МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт (факультет) Институт информационных технологий

Кафедра Математического и программного обеспечения ЭВМ

КУРСОВАЯ РАБОТА

Модуль Программирование .

по дисциплине Объектно-ориентированное программирование

на тему Объектно-ориентированное программирование на языке C++

Выполнил студент группы

1ПИб-02-2оп-22

01

*группа*

направления подготовки (специальности)

09.03.04 Программная инженерия

*шифр, наименование*

Тихомиров Владислав Валерьевич

*фамилия, имя, отчество*

Руководитель

Табунов Павел Александрович

*фамилия, имя, отчество*

Доцент

*должность*

Дата представления работы

« » 2024г. Заключение о допуске к защите

Оценка ,

количество баллов

Подпись преподавателя

Череповец, 2024

ОГЛАВЛЕНИЕ

[Введение 3](#_Toc196664143)

[1. Объектно-ориентированный анализ предметной области 4](#_Toc196664144)

[2. Проектирование классов 5](#_Toc196664145)

[3. Логическая структура программы 10](#_Toc196664146)

[3.1. Полиморфная обработка 10](#_Toc196664147)

[3.2. Файловый ввод и вывод 10](#_Toc196664148)

[3.3. Функция обработка данных 10](#_Toc196664149)

[3.4. Обработка исключений 10](#_Toc196664150)

[Заключение 13](#_Toc196664151)

[Список литературы 14](#_Toc196664152)

[Приложение 1. Техническое задание 15](#_Toc196664153)

# Введение

Объектно-ориентированное программирование или ООП — это одна из парадигм разработки. Помогает стандартизировать написание кода. Это снижает риск ошибок, ускоряет разработку и делает код более читабельным для других программистов.

Суть понятия объектно-ориентированного программирования в том, что все программы, написанные с применением этой парадигмы, состоят из объектов. Каждый объект — это определённая сущность со своими данными и набором доступных действий.

Объектно-ориентированное программирование используется, чтобы:

* структурировать информацию и не допускать путаницы;
* точно определять взаимодействие одних элементов с другими;
* повышать управляемость программы;
* быстрее масштабировать код под различные задачи;
* лучше понимать написанное;
* эффективнее поддерживать готовые программы;
* внедрять изменения без необходимости переписывать весь код.

Структура объектно-ориентированного программирования:

1. Класс — это «шаблон» для объекта, который описывает его свойства. Каждый объект — это экземпляр какого-нибудь класса.
2. Объект — это набор переменных и функций, как в традиционном функциональном программировании. Переменные и функции и есть его свойства.
3. Атрибуты — это переменные, конкретные характеристики объекта, такие как цвет поля или имя пользователя.
4. Методы — это функции, которые описаны внутри объекта или класса. Они относятся к определенному объекту и позволяют взаимодействовать с ними или другими частями кода.

Объектно-ориентированное программирование определяют через четыре принципа, по которым можно понять основы работы.

**Инкапсуляция:**

Каждый объект — независимая структура. Все, что ему нужно для работы, уже есть у него внутри. Если он пользуется какой-то переменной, она будет описана в теле объекта, а не снаружи в коде. Это делает объекты более гибкими. Даже если внешний код перепишут, логика работы не изменится. Внутреннее устройство одного объекта закрыто от других: извне «видны» только значения атрибутов и результаты выполнения методов.

**Наследование:**

Можно создавать классы и объекты, которые похожи друг на друга, но немного отличаются — имеют дополнительные атрибуты и методы. Более общее понятие в таком случае становится «родителем», а более специфичное и подробное — «наследником».

У одного «родителя» может быть несколько дочерних классов, которые наследуют его методы и атрибуты.

Наследование позволяет реализовывать сложные схемы с четкой иерархией «от общего к частному». Это облегчает понимание и масштабирование кода. Не нужно много раз переписывать в разных объектах одни и те же свойства. Достаточно унаследовать эти объекты от одного «родителя», и «родительские» свойства применятся автоматически.

**Полиморфизм:**

Одинаковые методы разных объектов могут выполнять задачи разными способами.

# Объектно-ориентированный анализ предметной области

Вычислительная техника представляет собой отрасль, занимающуюся разработкой и применением электронных вычислительных устройств, работающих в диапазоне тактовых частот от 1 ГГц до 5 ГГц. Создание систем с более высокой частотой относится к экспериментальным вычислительным технологиям. В нашей программе реализована иерархия классов, описывающих различные виды вычислительной техники.

Основу системы составляет абстрактный базовый класс IEVM, который определяет общие характеристики для всех устройств: название компании-производителя, количество ядер процессора и тактовую частоту. Наследником этого интерфейса является класс Personal, добавляющий специфические особенности персональных компьютеров.

Бытовые вычислительные устройства в нашей системе представлены несколькими специализированными классами. Класс Personal описывает стационарные компьютеры и включает информацию о типе используемого монитора - варианты asus, lenovo, samsung или xiaomi. Серверное оборудование, реализованное в классе Server, характеризуется важным параметром - максимальным количеством одновременно поддерживаемых пользователей, которое может варьироваться от нескольких десятков до тысяч подключений.

Особое внимание уделено мобильным устройствам. Смартфоны (класс Smartphone) отличаются наличием или отсутствием поддержки стилуса, а также различным количеством процессорных ядер. Ноутбуки (класс Laptop) дополнительно содержат информацию о весе устройства, что важно для портативных систем, и могут оснащаться разными типами экранов.

