Учреждение образования Федерации профсоюзов Беларуси

«Международный университет «МИТСО»

Рег. № \_\_\_\_\_\_\_\_\_\_ Кафедра информационных систем и технологий

Дата \_\_\_\_\_\_\_\_\_2020

**КУРСОВАЯ РАБОТА**

на тему \_\_\_\_Название\_\_\_Разработать программный модуль «Картотека недвижимости»\_\_\_\_\_\_\_\_

по дисциплине \_Название\_\_\_\_Основы конструирования программ\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| Основные замечания: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Отметка о допуске курсовой работы к  защите:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_  Подпись научного руководителя:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Студент: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись)  Урбанович Владислав Сергеевич (Ф.И.О. – полностью)  Курс \_\_1\_\_, группа \_\_\_1920\_\_\_\_  Факультет \_\_экономический\_\_\_  Специальность \_\_Информационные системы и технологии\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Научный руководитель:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (должность, ученая степень, ученое звание)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (Ф.И.О. – полностью) |

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc70793473)

[1.ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 4](#_Toc70793474)

[**1.1. Общие положения 4**](#_Toc70793475)

[**1.2. Постановка задачи 4**](#_Toc70793476)

[**1.3. Спецификация входных и выходных данных 4**](#_Toc70793477)

[**1.4. Выбор средства 5**](#_Toc70793478)

[**1.5.Описывается требуемое оборудование и программное обеспечение 6**](#_Toc70793479)

[2.ТЕХНОЛОГИЯ РАЗРАБОТКИ ПРИЛОЖЕНИЯ 7](#_Toc70793480)

[**2.1. Алгоритм решения 7**](#_Toc70793481)

[**2.2. Описание программы 7**](#_Toc70793482)

[**2.3. Результаты работы программы 9**](#_Toc70793483)

[РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 10](#_Toc70793484)

[ЗАКЛЮЧЕНИЕ 10](#_Toc70793485)

[ПРИЛОЖЕНИЕ 11](#_Toc70793486)

# ВВЕДЕНИЕ

Была поставлена задача разработать программный продукт «Бизнес процессы договорного отдела», предназначенный для использования работниками компании посредника. В программном продукте необходимо реализовать учёт заключённых договоров. Договор должен содержать название договора, дата заключения и спецификации к договору. Спецификации к договору содержат материал, количества, дата поставки.

Программа создана для учёта договоров и соблюдение договорённостей, автоматизировать информационные при заключении договора.

Цель разработки обеспечить работу программного продукта быстро и максимально эффективно, удовлетворять запросы компании, а также должен быть возможность пополнять базу данных, для учёта договоров. Основная же задача стоит в высокой оптимизация, чтобы заключающие договор могли запустить даже на самом слабом устройстве.

Задачи данного программного подукта: быстро и гибко редактироваться со стороны владельцев компании. С точки зрения использования пользователя программа должна иметь красивый, понятный и интуитивный интерфейс чтобы было пользователем приятно пользоваться программой. Так же требуется предусмотреть возможность добавления новых функций в программе.

Для разработки программы был использован текстовый файл и язык программирования C++ разработанный в среде Visual Studio компаний Microsoft.

Visual Studio-линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой Windows Forms, а также веб-сайты, веб-приложения, веб-службы, как в родном, так и в управляемом кодах для всех платформ, поддерживающих Windows, Windows Mobile, Windows Ce, .Net Framework, Xbox, Windows Phone .Net Compact Framework и Silverling.

# 1.ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

## 1.1. Общие положения

Составления договоров является очень важной частью любого бизнеса. Что привило к усложнению процесса учётов договоров. Для обеспечение подобного рода операций необходимо использовать технологии, которые будут обеспечивать возможность подобной работы.

Была поставлена задача разработать программный продукт «Бизнес процессы договорного отдела», предназначенный для использования работниками компании посредника. В программном продукте необходимо реализовать учёт заключённых договоров. Договор должен содержать название договора, дата заключения и спецификации к договору. Спецификации к договору содержат материал, количества, дата поставки.

Договор представляет собой два документа. Сам договор, имеющий примерную структуру и спецификации к договору, имеющей примерную структуру.

