|  |  |
| --- | --- |
| Институт (факультет) | Институт информационных технологий |
| Кафедра | Математического и программного обеспечения ЭВМ |

# КУРСОВАЯ РАБОТА

|  |
| --- |
| по дисциплине Объектно-ориентированное программирование |

|  |  |
| --- | --- |
| на тему | Объектно-ориентированное программирование на языке С++ |
|  |  |

|  |
| --- |
| Выполнил студент группы 1ИСб-00-21оп |
| *группа* |
| направления подготовки (специальности) |
| 09.03.02 Информационные системы и технологии |
| *шифр, наименование* |
| Дмитриев Максим Витальевич |
| *фамилия, имя, отчество* |

|  |
| --- |
| Руководитель |
| **Шаханов Никита Иванович** |
| *фамилия, имя, отчество* |
| аспирант |
| *должность* |

|  |
| --- |
| Дата представления работы |
| «\_\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20 \_\_\_ г. |
|  |
| Заключение о допуске к защите |
|  |
|  |
|  |
|  |
| Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| количество баллов |
| Подпись преподавателя\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Череповец, 2020

*год*

# Аннотация

Данная расчётно-пояснительная записка содержит описание выполнения задания к курсовой работе по дисциплине «Объектно-ориентированное программирование». В ходе выполнения работы была изучена предметная область в соответствии с заданием, а также разработана и реализована иерархия родственных классов, описывающая предметную область. Для моделирования и обработки данных были использованы методы работы в объектно-ориентированном программировании (в частности, полиморфная обработка родственных объектов). Результатом работы стало приложение на языке программирования С++.

Записка содержит 56 страниц, в том числе 6 источников литературы и 3 приложения

**Оглавление**

[1. Введение 3](#_Toc44057250)

[2. Основная часть 5](#_Toc44057251)

[1. Изучение и описание предметной области 5](#_Toc44057252)

[2. Проектирование классов 6](#_Toc44057253)

[3. Логическое проектирование 9](#_Toc44057254)

[4. Модульная структура программы 9](#_Toc44057255)

[5. Тестирование 13](#_Toc44057256)

[3. Заключение 18](#_Toc44057257)

[4. Список литературы 18](#_Toc44057258)

[Приложения: 18](#_Toc44057259)

[Приложение 1. Техническое задание 19](#_Toc44057260)

[Приложение 2. Руководство пользователя 26](#_Toc44057261)

[Приложение 3. Код программы 33](#_Toc44057262)

# 1. Введение

**Объектно-ориентированное программирование (ООП)** – это программирование, сфокусированное на данных, поведение и данные которых неразрывно связаны.

Сложность современного программного обеспечения требует от разработчиков владения перспективными технологиями его создания. Одной из таких технологий на данный момент является ООП. Применение ООП позволяет разрабатывать программное обеспечение повышенной сложности за счёт улучшения его технологичности (лучших механизмов разделения данных, увеличения повторяемости кодов, использования стандартизированных интерфейсов пользователя и т.д.).

В теории программирования ООП определяется как *технология создания сложного программного обеспечения, основанная на представлении программы в виде совокупности программных объектов, каждый из которых является экземпляром определённого типа (класса), а классы образуют иерархию с наследованием свойств*. Взаимодействие программных объектов в такой системе осуществляется путём передачи сообщений (рис. 1).

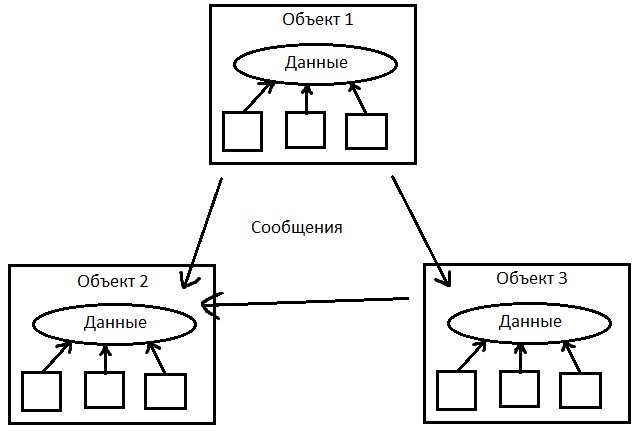


Рис. 1. Организация программы при использовании объектного подхода

Основное достоинство ООП – сокращение количества межмодульных вызовов и уменьшение объёмов информации, передаваемой между модулями, за счёт более полной локализации данных и интегрирования их с подпрограммами обработки, что позволяет вести практически независимую разработку отдельных частей (объектов) программы.

Наиболее существенный недостаток ООП – снижение быстродействия за счёт увеличения количества передач управления между подпрограммами.

ООП появилось в результате длительной эволюции технологии разработки программных продуктов, причиной которой стало стремление ускорить процесс создания надёжных программных средств. Существующие в настоящее время объектные модели различных версий универсальных языков программирования имеют свои особенности, но в их основе лежат единые концепции.

Концепции ООП – это основные идеи, заложенные в нём:

1. **Реальное моделирование** решаемых задач;
2. **Пользовательские типы данных**;
3. **Инкапсуляция** (сокрытие деталей реализации);
4. Повышение степени повторного использования кода благодаря **наследованию** и **шаблонам**;
5. Интерпретация вызова функции на этапе выполнения (**механизм замещения, переопределения, перекрытия**).

# 2. Основная часть

# 1. Изучение и описание предметной области

Согласно поставленному заданию необходимо сформировать иерархию родственных объектов для моделирования и обработки данных предметной области. Согласно индивидуальному варианту данной областью являются **Видеомониторы**. Для хранения объектов сформированной иерархии должна использоваться структура данных в виде контейнера. Согласно варианту контейнером является **линейный односвязный список**. В данной программе списком будет служить **стек**.

Главная функция видеомонитора – графическое отображение сигналов, поступающих на вход. На данном этапе все мониторы можно разделить на мониторы с **разъёмом VGA** для получения графической информации от компьютера и на **телевизоры**, получающих сигналы от антенны. Работа VGA-монитора может быть спроекцирована на полотно при помощи **проектора**, обладающего как минимум теми же свойствами, что и VGA-монитор. Работа телевизора может быть улучшена использованием **домашнего кинотеатра** с возможностью объёмного звучания. Также существуют монитор, объединяющий в себе свойства как VGA-мониторов для отображения данных компьютера, так и телевизора для наличия звука – это **HDMI-монитор**.

Таким образом, получившаяся иерархия родственных классов графически представлена на рис. 2.

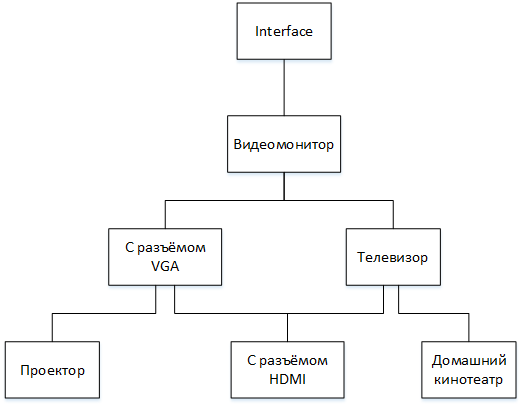


Рис. 2. Контекстная диаграмма классов

# 2. Проектирование классов

Для хранения объектов иерархии необходимо создать контейнерный класс. В данной работе таковым является стек. Для стека, как и для любого другого линейного односвязного списка, характерно присутствие внутренней структуры, содержащей, непосредственно, объект хранения и указатель на следующий элемент. Таким образом, диаграмма классов контейнера представлена на рис. 3.1 (см. след. стр.).

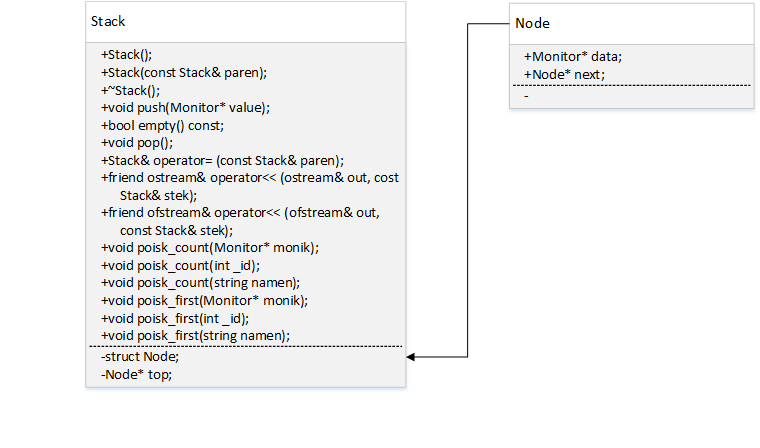


Рис. 3.1. Диаграмма классов контейнера

Структура **Node**:

- **Monitor\* data** – объект на хранении;

- **Node\* next** – указатель на следующий элемент списка;

Класс **Stack**:

- Конструкторы и деструктор;

- **Node\* top** – вершина стека;

- **void push(Monitor\* value)**, **void pop()** – добавление и удаление элемента соответственно;

- **bool empty()** – проверка списка на пустоту;

- Перегрузки операторов = и <<;

- **void poisk\_count(…)** – подсчёт интересующих мониторов;

- **void poisk\_first(…)** – поиск первого интересующего монитора;

Любому видеомонитору присущи как минимум три параметра – **ширина** экрана, **высота** экрана и **название** самого монитора. В рамках данной работы предполагается, что у всех видеомониторов также присутствует параметр в виде **индивидуального номера** (**ID**).

