = "Октябрь";

break;

}

case 11:

{

ourExpenses[amount].data = "Ноябрь";

break;

}

case 12:

{

ourExpenses[amount].data = "Декабрь";

break;

}

system("pause");

default:

cout << "Месяц не найден" << endl;

return;

break;

}

cout << endl;

cout << "Введите получателя: ";

cin >> ourExpenses[amount].recipient;

cout << "Введите сумму расхода: ";

cin >> ourExpenses[amount].symm;

cout << "Выберите категорию: " << endl;

cout << "1. ЖКХ" << endl;

cout << "2. Налог на недвижимость" << endl;

cout << "3. Страховка" << endl;

cout << "4. Принадлежности" << endl;

cout << "5. Ремонт" << endl;

cout << "6. Реклама" << endl;

cout << "Введите номер категорию: ";

cin >> categ;

switch (categ)

{

case 1:

{

ourExpenses[amount].categories = "ЖКХ";

break;

}

case 2:

{

ourExpenses[amount].categories = "Налог на недвижимость";

break;

}

case 3:

{

ourExpenses[amount].categories = "Страховка";

break;

}

case 4:

{

ourExpenses[amount].categories = "Принадлежности";

break;

}

case 5:

{

ourExpenses[amount].categories = "Ремонт";

break;

}

case 6:

{

ourExpenses[amount].categories = "Реклама";

break;

}

system("pause");

default:

cout << "Категория не найдена" << endl;

break;

}

}

void Expenses::showE(int amount) // Вывод таблицы расходов

{

cout << setw(10) << "Месяц " << setw(25) << "Получатель" << setw(25) << "Сумма" << setw(25) << "Категории" << endl;

cout << "--------------------------------------------------------------------------------------" << endl;

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

{

cout << setw(10) << ourExpenses[i].data << setw(25) << ourExpenses[i].recipient << setw(25) << ourExpenses[i].symm << setw(25) << ourExpenses[i].categories << endl;

cout << "--------------------------------------------------------------------------------------" << endl;

}

}

void Expenses::purch(Product& objectPr, int& amountPr, int amountEx) // Закупка товаров

{

string pName;

int pKolvo;

int date;

bool proverka = true;

cout << "Введите название товара: ";

cin >> pName;

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

{

if (pName == objectPr.ourProduct1[i].name) // Проверка, существует ли товар

{

cout << "Введите количество: ";

cin >> pKolvo;

objectPr.ourProduct1[i].kolvo = objectPr.ourProduct1[i].kolvo + pKolvo; // Добавляем количество товаров

proverka = false;

}

}

if (proverka == true) // Если переменная proverka = true, значит, что товара нет и мы создаем новый

{

objectPr.addStructPr1(amountPr); // Выделение памяти под временный объект структуры

objectPr.addStruct(amountPr); // Выделение памяти под основной объект структуры

objectPr.ourProduct1[amountPr].name = pName;

cout << "Введите вид: ";

cin >> objectPr.ourProduct1[amountPr].vid;

cout << "Введите количество: ";

cin >> objectPr.ourProduct1[amountPr].kolvo;

cout << "Введите цену товара для продажи: ";

cin >> objectPr.ourProduct1[amountPr].price;

amountPr++;

}

cout << "Месяц:" << endl;

cout << "1. Январь" << endl;

cout << "2. Февраль" << endl;

cout << "3. Март" << endl;

cout << "4. Апрель" << endl;

cout << "5. Май" << endl;

cout << "6. Июнь" << endl;

cout << "7. Июль" << endl;

cout << "8. Август" << endl;

cout << "9. Сентябрь" << endl;

cout << "10. Октябрь" << endl;

cout << "11. Ноябрь" << endl;

cout << "12. Декабрь" << endl;

cout << "Выберите месяц: ";

cin >> date;

switch (date)

{

case 1:

{

ourExpenses[amountEx].data = "Январь";

break;

}

case 2:

{

ourExpenses[amountEx].data = "Февраль";

break;

}

case 3:

{

ourExpenses[amountEx].data = "Март";

break;

}

case 4:

{

ourExpenses[amountEx].data = "Апрель";

break;

}

case 5:

{

ourExpenses[amountEx].data = "Май";

break;

}

case 6:

{

ourExpenses[amountEx].data = "Июнь";

break;

}

case 7:

{

ourExpenses[amountEx].data = "Июль";

break;

}

case 8:

{

ourExpenses[amountEx].data = "Август";

break;

}

case 9:

{

ourExpenses[amountEx].data = "Сентябрь";

break;

}

case 10:

{

ourExpenses[amountEx].data = "Октябрь";

break;

}

case 11:

{

ourExpenses[amountEx].data = "Ноябрь";

break;

}

case 12:

{

ourExpenses[amountEx].data = "Декабрь";

break;

}

system("pause");

default:

cout << "Месяц не найден" << endl;

break;

}

cout << endl;

cout << "Введите получателя: ";

cin >> ourExpenses[amountEx].recipient;

cout << "Введите сумму расхода: ";

cin >> ourExpenses[amountEx].symm;

ourExpenses[amountEx].categories = "Закупки";

}

void Expenses::annualReport() // Вывод годового отчета

{

cout << "Расход" << endl;

cout << " ЖКХ: " << report.JKX << endl;

cout << " Ремонт: " << report.remont << endl;

cout << " Реклама: " << report.reklama << endl;

cout << " Страховка: " << report.straxovka << endl;

cout << " Налог на недвижимость: " << report.nalog << endl;

cout << " Принадлежности: " << report.prinadlej << endl;

cout << " Сумма: " << report.symmaP << endl;

cout << "Доход" << endl;

cout << " Сумма: " << insymm << endl;

cout << "Остатки" << endl;

cout << " Сумма: " << ostatki << endl;

}

**Income.cpp**

#include "Income.h"

#include <iostream>

#include <iomanip>

#include "Struct.h"

using namespace std;

float insymm;

Income::Income() // Конструктор

{

ourIncome = 0; // Присваиваем указателю на структуру нулевое значение

prIdI = 0;

}

Income::~Income()

{

delete[] ourIncome;

}

void Income::addStructI(int amount) // Выделяем память под объект структуры

{

ourIncome = new income1[amount + 1]; // выделение памяти для первой структуры

}

void Income::addIncome(Product &objectPr, int amount) // Заполнение объекта структуры

{

int mes, idtov;

float kolvotov;

float symm;

bool proverka = true;

objectPr.showP(amount);

cout << "Введите id товара: ";

cin >> idtov;

cout << "Введите количество товара: ";

cin >> kolvotov;

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

{

if (idtov == objectPr.ourProduct1[i].id) // Существует ли такой товар

{

if (kolvotov > objectPr.ourProduct1[i].kolvo) // Хватает ли количества товаров

{

cout << "Недостаточно товара" << endl;

return;

}

else {

objectPr.ourProduct1[i].kolvo = objectPr.ourProduct1[i].kolvo - kolvotov; // Изменение количества товара

symm = kolvotov \* objectPr.ourProduct1[i].price; // Авторасчет суммы продаваемого товара

cout << "Сумма дохода: " << symm << endl;

proverka = false;

cout << endl;

cout << "1. Январь" << endl;

cout << "2. Февраль" << endl;

cout << "3. Март" << endl;

cout << "4. Апрель" << endl;

cout << "5. Май" << endl;

cout << "6. Июнь" << endl;

cout << "7. Июль" << endl;

cout << "8. Август" << endl;

cout << "9. Сентябрь" << endl;

cout << "10. Октябрь" << endl;

cout << "11. Ноябрь" << endl;

cout << "12. Декабрь" << endl;

cout << "Выберите месяц: ";

cin >> mes;

switch (mes)

{

case 1:

{

ourIncome[prIdI].Jan = symm + ourIncome[prIdI].Jan;

break;

}

case 2:

{

ourIncome[prIdI].Feb = symm + ourIncome[prIdI].Feb;

break;

}

case 3:

