Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Петрозаводский государственный университет»

Физико-технический институт

Кафедра информационно-измерительных систем и физической электроники

ПЕКАРНЯ ИНДИВИДУАЛЬНЫЙ ОТЧЁТ

Командный проект по дисциплине «Технология программирования»

Выполнил:

студент группы 21312

А. Г. Смирнов

Научный руководитель:

канд. физ.-мат. наук, доцент КИИСиФЭ А. В. Бульба

Петрозаводск 2023

История коммитов:

1. **Start project** (САГ) – создание репозитория

* Файл main.cpp:

#include <iostream>

using namespace std;

int main()

{

cout << "Here is a start of our project!" << endl;

return 0;

}

2. **Create a header files, update Produce.h** (САГ) – Созданы заголовочные файлы для всех классов. Создан прототип класса Produce

* Produce.h

#ifndef PRODUCE

#define PRODUCE

#include <iostream>

using namespace std;

Class Produce

{

private:

string name;

int id;

double cost;

public:

Produce(string, int, double);

~Produce();

string GetName();

int GetId();

double GetCost();

};

#endif // PRODUCE

3. **Update Produce.h, ProduceList.h and ProduceInputScreen.h, create GlobalMetods.h and libs.h** (САГ) – обновлены файлы Produce.h, ProduceList.h и ProduceInputScreen.h. Созданы заголовочный файл для GlobalMetods, а так же файл libs.h.

* GlobalMetods.h

#ifndef GLOBALMETODS

#define GLOBALMETODS

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

char getaChar(); // получение символа

#endif // GLOBALMETODS

* ProduceInputScreen.h

#ifndef PRODUCEINPUTSCREEN

#define PRODUCEINPUTSCREEN

#include "ProduceList.h"

class ProduceInputScreen

{

private:

ProduceList\* ptrProduceList;

string PrName;

int PrId;

double PrCost;

public:

ProduceInputScreen(ProduceList\* ptrPL) : ptrProduceList(ptrPL)

{

/\* тут пусто \*/

}

void setProduce(); // добавить данные о жильце

};

#endif // PRODUCEINPUTSCREEN

* ProduceList.h

#ifndef PRODUCELIST

#define PRODUCELIST

#include "Produce.h"

class ProduceList {

private:

// установить указатели на продукцию

list <Produce\*> setPtrsProds; // указатели на класс продукция

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

public:

~ProduceList(); // деструктор (удаление продукции из списка)

void insertProduce(Produce\*); // добавить продукцию в список

int getIdPr(string); // возвращают id и стоимость записи по имени позиции

double getCostPr(string);

void display(); // вывод списка продукции

};

#endif // PRODUCELIST

* libs.h

#ifndef LIBS

#define LIBS

#include <iostream>

#include <vector>

#include <list>

#include <string>

#include <numeric>

using namespace std;

#endif // LIBS

**4. Merge branch 'develop-Alex' into develop-general** (САГ) – Залил все наработки с 'develop-Alex' ветки в основную ветку разработки.

**5. Merge branch 'develop-Sasha' into develop-general** (САГ) – Залил все наработки с 'develop-Sasha' ветки в основную ветку разработки.

**6. Add including files** (САГ) – Добавлен файл, в котором подключены все библиотеки. Этот файл для упрощения подключается во всех остальных классах.

Сделал последовательное включение всех заголовочников в нужные места (libs.h - > Expense.h, Profit.h, Produce.h, эти классы -> в контейнерные классы -> в классы экраны)

**7. Случайно залез в чужую ветку, подправил некоторые имена в классах**

**Produce и PreduceList** (САГ) – Ошибочно изменён код на чужой ветке разработки. Были внесены правки в методы классов Produce и PreduceList

**8. Merge branch 'develop-general' into develop-Alex** (САГ) – Актуальный прототип программы был залит на личною ветку Алексея Германовича.