Инженер, должен на основе этих документов, составить консолидированный план поставок материалов на свое предприятие, отмечать, что выполнено по поставкам, что сорвано, формировать всевозможные отчеты.

Данный программный продукт выгодно отличается от других подобных тем что пользуясь им, пользователь может не только структурированно хранить данные договоров но и производить различные взаимодействия с ними (добавление/изменение удаление).

## 1.2. Постановка задачи

Исходя из задач, описанных выше главными требованиями к языку, на котором будет написана программы — это тонкая настройка и возможность использованием минимум вычислительных ресурсов добиваться максимум результата.

## 1.3. Спецификация входных и выходных данных

В данном программном продукте используются следующие типы данных типа данных: sting, double. Так же существуют пользовательские типы данных: ConsoleMenager, AgreementManager.

boll – используется только внутри программы, для проверки, подсчёта, иннерваций и так далее. Пользователь на данный этого типа влиять и пользоваться не может.

String – используется в двух случаях. Либо для наименования данных (введение информации о договоре). Во втором же случае данный тип данных используется для вывода всех данных на экран.

Int – используется для реализации различных алгоритм (пользователь не работает с этим типом данных) и так же для наименования данных.

ConsoleMenager – предназначен для работы с консолью, ввод различных данных, вывод данных, чтение данных и т.д.

AgreementManager – данный класс представляет собой бизнес-логику приложения.

## 1.4. Выбор средства

Для выполнения данных задач очень подходит такой язык программирования как с++. Так как его из-за того, что он является весьма низкоуровневым языком программирования, а значит он больше других приближен к машинному коду, из-за этого с++ дает возможность оптимизировать проект так что даже на очень старых устройствах он будет работать.

Среда разработки Microsoft Visual Studio. Вид приложения – консольное. Программа построена на базе объектно-ориентированного программирования. Способ организации данных поля классов. Способ хранения данных – динамические массивы. Каждая логическая завершенная данных подзадача программы реализована в виде методов. Построение программного кода соответствует с <<C++ Code Convention>>. К защите курсовой предоставляется: консольное приложение и пояснительная записка

Windows 10 – операционная система для персональных компьютеров и рабочих станций, разработанная корпорацией Microsoft в рамках семейства Windows NT. После Windows 8.1 система получила номер 10, минуя 9. Серверный аналог Windows 10 – Windows Server 2016. Система призвана стать единой для разных устройств, таких как персональные компьютеры, планшеты, смартфоны, консоли Xbox One и пр. Доступна единая платформа разработки и единый магазин универсальных приложений, совместимых со всеми поддерживаемыми устройствами. Windows 10 поставляется в качестве услуги с выпуском обновлений на протяжении всего цикла поддержки. В течение первого года после выхода системы пользователи могли бесплатно обновиться до Windows 10 на устройствах под управлением лицензионных копий Windows 7, Windows 8.1 и Windows Phone 8.1. Среди значимых нововведений – голосовая помощница Кортана, возможность создания и переключения нескольких рабочих столов и др. Windows 10 – последняя «коробочная» версия Windows, все последующие версии будут распространяться исключительно в цифровом виде.

## 1.5.Описывается требуемое оборудование и программное обеспечение

Для полноценного функционирования разработанного программного приложения, качественного выполнения всех поставленных задач, требуется ПК следующей конфигурации:

* процессор Intel Core i3;
* оперативная память DDR3 512 МБ или больше;
* минимально свободного места на жёстком диске 512 МБ;
* мышь;
* операционная система Windows 7/8/10.

# 2.ТЕХНОЛОГИЯ РАЗРАБОТКИ ПРИЛОЖЕНИЯ

## 2.1. Алгоритм решения

В текущей главе представлено описание алгоритма, который имеет начало алгоритма и конец, внутри алгоритма имеются циклы и иные операций.

Вначале программы происходит инициализация классов и необходимых переменных для дальнейшего взаимодействия с ними. Далее происходит бесконечный цикл while. Он необходим для реализации повторного выбора операции.

Далее в программе происходит выбор операции по нажатию необходимой цифры.