Отличием телевизора от простого монитора является наличие звука, поэтому добавляется параметр **максимальной громкости**, в то время как у VGA-мониторов имеется параметр **частоты обновления кадра**. Оба этих параметра присутствуют у HDMI-монитора.

Так как проектор предназначен для вывода изображения на полотно, то у него также имеются параметры **минимальной** и **максимальной дальностей** проекции. Домашний кинотеатр обладает параметром наличия объёмного звука.

Таким образом, детальная диаграмма классов иерархии представлена на рис. 3.2.

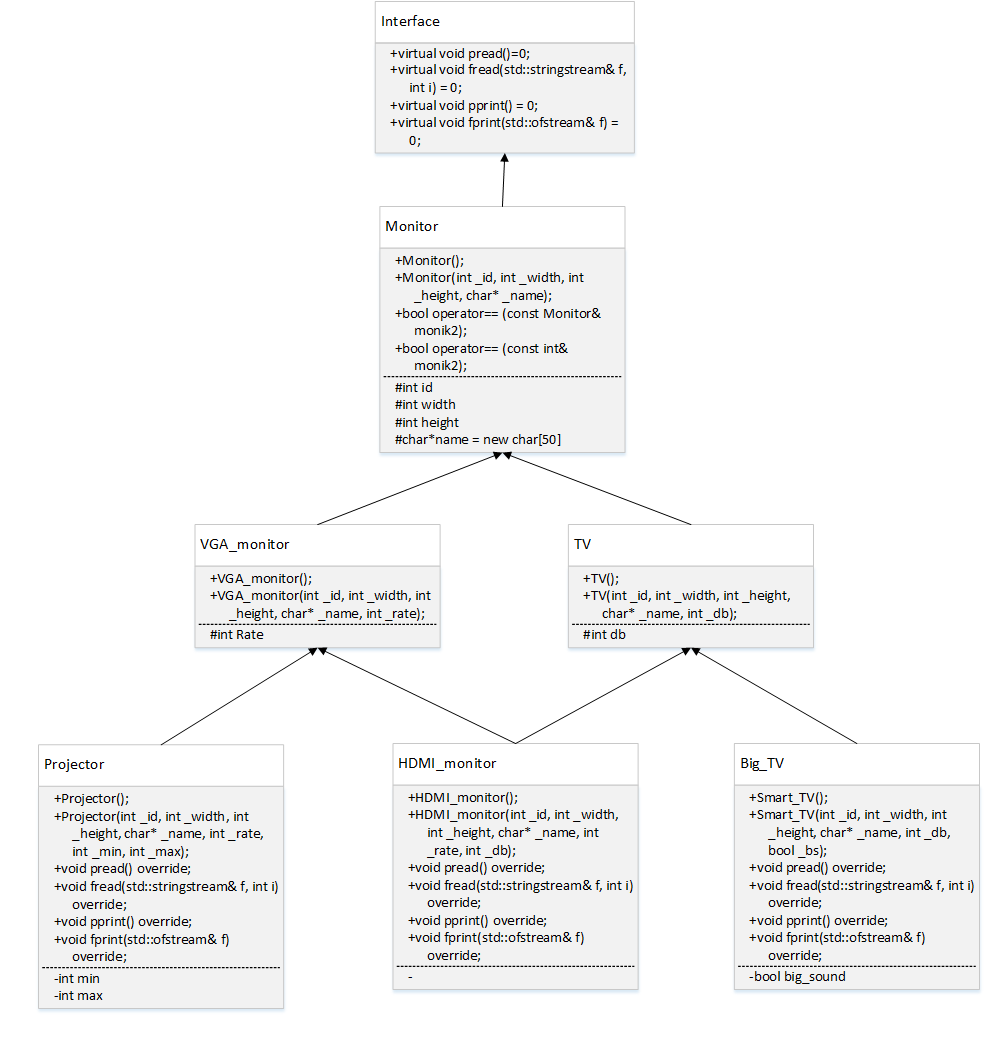


Рис. 3.2. Детальная диаграмма классов

Класс **Interface**:

* **void pread()** – ввод данных о мониторе с клавиатуры, виртуальный отложенный метод;
* **void pprint()** – ввод данных о мониторе из файла, виртуальный отложенный метод;
* **void fread(std::stringstream& f, int i)** – вывод данных о мониторе на экран, виртуальный отложенный метод;
* **void fprint(std::ofstream& f)** – вывод данных о мониторе в файл, виртуальный отложенный метод;

Класс **Monitor**:

* Конструкторы (по умолчанию и с параметрами);
* Переопределённые методы pread, fread, pprint, fprint;
* Поля:
  + **int id** – индивидуальный номер;
  + **int width**, **int height** – ширина и высота экрана;
  + **char\* name** – наименование;

Класс **VGA\_monitor**:

* Конструкторы (по умолчанию и с параметрами);
* Переопределённые методы pread, fread, pprint, fprint;
* Поля:
  + **int rate** – частота обновления изображения;

Класс **TV**:

* Конструкторы (по умолчанию и с параметрами);
* Переопределённые методы pread, fread, pprint, fprint;
* Поля:
  + **int db** – максимальная громкость;

Класс **HDMI\_monitor**:

* Конструкторы (по умолчанию и с параметрами);
* Переопределённые методы pread, fread, pprint, fprint;

Класс **Projector**:

* Конструкторы (по умолчанию и с параметрами);
* Переопределённые методы pread, fread, pprint, fprint;
* Поля:
  + **int min**, **int max** – минимальная и максимальная дальность проекции;

Класс **Big\_TV**:

* Конструкторы (по умолчанию и с параметрами);
* Переопределённые методы pread, fread, pprint, fprint;
* Поля:
  + **bool big\_sound** – наличие объёмного звучания;

# 3. Логическое проектирование

Логическая структура программы, рассматриваемой в работе, представлена в рис. 4.

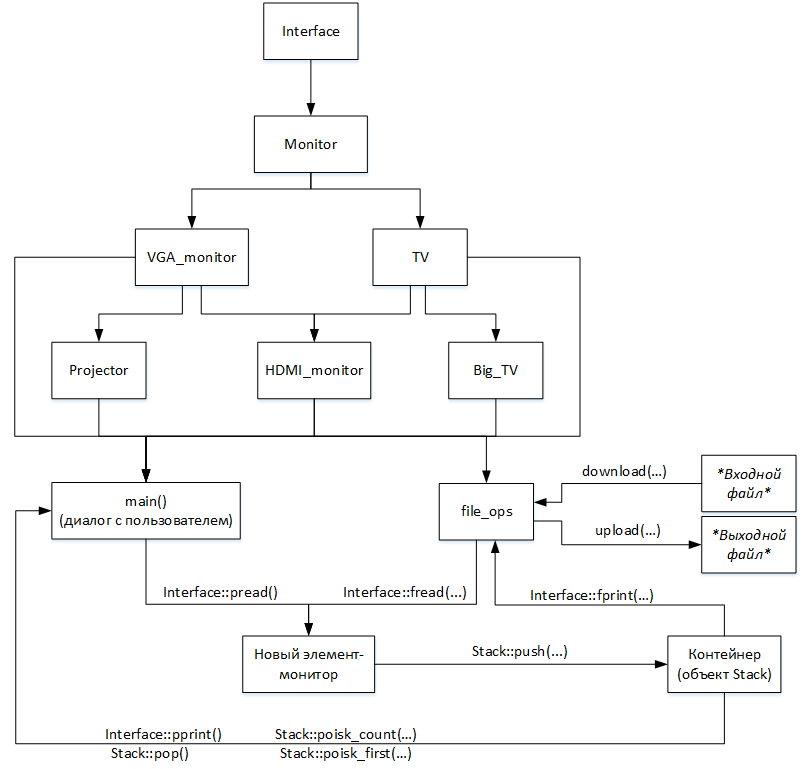


Рис. 4. Логическая структура программы

Для добавления нового элемента в список необходимо создать новый элемент-монитор. Это можно сделать либо из файла, используя метод fread при операции download, либо ввести данные с клавиатуры, используя метод pread. Новый монитор добавляется в список посредством метода списка push.

Вывести информацию о крайнем элементе списка можно либо на экран через метод pprint, либо в файл через метод fprint при операции upload.

При взаимодействии с программой пользователь может использовать операции списка типа удаления крайнего элемента (pop), подсчёта (poisk\_count) и поиска (poisk\_first) интересующего монитора.

# 4. Модульная структура программы

Модульная структура программы представлена на рис. 5.