Наиболее специализированным классом в нашей иерархии могли бы стать игровые ноутбуки, которые наследовали бы основные характеристики от класса Laptop, но добавляли бы такие уникальные особенности как наличие дискретной видеокарты, подсветку клавиатуры и усовершенствованную систему охлаждения. Такая многоуровневая система классификации позволяет точно и полно описывать параметры каждого типа вычислительной техники, начиная от общих характеристик процессора и заканчивая специализированными возможностями конкретных устройств.

Иерархия родственных типов представлена (рис. 1).



Рис. 1. Контекстная диаграмма классов

# Проектирование классов

Абстрактный базовый класс-интерфейс IEVM

|  |  |  |
| --- | --- | --- |
| Тип данных | Имя | Краткое описание |
| void | virtual SetNameOfCorpus | Установка значения для атрибута nameOfCorpus |
| string | virtual GetNameOfCorpus() | Получение значения атрибута nameOfCorpus |
| Int | virtual get\_power() const | Получения значения атрибута power |
| void | virtual SetJadra | Установка значения для атрибута jadra |
| int | virtual GetJadra | Получение значения атрибута jadra |
| void | virtual SetChast | Установка значения для адтрибута Chast |
| double | Virtual GetChast | Получение значения атрибута chast |
| void | virtual print | Вывод значений атрибутов класса |

Класс Laptop

|  |  |
| --- | --- |
| Атрибут | |
| Закрытая часть | |
| string\* nameOfCompany | Название компании |
| int jadra | Кол-во ядер |
| double chast | Частота процессора |
| int weight | Вес |
| static int count | Номер объекта |

|  |  |
| --- | --- |
| Метод | |
| Открытая часть | |
| Laptop() | Конструктор по умолчанию |
| Laptop(string NameOfCompany, int jadra, double chast, int weight) | Конструктор с параметрами |
| void SetWeight(int weight) | Сеттер атрибута вес |
| int GetWeight(); | Геттер атрибута вес |
| void SetNameOfCorpus(string nameOfCompany) | Сеттер атрибута название компании |
| string GetNameOfCorpus() | Геттер атрибута название компании |
| void SetJadra(int jadra) | Сеттер атрибута ядра |
| int GetJadra() | Геттер атрибута ядра |
| void SetChast(double chast) | Сеттер атрибута частота |
| double GetChast() | Геттер атрибута частота |
| void print() | Функция вывода |

Класс Stack содержит в закрытой части атрибуты:

* int capacity (емкость стека)
* Node\* first (указатель на первый элемент)
* Node\* last (указатель на последний элемент)

В открытой части:

1. Конструктор - Stack(int size) : capacity(size), count(0), first(nullptr), last(nullptr) {  
   if (size <= 0) {  
   throw invalid\_argument("Stack size must be positive");  
   }  
   }
2. Деструктор - ~Stack() {  
   while (first) {  
   Node\* temp = first;  
   first = first->next;  
   delete temp;  
   }  
   }
3. Функция - void push\_back(const T& val);
4. Функция - void at(int index) const;
5. Функция - void pop\_back();
6. Функция - void find(double targetFrequency);

Класс Laptop содержит в закрытой части атрибуты:

* string\* nameOfCompany (название компании)
* int jadra (количество ядер)
* double chast (частота)
* int weight (вес)
* static int count (количество созданных объектов)

В открытой части:

1. Конструктор - Laptop();
2. Конструктор - Laptop(string NameOfCompany, int jadra, double chast, int weight);
3. Функция - void SetWeight(int weight);
4. Функция - int GetWeight();
5. Функция - void SetNameOfCorpus(string nameOfCompany) override;
6. Функция - string GetNameOfCorpus() override;
7. Функция - void SetJadra(int jadra) override;
8. Функция - int GetJadra() override;
9. Функция - void SetChast(double chast) override;
10. Функция - double GetChast() override;
11. Функция - void print() override;

Класс Server содержит в закрытой части атрибуты:

* static int count (количество созданных объектов)
* string\* nameOfCompany (название компании)
* int jadra (количество ядер)
* double chast (частота)
* int MaxUser (максимальное количество пользователей)

В открытой части:

1. Конструктор - Server();
2. Конструктор - Server(string NameOfCompany, int jadra, double chast, int MaxUser );
3. Функция - void SetNameOfCorpus(string nameOfCompany) override;
4. Функция - string GetNameOfCorpus() override;
5. Функция - void SetJadra(int jadra) override;
6. Функция - int GetJadra() override;
7. Функция - void SetChast(double chast) override;
8. Функция - double GetChast() override;
9. Функция - void SetUser (int MaxUser );
10. Функция - int GetUser ();
11. Функция - void print() override;
12. Деструктор - ~Server();

Класс Smartphone содержит в закрытой части атрибуты:

* static int count (количество созданных объектов)
* string\* nameOfCompany (название компании)
* int jadra (количество ядер)
* double chast (частота)
* bool stelus (поддержка стилуса)

В открытой части:

