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 13](#_Toc70793484)

[ЗАКЛЮЧЕНИЕ 15](#_Toc70793485)

[ПРИЛОЖЕНИЕ 17](#_Toc70793486)

# ВВЕДЕНИЕ

Была поставлена задача разработать программный продукт «Бизнес- процессы подразделений, учитывающих здания и сооружения организации», предназначенный для использования работниками подразделения. В программном продукте необходимо реализовать паспортизация зданий и сооружений, учет капитальных и профилактических ремонтов. В программе должна содержать информацию о зданиях его состоянии и комендантах, которые следят за зданием.

Программа создана для паспортизации зданий и сооружений, учет капитальных и профилактических ремонтов, автоматизации работу подразделения.

Цель разработки обеспечить работу программного продукта быстро и максимально эффективно, удовлетворять запросы ресторана, а также должен быть возможность пополнять базу данных, для паспортизации зданий и сооружений, учет капитальных и профилактических ремонтов. Основная же задача стоит в высокой оптимизация, чтобы пользователь могли запустить даже на самом слабом устройстве.

Задачи данного программного продукта: быстро и гибко редактироваться со стороны пользователей подразделений. 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

Была поставлена задача разработать программный продукт «Бизнес- процессы подразделений, учитывающих здания и сооружения организации», предназначенный для использования работниками подразделения. В программном продукте необходимо реализовать паспортизация зданий и сооружений, учет капитальных и профилактических ремонтов. В программе должна содержать информацию о зданиях его состоянии и комендантах, которые следят за зданием.

Предприятия, имеющее в собственности здания, должно поддерживать ­его в целости и сохранности. Для этого составляется паспорт здания, закрепляются ответственные (коменданты), которые должны за этим следить и при необходимости, принимать меры. У них есть планы капитального ремонта и профилактических ремонтов. Для этих целей они либо приглашают бригады со стороны, либо привлекают собственные структуры

Данный программный продукт выгодно отличается от других подобных тем что пользуясь им, пользователь может не только структурированно хранить данные зданий, но и возможность вести паспорт объекта.

## 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: Commandant, Building, Passport.

**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 здании). 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.

**Commandant** - представляет собой коменданта который следит за зданием.

**Building** - представляет собой здание в котором есть данные о здании, его паспорт и список комендантов приписанных к зданию.

**Passport** - представляет собой паспорт объекта.

## 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.

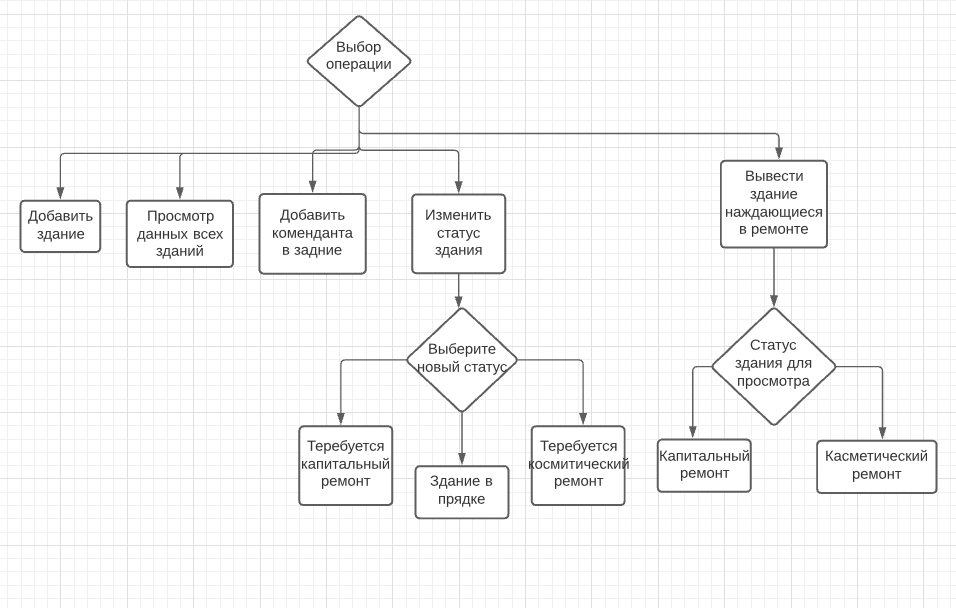
## 1.5.Описывается требуемое оборудование и программное обеспечение