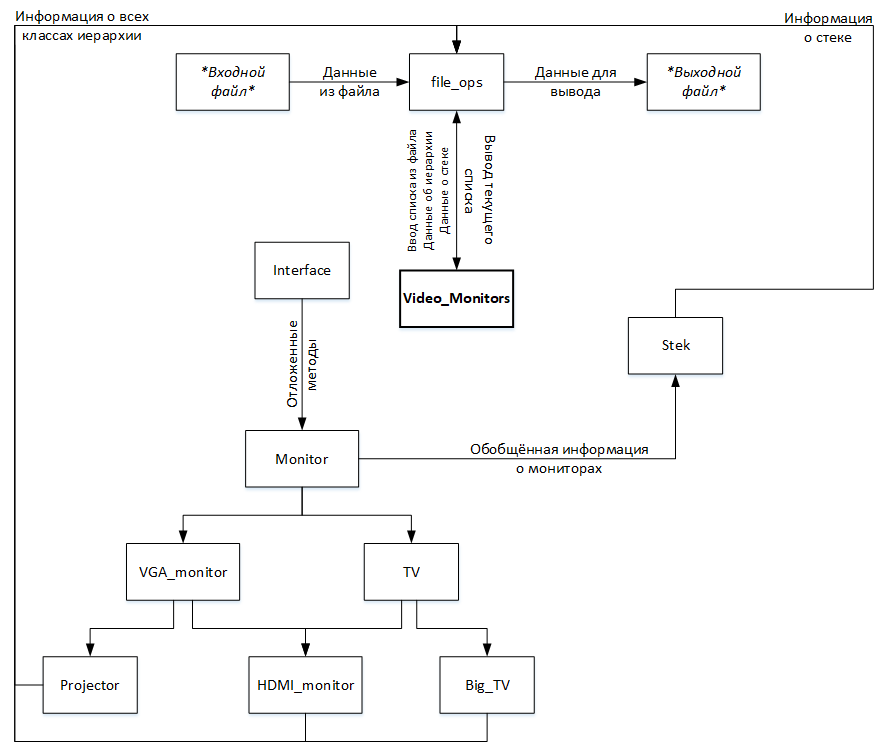


Рис. 5. Модульная структура программы

* **Interface** – содержит отложенные методы классов иерархии;
* **Monitor** – содержит обобщённые данные и методы видеомониторов;
* **VGA\_monitor**, **TV**, **HDMI\_monitor**, **Projector**, **Big\_TV** – содержат индивидуальную реализацию каждого класса иерархии;
* **Stek** – содержит контейнерный класс и методы работы со списком. Использует данные о видеомониторах, следовательно, вызывает модуль Monitor;
* **file\_ops** – содержит методы считывания и записи данных в файл. Использует данные обо всех объектах иерархии, а также методы работы со списком. Следовательно, вызывает модули Projector, HDMI\_monitor, Big\_TV и Stek;
* **Video\_Monitors** – главный модуль программы, ответственный за совместное использование данных и за взаимодействие с пользователем. Вызывает модуль file\_ops для взаимодействия программы с файлами. Через file\_ops вызываются модули Projector, HDMI\_monitor, Big\_TV и Stek.

Более детальная информация о каждом модуле представлена в табл. 1.

Табл. 1. Описание модулей программы

|  |  |  |  |
| --- | --- | --- | --- |
| Имя модуля | Заголовок процедуры или функции | Формальные параметры | Выполняемое действие |
| Monitor | Monitor | - | Конструктор по умолчанию |
| int \_id  int \_width  int \_height  char\* \_name | Конструктор с параметрами |
| VGA\_monitor | VGA\_monitor | - | Конструктор по умолчанию |
| int \_id  int \_width  int \_height  char\* \_name  int \_rate | Конструктор с параметрами |
| TV | TV | - | Конструктор по умолчанию |

Продолжение табл. 1

|  |  |  |  |
| --- | --- | --- | --- |
| Имя модуля | Заголовок процедуры или функции | Формальные параметры | Выполняемое действие |
| TV | TV | int \_id  int \_width  int \_height  char\* \_name  int \_db | Конструктор с параметрами |
| Projector | Projector | - | Конструктор по умолчанию |
| int \_id  int \_width  int \_height  char\* \_name  int \_rate  int \_min  int \_max | Конструктор с параметрами |
| HDMI\_monitor | HDMI\_monitor | - | Конструктор по умолчанию |
| int \_id  int \_width  int \_height  char\* \_name  int \_rate  int \_db | Конструктор с параметрами |
| Big\_TV | Big\_TV | - | Конструктор по умолчанию |
| int \_id  int \_width  int \_height  char\* \_name  int \_db  bool \_bs | Конструктор с параметрами |
| Interface | void pread | - | Ввод данных о мониторе с клавиатуры |
| Monitor |
| VGA\_monitor |
| TV |
| Projector |
| HDMI\_monitor |
| Big\_TV |

Продолжение табл. 1

|  |  |  |  |
| --- | --- | --- | --- |
| Имя модуля | Заголовок процедуры или функции | Формальные параметры | Выполняемое действие |
| Interface | void fread | std::stringstream& f  int i | Ввод данных о мониторе из файла |
| Monitor |
| VGA\_monitor |
| TV |
| Projector |
| HDMI\_monitor |
| Big\_TV |
| Interface | void pprint | - | Вывод данных о мониторе на экран |
| Monitor |
| VGA\_monitor |
| TV |
| Projector |
| HDMI\_monitor |
| Big\_TV |
| Interface | void fprint | std::ofstream& f | Вывод данных о мониторе в файл |
| Monitor |
| VGA\_monitor |
| TV |
| Projector |
| HDMI\_monitor |
| Big\_TV |
| Stek | void push | Monitor\* value | Добавление элемента-монитора в вершину стека |
| void pop | - | Удаление элемента с вершины стека |
| bool empty | - | Проверка стека на пустоту |

Продолжение табл. 1

|  |  |  |  |
| --- | --- | --- | --- |
| Имя модуля | Заголовок процедуры или функции | Формальные параметры | Выполняемое действие |
| Stek | void poisk\_count | Monitor\* monik | Подсчёт мониторов в стеке по габаритам |
| int \_id | Подсчёт мониторов в стеке по индивидуальному номеру |
| string namen | Подсчёт в стеке мониторов определённого типа |
|  | void poisk\_first | Monitor\* monik | Нахождение первого монитора, удовлетворяющего условию – габаритам |
| int \_id | Нахождение первого монитора, удовлетворяющего условию – индивидуальному номеру |
| string namen | Нахождение первого монитора, удовлетворяющего условию – типу |
| file\_ops | void download | string path  Stack& moniks | Считывание данных из файла, занесение считанных элементов в список |
| void upload | string path  Stack moniks | Запись элементов списка в файл |
| Video\_Monitors | int main | - | Взаимодействие с пользователем, создание и редактирование списка-стека, запись и вывод данных об элементах списка |

# 5. Тестирование

Для проверки работоспособности программы в целом и отдельных модулей в частности было проведено тестирвание. Результаты приведены в табл. 5.1 - 5.9.

Таблица 5.1. Тестирование модуля Video\_Monitors

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата и время тестирования | Тестируемая подпрограмма | Кто проводил тестирование | Описание теста | Результаты тестирования |
| 02.03.2020 | main() | Дмитриев М.В. | Диалог программы и пользователя | Успех |
| 29.04.2020 | main() | Дмитриев М.В. | Инициализация и изменение списка | Успех |
| 15.05.2020 | main() | Дмитриев М.В. | Файловый ввод-вывод списка | Успех |

Таблица 5.2. Тестирование модуля Monitor

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата и время тестирования | Тестируемая подпрограмма | Кто проводил тестирование | Описание теста | Результаты тестирования |
| 15.03.2020 | Monitor() | Дмитриев М.В. | Создание объекта конструктором по умолчанию | Успех |
| 15.03.2020 | Monitor(int \_id, int \_width, int \_height, char\* \_name) | Дмитриев М.В. | Создание объекта конструктором с параметрами | Успех |
| 15.03.2020 | pread() | Дмитриев М.В. | Считывание данных об объекте с клавиатуры | Успех |
| 15.03.2020 | pprint() | Дмитриев М.В. | Вывод данных об объекте на экран | Успех |
| 15.03.2020 | fread(std::stringstream& f, int i) | Дмитриев М.В. | Считвание данных об объекте из строки файла | Успех |
| 15.03.2020 | fprint(std::ofstream& f) | Дмитриев М.В. | Вывод данных об объекте в файл | Успех |

Таблица 5.3. Тестирование модуля VGA\_monitor

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата и время тестирования | Тестируемая подпрограмма | Кто проводил тестирование | Описание теста | Результаты тестирования |
| 15.03.2020 | VGA\_monitor() | Дмитриев М.В. | Создание объекта конструктором по умолчанию | Успех |
| 15.03.2020 | VGA\_monitor(int \_id, int \_width, int \_height, char\* \_name, int \_rate) | Дмитриев М.В. | Создание объекта конструктором с параметрами | Успех |
| 15.03.2020 | pread() | Дмитриев М.В. | Считывание данных об объекте с клавиатуры | Успех |
| 15.03.2020 | pprint() | Дмитриев М.В. | Вывод данных об объекте на экран | Успех |
| 15.03.2020 | fread(std::stringstream& f, int i) | Дмитриев М.В. | Считвание данных об объекте из строки файла | Успех |
| 15.03.2020 | fprint(std::ofstream& f) | Дмитриев М.В. | Вывод данных об объекте в файл | Успех |