1. Конструктор - Smartphone();
2. Конструктор - Smartphone(string NameOfCompany, int jadra, double chast, bool stelus);
3. Функция - void SetNameOfCorpus(string nameOfCompany) override;
4. Функция - string GetNameOfCorpus() override;
5. Функция - void SetJadra(int jadra) override;
6. Функция - int GetJadra() override;
7. Функция - void SetChast(double chast) override;
8. Функция - double GetChast() override;
9. Функция - void SetStelus(bool stelus);
10. Функция - bool GetStelus();
11. Функция - void print() override;
12. Деструктор - ~Smartphone();



# Логическая структура программы

## Полиморфная обработка

## Файловый ввод и вывод

## Функция обработка данных

## Обработка исключений

В классе Stack реализована обработка исключений с помощью операторов throw, try и catch. В конструкторе класса проверяется размер стека, и если он меньше или равен нулю, выбрасывается исключение:

if (size <= 0) {

throw invalid\_argument("Stack size must be positive");

}

Если при создании объекта стека был задан размер меньше или равный нулю, то вызывается исключение, и программа завершит выполнение с сообщением:

Error: Stack size must be positive

Примеры обработки исключений в методах класса

1. Метод push\_back:
   * Если стек переполнен, выводится сообщение:
2. Stack overflow
3. Метод at:
   * Если индекс выходит за пределы допустимого диапазона, выбрасывается исключение:
4. throw out\_of\_range("Index out of range");
   * При этом выводится сообщение:
5. Error: Index out of range
6. Метод pop\_back:
   * Если стек пуст, выводится сообщение:
7. Stack is empty!
8. Метод find:
   * Если стек пуст, выводится сообщение:
9. Stack is empty!
   * Если не найдено ни одного элемента с заданной частотой, выводится сообщение:

No elements with frequency [targetFrequency] found.

.

1. Модульная структура программы

Модульная структура программы представлена (рис.4).