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



## 2.2. Описание программы

В данной программы используются следующие библиотеки:

**#include <math.h>** - заголовочный файл стандартной библиотеки языка программирования С, разработанный для выполнения простых математических операций. Большинство функций привлекают использование чисел с плавающей точкой. C++ также реализует данные функции для обеспечения совместимости, все они содержатся в заголовочном файле cmath.

Все эти функции принимают double, если не определено иначе. Для работы с типами float и long double используются функции с постфиксами f и l соответственно. Все функции, принимающие или возвращающие угол, работают с радианами.

**#include <stdio.h>** — стандартный заголовочный файл ввода-вывода) заголовочный файл стандартной библиотеки языка Си, содержащий определения макросов, константы и объявления функций и типов, используемых для различных операций стандартного ввода и вывода. Функциональность унаследована от «портативного пакета ввода-вывода».

**#include <conio.h>** - консольный ввод-вывод) — заголовочный файл, используемый в старых компиляторах, работающих в операционных системах MS-DOS, для создания текстового интерфейса пользователя. Тем не менее, он не является частью языка программирования Си, стандартной библиотеки языка Си, ISO C или требуемой стандартом POSIX.

**#include <locale.h>** — заголовочный файл стандартной библиотеки языка программирования С, который используется для задач, связанных с локализацией.

**#include <cstdlib>** - заголовочный файл стандартной библиотеки языка Си, который содержит в себе функции, занимающиеся выделением памяти, контролем процесса выполнения программы, преобразованием типов и другие. Заголовок вполне совместим с C++ и известен в нём как cstdlib. Название «stdlib» расшифровывается как «standard library» (стандартная библиотека).

**#include <string>** - класс с методами и переменными для организации работы со строками в языке программирования C++. Он включён в стандартную библиотеку C++. Название образовано от имени строчного типа данных (англ. string; с англ. — «строка»). В языке C++ и его предшественнике, языке программирования Си, нет встроенной поддержки строкового типа данных, вместо этого используется массив символов. string управляет строками, как и string.h в Си. string использует единственный объект string для организации работы со строками. Являясь частью стандартной библиотеки C++, эти объекты также являются частью стандартного пространства имён — std.

**#include <iostream>** - заголовочный файл с классами, функциями и переменными для организации ввода-вывода в языке программирования C++. Он включён в стандартную библиотеку C++. Название образовано от Input/Output Stream («поток ввода-вывода»).

**#include <algorithm>** - заголовочный файл в стандартной библиотеке языка программирования C++, включающий набор функций для выполнения алгоритмических операций над контейнерами и над другими последовательностями. Все функции библиотеки расположены в пространстве имён std.

**Commandant** – представляет собой коменданта который следит за зданием (рис 2.2.1).

Переменные:

*Surname* - фамилия коменданта.

*Name* - имя коменданта.

*Patronymic* - отчество коменданта.