Таблица 4. Тестирование модуля TV

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата и время тестирования | Тестируемая подпрограмма | Кто проводил тестирование | Описание теста | Результаты тестирования |
| 15.03.2020 | TV() | Дмитриев М.В. | Создание объекта конструктором по умолчанию | Успех |
| 15.03.2020 | TV(int \_id, int \_width, int \_height, char\* \_name, int \_db) | Дмитриев М.В. | Создание объекта конструктором с параметрами | Успех |
| 15.03.2020 | pread() | Дмитриев М.В. | Считывание данных об объекте с клавиатуры | Успех |
| 15.03.2020 | pprint() | Дмитриев М.В. | Вывод данных об объекте на экран | Успех |
| 15.03.2020 | fread(std::stringstream& f, int i) | Дмитриев М.В. | Считвание данных об объекте из строки файла | Успех |
| 15.03.2020 | fprint(std::ofstream& f) | Дмитриев М.В. | Вывод данных об объекте в файл | Успех |

Таблица 5. Тестирование модуля HDMI\_monitor

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата и время тестирования | Тестируемая подпрограмма | Кто проводил тестирование | Описание теста | Результаты тестирования |
| 15.03.2020 | HDMI\_monitor() | Дмитриев М.В. | Создание объекта конструктором по умолчанию | Успех |
| 15.03.2020 | HDMI\_monitor(int \_id, int \_width, int \_height, char\* \_name, int \_rate, int \_db, ) | Дмитриев М.В. | Создание объекта конструктором с параметрами | Успех |
| 15.03.2020 | pread() | Дмитриев М.В. | Считывание данных об объекте с клавиатуры | Успех |
| 15.03.2020 | pprint() | Дмитриев М.В. | Вывод данных об объекте на экран | Успех |
| 15.03.2020 | fread(std::stringstream& f, int i) | Дмитриев М.В. | Считвание данных об объекте из строки файла | Успех |
| 15.03.2020 | fprint(std::ofstream& f) | Дмитриев М.В. | Вывод данных об объекте в файл | Успех |

Таблица 6. Тестирование модуля Projector

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата и время тестирования | Тестируемая подпрограмма | Кто проводил тестирование | Описание теста | Результаты тестирования |
| 15.03.2020 | Projector() | Дмитриев М.В. | Создание объекта конструктором по умолчанию | Успех |
| 15.03.2020 | Projector(int \_id, int \_width, int \_height, char\* \_name, int \_rate, int \_min, int \_max) | Дмитриев М.В. | Создание объекта конструктором с параметрами | Успех |
| 15.03.2020 | pread() | Дмитриев М.В. | Считывание данных об объекте с клавиатуры | Успех |
| 15.03.2020 | pprint() | Дмитриев М.В. | Вывод данных об объекте на экран | Успех |
| 15.03.2020 | fread(std::stringstream& f, int i) | Дмитриев М.В. | Считвание данных об объекте из строки файла | Успех |
| 15.03.2020 | fprint(std::ofstream& f) | Дмитриев М.В. | Вывод данных об объекте в файл | Успех |

Таблица 7. Тестирование модуля Big\_TV

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата и время тестирования | Тестируемая подпрограмма | Кто проводил тестирование | Описание теста | Результаты тестирования |
| 15.03.2020 | Big\_TV() | Дмитриев М.В. | Создание объекта конструктором по умолчанию | Успех |
| 15.03.2020 | Big\_TV(int \_id, int \_width, int \_height, char\* \_name, int \_db, bool \_bs) | Дмитриев М.В. | Создание объекта конструктором с параметрами | Успех |
| 15.03.2020 | pread() | Дмитриев М.В. | Считывание данных об объекте с клавиатуры | Успех |
| 15.03.2020 | pprint() | Дмитриев М.В. | Вывод данных об объекте на экран | Успех |
| 15.03.2020 | fread(std::stringstream& f, int i) | Дмитриев М.В. | Считвание данных об объекте из строки файла | Успех |
| 15.03.2020 | fprint(std::ofstream& f) | Дмитриев М.В. | Вывод данных об объекте в файл | Успех |

Таблица 8. Тестирование модуля Stek

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата и время тестирования | Тестируемая подпрограмма | Кто проводил тестирование | Описание теста | Результаты тестирования |
| 18.04.2020 | Stack() | Дмитриев М.В. | Создание объекта конструктором по умолчанию | Успех |
| 18.04.2020 | Stack(const Stack& paren) | Дмитриев М.В. | Создание объекта конструктором с параметрами | Успех |
| 18.04.2020 | ~Stack() | Дмитриев М.В. | Уничтожение списка | Успех |
| 18.04.2020 | push(Monitor\* value) | Дмитриев М.В. | Добавление элемента в список | Успех |
| 18.04.2020 | empty() | Дмитриев М.В. | Проверка списка на пустоту | Успех |
| 18.04.2020 | pop() | Дмитриев М.В. | Удаление последнего элемента из списка | Успех |

Продолжение таблицы 8.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата и время тестирования | Тестируемая подпрограмма | Кто проводил тестирование | Описание теста | Результаты тестирования |
| 18.04.2020 | poisk\_count(Monitor\* monik) | Дмитриев М.В. | Подсчёт элементов в списке по габаритам | Успех |
| 18.04.2020 | poisk\_count(int \_id) | Дмитриев М.В. | Подсчёт элементов в списке по id | Успех |
| 18.04.2020 | poisk\_count(string namen) | Дмитриев М.В. | Подсчёт элементов в списке по типу | Успех |
| 18.04.2020 | poisk\_first(Monitor\* monik) | Дмитриев М.В. | Поиск элемента в списке по габаритам | Успех |
| 18.04.2020 | poisk\_first(int \_id) | Дмитриев М.В. | Поиск элемента в списке по id | Успех |
| 18.04.2020 | poisk\_first(string namen) | Дмитриев М.В. | Поиск элемента в списке по типу | Успех |

Таблица 9. Тестирование модуля file\_ops

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата и время тестирования | Тестируемая подпрограмма | Кто проводил тестирование | Описание теста | Результаты тестирования |
| 28.04.2020 | download(string path, Stack& moniks) | Дмитриев М.В. | Загрузка списка из файла | Успех |
| 28.04.2020 | upload(string path, Stack moniks) | Дмитриев М.В. | Выгрузка текущего списка в файл | Успех |

# 3. Заключение

В ходе выполнения работы была изучена предметная область в соответствии с заданием, а также разработана и реализована иерархия родственных классов, описывающая предметную область. Для моделирования и обработки данных были использованы методы работы в объектно-ориентированном программировании (в частности, полиморфная обработка родственных объектов). Результатом работы стало приложение на языке программирования С++.

Данная работа была направлена на закрепление знаний, полученных за курс по дисциплине "Объектно-ориентированное программирование".

# 4. Список литературы

1. «Объектно-ориентированное программирование» – Иванова Г.С., Ничушкина Т.Н.;
2. «Указания к лабораторным работам по дисциплине «Объектно-ориентированное программирование». Часть 1» – ЧГУ;
3. «Указания к лабораторным работам по дисциплине «Объектно-ориентированное программирование». Часть 2» – ЧГУ;
4. «Объектно-ориентированное программирование в С++» – Лафоре Р.;
5. Онлайн-ресурс https://skillbox.ru/;
6. Онлайн-ресурс https://habr.com/;

# Приложения:

# Приложение 1

МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

|  |
| --- |
| Институт информационных технологий |
| наименование института(факультета) |
| Математически-программное обеспечение ЭВМ |
| наименование кафедры |
| Объектно-ориентированное программирование |
| наименование дисциплины в соответствии с учебным планом |

УТВЕРЖДАЮ

Зав. кафедрой ПО ВТ и ИС,

д. т.н., профессор\_\_\_\_\_\_\_\_\_\_ Ершов Е.В.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ С++

Техническое задание на курсовой проект (работу)

Листов 6

Руководитель Шаханов Никита Иванович

ФИО преподавателя

Исполнитель

студент 1ИСб-00-21оп

группа

Дмитриев Максим Витальевич

Фамилия, имя, отчество

2020 год

**Введение**

**Объектно-ориентированное программирование (ООП)** – это программирование, сфокусированное на данных, поведение и данные которых неразрывно связаны.

Сложность современного программного обеспечения требует от разработчиков владения перспективными технологиями его создания. Одной из таких технологий на данный момент является ООП. Применение ООП позволяет разрабатывать программное обеспечение повышенной сложности за счёт улучшения его технологичности (лучших механизмов разделения данных, увеличения повторяемости кодов, использования стандартизированных интерфейсов пользователя и т.д.).

ООП появилось в результате длительной эволюции технологии разработки программных продуктов, причиной которой стало стремление ускорить процесс создания надёжных программных средств. Существующие в настоящее время объектные модели различных версий универсальных языков программирования имеют свои особенности, но в их основе лежат единые концепции.

Концепции ООП – это основные идеи, заложенные в нём:

1. Реальное моделирование решаемых задач;
2. Пользовательские типы данных;
3. Инкапсуляция (сокрытие деталей реализации);
4. Повышение степени повторного использования кода благодаря наследованию и шаблонам;
5. Интерпретация вызова функции на этапе выполнения (механизм замещения, переопределения, перекрытия).

Настоящее техническое задание относится к курсовой работе и является демонстрацией процесса разработки и сдачи программной продукции, назначение которой описано в расчётно-пояснительной записке и далее в данном приложении.

**Основания для разработки**

Разработка продукта ведётся с задания преподавателя кафедры Математического и программного обеспечения ЭВМ по дисциплине «Объектно-ориентированное программирование» Шаханова Никиты Ивановича.