**9. Создал .cpp файлы и добавил реализацию деструкторов для классов Expense, Produce, Profit и их классов-контейнеров. Полностью описал реализацию UserInterface и создал работающий скелет будущей программы** (САГ) – Были добавлены исполняемые файлы для всех существующих классов, реализованы деструкторы для них. Создан рабочий прототип программы.

Expense::~Expense() // деструктор

{ }

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

ExpenseList::~ExpenseList() // деструктор

{ // удалить объекты expense

// удалить указатели на вектор

while (!vectPtrsExpenses.empty())

{

iter = vectPtrsExpenses.begin();

delete \*iter;

vectPtrsExpenses.erase(iter);

}

}

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

Produce::~Produce() // деструктор

{ }

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

ProduceList::~ProduceList() // деструктор

{ // удалить строки с продукцией,

// удалить указатели из списка.

while (!setPtrsProds.empty())

{

iter = setPtrsProds.begin();

delete \*iter;

setPtrsProds.erase(iter);

}

}

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

Profit::~Profit() // деструктор

{ }

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

ProfitList::~ProfitList() // деструктор

{ // удалить строки с доходами,

// удалить указатели из списка.

while (!setPtrsProfits.empty())

{

iter = setPtrsProfits.begin();

delete \*iter;

setPtrsProfits.erase(iter);

}

}

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

UserInterface.cpp

UserInterface::UserInterface()

{

ptrProduceList = new ProduceList;

ptrProfitList = new ProfitList;

ptrExpenseList = new ExpenseList;

}

UserInterface::~UserInterface()

{

delete ptrProduceList;

delete ptrProfitList;

delete ptrExpenseList;

}

void UserInterface::interact()

{

while (true)

{

cout << "To enter data press 'i', \n"

<< " to display the report 'd', \n"

<< " to exit 'q': \n";

char ch;

cin >> ch;

if (ch == 'i') // input data

{

cout << " press 't' to add products, \n"

<< " press 'r' to record income, \n"

<< " press 'e' to record expenses: \n";

cin >> ch;

switch (ch)

{

// input screens only exist during their use

case 't': cout << "Function is under development. Please come back later!" << endl;

break;

case 'r': cout << "Function is under development. Please come back later!" << endl;

break;

case 'e': cout << "Function is under development. Please come back later!" << endl;

break;

default: cout << "Unknown function\n";

break;

} // end of switch section

} // end of if condition

else if (ch == 'd') // output data

{

cout << "Press 't' to display the list of products, \n"

<< " press 'r' to display income, \n"

<< " press 'e' to display expenses, \n"

<< " press 'a' to display the annual report: \n";

cin >> ch;

switch (ch)

{

case 't': cout << "Function is under development. Please come back later!" << endl;

break;

case 'r': cout << "Function is under development. Please come back later!" << endl;

break;

case 'e': cout << "Function is under development. Please come back later!" << endl;

break;

case 'a': cout << "Function is under development. Please come back later!" << endl;

default: cout << "Unknown display function\n";

break;

} // end switch

} // end elseif

else if (ch == 'q') {

system("pause");

return; // exit

}

else

cout << "Unknown function. Press only 'i', 'd', or 'q'\n";

} // end while

}

* main.cpp

#include "UserInterface.h"

int main()

{

setlocale(LC\_ALL, "rus");

UserInterface theUserInterface;

theUserInterface.interact();

return 0;

}

**10. Доработан скелет** (САГ) – Внесены правки в существующий прототип программы.

**11. Merge branch 'develop-Alex' into develop-general** (САГ) – Залил все наработки с ветки 'develop-Alex' в основную ветку разработки.

**12. Merge branch 'release-1.0'** (САГ) – Первый релиз программы.

**13. Merge branch 'develop-general' into develop-Alex** (САГ) – Залил все наработки с 'develop-general' ветки в свою ветку разработки.

**14. Созданы исполняемые файлы, расставлены включения** (САГ) – Были созданы исполняемые файлы, так же в них были расставлены включения заголовочных файлов.

**15. Merge branch 'develop-general' into develop-Alex** (САГ) – Залил все наработки с 'develop-general' ветки в свою ветку разработки.