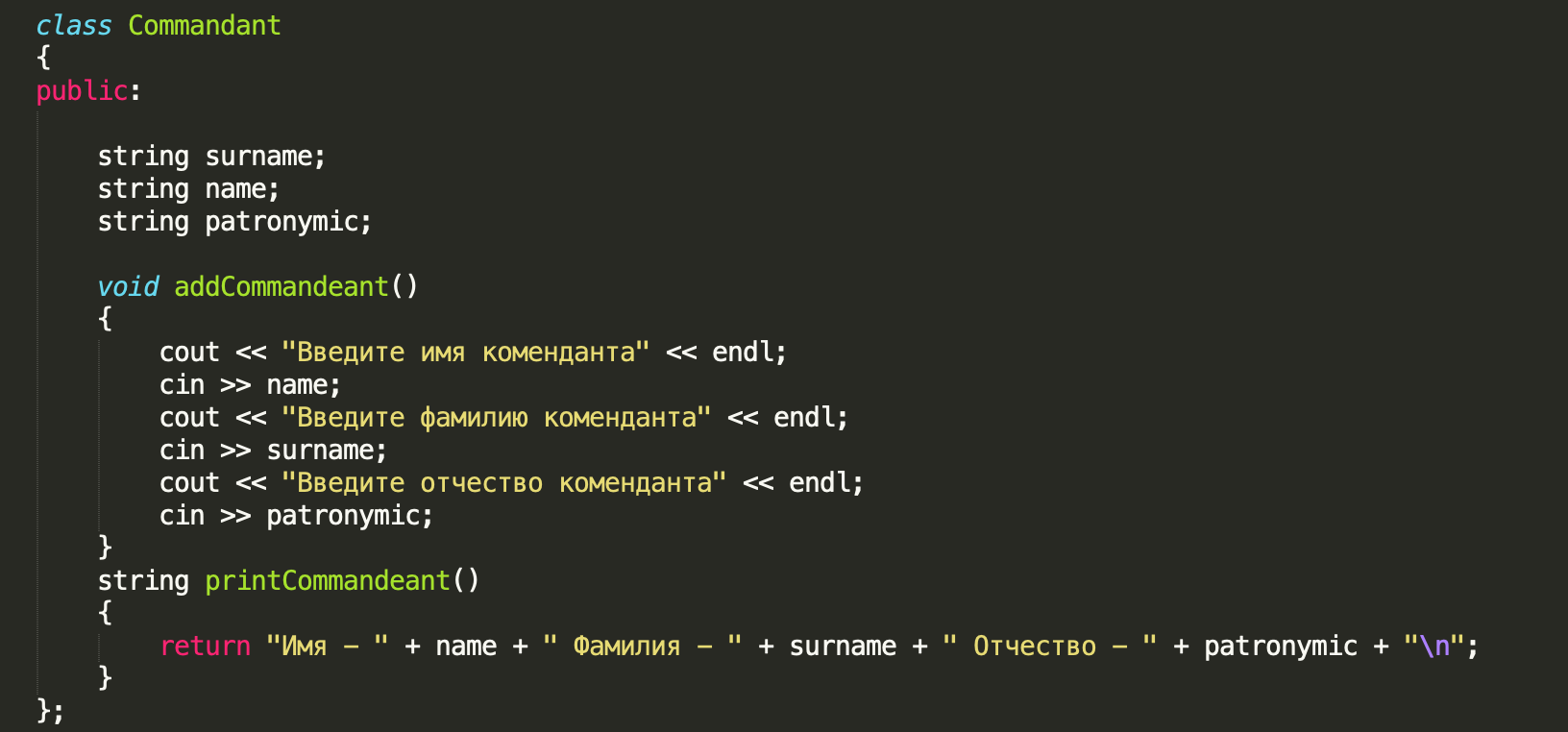


рис 2.2.1

void **addCommandeant()** - реализует механизм присвоение данных объекту класса.

string **printCommandeant()** - реализует механизм вывода данных объекта класса путём чтение полей класса и возвращение строковой переменной.