Задание:

1. Разработать иерархию родственных типов, корневой класс которой абстрактный базовый класс (**класс-интерфейс)**, для моделирования и обработки данных предметной области набором отложенных методов - полиморфная обработка родственных объектов (согласно варианта «Видеомониторы»).
2. Для хранения объектов каждого производного класса использовать структуру данных (согласно варианта «Линейный односвязный список»).
3. Реализовать функции обработки данных (сортировка и поиск по выбранным полям и задаваемым диапазонам значений, другие функции, в том числе перегруженные).
4. Реализовать файловый ввод/вывод и ввод данных с клавиатуры, вывод данных на дисплей.
5. Предусмотреть обработку различных исключительных ситуаций.

**Назначение разработки**

Назначение программного продукта – моделирование и обработка данных предметной области - видеомониторов, хранение в линейном односвязном списке объектов родственных классов, обработка данных, ввод/вывод данных, а также обработка различных исключительных ситуаций

**Требования к программе**

1. **Требования к функциональным характеристикам**
2. Моделирование и обработка данных предметной области набором отложенных методов - полиморфная обработка родственных объектов (согласно варианта «Видеомониторы»);
3. Хранение объектов каждого производного класса в линейном односвязном списке;
4. Обработка данных (сортировка и поиск по выбранным полям и задаваемым диапазонам значений, другие функции, в том числе перегруженные);
5. Файловый ввод/вывод и ввод данных с клавиатуры, вывод данных на дисплей;
6. Обработка различных исключительных ситуаций.
7. **Требования к надёжности**

Ввод корректных данных, как с клавиатуры, так и из файла

1. **Условия эксплуатации**

В соответствии с требованиями носителя

1. **Требования к составу и параметрам технических средств**

* Персональный компьютер
* Мышь и/или клавиатура
* Монитор

1. **Требования к информационной и программной совместимости**

* Операционная система MS Windows XP или новее
* Пакет MS Visual Studio 2015 (не обязательно)

1. **Требования к маркировке и упаковке**

Требования не предъявляются

1. **Требования к транспортировке и хранению**

Транспортировка и хранение приложения и прилагающейся программы производятся с помощью физических носителей информации

**Требования к программной документации**

* Расчётно-пояснительная записка;
* Техническое задание;
* Результаты тестирования;
* Руководство пользователя;
* Код программы.

**Технико-экономические показатели**

Требования не предъявляются

**Стадии и этапы разработки**

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапа разработки | Сроки разработки | Результат выполнения | Отметка о выполнении |
| Создание технического задания | 03.02.2020 | Техническое задание оформлено | Выполнено |
| Анализ предметной области | 10.02.2020 | Предметная область «Видеомониторы» проанализирована | Выполнено |
| Создание иерархии родственных классов | 24.02.2020 | Создана иерархия родственных классов предметной области «Видеомониторы» | Выполнено |
| Программирование иерархии родственных объектов | 15.03.2020 | Создание модулей и классов, моделирующих созданную иерархию | Выполнено |
| Программирование контейнерного класса | 19.04.2020 | Создание модуля и класса Stek, предназначенного для хранения объектов предметной области | Выполнено |
| Добавление файловых операций загрузки и выгрузки | 29.04.2020 | Создание модуля file\_ops, предназначенного для файловых загрузки и выгрузки списка | Выполнено |
| Оформление РПЗ | 29.05.2020 | Расчётно-пояснительная записка готова к сдаче | Выполнено |

**Порядок контроля и приёмки**

Контроль выполнения работы осуществляется преподавателем. В случае обнаружения недостатков исполнитель работ обязан исправить их в срок, согласованный с преподавателем. Контроль разработки продукта осуществляется поэтапно, в соответствии с графиком, представленным в таблице ниже:

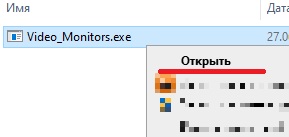
Порядок контроля и приёмки

|  |  |  |
| --- | --- | --- |
| Наименование контрольного этапа выполнения работы | Результат выполнения | Отметка и дата приёмки результата |
| Разработка технического задания | Техническое задание проверено преподавателем | Выполнено  03.02.2020 |
| Создание иерархии родственных объектов | Иерархия родственных объектов проверена преподавателем | Выполнено  27.02.2020 |
| Программирование иерархии родственных объектов | Программная реализация классов иерархии проверена преподавателем | Выполнено  17.03.2020 |
| Программирование контейнерного класса | Программная реализация контейнера проверена преподавателем | Выполнено  21.04.2020 |
| Подготовка документации | Расчётно-пояснительная записка проверена преподавателем |  |
| Защита курсовой работы | Курсовая работа защищена |  |

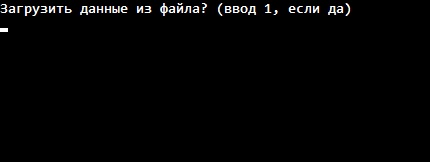
# Приложение 2

Приложение **Video\_Monitors** является обработчиком данных о видеомониторах, производя хранение таковых на так называемой «полке». На «полке» можно осуществить такие действия с мониторами, как добавление, удаление, подсчёт и поиск.

Для запуска приложения нужно открыть одноимённый файл **Video\_Monitors.exe**. Для этого нужно либо дважды нажать *Левую кнопку мыши* при наведённом на файл курсоре, либо нажать клавишу *Enter* при выделенном файле. Также можно вызвать контекстное меню (*Правая кнопка мыши*) и выбрать пункт *Открыть*:

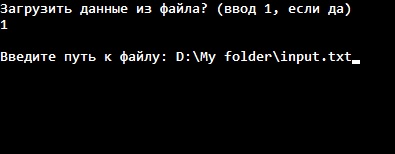


После открытия файла на экране появится следующее:



Мигающая белая черта говорит об ожидании ввода информации. На данном этапе программа уточняет, нужно ли производить загрузку имеющегося списка из файла. Если загрузка необходима, введите *1* с клавиатуры, иначе введите любой другой символ. После ввода нажмите *Enter*.

В случае подтверждения программа запросит адрес файла. Если файл находится в одном каталоге с Video\_Monitor.exe, то достаточно ввести название с расширением. Иначе необходимо ввести полный путь к файлу:

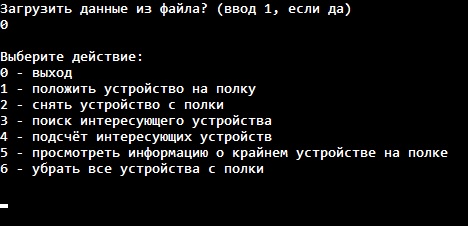


Внимание! Данные в файле должны быть записаны строго по форме (скобки писать не нужно):

|  |  |
| --- | --- |
| ***VGA-монитор*** | VGA\_monitor <*id*> <*ширина*> <*высота*> <*наименование\**> <*частота*> |
| ***Телевизор*** | TV <*id*> <*ширина*> <*высота*> <*наименование\**> <*макс*. *громкость*> |
| ***HDMI-монитор*** | HDMI\_monitor <*id>* <*ширина*> <*высота*> <*наименование\**> <*частота*> <*макс*. *громкость*> |
| ***Проектор*** | Projector <*id*> <*ширина*> <*высота*> <*наименование\**> <*частота*> <*мин*. *дистанция*> <*макс*. *дистанция*> |
| ***Домашний кинотеатр*** | Big\_TV <*id*> <*ширина*> <*высота*> <*наименование\**> <*макс*. *громкость*> <*наличие объёмного звука*> |

*\*если наименование содержит несколько слов – то оно должно быть в кавычках*

Вне зависимости от ответа далее появляется следующее меню:



Здесь на выбор предлагается несколько действий:

*1* – *положить устройство на полку*;

*2* – *снять устройство с полки*;

*3* – *поиск интересующего устройства*;

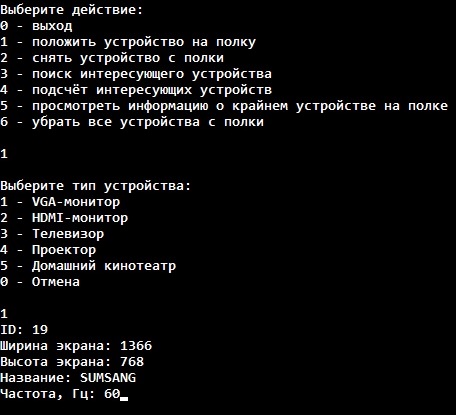
*4* – *подсчёт интересующих устройств*;

*5* – *просмотреть информацию о крайнем устройстве на полке*;

*6* – *убрать все устройства с полки*;

*0* или *др. символ* – *выход*.

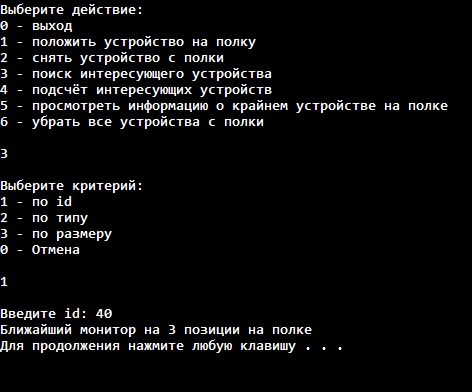
При выборе *1* предлагается выбрать тип заносимого устройства. После выбора типа необходимо ввести характеристики объекта, после чего нажать *Enter* (см. след. стр):



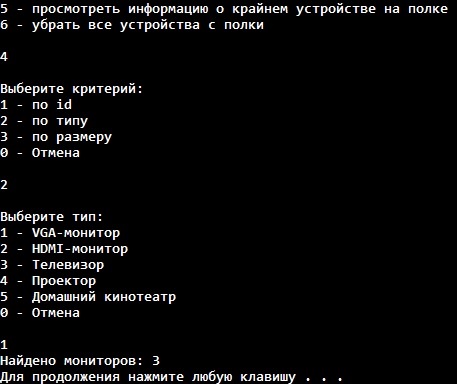
Устройство будет поставлено на «полку» перед крайним стоявшим до этого устройством.