{

ourIncome[prIdI].Mar = symm + ourIncome[prIdI].Mar;

break;

}

case 4:

{

ourIncome[prIdI].Apr = symm + ourIncome[prIdI].Apr;

break;

}

case 5:

{

ourIncome[prIdI].May = symm + ourIncome[prIdI].May;

break;

}

case 6:

{

ourIncome[prIdI].Jun = symm + ourIncome[prIdI].Jun;

break;

}

case 7:

{

ourIncome[prIdI].Jul = symm + ourIncome[prIdI].Jul;

break;

}

case 8:

{

ourIncome[prIdI].Aug = symm + ourIncome[prIdI].Aug;

break;

}

case 9:

{

ourIncome[prIdI].Sep = symm + ourIncome[prIdI].Sep;

break;

}

case 10:

{

ourIncome[prIdI].Oct = symm + ourIncome[prIdI].Oct;

break;

}

case 11:

{

ourIncome[prIdI].Nov = symm + ourIncome[prIdI].Nov;

break;

}

case 12:

{

ourIncome[prIdI].Dec = symm + ourIncome[prIdI].Dec;

break;

}

system("pause");

default:

cout << "Месяц не найден" << endl;

break;

}

}

}

}

if (proverka == true)

{

cout << "id товара не найден" << endl;

}

}

void Income::showI() // Вывод таблицы доходов

{

cout << setw(10) << "Jan" << setw(10) << "Feb" << setw(10) << "Mar" << setw(10) << "Apr" << setw(10) << "May" << setw(10) << "Jun" << setw(10) << "Jul" << setw(10) << "Aug" << setw(10) << "Sep" << setw(10) << "Oct" << setw(10) << "Nov" << setw(10) << "Dec" << endl;

cout << endl;

cout << endl;

cout << setw(10) << ourIncome[prIdI].Jan << setw(10) << ourIncome[prIdI].Feb << setw(10) << ourIncome[prIdI].Mar << setw(10) << ourIncome[prIdI].Apr << setw(10) << ourIncome[prIdI].May << setw(10) << ourIncome[prIdI].Jun << setw(10) << ourIncome[prIdI].Jul << setw(10) << ourIncome[prIdI].Aug << setw(10) << ourIncome[prIdI].Sep << setw(10) << ourIncome[prIdI].Oct << setw(10) << ourIncome[prIdI].Nov << setw(10) << ourIncome[prIdI].Dec << endl;

}

void Income::itSymm() // Общая сумма доходов

{

insymm = 0;

insymm = ourIncome[prIdI].Jan + ourIncome[prIdI].Feb + ourIncome[prIdI].Mar + ourIncome[prIdI].Apr + ourIncome[prIdI].May + ourIncome[prIdI].Jun + ourIncome[prIdI].Jul + ourIncome[prIdI].Aug + ourIncome[prIdI].Sep + ourIncome[prIdI].Oct + ourIncome[prIdI].Nov + ourIncome[prIdI].Dec;

}

**Product.cpp**

#include "Product.h"

#include <iostream>

#include "Struct.h"

using namespace std;

float ostatki;

Product::Product()

{

ourProduct = 0;

prId = 0;

}

Product::~Product()

{

delete[] ourProduct;

}

void Product::ostatkiP(int amount) // Сумма остатков товара

{

ostatki = 0;

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

{

ostatki = ostatki + (ourProduct[i].kolvo \* ourProduct[i].price);

}

}

void Product::copyProd(int amount) // Копирование данных основного объекта структуры во временный

{

ourProduct1 = new product1[amount];

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

{

ourProduct1[i].id = ourProduct[i].id;

ourProduct1[i].name= ourProduct[i].name;

ourProduct1[i].vid = ourProduct[i].vid;

ourProduct1[i].kolvo = ourProduct[i].kolvo;

ourProduct1[i].price = ourProduct[i].price;

}

}

void Product::recopyProd(int amount) // Копируем измененные данные временного объекта структуры в основной объект структуры

{

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

{

ourProduct[i].id = ourProduct1[i].id;

ourProduct[i].name = ourProduct1[i].name;

ourProduct[i].vid = ourProduct1[i].vid;

ourProduct[i].kolvo = ourProduct1[i].kolvo;

ourProduct[i].price = ourProduct1[i].price;

}

delete[] ourProduct1;

}

void Product::addStructPr1(int amount) // Выделение памяти под временный объект структуры

{

if (amount == 0)

{

ourProduct1 = new product1[amount + 1]; // выделение памяти для первой структуры

ourProduct1[amount].id = 0;

}

else

{

product1\* tempObjEx = new product1[amount + 1];

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

{

tempObjEx[i] = ourProduct1[i]; // копируем во временный объект

}

delete[] ourProduct1;

ourProduct1 = tempObjEx;

prId++;

ourProduct1[amount].id = prId;

}

}

void Product::addStruct(int amount) // Выделение памяти под основной объект структуры

{

if (amount == 0)

{

ourProduct = new product1[amount + 1]; // выделение памяти для первой структуры

ourProduct[amount].id = 0;

}

else

{

product1\* tempObj = new product1[amount + 1];

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

{

tempObj[i] = ourProduct[i]; // копируем во временный объект

}

delete[] ourProduct;

ourProduct = tempObj;

prId++;

ourProduct[amount].id = prId;

}

}

void Product::addProduct(int amount) // Заполняем объект структуры

{

cout << "Введите название товара: ";

cin >> ourProduct[amount].name;

cout << endl;

cout << "Введите вид товара: ";

cin >> ourProduct[amount].vid;

cout << endl;

cout << "Введите количество товарв: ";

cin >> ourProduct[amount].kolvo;

cout << endl;

cout << "Введите цену товара: ";

cin >> ourProduct[amount].price;

}

void Product::showP(int amount) // Вывод объектов структуры

{

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

{

cout

<< "id " << ourProduct[i].id << '\n'

<< "Наименование товара: " << ourProduct[i].name << '\n'

<< "Вид товара: " << ourProduct[i].vid << '\n'

<< "Количество товара: " << ourProduct[i].kolvo << '\n'

<< "Цена товара: " << ourProduct[i].price << '\n'

<< "=================================\n";

}

}

void Product::editP(int amount) // Редактирование объета структуры

{

int idred, punct;

cout << "|---------------------------|" << endl;

cout << "|Пункты редактирования: |" << endl;

cout << "|---------------------------|" << endl;

cout << "|1. Наименование |" << endl;

cout << "|---------------------------|" << endl;

cout << "|2. Вид |" << endl;

cout << "|---------------------------|" << endl;

cout << "|3. Количество |" << endl;

cout << "|---------------------------|" << endl;

cout << "|4. Цена |" << endl;

cout << "|---------------------------|" << endl;

cout << "|0. Вернуться в меню |" << endl;

cout << "|---------------------------|" << endl;

cout << "Введите пункт, который хотите отредактировать у товара: ";

cin >> punct;

system("cls");

switch (punct)

{

case 1:

{

string rename;

bool proverka = true;

showP(amount);

cout << "Введите id товара: ";

cin >> idred;

cout << endl;

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

{

if (idred == ourProduct[i].id) // Существует ли товар

{

cout << "Введите новое наименование товара: ";

cin >> rename;

ourProduct[i].name = rename;

cout << "Наименование изменено" << endl;

proverka = false;

}

}

if (proverka == true)

{

cout << "id не найден" << endl;

}

break;

}

case 2:

{

string revid;

bool proverka = true;

showP(amount);

cout << "Введите id товара: ";

cin >> idred;

cout << endl;

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

{

if (idred == ourProduct[i].id)

{

cout << "Введите новый вид товара: ";

cin >> revid;

ourProduct[i].vid = revid;

cout << "Вид товара изменен" << endl;

proverka = false;

}

}

if (proverka == true)

{

cout << "id не найден" << endl;

}

break;

}

case 3:

{

float rekolvo;

bool proverka = true;

showP(amount);

cout << "Введите id товара: ";

cin >> idred;

cout << endl;

