МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

|  |
| --- |
| Институт информационных технологий |
| Кафедра математического и программного обеспечения |
| Объектно-ориентированное программирование |
|  |

ЛАБОРАТОРНАЯ РАБОТА №1

|  |  |
| --- | --- |
| Исполнитель: | студент  группы 1ПИб-02-3оп-23  Богданов  Ренат Алексеевич |
| Руководитель: | Кустов  Максим Александрович |
| Оценка: |  |
| Подпись: |  |

2024 год

**ТРЕБОВАНИЯ К РАЗРАБОТКЕ**

1. Сначала спроектируйте класс.
2. Каждый класс должен быть оформлен в отдельных файлах: заголовочный (.h) и файл с кодом (.cpp).
3. Запрещается использовать обработку исключительных ситуаций и генерировать исключения.
4. Придерживайтесь принципа DRY (Don’t repeat yourself).
5. Обязательно наличие комментариев.
6. Внимательно прочитать примечания (см. ниже).

**ЦЕЛЬ РАБОТЫ**

Изучить способы определения классов, правила доступа к элементам; приобрести практические навыки работы с объектами класса; изучить принципы и механизмы создания абстрактных типов данных.

Изучить синтаксис и семантику определения и вызова конструкторов и деструкторов. Приобрести практические навыки создания и уничтожения объектов. Изучить особенности применения различных видов конструкторов.

**ЗАДАНИЯ**

1. Ознакомьтесь с УМП по ООП часть 1 раздел IV.
2. Спроектируйте и разработайте алгоритм и программу, реализующую абстрактный тип данных (АТД) – класс, согласно варианту задания (А4. Инструментальные среды для разработки программного обеспечения):
   1. Предусмотрите закрытую реализацию и открытый интерфейс.
   2. Интерфейс должен содержать функции вывода содержимого и функции доступа к некоторым полям.
   3. Элементы данных должны быть представлены из 5 полей, один из которых должен быть статическим, второй иметь тип указатель, а третий тип перечисление, остальные на свое усмотрение.
   4. Примените основные виды конструкторов и определитесь с деструктором класса.
   5. Проверьте работоспособность АТД на тестовом наборе данных.

**Код программы**

ide.h

#pragma once

#include <string>

#include <iostream>

enum class IdeType {

TextEditor,

FullFeatured,

CloudBased

};

class Ide {

public:

Ide();

Ide(const std::string& name, const std::string& version, IdeType type);

Ide(const Ide& other);

~Ide();

void showInfo() const;

Ide\* getAdress() const;

IdeType getType() const;

const std::string& getName() const;

const std::string& getVersion() const;

static int getId();

private:

static int id;

Ide\* adress;

IdeType type;

std::string name;

std::string version;

void initialize();

};

ide.cpp

#include "ide.h"

int Ide::id = 0;

// Конструктор по умолчанию

Ide::Ide() : name("Noname IDE"), version("0.1"), type(IdeType::FullFeatured) {

initialize();

}

// Конструктор с параметрами

Ide::Ide(const std::string& name, const std::string& version, IdeType type) :

name(name), version(version), type(type) {

initialize();

}

// Конструктор копирования

Ide::Ide(const Ide& other) :

name(other.name), version(other.version), type(other.type) {

initialize();

}

// Деструктор

Ide::~Ide() {}

// Метод для вывода информации

void Ide::showInfo() const {

std::cout << "Name: " << name << std::endl;

std::cout << "Version: " << version << std::endl;

std::cout << "Type: ";

switch (type) {

case IdeType::TextEditor:

std::cout << "Text Editor" << std::endl;

break;

case IdeType::FullFeatured:

std::cout << "Full Featured" << std::endl;

break;

case IdeType::CloudBased:

std::cout << "Cloud Based" << std::endl;

break;

default:

std::cout << "Unknown Type" << std::endl;

break;

}

std::cout << "ID: " << id << std::endl;

std::cout << "Address: " << adress << std::endl;

}

// Методы доступа к полям

const std::string& Ide::getName() const { return name; }

Ide\* Ide::getAdress() const { return adress; }

const std::string& Ide::getVersion() const { return version; }

IdeType Ide::getType() const { return type; }

int Ide::getId() { return id; }

void Ide::initialize()

{

++id;

adress = this;

}

*main.cpp*

#include <iostream>

#include "ide.h"

int main() {

Ide ide1;

ide1.showInfo();

std::cout << std::endl;

Ide ide2("Sublime Text", "4180", IdeType::TextEditor);

ide2.showInfo();

std::cout << std::endl;

Ide ide3("Visual Studio Code", "1.94.0", IdeType::FullFeatured);

ide3.showInfo();

std::cout << std::endl;

Ide ide4(ide3);

ide4.showInfo();

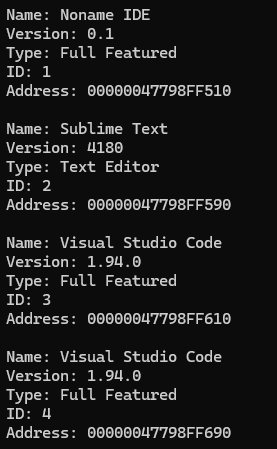
std::cout << std::endl;

std::cin.get();

return 0;

}

**Результат работы программы**

****

**Контрольные вопросы**

1. *Дайте определение понятия «класс». Сформулируйте правила доступа к его элементам.*

Класс – это гетерогенный агрегатный тип, имеющий элементы, данные и элементы функций, обеспечивающий сокрытие данных и наследование.