При выборе *2* крайнее устройство снимается с «полки» – теперь крайним является стоявшее за первым.

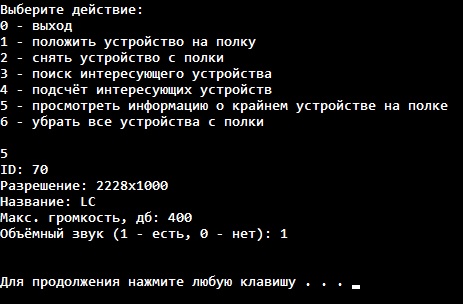
При выборе *3* происходит поиск монитора, а именно позиции, на которой находится первый монитор, удовлетворяющий условиям поиска:



Выбор *4* аналогичен выбору *3*, с той лишь разницей, что происходит подсчёт мониторов, удовлетворяющих условиям поиска:

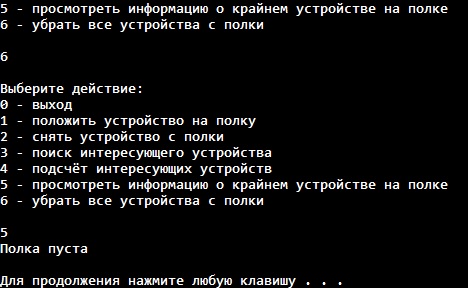


При выборе *5* на экран выводится информация о крайнем стоящем на «полке» устройстве:

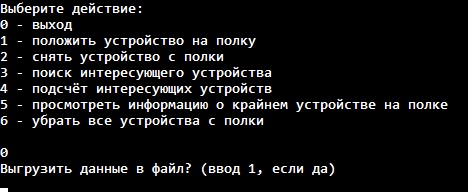


Чтобы просмотреть информацию о стоящем за крайним, необходимо снять просматриваемый монитор (см. выбор 2).

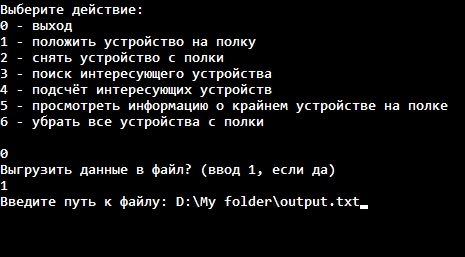
При выборе *6* список, т.е. «полка», очищается – данные обо всех мониторах становятся более недоступны:



При выборе *выхода* предлагается произвести вывод информации из текущего списка в файл:



Ответ даётся аналогично вопросу в самом начале программы, с той лишь разницей, что можно указывать несуществующий файл:



Внимание! При указании полного пути все указанные каталоги должны существовать!

Внимание! При выборе существующего файла все данные будут удалены!

# Приложение 3

**Interface.h**

#pragma once

#include <fstream>

#include <sstream>

class Interface

{

public:

virtual void pprint() = 0;

virtual void fprint(std::ofstream& f) = 0;

virtual void pread() = 0;

virtual void fread(std::stringstream& f, int i) = 0;

};

**\Monitor\Monitor.h**

#pragma once

#include <string>

#include <iostream>

#include "..\Interface.h"

class Monitor : public Interface

{

protected:

int id, width, height;

char\* name = new char[50];

public:

void pprint();

void fprint(std::ofstream& f);

void pread();

void fread(std::stringstream& f, int i);

Monitor();

Monitor(int \_id, int \_width, int \_height, char\* \_name);

bool operator== (const Monitor & monik2);

bool operator== (const int & monik2);

};

**\Monitor\Monitor.cpp**

#include "stdafx.h"

#include "Monitor.h"

using namespace std;

void Monitor::pprint() {

cout << "ID: " << id << endl

<< "Разрешение: " << width << "x" << height << endl;

printf("Название: %s\n", name);

}

void Monitor::fprint(ofstream& f) {

string type = typeid(\*this).name();

f << type.erase(0, 6) << " "

<< id << " "

<< width << " " << height << " "

<< '\"' << name << '\"';

}

void Monitor::pread() {

cout << "ID: "; scanf("%d", &id);

if (id < 0)

throw "Ошибка: введён неверный id\n";

cout << "Ширина экрана: "; scanf("%d", &width);

if (width < 0)

throw "Ошибка: введена некорректная ширина\n";

cout << "Высота экрана: "; scanf("%d", &height);

if (height < 0)

throw "Ошибка: введена некорректная высота\n";

cout << "Название: "; cin.get(); cin.getline(name, 50);

if (name[0] == 0)

throw "Ошибка: введено пустое название\n";

}

void Monitor::fread(stringstream& f, int i) {

char\* name2 = new char[50];

f >> id;

if (id < 0) {

string ext = "В записи №";

ext += \_itoa(i, new char[5], 10);

ext += " указан некорректный id\n";

throw ext;

}

f >> width;

if (width < 0) {

string ext = "В записи №";

ext += \_itoa(i, new char[5], 10);

ext += " указана некорректная ширина\n";

throw ext;

}

f >> height;

if (height < 0) {

string ext = "В записи №";

ext += \_itoa(i, new char[5], 10);

ext += " указана некорректная высота\n";

throw ext;

}

f >> name;

if (name[0] == 34)

{

if (name[strlen(name) - 1] == 34) {

string namen = (string)name;

namen = namen.substr(1, namen.length() - 2);

strcpy(name, namen.c\_str());

}

else {

char\* name2 = new char[50];

f >> name2;

while (name2[0] != 0 && name2[strlen(name2) - 1] != 34) {

strcat(name, " ");

strcat(name, name2);

f >> name2;

}

if (name2[strlen(name2) - 1] == 34) {

strcat(name, " ");

strcat(name, name2);

string namen = (string)name;

namen = namen.substr(1, namen.length() - 2);

strcpy(name, namen.c\_str());

}

else {

string ext = "В записи №";

ext += \_itoa(i, new char[5], 10);

ext += " не указано наименование\n";

throw ext;

}

}

}

if (name[0] == 0) {

string ext = "В записи №";

ext += \_itoa(i, new char[5], 10);

ext += " не указано наименование\n";

throw ext;

}

}

Monitor::Monitor()

{

id = -1;

width = -1;

height = -1;

}

Monitor::Monitor(int \_id, int \_width, int \_height, char\* \_name)

{

id = \_id;

width = \_width;

height = \_height;

name = \_name;

}

bool Monitor::operator== (const Monitor & monik2)

{

return (height == monik2.height && width == monik2.width);

}

bool Monitor::operator== (const int & \_id)

{

return (id == \_id);

}

**\Monitor\VGA\_monitor\VGA\_monitor.h**

#pragma once

#include "..\Monitor.h"

class VGA\_monitor : virtual public Monitor

{

protected:

int Rate;

public:

void pprint();

void fprint(std::ofstream& f);

void pread();

void fread(std::stringstream& f, int i);

VGA\_monitor();

VGA\_monitor(int \_id, int \_width, int \_height, char\* \_name, int \_rate);

};

**\Monitor\VGA\_monitor\VGA\_monitor.cpp**

#include "stdafx.h"

#include "VGA\_monitor.h"

using namespace std;

void VGA\_monitor::pprint() {

Monitor::pprint();

cout << "Частота, Гц: " << Rate << endl;

}

void VGA\_monitor::fprint(ofstream& f) {

Monitor::fprint(f);

f << " " << Rate;

}

void VGA\_monitor::pread() {

Monitor::pread();

cout << "Частота, Гц: "; scanf("%d", &Rate);

if (Rate < 0)

throw "Ошибка: введена некорректная частота\n";

}

void VGA\_monitor::fread(stringstream& f, int i) {

Monitor::fread(f, i);

f >> Rate;

if (Rate < 0) {

string ext = "В записи №";

ext += \_itoa(i, new char[5], 10);

ext += " указана некорректная частота\n";

throw ext;

}

}

VGA\_monitor::VGA\_monitor() :Monitor()

{

Rate = -1;

}

VGA\_monitor::VGA\_monitor(int \_id, int \_width, int \_height, char\* \_name, int \_rate) :Monitor(\_id, \_width, \_height, \_name)

{

Rate = \_rate;

}

**\Monitor\TV\TV.h**

#pragma once

#include "..\Monitor.h"

class TV :virtual public Monitor

{

protected:

int db;

public:

void pprint();

void fprint(std::ofstream& f);

void pread();

void fread(std::stringstream& f, int i);

TV();

TV(int \_id, int \_width, int \_height, char\* \_name, int \_db);

};

**\Monitor\TV\TV.cpp**

#include "stdafx.h"

#include "TV.h"

using namespace std;

void TV::pprint() {

Monitor::pprint();

cout << "Макс. громкость, дб: " << db << endl;