**Passport** – представляет собой паспорт объекта (рис 2.2.2).

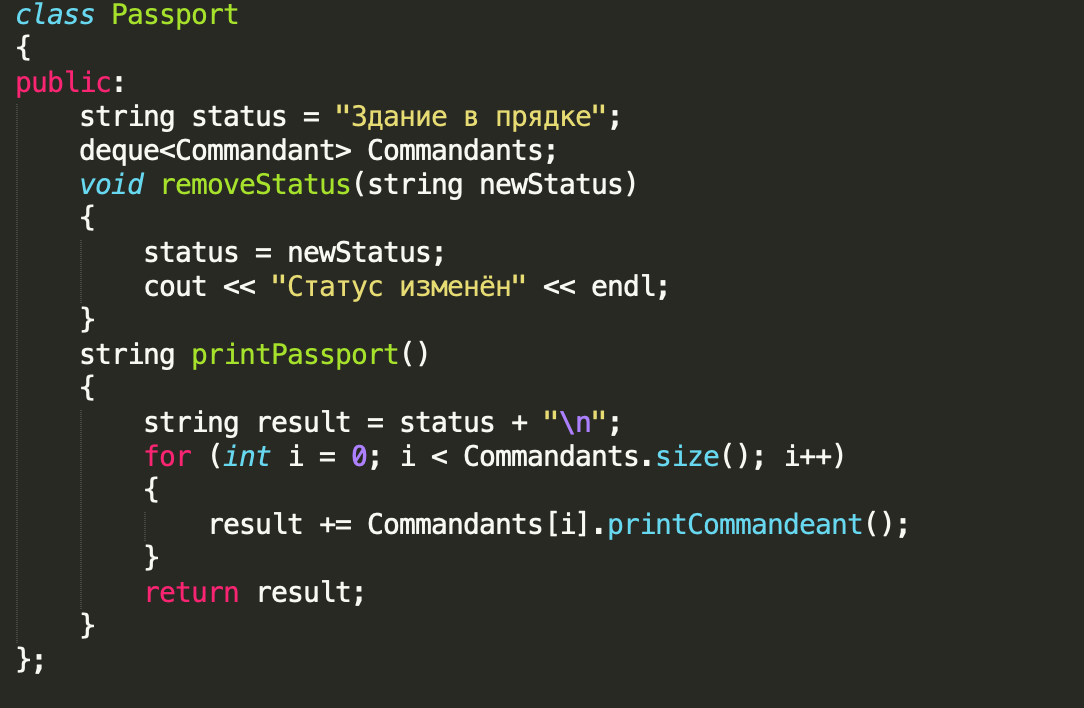


рис 2.2.2

*nameMenu* – название меню.

*listOfDishes* – список блюд в меню.

void **removeStatus**(string newStatus) - реализует механизм изменение статуса объекта.

string **printPassport()** - реализует механизм вывода данных объекта класса путём чтение полей класса и возвращение строковой переменной.

**Building** – представляет собой здание в котором есть данные о здании, его паспорт и список комендантов приписанных к зданию (рис 2.2.3).



рис 2.2.3

*sizeFloor* - количество этажей.

*sizeApartments* - количество комнат.

*Area* - площадь здания.

*Passport* - паспорт объекта.

void **printBuilding ()** - реализует механизм присвоение данных объекту класса.

string **printCommandeant()** - реализует механизм вывода данных объекта класса путём чтение полей класса и вывод их на консоль.

## 2.3. Результаты работы программы

Результат работы программы представляет собой вывод данных из структур, которые были описаны в предыдущий главе.

# 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 (рис 3.1). Для осуществления выбора необходимо ввести соответствующую цифру в консоль.

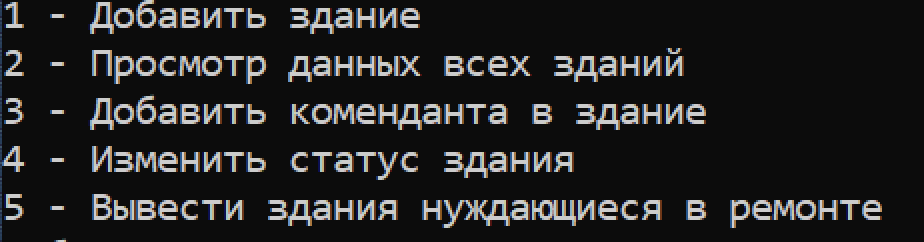


Рис 3.1

При выборе пункта № 1 в консоль необходимо ввести данные здания (рис 3.2)

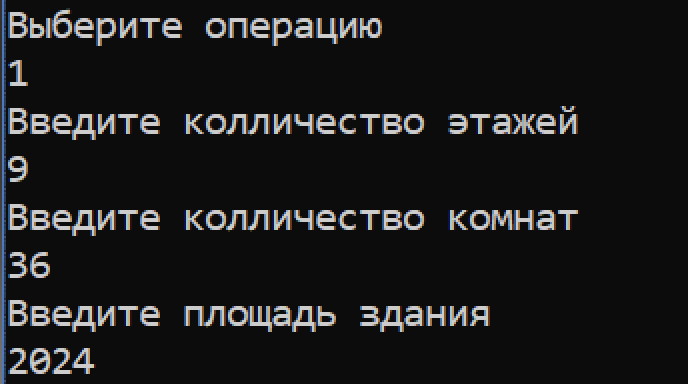


Рис 3.2

При выборе пункта № 2 на консоль выводиться все данные о зданиях (информация о здании, паспорт здания, коменданты, приписанные к зданию) (рис 3.3).