1 - Создать договор

2 - Список всех договоров

1. – Найти договор
2. - Выход
3. При помощи функции createAgreemet() добавляются данные на договора. В данной функции вызываются другие функции: printRequest, readString, setOrganizationData, setRegistrationData.
4. При помощи функции redrawAgreementsMenu производиться вывод всех договоров. В данной функции вызываются другие функции: getId, getOrganization, clearScreen.
5. При помощи функции resolveAgreementSearch производиться поиск договоров. В данной функции вызываются другие функции: redrawAgreementSearchMenu(), printRequest(), readInt(), findPositionById(), findPositionsByOrganizftion(). Так же логика данной функции зависит от выбранной переменной choice. Она является целочисленной переменной при её помощи происходит выбор.
6. Происходит выход из программы и конец работы программы.

## 2.2. Описание программы

В данной программы используются следующие библиотеки:

#include <iostream> - основная библиотека, которая реализует основные возможность языка С++.

#include <string> - библиотека для возможности работы с типом данных string.

#include <ctime> - данная библиотека необходима для реализации работы с датой и времени.

#include <deque> - данная библиотека необходима для реализации работы с двухсторонней очередью.

Функций:

**createAgreement()** – используется для создание договора(рис 2.2.1).



рис 2.2.1

**resolveAgreementSearch()** – необходима для реализации алгоритма поиска договора (рис 2.2.2)

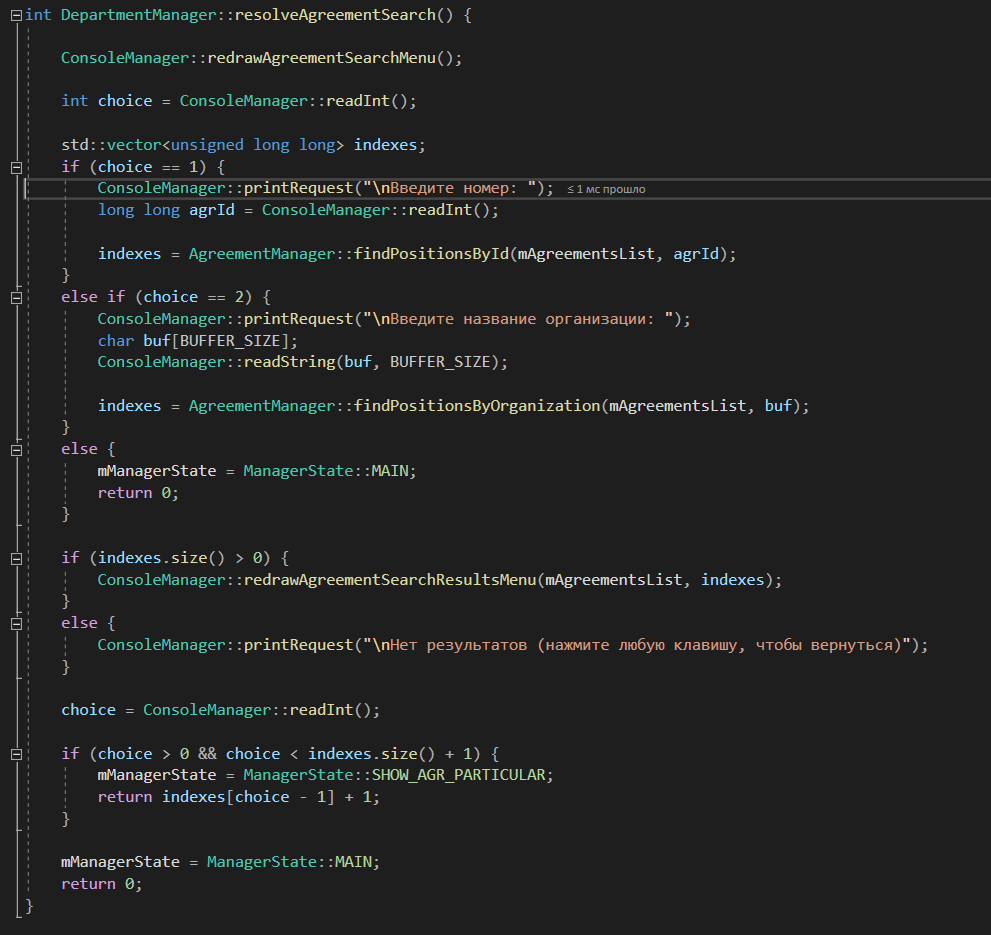