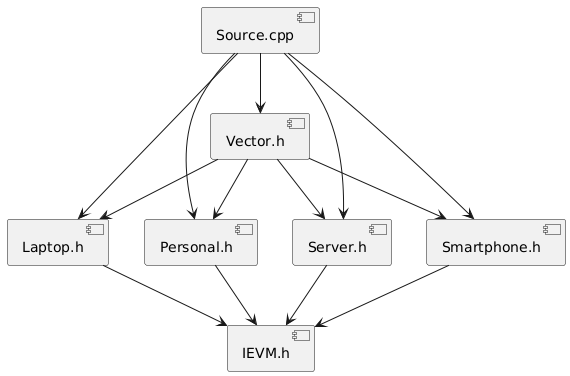


Рис. 4. Модульная диаграмма

1. Тестирование программы

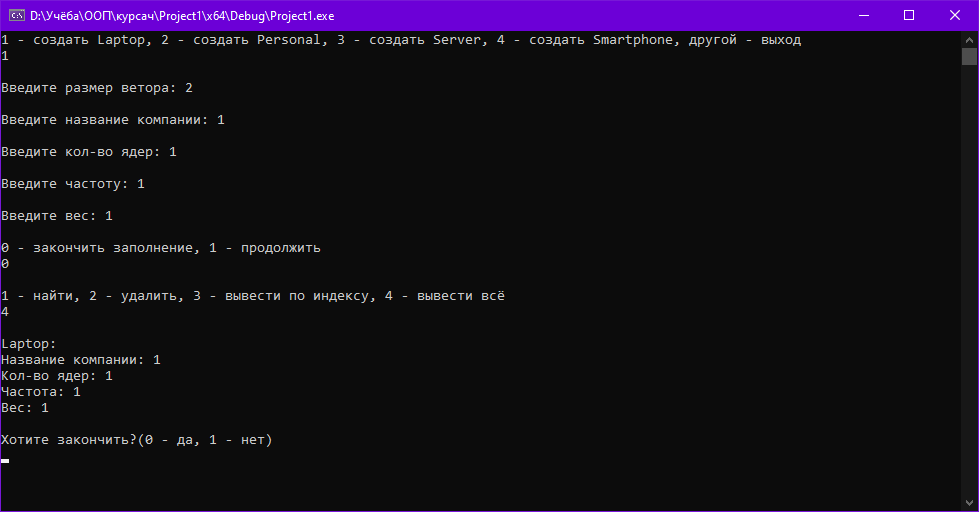


Рис. 5. Создание Laptop

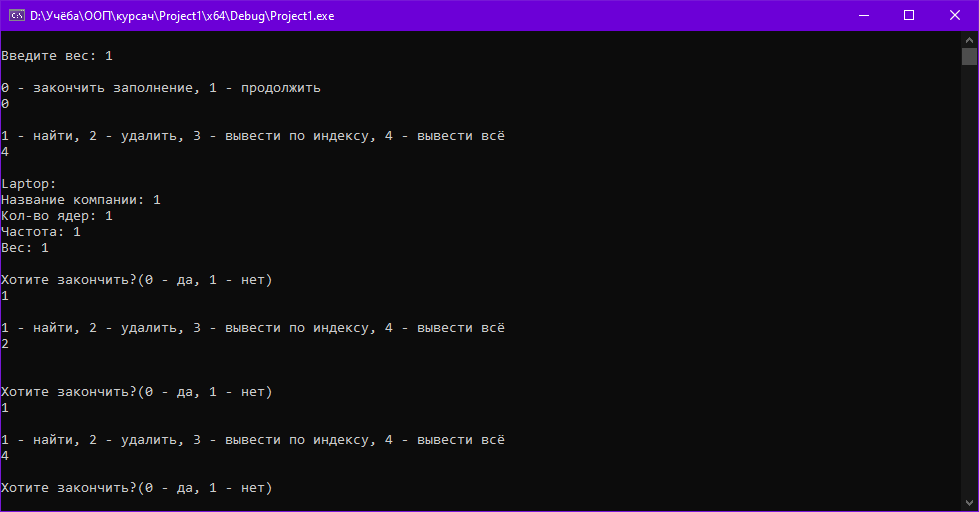


Рис. 6. Удаление объекта типа Laptop

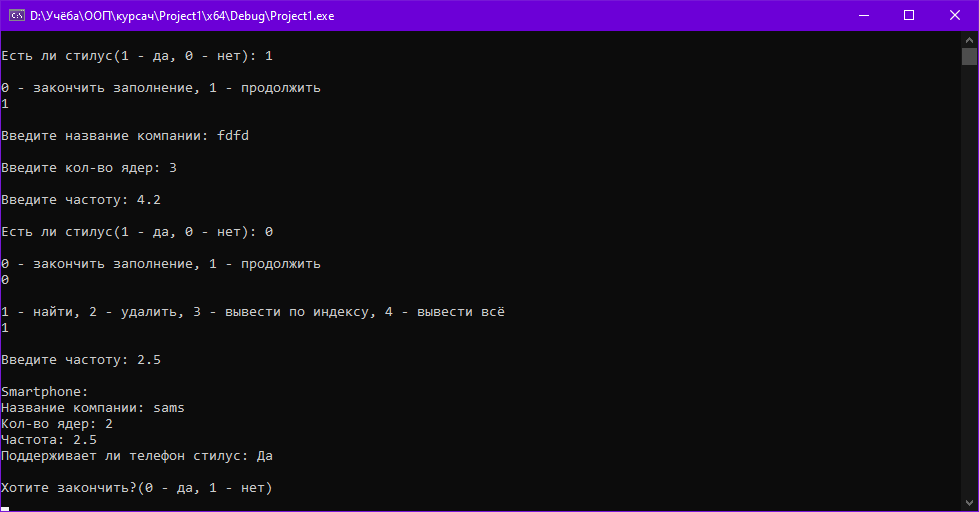


Рис. 7. Создание и поиск элемента типа Smartphone

# Заключение

В ходе выполнения курсовой работы на тему: объектно-ориентированное программирование на языке С++ закрепил способы создания производного класса и особенности работы с ним, правила инициализации и доступа к элементам производного класса, приобрел практические навыки наследования.

# Список литературы

1. Ашарина, И.В. Объектно-ориентированное программирование в С++: лекции и упражнения: Учебное пособие для вузов / И.В. Ашарина. - М.: РиС, 2015. - 336 c.
2. Васильев, А.Н. Объектно-ориентированное программирование на C++ / А.Н. Васильев. - СПб.: Наука и техника, 2016. - 544 c.
3. Кирютенко, Ю.А. Объектно-ориентированное программирование. Язык Smalltalk / Ю.А. Кирютенко, В.А. Савельев. - М.: Вузовская книга, 2007. - 328 c.
4. Лафоре, Р. Объектно-ориентированное программирование в C++. Классика Computer Science / Р. Лафоре. - СПб.: Питер, 2013. - 928 c.
5. Лесневский, А.С. Объектно-ориентированное программирование для начинающих / А.С. Лесневский. - М.: Бином. Лаборатория знаний, 2005. - 232 c.
6. Павловская, Т. С/С++.Процедурное и объектно-ориентированное программирование / Т. Павловская. - СПб.: Питер, 2018. - 496 c.
7. Хорев, П.Б. Объектно-ориентированное программирование / П.Б. Хорев. - М.: Academia, 2018. - 352 c.
8. Страуструп Б. Язык программирования С++: В 2 ч. – К.: ДиаСофт, 1993.
9. Программирование. Базовые средства языка программирования С++: Учеб. пособие / Е.В. Ершов, О.Г. Ганичева, В.В. Селивановских, Л.Н. Виноградова. – Череповец: ЧГУ, 2011. – 181 с. – ISBN 978 – 5 – 85341 – 467 – 9.
10. Ершов Е.В., д-р техн. наук, проф.; Виноградова Л.Н. и др. Методика и организация самостоятельной работы студентов − Коллектив авторов, ФГБОУ ВПО «Череповецкий государственный университет», 2012. − 208 с.

# Приложение 1. Техническое задание

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт информационных технологий

наименование института (факультета)

Математическое и программное обеспечение ЭВМ

наименование кафедры

Объектно-ориентированное программирование

наименование дисциплины в соответствии с учебным планом

УТВЕРЖДАЮ

Зав. кафедрой МПО ЭВМ ,

д.т.н., профессор Ершов Е.В.

« » 2024 г.

Объектно-ориентированное программирование на языке С++

Техническое задание на курсовую работу

Листов 5

Руководитель: Табунов П.А.

Ф.И.О. преподавателя

Исполнитель:

студент 1Пиб-01-1оп-21

группа

Тихомиров В.В.