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

{

if (idred == ourProduct[i].id)

{

cout << "Введите новое количество товара: ";

cin >> rekolvo;

ourProduct[i].kolvo = rekolvo;

cout << "Количество товара изменено" << endl;

proverka = false;

}

}

if (proverka == true)

{

cout << "id не найден" << endl;

}

break;

}

case 4:

{

float reprice;

bool proverka = true;

showP(amount);

cout << "Введите id товара: ";

cin >> idred;

cout << endl;

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

{

if (idred == ourProduct[i].id)

{

cout << "Введите новую цену товара: ";

cin >> reprice;

ourProduct[i].price = reprice;

cout << "Цена товара изменена" << endl;

proverka = false;

}

}

if (proverka == true)

{

cout << "id не найден" << endl;

}

break;

}

system("pause");

default:

cout << "Такого действия не существует" << endl;

break;

}

}

**Function.cpp**

#include "function.h"

#include <iostream>

using namespace std;

void menu() // Вывод основного меню программы

{

cout << "|-------------------------------------------------|" << endl;

cout << "|Операции |" << endl;

cout << "|-------------------------------------------------|" << endl;

cout << "| 1. Добавить товар |" << endl;

cout << "|-------------------------------------------------|" << endl;

cout << "| 2. Ввести значения в таблицу расходов |" << endl;

cout << "|-------------------------------------------------|" << endl;

cout << "| 3. Ввести сумму оплаты в таблицу доходов |" << endl;

cout << "|-------------------------------------------------|" << endl;

cout << "| 4. Закупки |" << endl;

cout << "|-------------------------------------------------|" << endl;

cout << "| 5. Вывести список товаров/Редактировать |" << endl;

cout << "|-------------------------------------------------|" << endl;

cout << "| 6. Вывести таблицу доходов |" << endl;

cout << "|-------------------------------------------------|" << endl;

cout << "| 7. Вывести таблицу расходов |" << endl;

cout << "|-------------------------------------------------|" << endl;

cout << "| 8. Вывести Годовой отчет |" << endl;

cout << "|-------------------------------------------------|" << endl;

cout << "| 0. Закрыть программу |" << endl;

cout << "|-------------------------------------------------|" << endl;

cout << "| Выберите операцию: ";

}

void punkt5() // Вывод списка действий с товарами

{

cout << "|---------------------------|" << endl;

cout << "|Действия: |" << endl;

cout << "|---------------------------|" << endl;

cout << "|1. Вывести список товаров |" << endl;

cout << "|---------------------------|" << endl;

cout << "|2. Редактировать товар |" << endl;

cout << "|---------------------------|" << endl;

cout << "|3. Удалить товар |" << endl;

cout << "|---------------------------|" << endl;

cout << "|0. Вернуться в меню |" << endl;

cout << "|---------------------------|" << endl;

cout << "|Выберите действие: ";

}

**main.cpp**

#include "Product.h"

#include "Income.h"

#include "Expenses.h"

#include "function.h"

#include <iostream>

#include <windows.h>

using namespace std;

int main()

{

setlocale(LC\_ALL, "rus");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int a;

Product objectPr; // Создаем объект класса Product

Income objectIn; // Создаем объект класса Income

Expenses objectEx; // Создаем объект класса Expenses

int productAmount = 0; // Присваиваем количество товара

int incomeAmount = 0;

int expensesAmount = 0;

objectIn.addStructI(incomeAmount);

do{ // Цикл меню программю

menu(); // Вызов функции, отвечающей выводу основного меню программы

cin >> a;

system("cls"); // Очистка экрана

switch (a)

{

case 1:

{

objectPr.addStruct(productAmount); // Вызов метода объекта класса Product

objectPr.addProduct(productAmount);

productAmount++;

break;

}

case 2:

objectEx.addStructE(expensesAmount); //Вызов метода объекта класса Expenses

objectEx.addExpenses(expensesAmount);

expensesAmount++;

break;

case 3:

objectPr.copyProd(productAmount);

objectIn.addIncome(objectPr, productAmount); //Вызов метода объекта класса Income

objectPr.recopyProd(productAmount);

break;

case 4:

objectEx.addStructE(expensesAmount);

objectPr.copyProd(productAmount);

objectEx.purch(objectPr, productAmount, expensesAmount);

objectPr.recopyProd(productAmount);

expensesAmount++;

break;

case 5:

{

int deistvie;

punkt5(); // Вызов функции, отвечающей за вывод меню действия с товарами

cin >> deistvie;

system("cls");

switch (deistvie)

{

case 1:

{

if (productAmount != 0){

objectPr.showP(productAmount);

}

else {

cout << "Нет товаров" << endl;

}

break;

}

case 2:

{

objectPr.editP(productAmount);

break;

}

case 0:

{

break;

}

system("pause");

default:

cout << "Такой операции не существует" << endl;

break;

}

}

break;

case 6:

objectIn.showI();

cout << endl;

break;

case 7:

objectEx.showE(expensesAmount);

break;

case 8:

objectEx.dannOtch(expensesAmount);

objectIn.itSymm();

objectPr.ostatkiP(productAmount);

objectEx.annualReport();

break;

case 0:

return 0;

break;

default:

cout << "Такой операции не существует" << endl;

break;

}

system("pause");

system("cls");

} while (a != 0); // Условие выхода из цикла

return 0;

}

15. Составьте руководство пользователя

Сначала пользователь запускает программу, после чего на экране появляется основное меню, где пользователь выбирает нужную ему операцию.

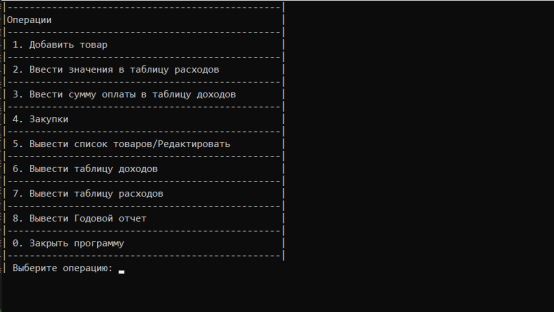


Рис.12 Основное меню

При выборе операции добавить товар, пользователю выводится окно ввода данных.

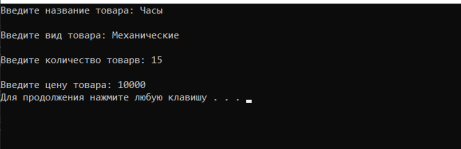


Рис.13 Добавление товара

При выборе операции ввести значения в таблицу расходов

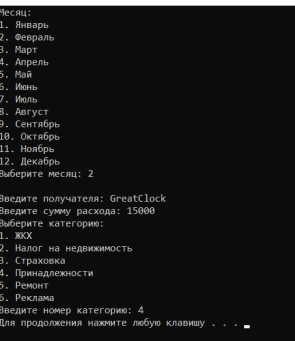


Рис.14 Ввести значения в таблицу расходов

При выборе операции ввести сумму оплаты в таблицу доходов

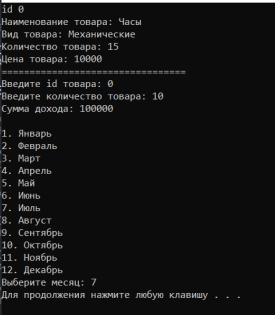


Рис.15 Ввести значения в таблицу доходов

При выборе операции закупки

Если товар существует

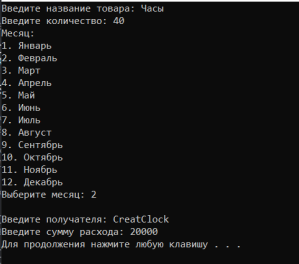


Рис.16 Закупки при существующем товаре

Если товар не существует

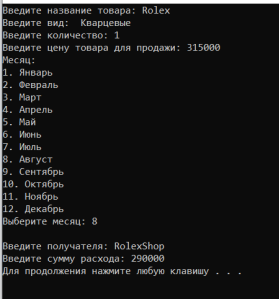


Рис.17 Закупки при не существующем товаре.