рис 2.2.2

**redrewAgreementsMenu() –** выводит меню программы(рис 2.2.3).

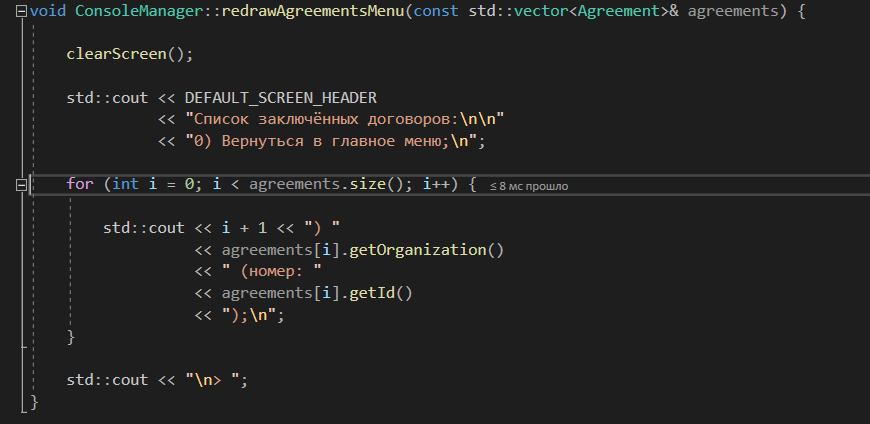


рис 2.2.3

## 2.3. Результаты работы программы

Для описания данного я буду опираться на главу “Алгоритм программы” так как основная реализация программного продукта за счёт механизма switch.

Переменные описывающие договр:

1. Agr.serOrgsnization – название организации.
2. Arg.serExpirationDate – дата истечения действия договора.

Переменные описывающие спецификацию к договору:

1. Sup.material – данные материала.
2. Sup.amount – данные о количестве материала.
3. Sup.measureUnit – единицы измерения материала.
4. Sup.arrivalDate – дата поставки материала.
5. Sup.status – статус поставки материала.

# 3.РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Так как была поставлена задача разработать консольное приложение, а значит при разработке интерфейса возникают все проблемы для консольных приложений. Этот факт необходимо учитывать при разработке интерфейса.

При запуске программы на консоль выводиться выбор операции. Для осуществления выбора необходимо ввести соответствующую цифру в консоль.

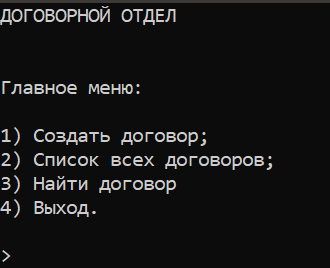


Рис 3.1

При выборе пункта № 1 на консоль необходимо ввести данные договора поочерёдно и так же реализована добавления спецификации договора (рис 3.2).

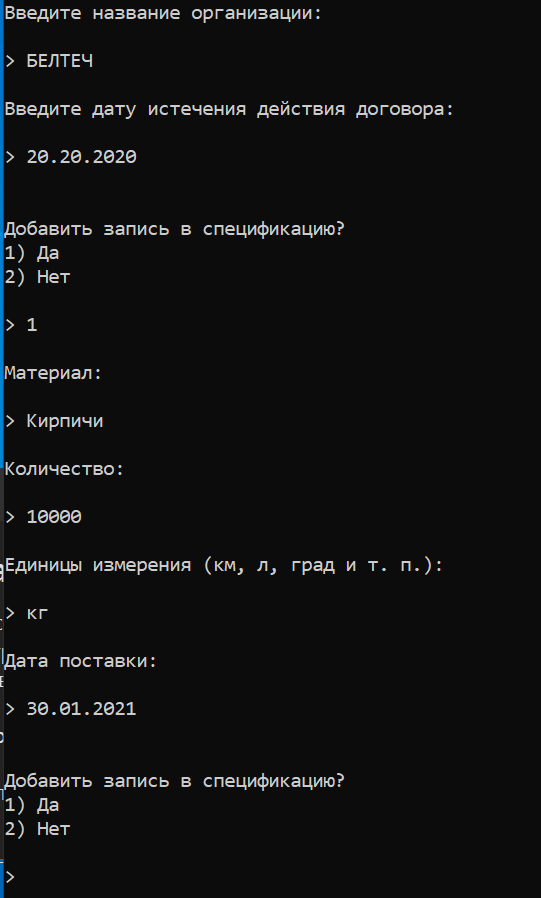


Рис 3.2

Пункт № 2 дает возможность просмотреть данные о договорах путём ввода номера номер желаемого договора для просмотра (рис 3.2).