Фамилия, имя, отчество

2024 год

Введение

Курсовая работа на тему: объектно-ориентированное программирование на языке С++. Закрепить способы создания производного класса и особенности работы с ним, правила инициализации и доступа к элементам производного класса, приобрести практические навыки наследования.

1. Основания для разработки

Основанием для разработки является задание на курсовую работу ООП\_2 по дисциплине "Объектно-ориентированное программирование", выданное на кафедре МПО ЭВМ ИИТ ЧГУ.

Дата утверждения: 13 февраля 2024 года.

Наименование темы разработки: объектно-ориентированное программирование на языке С++.

1. Назначение разработки

Курсовая работа направлена на разработку иерархии родственных типов, корневой класс которой абстрактный базовый класс.

1. Требования к программе
   1. Требования к функциональным характеристикам

Программа должна содержать следующие параметрам:

* 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 А.19 - холодильная техника);
* Создать обобщенный (void**\***) контейнерный класс (базовый) и от него, используя закрытое наследование, производный класс – шаблон для хранения указателей на абстрактный базовый класс-интерфейс (согласно варианта В.1 - статический вектор);
* Для хранения объектов каждого производного класса использовать структуру данных (согласно варианта С.6 - линейный двусвязный список)
* Реализовать функции обработки данных (сортировка и поиск по выбранным полям и задаваемым диапазонам значений, другие функции, в том числе перегруженные);
* Реализовать файловый ввод/вывод и ввод данных с клавиатуры, вывод данных на дисплей.
  1. Требования к надежности

Программа должна:

1. Предусматривать обработку различных исключительных ситуаций;
2. Работа всех функций должна быть проверена и результаты проверки оформлены протоколом тестирования.
   1. Условия эксплуатации

Требования не предъявляются.

* 1. Требования к составу и параметрам технических средств

Рекомендуемые характеристики технических средств:

1. Тактовая частота процессора: не ниже 2,6 ГГц;
2. Оперативная память: 4 ГБ;
3. Свободное пространство на жестком диске: не менее 10 ГБ;
4. Монитор;
5. Клавиатура;
6. Мышь.
   1. Требования к информационной и программной совместимости

Рекомендуемые требования:

1. Microsoft Windows 10, 32/64 бита и новее;
2. Среда разработки Visual Studio 2019 и новее.
   1. Требования к маркировке и упаковке

Программа должна распространяться на CD-диске.

* 1. Требования к транспортированию и хранению

Требования не предъявляются.

* 1. Специальные требования

Требования не предъявляются.

1. Требования к программной документации
   1. Содержание расчётно-пояснительной записки

Программная документация должна содержать расчётно-пояснительную записку с содержанием:

Титульный лист

Оглавление

Введение

1. Объектно-ориентированный анализ предметной области
2. Проектирование классов
3. Логическая структура программы
   1. Полиморфная обработка
   2. Файловый ввод и вывод
   3. Функция обработки данных
   4. Обработка исключений
4. Модульная структура программы
5. Тестирование программы

Заключение

Список литературы

Приложение 1. Техническое задание

Приложение 2. Текст программы

Приложение 3. Руководство пользователя

* 1. Требования к содержательной части

Требования к оформлению, установленные ГОСТ, должны быть выполнены на протяжении всей работы без каких-либо изменений (в табл. П1.1).

Таблица П1.1

Требования к оформлению

|  |  |
| --- | --- |
| Документ | Печать на отдельных листах формата А4 (210х297 мм); оборотная сторона не заполняется; листы нумеруются. Печать возможна ч/б. |
| Страницы | Ориентация — книжная; отдельные страницы, при необходимости, альбомная. Поля: верхнее, нижнее — по 2 см, левое — 3 см, правое — 1 см. |
| Абзацы | Межстрочный интервал — 1,5, перед и после абзаца — 0. |
| Шрифты | Кегль — 14. В таблицах шрифт 12. Шрифт листинга — 10 (возможно в 2 колонки). |
| Рисунки | Подписывается под ним по центру: «Рис.Х. Название В» приложениях: «Рис.П1.3. Название» |
| Таблицы | Подписывается: над таблицей, выравнивание по правому: «Таблица Х». В следующей строке по центру Название Надписи в «шапке» (имена столбцов, полей) — по центру. В теле таблицы (записи) текстовые значения — выравнены по левому краю, числа, даты — по правому. |

1. Стадии и этапы разработки

Стадии и этапы разработки, представлены в таблице (табл.П.1.).

Таблица П 1.2

Стадии и этапы разработки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование  этапа разработки | Сроки разработки | Результат выполнения | Отметка о выполнении |
| Изучение предметной области | 10.02.2024 – 15.02.2024 | Предметная область изучена | Выполнено |
| Проектирование классов | 16.02.2024-28.02.2024 | Сделано проектирование классов | Выполнено |
| Логическое проектирование | 01.03.2024-30.03.2024 | Сделано логическое проектирование | Выполнено |
| Модульное проектирование | 01.04.2024-15.04.2024 | Сделано модульное проектирование | Выполнено |
| Тестирование программы | 10.05.2024-20.05.2024 | Успешно пройденные тесты | Выполнено |