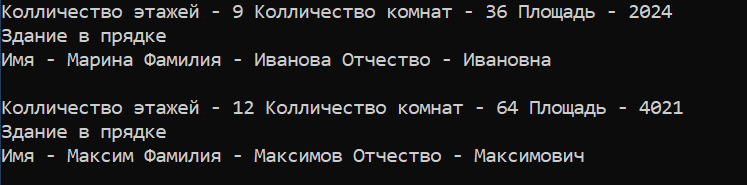


рис 3.3

При выборе пункта № 3 вначале необходимо произвести выбор здание, в которое будет добавляться комендант. После успешного выбора, необходимо ввести данные самого коменданта (Рис 3.4).

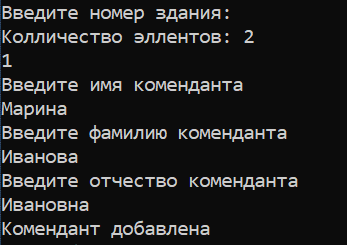


Рис 3.4

При выборе пункта № 4 вначале необходимо произвести выбор здание, в котором будет происходить изменение статуса здания. После успешного выбора, необходимо произвести выбор какой именно статус присвоить зданию: Требуется капитальный ремонт, требуется косметический ремонт Здание в прядке (Рис 3.5).

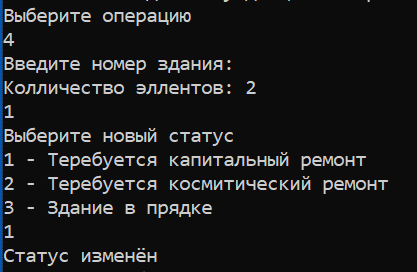


Рис 3.5

При выборе пункта № 3 вначале необходимо произвести выбор, по какому критерию необходимо вывести данные (капитальный ремонт, косметический ремонт). После успешного выбора, происходит вывод данных (Рис 3.6).

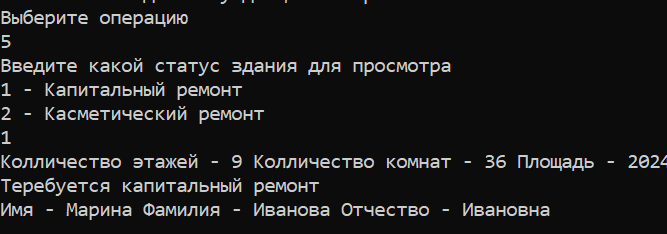


Рис 3.5

# ЗАКЛЮЧЕНИЕ

Исходя из поставленных целей и задач было сконструировано, разработана и протестирован программный модуль «Бизнес- процессы подразделений, учитывающих здания и сооружения организации», предназначенный для использования работниками подразделения.

Были реализованы задачи:

1. Паспортизация зданий и сооружений
2. Учет капитальных и профилактических ремонтов

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.
3. Так же программы имеет высокую оптимизацию. Из-за этого программа работает весьма быстра и запускается даже на самом слабом устройстве.

Данный программный продукт выгодно отличается от других подобных тем что пользуясь им, пользователь может не только структурированно хранить данные зданий, но и возможность вести паспорт объекта и 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 <iostream>

#include <string>

#include <deque>

#include <Windows.h>

using namespace std;

class Commandant

{

public:

string surname;

string name;

string patronymic;

void addCommandeant()

{

cout << "Введите имя коменданта" << endl;

cin >> name;

cout << "Введите фамилию коменданта" << endl;

cin >> surname;

cout << "Введите отчество коменданта" << endl;

cin >> patronymic;

}

string printCommandeant()

{

return "Имя - " + name + " Фамилия - " + surname + " Отчество - " + patronymic + "\n";