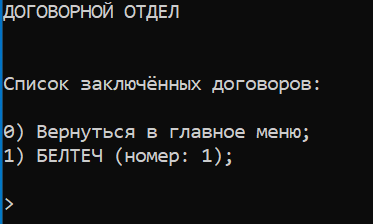


рис 3.2

При выборе 0 происходит переход в главное меню. При выборе же конкретного номера договора происходит печать данных договора на консоль (Рис 3.3).

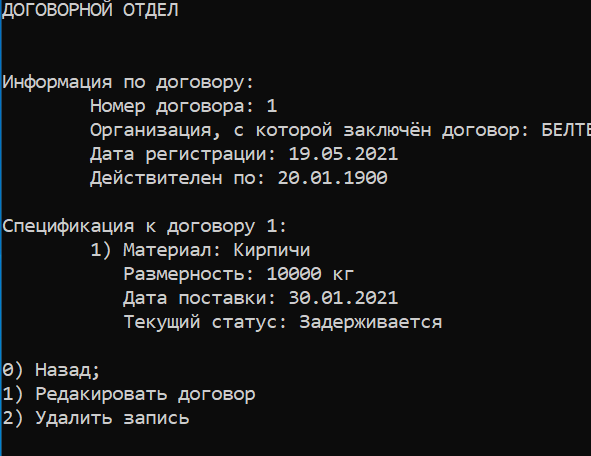


Рис 3.3

После введения данных пользователь получает возможность удалять или редактировать данные договора.

Редактирование данных происходит так же путём выбора желаемого пункта изменения (рис 3.4 ).

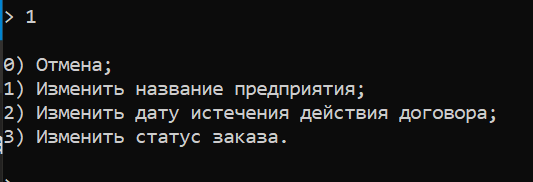


Рис 3.4

В пункте № 3 происходит поиск договора. Поиск реализуется по имени и по номеру договора(рис 3.5).

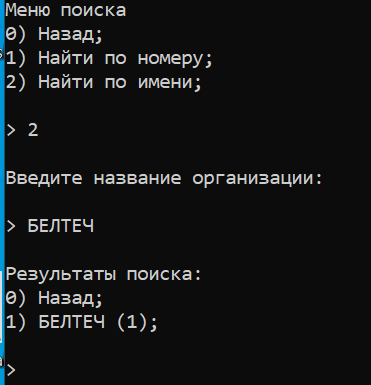


рис 3.5

При выборе пункта № 4 происходит выход из программы.

# ЗАКЛЮЧЕНИЕ

Исходя из поставленных целей и задач было сконструировано, разработана и протестирован программный модуль «Учёт клиентов санатория», предназначенный для использования работниками санатория. Были реализованы задачи:

1. Хранение данных договора.
2. Реализация алгоритм изменения договора.
3. Вывод данных
4. Добавление спецификации

Цель была выполнены, программный продукт имеет следующие преимущества:

1. Работа программного продукта происходит быстро и максимально эффективно. Программа удовлетворяет запросам пользователя.
2. Имеет возможность пополнять базу данных, для расширения ассортимента товаров.
3. Так же программы имеет высокую оптимизацию. Из-за этого программа работает весьма быстра и запускается даже на самом слабом устройстве.

Так же данный программный продукт выгодно отличается от других представленных на рынке, возможностью не только хранить данные договоров, но и производить различные взаимодействия с ними, ещё она снабжена интуитивным интерфейс.

Таким образом программа работает без сбоев и все задачи были выполнены.