При выборе операции вывести список товаров/ редактировать

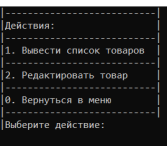


Рис.18 Операция вывести список товаров/ редактировать

Операция в пункте “вывести список товаров редактировать” - вывести список товаров

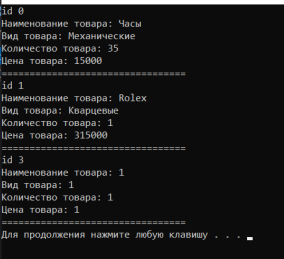


Рис.19 Список товаров.

Операция в пункте “вывести список товаров редактировать” - редактировать

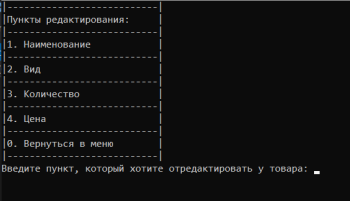


Рис.20 Меню редактирования товаров.

Операция наименование

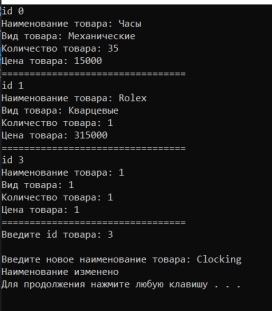


Рис.21 Редактирование наименования товара.

Остальные операции меню редактирования выглядят аналогично.

При выборе операции вывести таблицу доходов

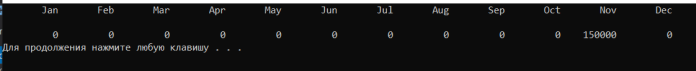


Рис.22 Таблица доходов.

При выборе операции вывести таблицу расходов

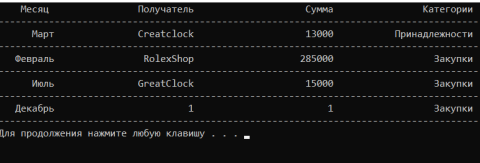


Рис.23 Таблица расходов.

При выборе операции годовой отчет

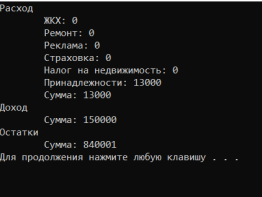


Рис.24 Годовой отчет.

При выборе операции закрыть программу, программа закрывается

Ссылка на репозиторий GitHub: https://github.com/AnTikO26/ClockShop

История проекта на GitHub

Ветка master :

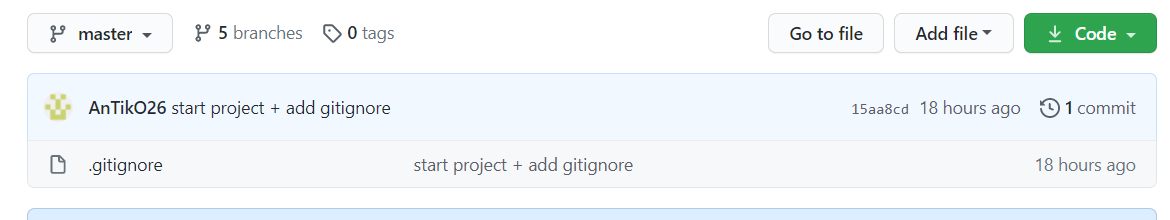


Рис. 25 Коммит ветки master

Создал Петкун Ю.А. Начало проекта и внесение файл gitignore

Ветка develop:

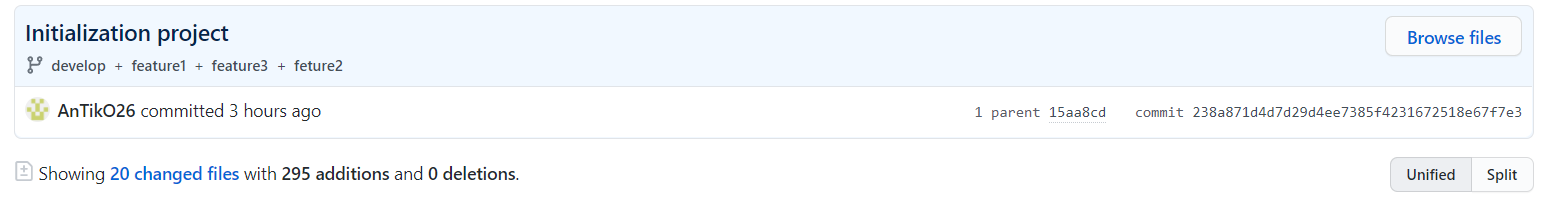


Рис. 26 Первый коммит ветки develop

Создал Петкун Ю.А. Инициализация проекта. Создание заголовочных и исполняемых файлов.

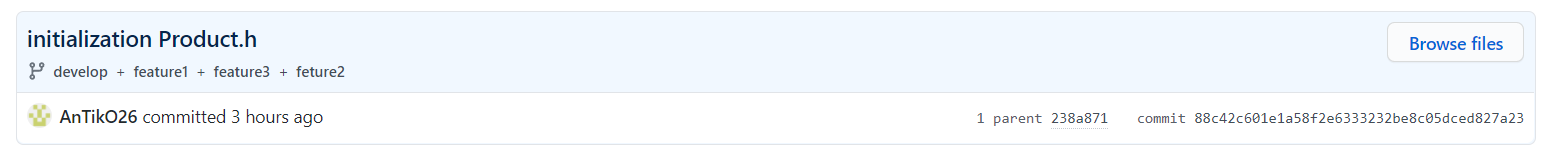


Рис. 27 Второй коммит ветки develop

Создал Петкун Ю.А. Инициализировал заголовочный файл Product.h. Объявил класс Product

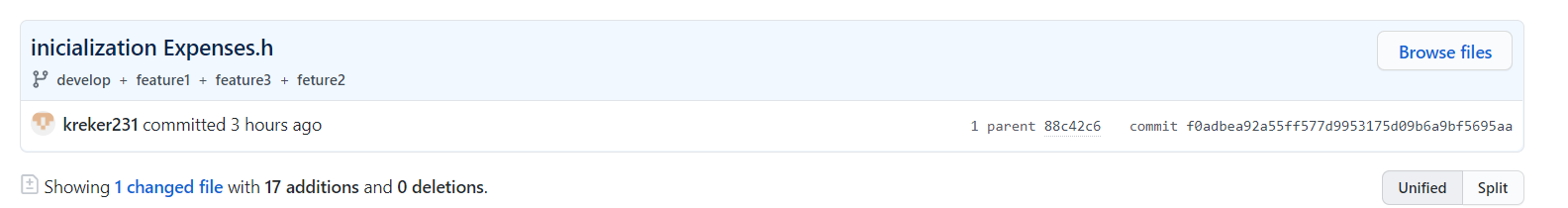


Рис 28. Третий коммит ветки develop

Создал Коханович А.С. Инициализировал заголовочный файл Expenses.h. Объявил класс Expenses

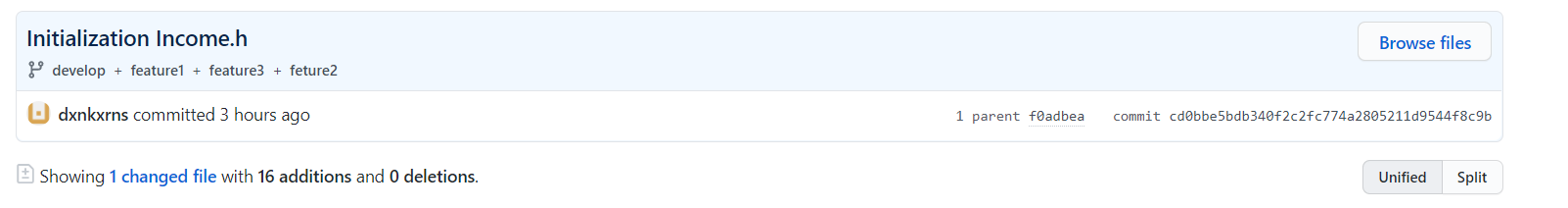


Рис. 29 Четвертый коммит ветки develop

Создал Бровко Д.Р. Инициализировал заголовочный файл Income.h. Объявил класс Income

