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 №3

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

2024 год

**Цель работы**

Изучить понятие перегрузки, приобрести практические навыки перегрузки отдельных операторов.

## Задания

1. Модифицируйте абстрактный тип данных, реализованный по заданию раздела V, согласно варианту задания (А4. Инструментальные среды для разработки программного обеспечения), перегрузив все доступные операторы.

2. 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();

// Перегруженный оператор присваивания (=)

Ide& operator=(const Ide& other) {

if (this != &other) { // Проверка на самоприсваивание

name = other.name;

version = other.version;

}

return \*this; // Возвращаем ссылку на текущий объект

}

// Перегруженный оператор сравнения (==)

bool operator==(const Ide& other) const {

return (name == other.name) && (type == other.type) && (version == other.version);

}

// Перегруженный оператор сравнения (!=)

bool operator!=(const Ide& other) const {

return !((name == other.name) && (type == other.type) && (version == other.version));

}

// Перегруженный оператор сложения (+)

Ide operator+(const Ide& other) const {

return Ide(name + " + " + other.name, version, type);

}

// Перегруженный оператор вычитания (-)

Ide operator-(const Ide& other) const {

return Ide(name + " - " + other.name, version, type);

}

// Перегрузка логического И (&&)

bool operator&&(const Ide& other) const {

return (type == other.type) && (version == other.version);

}

// Перегрузка оператора ввода (<<)

friend std::ostream& operator<<(std::ostream& out, const Ide& ide) {

out << "Name: " << ide.name << std::endl;

out << "Version: " << ide.version << std::endl;

out << "Type: " << (int)ide.type << std::endl;

return out;

}

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() {

setlocale(LC\_ALL, "RU");

Ide ide1;

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

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

Ide ide4(ide3);

std::cout << "IDE 1:" << std::endl;

ide1.showInfo();

std::cout << std::endl;

std::cout << "IDE 2:" << std::endl;

ide2.showInfo();

std::cout << std::endl;

// Присваивание

ide4 = ide1;

std::cout << "IDE 4 (после присваивания IDE 1):" << std::endl;

ide4.showInfo();

std::cout << std::endl;

// Сравнение

std::cout << "IDE 1 == IDE 2: " << (ide1 == ide2) << std::endl;

std::cout << "IDE 1 == IDE 4: " << (ide1 == ide4) << std::endl;

std::cout << "IDE 1 != IDE 2: " << (ide1 != ide2) << std::endl;

std::cout << std::endl;

// Сложение

ide2 = ide2 + ide3;

std::cout << "IDE 2 (после сложения с IDE 3):" << std::endl;

ide2.showInfo();

std::cout << std::endl;

// Вычитание

ide2 = ide2 - ide3;

std::cout << "IDE 2 (после вычитания IDE 3):" << std::endl;

ide2.showInfo();

std::cout << std::endl;

// Логическое И

std::cout << "IDE 1 && IDE 4: " << (ide1 && ide4) << std::endl;

std::cout << "IDE 2 && IDE 4: " << (ide2 && ide4) << std::endl;

std::cout << std::endl;

// Оператор ввода (<<)

std::cout << "IDE 3:" << std::endl;

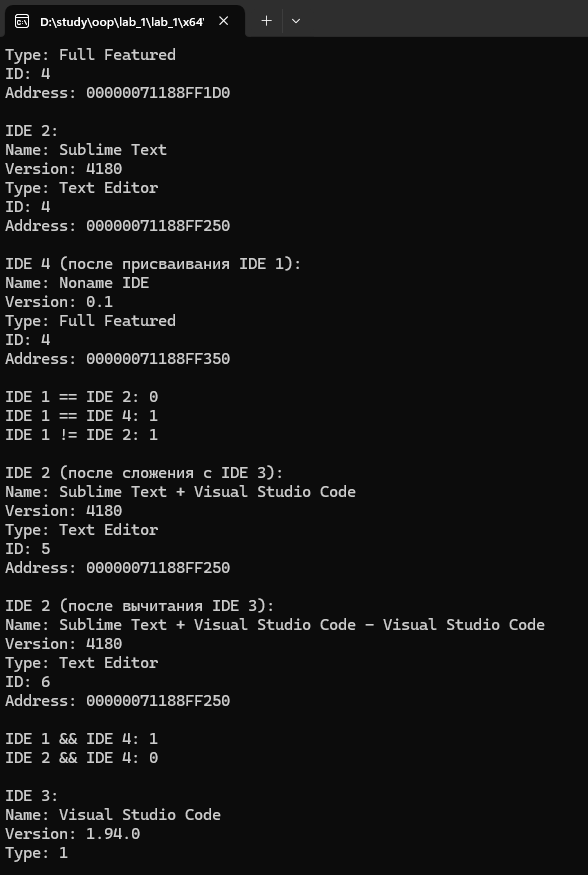
std::cout << ide3 << std::endl;

std::cin.get();

return 0;

}

Результат работы программы



Ответы на контрольные вопросы

1. *Назовите алгоритм соответствия для каждого аргумента.*

Алгоритм соответствия для каждого аргумента следующий:

1. Использовать точное соответствие, если оно найдено.
2. Проверить поддержку стандартных типов.
3. Проверить стандартные преобразования типов.
4. Проверить преобразования, определяемые пользователем.
5. Использовать соответствие для аргументов, если оно найдено.
6. *С какой целью применяется перегрузка операторов?*

Перегрузка операторов позволяет абстрактному типу данных использовать синтаксис выражений С++, что существенно облегчает написание программ.

1. *Для каких операторов нельзя использовать перегрузку?*

Троичный условный оператор выражения «?:», оператор разрешения области видимости «:: », оператор членства «.» и оператор селектора члена объекта «.\*».

1. *В каком случае используется функция-член, имеющая пустой список аргументов?*

Для перегрузки унарного оператора используется функция-член, имеющая пустой список аргументов, так как единственный аргумент оператора является неявным аргументом.

1. *Что произойдет, если перегруженная операция [ ] будет иметь тип возвращаемого значения int, а не int&?*

Если перегруженная операция [ ] будет иметь тип возвращаемого значения int, а не int&, то возвращаемое значение будет являться копией элемента массива. При изменении возвращаемого значения исходный элемент массива не изменится.

Вывод

В ходе выполнения лабораторной работы был изучен механизм перегрузки операторов в языке С++ и приобретены практические навыки применения этого механизма.