# СПИСОК ЛИТЕРАТУРЫ

1. Коплиен Дж. Мультипарадигменное проектирование для С++ / Коплиен Дж. – Питер, 2005.
2. Обзор обновлений и новых функций Windows 10 [Электронный ресурс]. – Microsoft, 2020. – Режим доступа: https://www.microsoft.com/ru-ru/windows/features. – Дата доступа: 22.05.2020.
3. Объектно-ориентированный анализ и проектирование с примерами приложений / Гради Буч [и др.]. – 3-е изд. – М.: ООО «И.Д. Вильямс», 2008. – 720 с.
4. Стивен Прата Язык программирования C++ / Стивен Прата – Вильямс, 2012.
5. Страуструп Б. Программирование. Принципы и практика с использованием C++ / Страуструп Б. – 2-е изд. – Вильямс, 2016.
6. Model-View-Presenter [Электронный ресурс]. – Википедия, 2020. – Режим доступа: https://ru.wikipedia.org/wiki/Model-View-Presenter. – Дата доступа: 22.04.2020.
7. SFML [Электронный ресурс]. – SFML, 2020. – Режим доступа: https://www.sfml-dev.org. – Дата доступа: 22.04.2020.

# ПРИЛОЖЕНИЕ

#include "department\_manager.h"

DepartmentManager::DepartmentManager() {

mManagerState = ManagerState::MAIN;

}

DepartmentManager::~DepartmentManager() {

std::ofstream file;

file.open(AGREEMENTS\_FILEPATH, std::ios::out | std::ios::trunc);

if (file.is\_open()) {

for (Agreement agr : mAgreementsList) {

std::string out(agr.toString());

file.write(out.c\_str(), out.size());

file.write("\n", 1);

std::ofstream specFile;

std::ostringstream ss;

ss << SPECIFICATION\_FOLDER << SPECIFICATION\_PREFIX << agr.getId() << DEFAULT\_FILE\_SUFFIX;

specFile.open(ss.str().c\_str(), std::ios::out | std::ios::trunc);

if (specFile.is\_open()) {

std::string out\_(agr.getSpecification().toString());

specFile.write(out\_.c\_str(), out\_.size());

specFile.close();

}

}

file.close();

}

}

int DepartmentManager::startManagerLoop() {

mAgreementsList = AgreementManager::fetchAgreements();

int exitCode = 0;

while (true) {

switch (mManagerState) {

case ManagerState::MAIN:

exitCode = resolveMainMenuChoice();

break;

case ManagerState::CREATE\_AGR:

exitCode = resolveAgreementCreation();

break;

case ManagerState::SHOW\_AGR\_LIST:

exitCode = resolveAgreementsMenuChoice();

break;

case ManagerState::SHOW\_AGR\_PARTICULAR:

exitCode = resolveAgreementPreviewMenuChoice(exitCode);

break;

case ManagerState::DELETE\_AGR:

exitCode = resolveAgreementRemoval(exitCode);

break;

case ManagerState::EDIT\_AGR:

exitCode = resolveAgreementEditing(exitCode);

break;

case ManagerState::FIND\_AGR:

exitCode = resolveAgreementSearch();

break;

case ManagerState::EDIT\_SPEC:

exitCode = resolveSpecificationEditing(exitCode);

break;

case ManagerState::TERMINATE:

return 0;

default:

return -1;

}

}

}

int DepartmentManager::resolveMainMenuChoice() {

ConsoleManager::redrawMainMenu();

int choice = ConsoleManager::readInt();

switch (choice) {

case 1:

mManagerState = ManagerState::CREATE\_AGR;

break;

case 2:

mManagerState = ManagerState::SHOW\_AGR\_LIST;

break;

case 3:

mManagerState = ManagerState::FIND\_AGR;

break;

case 4:

mManagerState = ManagerState::TERMINATE;

break;

default:

break;

}

return 0;

}

int DepartmentManager::resolveAgreementsMenuChoice() {

ConsoleManager::redrawAgreementsMenu(mAgreementsList);

int choice = ConsoleManager::readInt();

if (choice == 0) {

mManagerState = ManagerState::MAIN;

}

else if (choice > 0 && choice <= mAgreementsList.size()) {

mManagerState = ManagerState::SHOW\_AGR\_PARTICULAR;

return choice;

}

return 0;