}

void TV::fprint(ofstream& f) {

Monitor::fprint(f);

f << " " << db;

}

void TV::pread() {

Monitor::pread();

cout << "Макс. громкость, дб: "; scanf("%d", &db);

if (db < 0)

throw "Ошибка: введена некорректная макс. громкость\n";

}

void TV::fread(stringstream& f, int i) {

Monitor::fread(f, i);

f >> db;

if (db < 0) {

string ext = "В записи №";

ext += \_itoa(i, new char[5], 10);

ext += " указана некорректная макс. громкость\n";

throw ext;

}

}

TV::TV() :Monitor()

{

db = -1;

}

TV::TV(int \_id, int \_width, int \_height, char\* \_name, int \_db) :Monitor(\_id, \_width, \_height, \_name)

{

db = \_db;

}

**\Monitor\VGA\_monitor\Projector\Projector.h**

#pragma once

#include "..\VGA\_monitor.h"

class Projector : public VGA\_monitor

{

private:

int min, max;

public:

void pprint() override;

void fprint(std::ofstream& f) override;

void pread() override;

void fread(std::stringstream& f, int i) override;

Projector();

Projector(int \_id, int \_width, int \_height, char\* \_name, int \_rate, int \_min, int \_max);

};

**\Monitor\VGA\_monitor\Projector\Projector.cpp**

#include "stdafx.h"

#include "Projector.h"

using namespace std;

void Projector::pprint() {

VGA\_monitor::pprint();

cout << "Минимальная дальность отображения: " << min << endl;

cout << "Максимальная дальность отображения: " << max << endl;

}

void Projector::fprint(ofstream& f) {

VGA\_monitor::fprint(f);

f << " " << min << " " << max;

}

void Projector::pread() {

VGA\_monitor::pread();

cout << "Минимальная дальность отображения: "; scanf("%d", &min);

if (min < 0)

throw "Ошибка: введена некорректная мин. дальность\n";

cout << "Максимальная дальность отображения: "; scanf("%d", &max);

if (max < 0)

throw "Ошибка: введена некорректная макс. дальность\n";

}

void Projector::fread(stringstream& f, int i) {

VGA\_monitor::fread(f, i);

f >> min;

if (min < 0) {

string ext = "В записи №";

ext += \_itoa(i, new char[5], 10);

ext += " указана некорректная мин. дистанция\n";

throw ext;

}

f >> max;

if (max < 0) {

string ext = "В записи №";

ext += \_itoa(i, new char[5], 10);

ext += " указана некорректна макс. дистанция\n";

throw ext;

}

}

Projector::Projector() :VGA\_monitor()

{

min = max = -1;

}

Projector::Projector(int \_id, int \_width, int \_height, char\* \_name, int \_rate, int \_min, int \_max) : Monitor(\_id, \_width, \_height, \_name), VGA\_monitor(\_id, \_width, \_height, \_name, \_rate)

{

min = \_min;

max = \_max;

}

**\Monitor\TV\Big\_TV\Big\_TV.h**

#pragma once

#include "..\TV.h"

class Big\_TV :public TV

{

private:

bool big\_sound;

public:

void pprint() override;

void fprint(std::ofstream& f) override;

void pread() override;

void fread(std::stringstream& f, int i) override;

Big\_TV();

Big\_TV(int \_id, int \_width, int \_height, char\* \_name, int \_db, bool \_bs);

};

**\Monitor\TV\Big\_TV\Big\_TV.cpp**

#include "stdafx.h"

#include "Big\_TV.h"

using namespace std;

void Big\_TV::pprint() {

TV::pprint();

cout << "Объёмный звук (1 - есть, 0 - нет): " << (int)big\_sound << endl;

}

void Big\_TV::fprint(ofstream& f) {

TV::fprint(f);

f << " " << big\_sound;

}

void Big\_TV::pread() {

TV::pread();

cout << "Объёмный звук (1 - есть, 0 - нет): "; scanf("%d", &big\_sound);

if (db < 0)

throw "Ошибка: введено некорректное значение наличия объёмного звука (должно быть 0 или 1)\n";

}

void Big\_TV::fread(stringstream& f, int i) {

TV::fread(f, i);

f >> big\_sound;

if (big\_sound < 0) {

string ext = "В записи №";

ext += \_itoa(i, new char[5], 10);

ext += " указано некорректное значение наличия объёмного звука\n";

throw ext;

}

}

Big\_TV::Big\_TV() :TV()

{

big\_sound = -1;

}

Big\_TV::Big\_TV(int \_id, int \_width, int \_height, char\* \_name, int \_db, bool \_bs) :Monitor(\_id, \_width, \_height, \_name), TV(\_id, \_width, \_height, \_name, \_db)

{

big\_sound = \_bs;

}

**\Monitor\HDMI\_monitor\HDMI\_monitor.h**

#pragma once

#include "..\VGA\_monitor\VGA\_monitor.h"

#include "..\TV\TV.h"

class HDMI\_monitor : public VGA\_monitor, public TV

{

public:

void pprint() override;

void fprint(std::ofstream& f) override;

void pread() override;

void fread(std::stringstream& f, int i) override;

HDMI\_monitor();

HDMI\_monitor(int \_id, int \_width, int \_height, char\* \_name, int \_rate, int \_db);

};

**\Monitor\HDMI\_monitor\HDMI\_monitor.cpp**

#include "stdafx.h"

#include "HDMI\_monitor.h"

using namespace std;

void HDMI\_monitor::pprint() {

VGA\_monitor::pprint();

cout << "Макс. громкость, дб: " << db << endl;

}

void HDMI\_monitor::fprint(ofstream& f) {

VGA\_monitor::fprint(f);

f << " " << db;

}

void HDMI\_monitor::pread() {

VGA\_monitor::pread();

cout << "Макс. громкость, дб: "; scanf("%d", &db);

if (db < 0)

throw "Ошибка: введена некорректная макс. громкость\n";

}

void HDMI\_monitor::fread(stringstream& f, int i) {

VGA\_monitor::fread(f, i);

f >> db;

if (db < 0) {

string ext = "В записи №";

ext += \_itoa(i, new char[5], 10);

ext += " указана некорректная макс. громкость\n";

throw ext;

}

}

HDMI\_monitor::HDMI\_monitor() :VGA\_monitor(), TV() {};

HDMI\_monitor::HDMI\_monitor(int \_id, int \_width, int \_height, char\* \_name, int \_rate, int \_db) : Monitor(\_id, \_width, \_height, \_name), VGA\_monitor(\_id, \_width, \_height, \_name, \_rate), TV(\_id, \_width, \_height, \_name, \_db) {};

**\Stek\Stek.h**

#pragma once

#include "..\Monitor\Monitor.h"

using namespace std;

class Stack {

//данные, скрытые от пользователя

private:

struct Node {

Monitor\* data;

Node\* next;

};

Node\* top;

//методы для редактирования стэка

public:

Stack();

Stack(const Stack& paren);

~Stack();

void push(Monitor\* value);

bool empty() const;

void pop();

//перегруженные операторы

Stack& operator = (const Stack& paren);

friend ostream& operator << (ostream& out, const Stack& stek)

{

if (!stek.empty())

{

stek.top->data->pprint();

out << endl << endl;

}

else

out << "Полка пуста" << endl << endl;

return out;

}

friend ofstream& operator << (ofstream& out, const Stack& stek)

{

if (!stek.empty())

{

stek.top->data->fprint(out);

out << endl;

}

return out;

}

//методы для обработки стэка

void poisk\_count(Monitor\* monik); //поиск по габаритам

void poisk\_count(int \_id); //поиск по id

void poisk\_count(string namen); //поиск по типу

void poisk\_first(Monitor\* monik);

void poisk\_first(int \_id);

void poisk\_first(string namen);

};

**\Stek\Stek.cpp**

#include "stdafx.h"

#include "Stek.h"

Stack::Stack()

{

this->top = NULL;

}

Stack::Stack(const Stack& paren)

{

this->top = paren.top;

}

Stack::~Stack()

{

top = NULL;

}

void Stack::push(Monitor\* value) {

Node\* tmp = new Node;

tmp->next = top;

top = tmp;

top->data = value;

}

bool Stack::empty() const {

return top == NULL;

}

void Stack::pop() {

if (!empty())

{

Node \*tmp = top;

top = top->next;

delete(tmp);

}

else

cout << "Полка пуста" << endl << endl;

}

//перегруженные операторы

Stack& Stack::operator = (const Stack& paren)

{

this->~Stack();

this->top = paren.top;

return \*this;

}

//методы для обработки стэка

void Stack::poisk\_count(Monitor\* monik) //поиск по габаритам

{

int i = 0;

Node\* tmp = top;

while (tmp)

{

if (\*tmp->data == \*monik)

i++;

tmp = tmp->next;

}

if (i) cout << "Найдено мониторов: " << i << endl;

else cout << "Мониторов не найдено" << endl;

}

void Stack::poisk\_count(int \_id) //поиск по id

{

int i = 0;

Node\* tmp = top;

while (tmp)

{

if (\*tmp->data == \_id)

i++;

tmp = tmp->next;

}

if (i) cout << "Найдено мониторов: " << i << endl;

else cout << "Мониторов не найдено" << endl;

}

void Stack::poisk\_count(string namen) //поиск по типу

{

int i = 0;