1. Порядок контроля и приемки

Порядок контроля и приемки, представлен в таблице

Таблица П 1.3

Порядок контроля и приемки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование  контрольного этапа  выполнения  курсовой работы | Сроки  контроля | Результат выполнения | Отметка о приемке  результата  контрольного этапа |
| Оформление Технического задания | 13.02.2024 | Готовое оформленное ТЗ |  |
| Оформление Расчётной пояснительной записки | 26.05.2024 | Готовое оформленное РПЗ |  |
| Сдача курсовой работы | 10.06.2024 | Получение оценки за выполненную работу |  |

# Приложение 2. Текст программы

Файл IEVM.h

#pragma once

#include <iostream>

#include <string>

using namespace std;

class IEVM {

public:

virtual void SetNameOfCorpus(string nameOfCorpus) = 0;

virtual string GetNameOfCorpus() = 0;

virtual void SetJadra(int jadra) = 0;

virtual int GetJadra() = 0;

virtual void SetChast(double chast) = 0;

virtual double GetChast() = 0;

virtual void print() = 0;

virtual ~IEVM() = default;

};

Файл Personal.h

#pragma once

#include "IEVM.h"

using namespace std;

enum screenName {

asus,

lenovo,

samsung,

xiaomi

};

const string NameOfScreen[] = {

"asus",

"lenovo",

"samsung",

"xiaomi"

};

class Personal : public IEVM {

screenName screen;

static int count;

string\* nameOfCompany;

int jadra;

double chast;

public:

Personal();

Personal(string NameOfCompany, int jadra, double chast, int screen);

void SetNameOfCorpus(string nameOfCompany) override;

string GetNameOfCorpus() override;

void SetJadra(int jadra) override;

int GetJadra() override;

void SetChast(double chast) override;

double GetChast() override;

void SetScreen(int screen);

screenName GetScreen();

void print() override;

~Personal();

};

int Personal::count = 0;

Personal::Personal() {

count++;

}

Personal::Personal(string NameOfCompany, int jadra, double chast, int screen) {

count++;

this->nameOfCompany = new string(NameOfCompany);

this->jadra = jadra;

this->chast = chast;

switch (screen)

{

case 1:

this->screen = screenName::asus;

break;

case 2:

this->screen = screenName::lenovo;

break;

case 3:

this->screen = screenName::samsung;

break;

case 4:

this->screen = screenName::xiaomi;

break;

default:

this->screen = screenName::samsung;

break;

}

}

void Personal::SetJadra(int jadra) {

this->jadra = jadra;

}

int Personal::GetJadra() {

return jadra;

}

void Personal::SetNameOfCorpus(string nameOfCompany) {

this->nameOfCompany = new string(nameOfCompany);

}

string Personal::GetNameOfCorpus() {

return \*nameOfCompany;

}

void Personal::SetChast(double chast) {

this->chast = chast;

}

double Personal::GetChast() {

return chast;

}

void Personal::SetScreen(int screen) {

switch (screen)

{

case 1:

this->screen = screenName::asus;

break;

case 2:

this->screen = screenName::lenovo;

break;

case 3:

this->screen = screenName::samsung;

break;

case 4:

this->screen = screenName::xiaomi;

break;

default:

this->screen = screenName::samsung;

break;

}

}

screenName Personal::GetScreen() {

return screen;

}

void Personal::print() {

cout << "Personal:" << endl;

cout << "Название компании: " << \*nameOfCompany << endl;

cout << "Кол-во ядер: " << jadra << endl;

cout << "Частота: " << chast << endl;

cout << "Монитор: " << NameOfScreen[screen] << endl;

}

Personal::~Personal() {

//delete nameOfCompany;

}

Файл Laptop.h

#pragma once

#include "IEVM.h"

class Laptop : public IEVM {

string\* nameOfCompany;

int jadra;

double chast;

int weight;

static int count;

public:

Laptop();

Laptop(string NameOfCompany, int jadra, double chast, int weight);

void SetWeight(int weight);

int GetWeight();

void SetNameOfCorpus(string nameOfCompany) override;

string GetNameOfCorpus() override;

void SetJadra(int jadra) override;

int GetJadra() override;

void SetChast(double chast) override;

double GetChast() override;

void print() override;

};

int Laptop::count = 0;

Laptop::Laptop() {

count++;

}

Laptop::Laptop(string NameOfCompany, int jadra, double chast, int weight) {

count++;

this->nameOfCompany = new string(NameOfCompany);

this->jadra = jadra;

this->chast = chast;

this->weight = weight;

}

void Laptop::SetWeight(int weight) {

this->weight = weight;

}

int Laptop::GetWeight() {

return weight;

}

void Laptop::SetJadra(int jadra) {

this->jadra = jadra;

}

int Laptop::GetJadra() {

return jadra;

}

void Laptop::SetNameOfCorpus(string nameOfCompany) {

this->nameOfCompany = new string(nameOfCompany);

}

string Laptop::GetNameOfCorpus() {

return \*nameOfCompany;

}

void Laptop::SetChast(double chast) {

this->chast = chast;

}

double Laptop::GetChast() {

return chast;

}

void Laptop::print() {

cout << "Laptop:" << endl;

cout << "Название компании: " << \*nameOfCompany << endl;

cout << "Кол-во ядер: " << jadra << endl;

cout << "Частота: " << chast << endl;

cout << "Вес: " << weight << endl;