}

int DepartmentManager::resolveAgreementPreviewMenuChoice(int index) {

ConsoleManager::redrawAgreementPreviewMenu(mAgreementsList[index - 1]);

int choice = ConsoleManager::readInt();

switch (choice) {

case 0:

mManagerState = ManagerState::SHOW\_AGR\_LIST;

return 0;

case 1:

mManagerState = ManagerState::EDIT\_AGR;

break;

case 2:

mManagerState = ManagerState::DELETE\_AGR;

break;

default:

break;

}

return index;

}

int DepartmentManager::resolveAgreementCreation() {

ConsoleManager::clearScreen();

Agreement agr = AgreementManager::createAgreement();

Specification spec;

spec.setId(agr.getId());

int choice;

while (true) {

choice = ConsoleManager::resolveYesNoChoice("\nÄîáàâèòü çàïèñü â ñïåöèôèêàöèþ?");

if (choice != 1) {

break;

}

spec.appendSupply(

SpecificationManager::createSupply()

);

}

agr.setSpecification(spec);

mAgreementsList.push\_back(agr);

mManagerState = ManagerState::MAIN;

return 0;

}

int DepartmentManager::resolveAgreementRemoval(int index) {

int choice = ConsoleManager::resolveYesNoChoice("\nÂû óâåðåíû?");

if (choice == 1) {

mAgreementsList.erase(mAgreementsList.begin() + index - 1);

mManagerState = ManagerState::SHOW\_AGR\_LIST;

return 0;

}

else {

mManagerState = ManagerState::SHOW\_AGR\_PARTICULAR;

return index;

}

}

int DepartmentManager::resolveAgreementEditing(int index) {

ConsoleManager::redrawAgreementEditingMenu();

int choice = ConsoleManager::readInt();

if (choice == 1) {

AgreementManager::changeOrganiztion(mAgreementsList[index - 1]);

mManagerState = ManagerState::SHOW\_AGR\_PARTICULAR;

}

else if (choice == 2) {

AgreementManager::changeExpirationDate(mAgreementsList[index - 1]);

mManagerState = ManagerState::SHOW\_AGR\_PARTICULAR;

}

else if (choice == 3) {

mManagerState = ManagerState::EDIT\_SPEC;

}

return index;

}

int DepartmentManager::resolveAgreementSearch() {

ConsoleManager::redrawAgreementSearchMenu();

int choice = ConsoleManager::readInt();

std::vector<unsigned long long> indexes;

if (choice == 1) {

ConsoleManager::printRequest("\nÂâåäèòå íîìåð: ");

long long agrId = ConsoleManager::readInt();

indexes = AgreementManager::findPositionsById(mAgreementsList, agrId);

}

else if (choice == 2) {

ConsoleManager::printRequest("\nÂâåäèòå íàçâàíèå îðãàíèçàöèè: ");

char buf[BUFFER\_SIZE];

ConsoleManager::readString(buf, BUFFER\_SIZE);

indexes = AgreementManager::findPositionsByOrganization(mAgreementsList, buf);

}

else {

mManagerState = ManagerState::MAIN;

return 0;

}

if (indexes.size() > 0) {

ConsoleManager::redrawAgreementSearchResultsMenu(mAgreementsList, indexes);

}

else {

ConsoleManager::printRequest("\nÍåò ðåçóëüòàòîâ (íàæìèòå ëþáóþ êëàâèøó, ÷òîáû âåðíóòüñÿ)");

}

choice = ConsoleManager::readInt();

if (choice > 0 && choice < indexes.size() + 1) {

mManagerState = ManagerState::SHOW\_AGR\_PARTICULAR;

return indexes[choice - 1] + 1;

}

mManagerState = ManagerState::MAIN;

return 0;

}

int DepartmentManager::resolveSpecificationEditing(int index) {

Specification& spec = mAgreementsList[index - 1].getSpecification();

ConsoleManager::redrawSpecificationEditingMenu(spec);

int choice = ConsoleManager::readInt() - 1;

if (choice >= 0 && choice < spec.getSuppliesList().size()) {

Specification::Supply& sup = spec.getSuppliesList()[choice];

SpecificationManager::changeStatus(sup);

}

mManagerState = ManagerState::SHOW\_AGR\_PARTICULAR;

return index;

}