Node\* tmp = top;

while (tmp)

{

if (typeid(\*tmp->data).name() == namen)

i++;

tmp = tmp->next;

}

if (i) cout << "Найдено мониторов: " << i << endl;

else cout << "Мониторов не найдено" << endl;

}

void Stack::poisk\_first(Monitor\* monik)

{

int i = 1;

Node\* tmp = top;

bool fl = false;

while (tmp && !fl)

{

if (\*tmp->data == \*monik)

{

fl = true;

break;

}

else

{

i++;

tmp = tmp->next;

}

}

if (fl) cout << "Ближайший монитор на " << i << " позиции на полке" << endl;

else cout << "Данного монитора на полке нет" << endl;

}

void Stack::poisk\_first(int \_id)

{

int i = 1;

Node\* tmp = top;

bool fl = false;

while (tmp)

{

if (\*tmp->data == \_id)

{

fl = true;

break;

}

else

{

i++;

tmp = tmp->next;

}

}

if (fl) cout << "Ближайший монитор на " << i << " позиции на полке" << endl;

else cout << "Данного монитора на полке нет" << endl;

}

void Stack::poisk\_first(string namen)

{

int i = 1;

Node\* tmp = top;

bool fl = false;

while (tmp)

{

if (typeid(\*tmp->data).name() == namen)

{

fl = true;

break;

}

else

{

i++;

tmp = tmp->next;

}

}

if (fl) cout << "Ближайший монитор на " << i << " позиции на полке" << endl;

else cout << "Данного монитора на полке нет" << endl;

}

**file\_ops.h**

#pragma once

#include "Stek\Stek.h"

#include "Monitor\HDMI\_monitor\HDMI\_monitor.h"

#include "Monitor\TV\Big\_TV\Big\_TV.h"

#include "Monitor\VGA\_monitor\Projector\Projector.h"

void download(string path, Stack& moniks);

void upload(string path, Stack moniks);

**file\_ops.cpp**

#include "stdafx.h"

#include "file\_ops.h"

void download(string path, Stack& moniks) {

ifstream f(path);

if (f)

{

int i = 1;

string line;

while (!f.eof())

{

Monitor\* monik;

string name;

getline(f, line);

if (line != "")

{

stringstream ss;

ss << line;

ss >> name;

if (name == "VGA\_monitor")

monik = new VGA\_monitor;

else if (name == "HDMI\_monitor")

monik = new HDMI\_monitor;

else if (name == "TV")

monik = new TV;

else if (name == "Projector")

monik = new Projector;

else if (name == "Big\_TV")

monik = new Big\_TV;

else

{

string ext = "В записи №";

ext += \_itoa(i, new char[5], 10);

ext += " указан некорректный тип\n";

throw ext;

}

monik->fread(ss, i);

moniks.push(monik);

i++;

}

}

f.close();

}

else

{

throw "Ошибка. Файл не найден или повреждён\n";

}

}

void upload(string path, Stack moniks) {

ofstream f(path);

Stack tmp = moniks;

while (!tmp.empty())

{

f << tmp;

tmp.pop();

}

f.close();

}

**Video\_Monitors.cpp**

#include "stdafx.h"

#include "file\_ops.h"

int main()

{

setlocale(LC\_ALL, "");

Stack moniks;

int doin = -1;

printf("Загрузить данные из файла? (ввод 1, если да)\n");

scanf("%d", &doin);

if (doin == 1)

{

printf("\nВведите путь к файлу: ");

string path;

cin.get();

getline(cin, path);

try

{

download(path, moniks);

}

catch (string \_wha)

{

cout << \_wha;

}

catch (char\* \_wha)

{

printf("%s", \_wha);

}

}

doin = 1;

while (doin > 0 && doin <= 6)

{

cout << "\nВыберите действие:" << endl

<< "0 - выход" << endl

<< "1 - положить устройство на полку" << endl

<< "2 - снять устройство с полки" << endl

<< "3 - поиск интересующего устройства" << endl

<< "4 - подсчёт интересующих устройств" << endl

<< "5 - просмотреть информацию о крайнем устройстве на полке" << endl

<< "6 - убрать все устройства с полки" << endl << endl;

scanf("%d", &doin);

switch (doin)

{

case 1:

{

try

{

cout << "\nВыберите тип устройства: " << endl

<< "1 - VGA-монитор" << endl

<< "2 - HDMI-монитор" << endl

<< "3 - Телевизор" << endl

<< "4 - Проектор" << endl

<< "5 - Домашний кинотеатр" << endl

<< "0 - Отмена" << endl << endl;

int chs;

scanf("%d", &chs);

while (!(chs >= 0 && chs <= 5))

{

cout << "Введите корректный номер" << endl;

scanf("%d", &chs);

}

Monitor\* monik;

if (chs == 1)

monik = new VGA\_monitor;

else if (chs == 2)

monik = new HDMI\_monitor;

else if (chs == 3)

monik = new TV;

else if (chs == 4)

monik = new Projector;

else if (chs == 5)

monik = new Big\_TV;

monik->pread();

moniks.push(monik);

}

catch (char\* \_wha)

{

printf("\n%s\n", \_wha);

}

break;

}

case 2:

moniks.pop();

cout << endl;

break;

case 3:

{

try

{

if (moniks.empty()) cout << "Полка пуста" << endl << endl;

else

{

cout << "\nВыберите критерий: " << endl

<< "1 - по id" << endl

<< "2 - по типу" << endl

<< "3 - по размеру" << endl

<< "0 - Отмена" << endl << endl;

int chs;

scanf("%d", &chs);

while (!(chs >= 0 && chs <= 3))

{

cout << "Введите корректный номер" << endl;

scanf("%d", &chs);

}

if (chs == 1)

{

cout << "\nВведите id: ";

int id;

scanf("%d", &id);

if (id < 0) throw "Ошибка: введён неверный id\n\n";

moniks.poisk\_first(id);

}

else if (chs == 2)

{

cout << "\nВыберите тип: " << endl

<< "1 - VGA-монитор" << endl

<< "2 - HDMI-монитор" << endl

<< "3 - Телевизор" << endl

<< "4 - Проектор" << endl

<< "5 - Домашний кинотеатр" << endl

<< "0 - Отмена" << endl << endl;

int chs;

scanf("%d", &chs);

while (!(chs >= 0 && chs <= 5))

{

cout << "Введите корректный номер" << endl;

scanf("%d", &chs);

}

if (chs == 1)

{

moniks.poisk\_first("class VGA\_monitor");

}

else if (chs == 2)

{

moniks.poisk\_first("class HDMI\_monitor");

}

else if (chs == 3)

{

moniks.poisk\_first("class TV");

}

else if (chs == 4)

{

moniks.poisk\_first("class Projector");

}

else if (chs == 5)

{

moniks.poisk\_first("class Big\_TV");

}

}

else if (chs == 3)

{

Monitor monik;

monik.pread();

moniks.poisk\_first(&monik);

}

}

system("pause");

cout << endl;

}

catch (char\* \_wha)

{

printf("\n%s\n", \_wha);

}

break;

}

case 4:

{

try

{

if (moniks.empty()) cout << "Полка пуста" << endl << endl;

else

{

cout << "\nВыберите критерий: " << endl

<< "1 - по id" << endl

<< "2 - по типу" << endl

<< "3 - по размеру" << endl

<< "0 - Отмена" << endl << endl;

int chs;

scanf("%d", &chs);

while (!(chs >= 0 && chs <= 3))

{

cout << "Введите корректный номер" << endl;

scanf("%d", &chs);

}

if (chs == 0);

else if (chs == 1)

{

cout << "\nВведите id: ";

int id;

scanf("%d", &id);

if (id < 0) throw "Ошибка: введён неверный id\n\n";

moniks.poisk\_count(id);

}

else if (chs == 2)

{

cout << "\nВыберите тип: " << endl

<< "1 - VGA-монитор" << endl

<< "2 - HDMI-монитор" << endl

<< "3 - Телевизор" << endl

<< "4 - Проектор" << endl

<< "5 - Домашний кинотеатр" << endl

<< "0 - Отмена" << endl << endl;

int chs;

scanf("%d", &chs);

while (!(chs >= 0 && chs <= 5))

{

cout << "Введите корректный номер" << endl;

scanf("%d", &chs);

}

if (chs == 1)

{

moniks.poisk\_count("class VGA\_monitor");

}

else if (chs == 2)

{

moniks.poisk\_count("class HDMI\_monitor");

}

else if (chs == 3)

{

moniks.poisk\_count("class TV");

}

else if (chs == 4)

{

moniks.poisk\_count("class Projector");

}

else if (chs == 5)

{

moniks.poisk\_count("class Big\_TV");

}

}

else if (chs == 3)

{

Monitor monik;

monik.pread();

moniks.poisk\_count(&monik);

}

}

system("pause");

cout << endl;

}

catch (char\* \_wha)

{

printf("\n%s\n", \_wha);

}

break;

}

case 5:

cout << moniks;

system("pause");

cout << endl;

break;

case 6:

moniks.~Stack();

break;

}

}

printf("Выгрузить данные в файл? (ввод 1, если да)\n");

scanf("%d", &doin);

if (doin == 1)

{

printf("Введите путь к файлу: ");

string path;

cin.get();

getline(cin, path);

upload(path, moniks);

}

return 0;

}