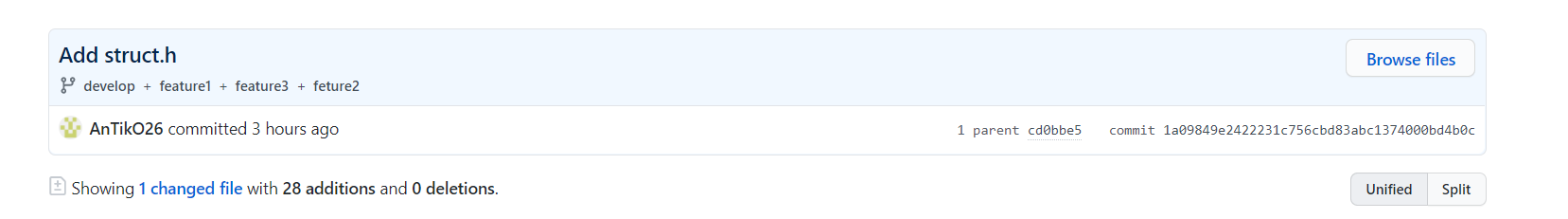


Рис.30 Пятый коммит ветки develop

Создал Петкун Ю.А. Добавил в заголовочный файл Struct.h. Объявил структуры: product1, income1, expenses1.

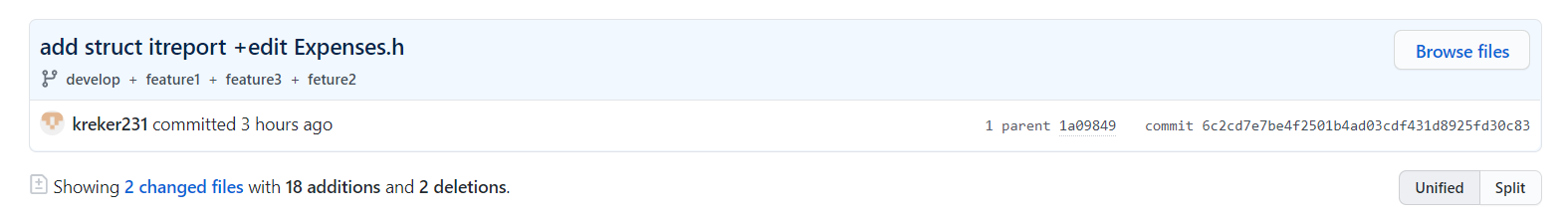


Рис 31. Шестой коммит ветки develop

Создал Коханович А.С. Объявил в заголовочном файле Struct.h структуру itreport. Объявил поля и методы класса Expenses

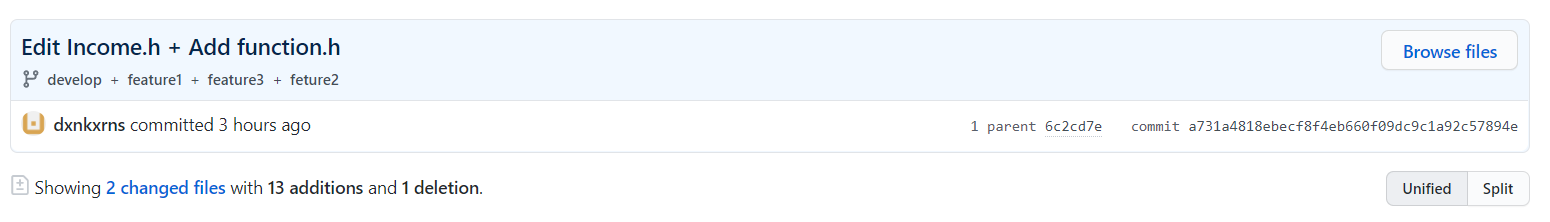


Рис 32. Седьмой коммит ветки develop

Создал Бровко Д.Р. Объявил поля и методы класса Income, объявил функции “Меню” программы

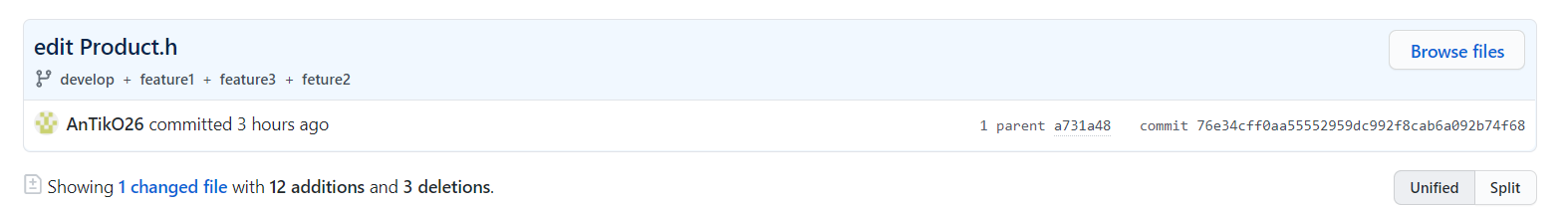


Рис. 33 Восьмой коммит ветки develop

Создал Петкун Ю.А. Объявил поля и методы класса Product

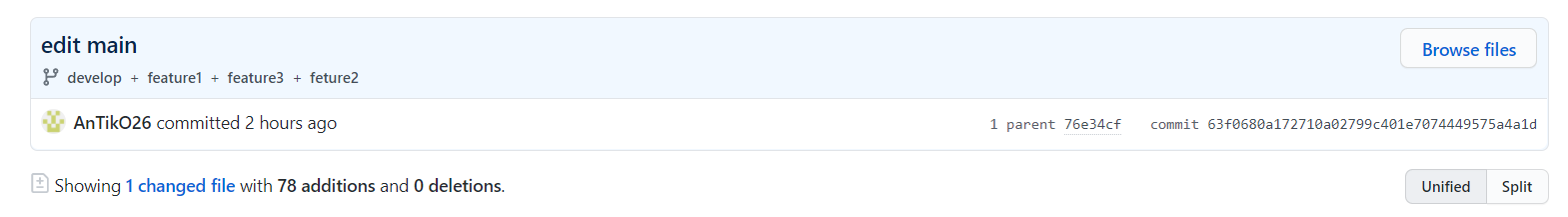


Рис. 34 Девятый коммит ветки develop

Создал Петкун Ю.А. Создал контейнер switch, подключил заголовочные файлы, определил переменные и создал по 1 объекту класса Product, Income. Expenses

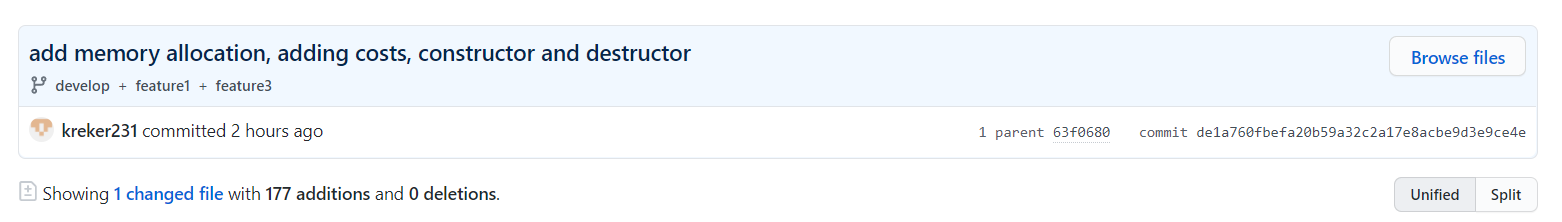


Рис. 35 Десятый коммит ветки develop

Создал Коханович А.С. Определил конструктор, деструктор, методы: addStruct, addExpenses