Правила доступа к членам класса в C++ определяются с помощью модификаторов доступа:

* Модификатор public (общий). Указывает на то, что свойства и методы доступны без ограничений. 4
* Модификатор private (закрытый). Указывает на то, что свойства и методы доступны только внутри класса.
* Модификатор protected (защищённый). Указывает на то, что свойства и методы доступны только наследникам класса.

1. *С какой целью в классе объединены компонентные данные и компонентные функции?*

Компонентные данные и функции объединены в классе для реализации инкапсуляции. Это означает, что они хранятся вместе, и доступ к данным осуществляется только через функции-члены класса. Это обеспечивает:

* Защиту данных: Скрытие внутренних деталей реализации от внешнего кода.
* Контроль над модификацией данных: Установка ограничений на доступ и изменения данных через функции-члены.
* Повышение модульности и повторного использования кода: Класс становится самостоятельной единицей, которую можно использовать повторно в разных частях программы.

1. Каким образом осуществляется доступ к открытым и закрытым элементам?

Открытые элементы: к открытым элементам можно обращаться непосредственно по имени из любой части программы, где объявлен объект класса.

Закрытые элементы: к закрытым элементам можно обращаться только изнутри самого класса. Для доступа к ним извне необходимо использовать публичные функции-члены, которые предоставляют контролируемый интерфейс для взаимодействия с закрытыми данными.

1. *Опишите назначение дружественных функций, назовите их разновидности.*

Дружественная функция должна быть объявлена внутри объявления класса, по отношению к которому она является дружественной, с использованием спецификатора friend. Такая функция имеет доступ к закрытым элементам класса наряду с компонентными функциями. Дружественными могут быть как внешние, так и компонентные функции, в этом случае для указания имени функции в дружественном классе используется оператор разрешения области видимости :: .

1. *Что понимается под указателем this?*

Ключевое слово this неявно объявляет указатель на себя. Может иметь место только в нестатических компонентных функциях. Заменяет объявление закрытого элемента:

1. *Каковы особенности использования статических компонентных данных?*

Принадлежность классу, а не объекту: Статические члены данных принадлежат всему классу, а не конкретному объекту.

Доступность из любого объекта: К статическим членам данных можно обращаться из любого объекта класса.

Инициализация только один раз: Статические данные инициализируются только один раз при первом использовании класса.

Доступ через имя класса: К статическим членам данных можно обращаться через имя класса, используя оператор разрешения области видимости (::).

1. *В чем заключается синтаксис и семантика компонентных функций static и const?*

Синтаксис:

static: Используется ключевое слово "static" перед именем функции-члена.

const: Используется ключевое слово "const" после объявления типа возвращаемого значения функции-члена.

Семантика:

static: Статические функции-члены не имеют доступа к объектам класса и не используют указатель this. Они могут использоваться для общих действий, которые не связаны с конкретным объектом.

const: Константы функции-члены не могут изменять состояние объекта, к которому они относятся. Они могут использоваться для чтения данных или для выполнения операций, не влияющих на состояние объекта.

1. *Каким образом могут изменяться компонентные данные объектов, объявленных константами?*

Объекты класса могут быть объявлены как константы, и это гарантирует постоянство компонентных данных. Отказаться можно с помощью const\_cast или индивидуально для отдельных компонентных данных с использованием ключевого слова mutable.

1. *Каковы особенности создания вложенных классов?*

Классы могут быть созданы внутри блоков, в том числе внутри других классов, как часть реализации более сложной конструкции. Использование таких классов невозможно за пределами соответствующих блоков.

Вложение классов позволяет вкладывать определения функций - компонентная функция должна быть определена внутри своего локального класса, и на данную функцию нельзя ссылаться вне области видимости ее локального класса.

**Вывод**

В ходе выполнения лабораторной работы были освоены навыки создания и обработки классов. Были изучены способы определения классов и правила доступа к их элементам: получены практические навыки работы с объектами класса, а также освоены принципы и механизмы создания абстрактных типов данных. Освоены синтаксис и семантика определения и вызова конструкторов и деструкторов: получены практические навыки создания и уничтожения объектов. Изучены особенности применения различных видов конструкторов.