}

};

class Passport

{

public:

string status = "Здание в прядке";

deque<Commandant> Commandants;

void removeStatus(string newStatus)

{

status = newStatus;

cout << "Статус изменён" << endl;

}

string printPassport()

{

string result = status + "\n";

for (int i = 0; i < Commandants.size(); i++)

{

result += Commandants[i].printCommandeant();

}

return result;

}

};

class Building

{

public:

string sizeFloor;

string sizeApartments;

string Area;

Passport passport;

void addBuilding()

{

cout << "Введите колличество этажей" <<endl;

cin >> sizeFloor;

cout << "Введите колличество комнат" << endl;

cin >> sizeApartments;

cout << "Введите площадь здания" << endl;

cin >> Area;

}

void printBuilding()

{

string result = "Колличество этажей - " + sizeFloor + " Колличество комнат - " + sizeApartments + " Площадь - " + Area + "\n";

result += passport.printPassport();

cout << result << endl;

}

};

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_ALL, "Russian");

deque<Building> Buildings;

Building building;

Commandant commandant;

while (true)

{

cout << "1 - Добавить здание" <<endl;

cout << "2 - Просмотр данных всех зданий" << endl;

cout << "3 - Добавить коменданта в здание" << endl;

cout << "4 - Изменить статус здания" << endl;

cout << "5 - Вывести здания нуждающиеся в ремонте" << endl;

int select;

int selectStatus;

int selectPrintStatus;

cout << "Выберите операцию" <<endl;

cin >> select;

int number;

bool selection = false;

switch (select)

{

case 1:

building.addBuilding();

Buildings.push\_back(building);

break;

case 2:

for (int i = 0; i < Buildings.size(); i++)

{

Buildings[i].printBuilding();

}

break;

case 3:

cout << "Введите номер здания:" << endl;

cout << "Колличество эллентов: " << Buildings.size() << endl;

cin >> number;

for (int i = 0; i < Buildings.size(); i++)

{

if (i == number - 1)

{

commandant.addCommandeant();

Buildings[i].passport.Commandants.push\_back(commandant);

selection = true;

cout << "Комендант добавлена" << endl;

}

}

if (!selection)

{

cout << "Данного номера не существует" << endl;

}

break;

case 4:

cout << "Введите номер здания:" << endl;

cout << "Колличество эллентов: " << Buildings.size() << endl;

cin >> number;

for (int i = 0; i < Buildings.size(); i++)

{

if (i == number - 1)

{

cout <<"Выберите новый статус" <<endl;

cout << "1 - Теребуется капитальный ремонт" << endl;

cout << "2 - Теребуется космитический ремонт" << endl;

cout << "3 - Здание в прядке" << endl;

cin >> selectStatus;

switch (selectStatus)

{

case 1:

Buildings[i].passport.removeStatus("Теребуется капитальный ремонт");

break;

case 2:

Buildings[i].passport.removeStatus("Теребуется космитический ремонт");

break;

case 3:

Buildings[i].passport.removeStatus("Здание в прядке");

break;

default:

cout << "Неверно введен пункт" << endl;

break;

}

selection = true;

cout << "Комендант добавлена" << endl;

}

}

if (!selection)

{

cout << "Данного номера не существует" << endl;

}

break;

case 5:

cout << "Введите какой статус здания для просмотра" <<endl;

cout << "1 - Капитальный ремонт" <<endl;

cout << "2 - Касметический ремонт" << endl;

cin >> selectPrintStatus;

switch (selectPrintStatus)

{

case 1:

for (int i = 0; i < Buildings.size(); i++)

{

if (Buildings[i].passport.status == "Теребуется капитальный ремонт")

{

Buildings[i].printBuilding();

}

}

break;

case 2:

for (int i = 0; i < Buildings.size(); i++)

{

if (Buildings[i].passport.status == "Теребуется космитический ремонт")

{

Buildings[i].printBuilding();

}

}

break;

default:

cout << "Неверно введен пункт" << endl;

break;

}

break;

default:

cout << "Неверно введен пункт" <<endl;

break;

}

}

}