}

Файл Server.h

#pragma once

#include "IEVM.h"

using namespace std;

class Server : public IEVM {

static int count;

string\* nameOfCompany;

int jadra;

double chast;

int MaxUser;

public:

Server();

Server(string NameOfCompany, int jadra, double chast, int MaxUser);

void SetNameOfCorpus(string nameOfCompany) override;

string GetNameOfCorpus() override;

void SetJadra(int jadra) override;

int GetJadra() override;

void SetChast(double chast) override;

double GetChast() override;

void SetUser(int MaxUser);

int GetUser();

void print() override;

~Server();

};

int Server::count = 0;

Server::Server() {

count++;

}

Server::Server(string NameOfCompany, int jadra, double chast, int MaxUser) {

count++;

this->nameOfCompany = new string(NameOfCompany);

this->jadra = jadra;

this->chast = chast;

this->MaxUser = MaxUser;

}

void Server::SetJadra(int jadra) {

this->jadra = jadra;

}

int Server::GetJadra() {

return jadra;

}

void Server::SetNameOfCorpus(string nameOfCompany) {

this->nameOfCompany = new string(nameOfCompany);

}

string Server::GetNameOfCorpus() {

return \*nameOfCompany;

}

void Server::SetChast(double chast) {

this->chast = chast;

}

double Server::GetChast() {

return chast;

}

void Server::SetUser(int MaxUser) {

this->MaxUser = MaxUser;

}

int Server::GetUser() {

return MaxUser;

}

void Server::print() {

cout << "Server:" << endl;

cout << "Название компании: " << \*nameOfCompany << endl;

cout << "Кол-во ядер: " << jadra << endl;

cout << "Частота: " << chast << endl;

cout << "Максимальное кол-во пользователей: " << MaxUser << endl;

}

Server::~Server() {

//delete nameOfCompany;

}

Файл Smartphone.h

#pragma once

#include "IEVM.h"

class Smartphone : public IEVM {

static int count;

string\* nameOfCompany;

int jadra;

double chast;

bool stelus;

public:

Smartphone();

Smartphone(string NameOfCompany, int jadra, double chast, bool stelus);

void SetNameOfCorpus(string nameOfCompany) override;

string GetNameOfCorpus() override;

void SetJadra(int jadra) override;

int GetJadra() override;

void SetChast(double chast) override;

double GetChast() override;

void SetStelus(bool stelus);

bool GetStelus();

void print() override;

~Smartphone();

};

int Smartphone::count = 0;

Smartphone::Smartphone() {

count++;

}

Smartphone::Smartphone(string NameOfCompany, int jadra, double chast, bool stelus) {

count++;

this->nameOfCompany = new string(NameOfCompany);

this->jadra = jadra;

this->chast = chast;

this->stelus = stelus;

}

void Smartphone::SetJadra(int jadra) {

this->jadra = jadra;

}

int Smartphone::GetJadra() {

return jadra;

}

void Smartphone::SetNameOfCorpus(string nameOfCompany) {

this->nameOfCompany = new string(nameOfCompany);

}

string Smartphone::GetNameOfCorpus() {

return \*nameOfCompany;

}

void Smartphone::SetChast(double chast) {

this->chast = chast;

}

double Smartphone::GetChast() {

return chast;

}

void Smartphone::SetStelus(bool stelus) {

this->stelus = stelus;

}

bool Smartphone::GetStelus() {

return stelus;

}

void Smartphone::print() {

cout << "Smartphone:" << endl;

cout << "Название компании: " << \*nameOfCompany << endl;

cout << "Кол-во ядер: " << jadra << endl;

cout << "Частота: " << chast << endl;

cout << "Поддерживает ли телефон стилус: ";

if (stelus) {

cout << "Да" << endl;

}

else {

cout << "Нет" << endl;

}

}

Smartphone::~Smartphone() {

//delete nameOfCompany;

}

Файл Source.cpp

#include "Vector.h"

#include "Laptop.h"

#include "Personal.h"

#include "Server.h"

#include "Smartphone.h"

void main() {

string name;

int jad;

double ch;

setlocale(LC\_ALL, "rus");

int d = 1;

int select, select1, index;

int size;

bool flag = 1;

//Laptop a[2];

//a[0] = Laptop("1", 1, 1, 1);

//a[1] = Laptop("2", 2, 2, 2);

//Stack<Laptop> st(2);

//st.push\_back(a[0]);

//st.push\_back(a[1]);

//st.pop\_back();

cout << "1 - создать Laptop, 2 - создать Personal, 3 - создать Server, 4 - создать Smartphone, другой - выход" << endl;

cin >> select;

cout << endl;

switch (select)

{

case 1: {

int wei;

cout << "Введите размер ветора: "; cin >> size; cout << endl;

int index1 = 0;

Stack<Laptop> st(size);

Laptop\* arr = new Laptop[size + 1];

while (flag) {

cout << "Введите название компании: "; cin >> name;

cout << endl << "Введите кол-во ядер: ";

cin >> jad; cout << endl << "Введите частоту: ";

cin >> ch; cout << endl << "Введите вес: ";

cin >> wei; cout << endl;

arr[index1] = Laptop(name, jad, ch, wei);

st.push\_back(arr[index1]);

index1++;

cout << "0 - закончить заполнение, 1 - продолжить" << endl;

cin >> flag;

cout << endl;