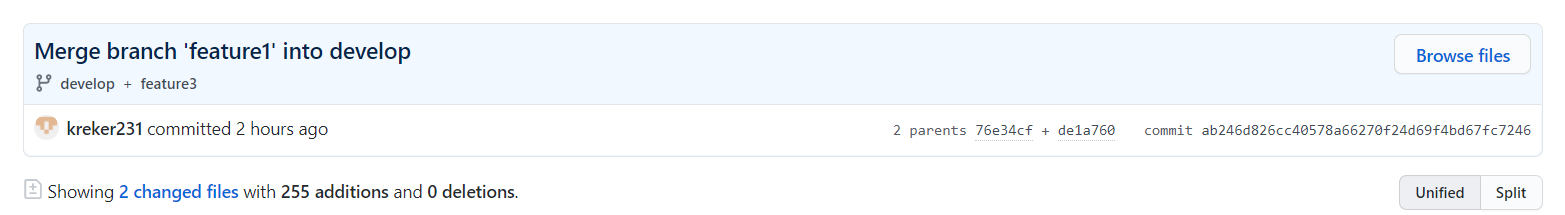


Рис. 36 Одиннадцатый коммит ветки develop

Создал Коханович А.С. Объединил ветку develop с feature1

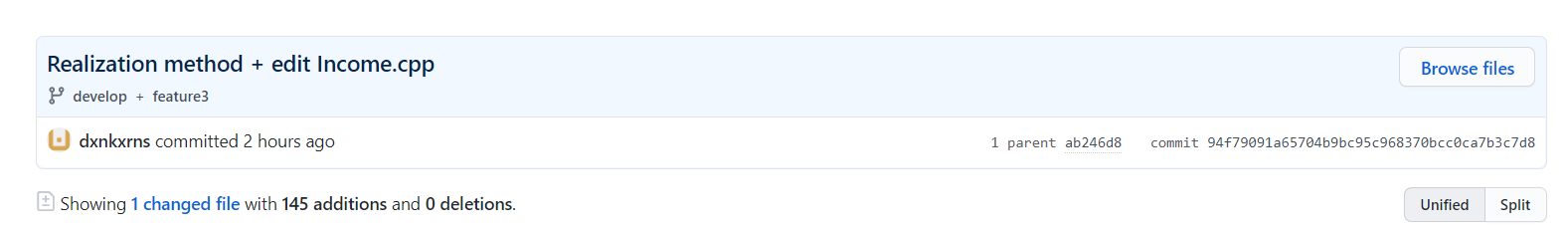


Рис. 37 Двенадцатый коммит ветки develop

Создал Бровко Д.Р. Определил конструктор, деструктор, методы: addStructI, addIncome

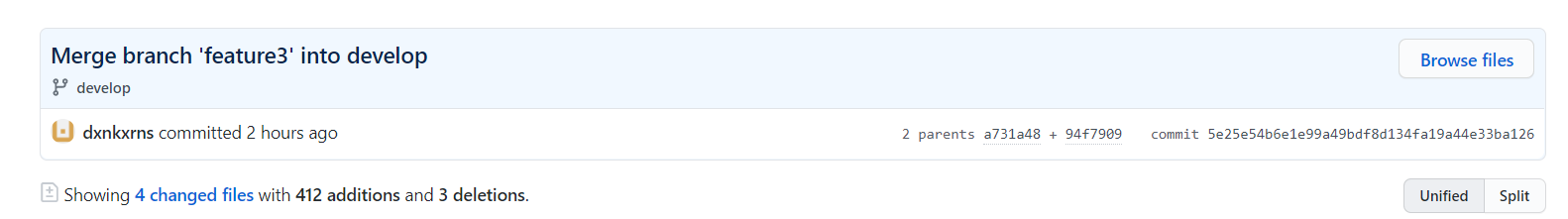


Рис. 38 Тринадцатый коммит ветки develop

Создал Бровко Д.Р. Объединил ветку develop с веткой feature2

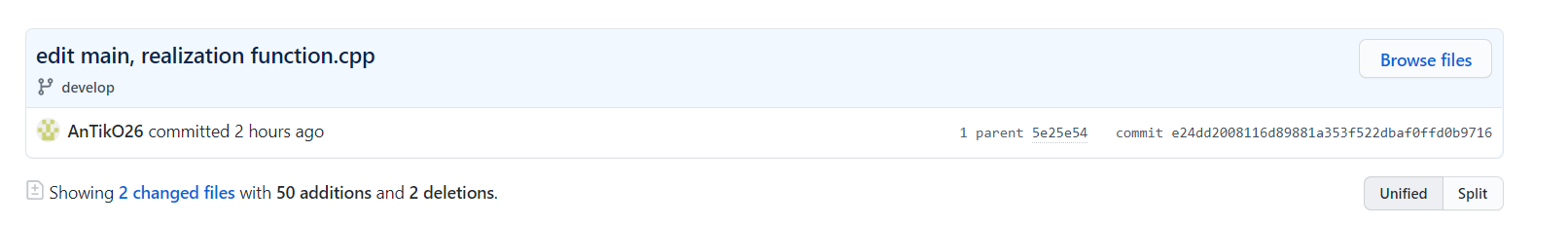


Рис. 39 Четырнадцатый коммит ветки develop

Создал Петкун Ю.А. Определил функции вывода меню. Реализовал работу операций “Добавить сумму в таблицу доходов” и “Добавить значения в таблицу расходов”

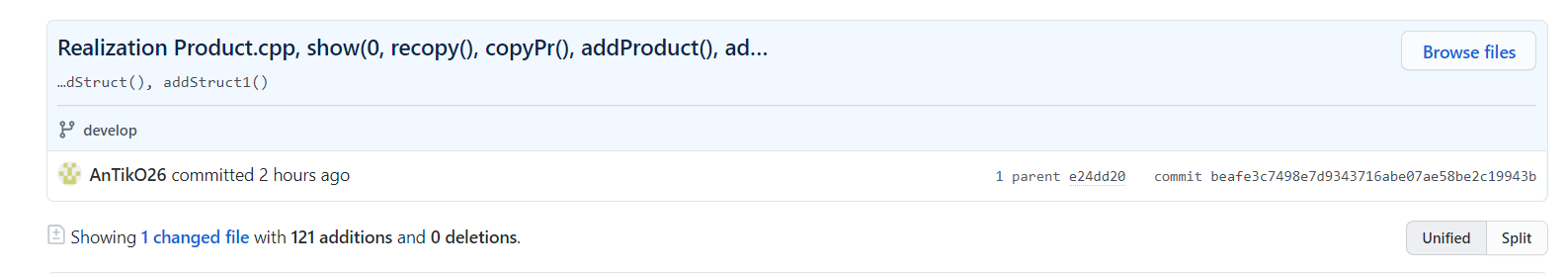


Рис. 40 Пятнадцатый коммит ветки develop

Создал Петкун Ю.А. Определил методы класса Product: show, recopy, copy, addProduct, addStruct, addStruct1. Определил конструктор и деструктор