**16. Полностью описана реализация класса UserInterface. Вернул файлы globalmetods ( ;( )** (САГ) – Была описана реализация класса UserInterface. Обратно добавлен файл globalmetods.

#include "UserInterface.h"

UserInterface::UserInterface()

{

ptrProduceList = new ProduceList;

ptrProfitList = new ProfitList;

ptrExpenseList = new ExpenseList;

}

UserInterface::~UserInterface()

{

delete ptrProduceList;

delete ptrProfitList;

delete ptrExpenseList;

}

void UserInterface::interact()

{

char ch;

while (true)

{

cout << "To enter data press 'e', \n"

<< " to display the report 'd', \n"

<< " to exit 'q': \n";

ch = getaChar();

if (ch == 'e') // enter data

{

cout << " press 'p' to add products, \n"

<< " press 'i' to record income, \n"

<< " press 'e' to record expenses: \n";

ch = getaChar();

switch (ch)

{

// input screens only exist during their use

case 'p':

ptrProduceInputScreen = new ProduceInputScreen(ptrProduceList);

ptrProduceInputScreen->setProduce();

delete ptrProduceInputScreen;

break;

case 'i':

ptrProfitInputScreen = new ProfitInputScreen(ptrProfitList);

ptrProfitInputScreen->setProfit();

delete ptrProfitInputScreen;

break;

case 'e':

ptrExpenseInputScreen = new ExpenseInputScreen(ptrExpenseList);

ptrExpenseInputScreen->SetExpense();

delete ptrExpenseInputScreen;

break;

default: cout << "Unknown function\n";

break;

} // end of switch section

} // end of if condition

else if (ch == 'd') // output data

{

cout << "Press 'l' to display the list of products, \n"

<< " press 'i' to display income, \n"

<< " press 'e' to display expenses, \n"

<< " press 'a' to display the annual report: \n";

ch = getaChar();

switch (ch)

{

case 'l': ptrProduceList->DisplayProd();

break;

case 'i': ptrProfitList->DisplayProf();

break;

case 'e': ptrExpenseList->DisplayExp();

break;

case 'a':

ptrAnnualReport = new AnnualReport(ptrProfitList, ptrExpenseList);

ptrAnnualReport->display();

delete ptrAnnualReport;

default: cout << "Unknown display function\n";

break;

} // end switch

} // end elseif

else if (ch == 'q') {

return; // exit

}

else

cout << "Unknown function. Press only 'e', 'd', or 'q'\n";

} // end while

}

* globalmetods.h

#ifndef GLOBALMETODS

#define GLOBALMETODS

#include "libs.h"

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

char getaChar(); // получение символа

#endif // GLOBALMETODS

* globalmetods.cpp

#include "globalmetods.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;

}

**17. (для прошлого коммита - Так же исправлены неточности в заголовочных файлах) Сейчас - описана реализация класса AnnualReport.** (САГ) – Были исправлены неточности в заголовочных файлах. Так же была написана реализация класса AnnualReport.

#include "AnnualReport.h"

#include "AnnualReport.h"

void AnnualReport::display()

{

cout << setw(20) << "Annual report" << endl;

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

cout << "Incomes\n" << endl;

cout << "Bakery sells:\t\t";

profits = ptrPL->DisplaySummProfits();

cout << profits << endl << endl;

cout << "Expenses\n\n";

expenses = ptrEL->DisplaySummExpenses();

cout << "Total expense:\t\t";

cout << expenses << endl << endl;

// вычисляем прибыльность

cout << "\nProfit:\t\t\t" << (profits - expenses) << endl;

}

**18. Merge branch 'develop-Alex' into develop-general** (САГ) – Залил все наработки с 'develop-Alex' ветки в основную ветку разработки.

**19. Отображение в таблице смещено вправо** (САГ) – Отображение информации в таблицах было смещено вправо.

В методах контейнерных классов, выводящих на экран информацию, добавлено использование метода setw(10) для потокового вывода.

**20. Merge branch 'release-2.0'** (САГ) – Второй релиз программы.