}

flag = 1;

while (flag) {

cout << "1 - найти, 2 - удалить, 3 - вывести по индексу, 4 - вывести всё" << endl;

cin >> select1; cout << endl;

if (select1 == 1) {

double fn;

cout << "Введите частоту: "; cin >> fn; cout << endl;

st.find(fn);

cout << endl;

}

else if (select1 == 2) {

st.pop\_back();

cout << endl;

}

else if (select1 == 3) {

cout << "Введите индекс элемента: "; cin >> index; cout << endl;

st.at(index);

cout << endl;

}

else {

for (int i = 0; i < st.count; i++) {

st.at(i);

cout << endl;

}

}

cout << "Хотите закончить?(0 - да, 1 - нет)" << endl; cin >> flag; cout << endl;

}

break;

}

case 2: {

int wei;

cout << "Введите размер ветора: "; cin >> size; cout << endl;

int index1 = 0;

Stack<Personal> st(size);

Personal\* arr = new Personal[size + 1];

while (flag) {

cout << "Введите название компании: "; cin >> name;

cout << endl << "Введите кол-во ядер: ";

cin >> jad; cout << endl << "Введите частоту: ";

cin >> ch; cout << endl << "Введите номер компании: ";

cin >> wei; cout << endl;

arr[index1] = Personal(name, jad, ch, wei);

st.push\_back(arr[index1]);

index1++;

cout << "0 - закончить заполнение, 1 - продолжить" << endl;

cin >> flag;

cout << endl;

}

flag = 1;

while (flag) {

cout << "1 - найти, 2 - удалить, 3 - вывести по индексу, 4 - вывести всё" << endl;

cin >> select1; cout << endl;

if (select1 == 1) {

double fn;

cout << "Введите частоту: "; cin >> fn; cout << endl;

st.find(fn);

cout << endl;

}

else if (select1 == 2) {

st.pop\_back();

cout << endl;

}

else if (select1 == 3) {

cout << "Введите индекс элемента: "; cin >> index; cout << endl;

st.at(index);

cout << endl;

}

else {

for (int i = 0; i < st.count; i++) {

st.at(i);

cout << endl;

}

}

cout << "Хотите закончить?(0 - да, 1 - нет)" << endl; cin >> flag; cout << endl;

}

break;

}

case 3: {

int wei;

cout << "Введите размер ветора: "; cin >> size; cout << endl;

int index1 = 0;

Stack<Server> st(size);

Server\* arr = new Server[size + 1];

while (flag) {

cout << "Введите название компании: "; cin >> name;

cout << endl << "Введите кол-во ядер: ";

cin >> jad; cout << endl << "Введите частоту: ";

cin >> ch; cout << endl << "Введите максимальное кол-во пользователей: ";

cin >> wei; cout << endl;

arr[index1] = Server(name, jad, ch, wei);

st.push\_back(arr[index1]);

index1++;

cout << "0 - закончить заполнение, 1 - продолжить" << endl;

cin >> flag;

cout << endl;

}

flag = 1;

while (flag) {

cout << "1 - найти, 2 - удалить, 3 - вывести по индексу, 4 - вывести всё" << endl;

cin >> select1; cout << endl;

if (select1 == 1) {

double fn;

cout << "Введите частоту: "; cin >> fn; cout << endl;

st.find(fn);

cout << endl;

}

else if (select1 == 2) {

st.pop\_back();

cout << endl;

}

else if (select1 == 3) {

cout << "Введите индекс элемента "; cin >> index; cout << endl;

st.at(index);

cout << endl;

}

else {

for (int i = 0; i < st.count; i++) {

st.at(i);

cout << endl;

}

}

cout << "Хотите закончить?(0 - да, 1 - нет)" << endl; cin >> flag; cout << endl;

}

break;

}

case 4: {

int wei;

cout << "Введите размер ветора: "; cin >> size; cout << endl;

int index1 = 0;

Stack<Smartphone> st(size);

Smartphone\* arr = new Smartphone[size + 1];

while (flag) {

cout << "Введите название компании: "; cin >> name;

cout << endl << "Введите кол-во ядер: ";

cin >> jad; cout << endl << "Введите частоту: ";

cin >> ch; cout << endl << "Есть ли стилус(1 - да, 0 - нет): ";

cin >> wei; cout << endl;

arr[index1] = Smartphone(name, jad, ch, wei);

st.push\_back(arr[index1]);

index1++;

cout << "0 - закончить заполнение, 1 - продолжить" << endl;

cin >> flag;

cout << endl;

}

flag = 1;

while (flag) {

cout << "1 - найти, 2 - удалить, 3 - вывести по индексу, 4 - вывести всё" << endl;

cin >> select1; cout << endl;

if (select1 == 1) {

double fn;

cout << "Введите частоту: "; cin >> fn; cout << endl;

st.find(fn);

cout << endl;

}

else if (select1 == 2) {

st.pop\_back();

cout << endl;

}

else if (select1 == 3) {

cout << "Введите индекс элемента "; cin >> index; cout << endl;

st.at(index);

cout << endl;

}

else {

for (int i = 0; i < st.count; i++) {

st.at(i);

cout << endl;

}

}

cout << "Хотите закончить?(0 - да, 1 - нет)" << endl; cin >> flag; cout << endl;

}

break;

}

default:

break;

}

}