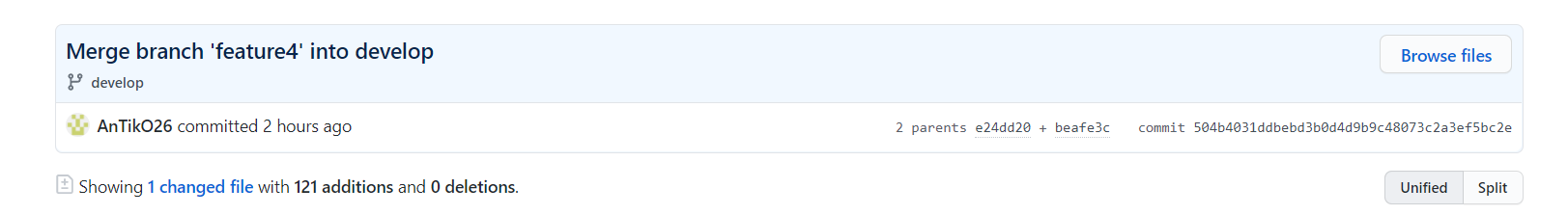


Рис. 41 Шестнадцатый коммит ветки develop

Создал Петкун Ю.А. Объединил ветку develop с веткой feature4

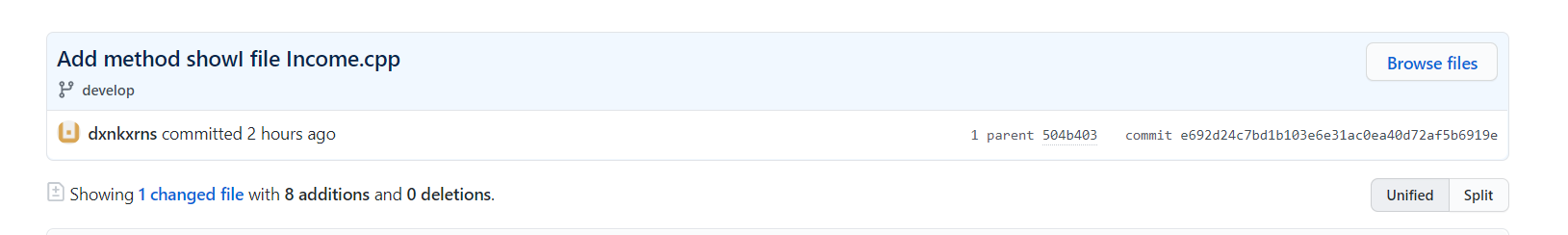


Рис. 42 Семнадцатый коммит ветки develop

Создал Бровко Д.Р. Определил метод showI

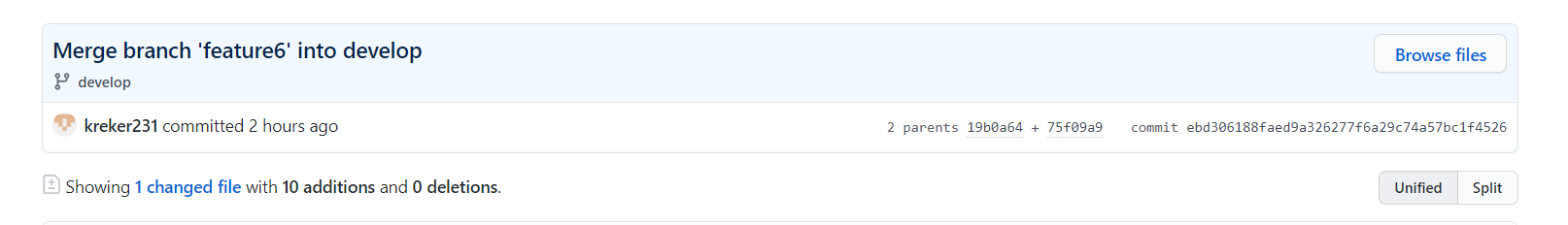


Рис. 43 Восемнадцатый коммит ветки develop

Создал Бровко Д.Р. Объединил ветку develop с feature6

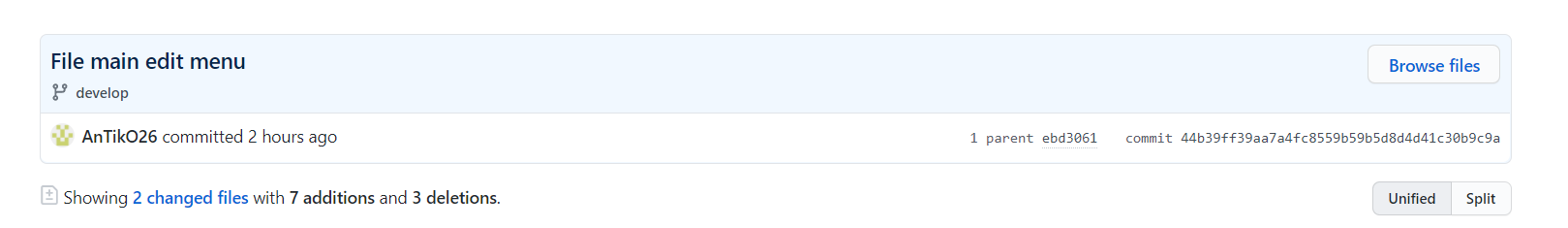


Рис. 44 Девятнадцатый коммит ветки develop

Создал Петкун Ю.А. Реализовал работу операций: “Добавить товар”, “Вывод таблицы расходов”, “Вывод таблицы доходов”

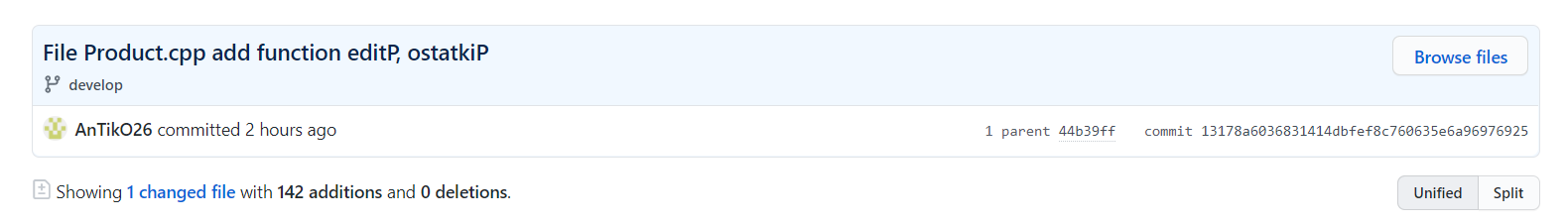


Рис. 45 Двадцатый коммит ветки develop

Создал Петкун Ю.А. Определил методы: editP, ostatkiP

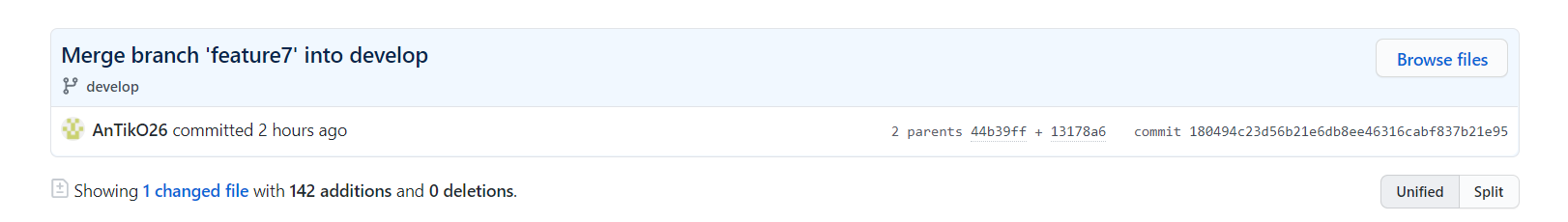


Рис. 46 Двадцать первый коммит ветки develop

Создал Петкун Ю.А. Объединил ветку develop с веткой feature7

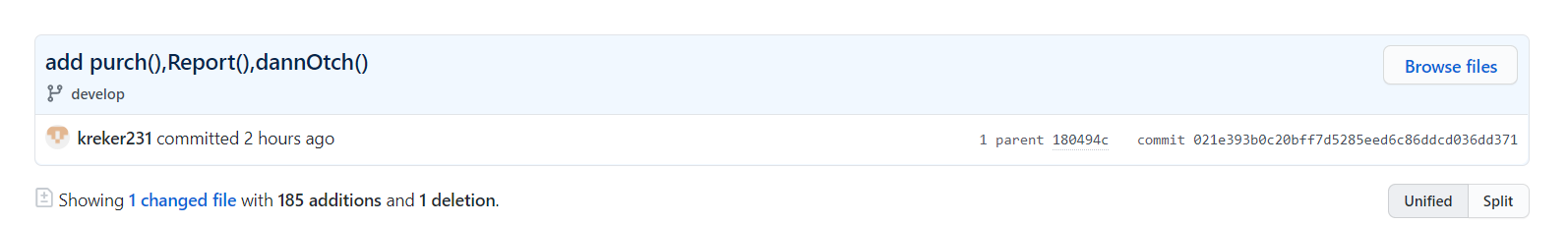


Рис. 47 Двадцать второй коммит ветки develop

Создал Коханович А.С. Определил методы: purch, Report, dannOtch

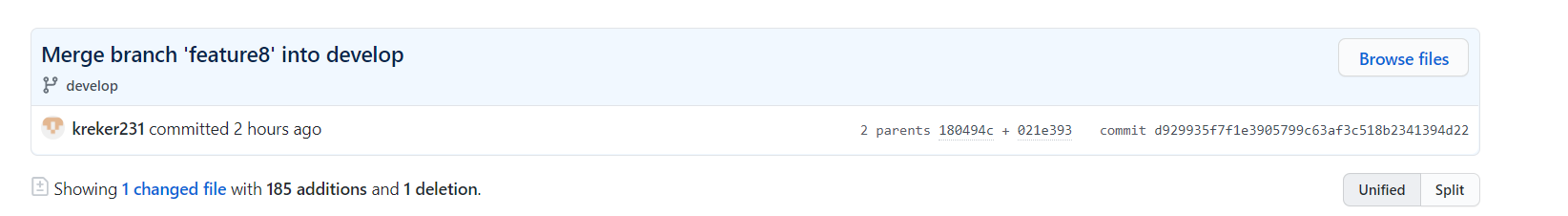


Рис. 48 Двадцать третий коммит ветки develop

Создал Коханович А.С. Объединил ветку develop с веткой feature8

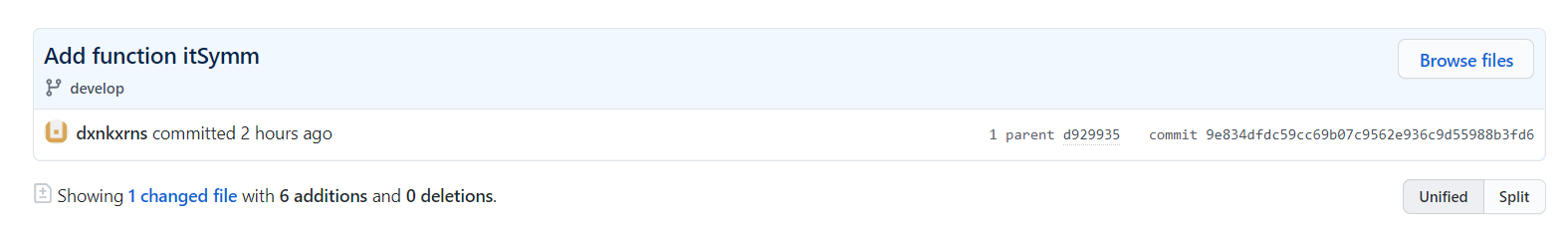


Рис. 49 Двадцать четвертый коммит ветки develop

Создал Бровко Д.Р. Определил метод itSymm

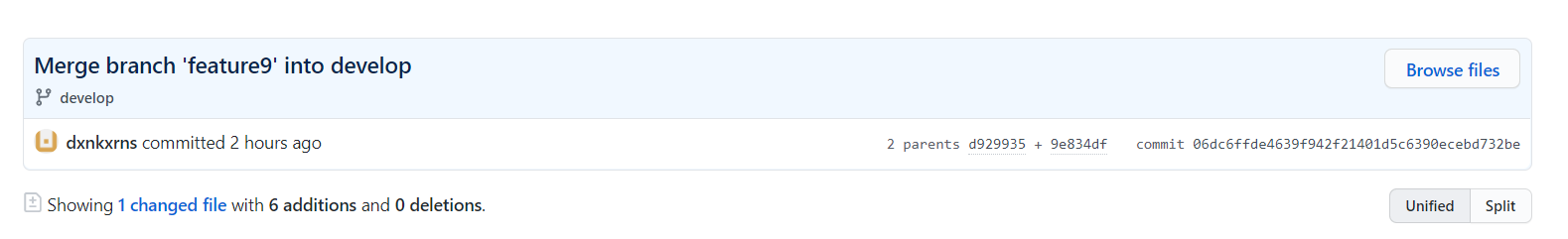


Рис. 50 Двадцать пятый коммит ветки develop

Создал Бровко Д.Р. Объединил ветку develop с веткой feature9

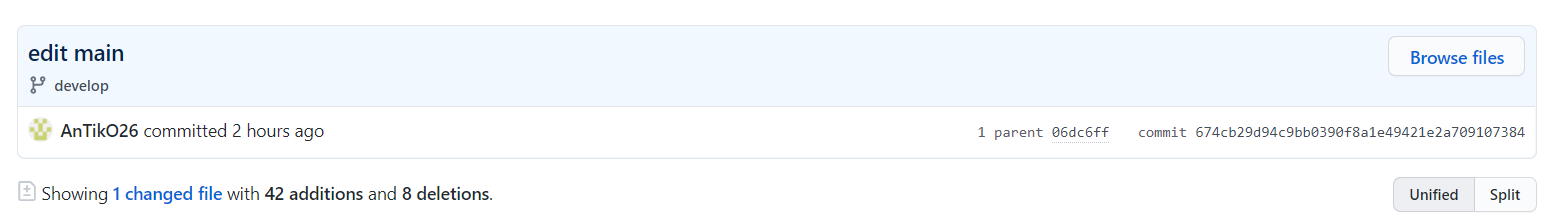


Рис. 51 Двадцать шестой коммит ветки develop

Создал Петкун Ю.А. Реализовал работу операций: “Закупки”, “Вывод годового отчеты”, “Вывести список товаров/Редактировать”

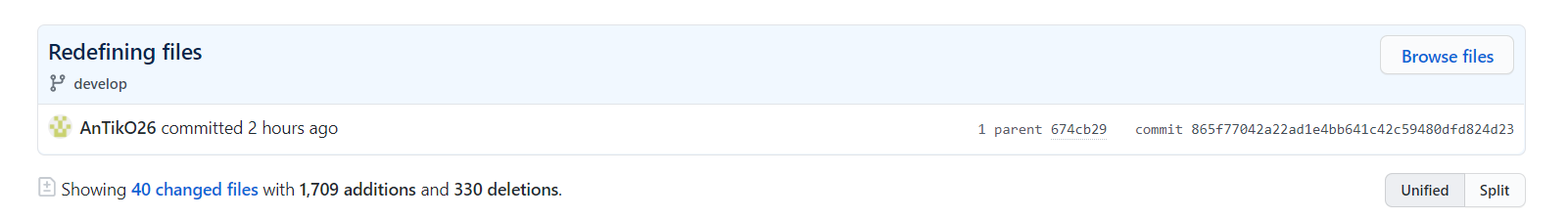


Рис. 52 Двадцать седьмой коммит ветки develop

Создал Петкун Ю.А. Возникли проблемы с проектом и пришлось его пересобрать в новый, копируя файлы с кодом с прошлого проекта

Заключение

Нами использовалась среда разработки Visual Studio 13, язык программирования С++. Система контроля версии нами использовалась. Все прецеденты реализованы. Зависаний программы у нас не наблюдается, а вот сбои наблюдаются при вводе некорректных данных, то есть при вводе данных несоответствующего типа. Принцип раздельной компиляции и очистка памяти полностью реализована. Не используемых переменных в программе нет. Может наблюдаться небольшая избыточность в некоторых условиях. Приложены: диаграмма списка вариантов использования, диаграммы действий, диаграмма классов, диаграммы последовательностей UML. Цель программы достигнута.