ЛАБОРАТОРНАЯ РАБОТА №2

Курс «Объектно-ориентированное программирование»

Тема: Классы и объекты. Инкапсуляция. Перегрузка операторов.

Цель: Научиться программировать на языке C++ классы, инкапсулировать данные и методы, перегружать унарные и бинарные операторы.

Темы для предварительной проработки [УСТНО]**:**

- Классы. Создание и удаление объектов классов.
- Принцип инкапсуляции в ООП. Спецификаторы доступа.
- Статические члены и функции класса.
- Перегрузка операторов.
- Файловый ввод-вывод.

Индивидуальные задания [КОД] **:**

1. Написать классы `Vector` и `Matrix` для хранения и обработки одномерных и двумерных массивов, соответственно. Реализовать задание 2 лабораторной работы №1 с помощью созданных классов. Все функции оформить в виде методов классов.
   В коде отразить следующее:
   - Выделение и освобождение динамической памяти производить в конструкторах и деструкторах классов, соответственно.
   - В классе Vector перегрузить оператор индексации []. В классе `Matrix` добавить методы `T at(int i, int j) const` и `setAt(int i, int j, T val)`, которые позволяют получить и установить значение элемента массива с индексом [i][j], T – это тип элементов массива по варианту (`int` или `double`).
   - Перегрузить операторы инкремента и декремента (как префиксного, так и постфиксного). Смысл инкремента / декремента всего массива заключается в инкременте / декременте каждого элемента массива.

2. Написать класс `Fraction` для представления обыкновенных дробей (как отношения двух целых чисел x/y). Перегрузить операторы +, -, *, / для дробей. Реализовать метод `void reduce()` для сокращения дроби, а также статические методы:
   - `int gcd(int n, int m)`
     функция для нахождения наибольшего общего делителя чисел `n` и `m`;
   - `void printAsFraction(double decFraction)`
   - `void printAsFraction(char* decFraction)`
     перегруженные методы вывода десятичной дроби в виде обыкновенной (например, при значении `decFraction = 0.43` на экране должно вывестись `43/100`, при `0.25` – `1/4` и т.д.).

   Также реализовать в виде статического члена класса счетчик всех созданных на данный момент в программе экземпляров дробей.
   Продемонстрировать работу созданного класса. Создать несколько объектов дробей. Произвести операции сложения, вычитания, умножения и деления дробей. Вывести на экран результаты. Показать также результаты работы статических методов класса.

3. Написать собственный класс, в соответствии с вариантом. Продемонстрировать в коде инкапсуляцию данных, применение конструкторов без параметров и с параметрами для инициализации данных. Класс должен содержать метод `serialize()` для сохранения данных класса в файл и метод `deserialize()` для чтения данных класса из файла по умолчанию в текущей директории, а также перегруженные методы `serialize(const std::string& filename)` и `deserialize(const std::string& filename )` для работы с файлом с указанным в параметре именем.

   Примечание. Вы можете вводить дополнительные закрытые свойства и метода класса, не указанные явно в задании, но помогающие решить поставленные задачи.

Контрольные вопросы [ОТЧЕТ]**:**

1. Чем отличаются классы и структуры в C++ технически и концептуально?
2. Чем отличаются понятия «класс» и «объект»? Приведите примеры.
3. Опишите традиционную файловую структуру ООП проекта.
4. Что такое header guards (include guards)? Какую проблему они позволяют решить, и какой есть второй способ ее решить?
5. Что такое инкапсуляция? Какие есть спецификаторы доступа и что они означают?
6. Что такое конструктор класса? Деструктор класса? Типы конструкторов.
7. Что означает указатель this?
8. В чем заключаются особенности статических членов и функций классов?
9. Как производится перегрузка методов и перегрузка операторов?
10. Укажите особенности работы с константными объектами классов.
11. Как можно запретить создание объекта класса?
12. В чем отличие глубокого копирования от поверхностного (deep and shallow copying)?

Рекомендуемые источники:

[1] Страуструп Б. Язык программирования C++. 3-е изд. / Б. Страуструп. – СПб.; М.: Невский диалект; Издательство «БИНОМ», 1999. – 991 с.

[2] Лафоре Р. Объектно-ориентированное программирование в C++ / Р. Лафоре. – СПб.: Питер, 2013. – 928 с.

[3] Прата С. Язык программирования C++ / С. Прата; пер. с англ. Ю. Н. Артеменко. – Лекции и упражнения, 6-е изд.: Пер. с англ. – М.: Вильямс, 2012. – 1248 с.

[4] Дейтел Х. М. Как программировать на C++ / Х. М. Дейтел, П. Дж. Дейтел. – М.: Издательство «БИНОМ», 2000. – 1024 с.

[5] Мейерс С. Эффективное использование STL / С. Мейерс. – СПб: Питер, 2002. – 224 с.

[6] Александреску А. Современное проектирование на C++: обобщенное программирование и прикладные шаблоны проектирования / А. Александреску; пер. с англ. и ред. канд. физ.-мат. наук Д. А. Клюшина. – СПб.: Вильямс, 2008. – 336 с.

Приложение А. Варианты индивидуальных заданий.

Вариант 0 (введен для примера, фрагменты кода можно найти в приложении В)

Класс РОБОТ.
Данные: имя (название), текущие координаты робота, уровень заряда батареи, скорость передвижения. Робот может двигаться в плоскости влево, вправо, вперед, назад, а также ускоряться или замедляться. При увеличении скорости каждый шаг робота увеличивается на единицу и отнимает на 3 единицы больше заряда. В основной функции создать 3 роботов со 100%-ным уровнем заряда. Первому роботу задать начальные координаты при создании, второму – в сеттере после создания, третьему – присвоить по умолчанию при создании. Пять раз переместить всех роботов вправо, при этом первого робота ускорить на первых трех шагах, второго – на втором и дважды на четвертом шагах, а третьего – на пятом шаге. Вывести на консоль уровень заряда и координаты всех роботов до и после перемещения.

Вариант 1

Класс СТУДЕНТ.
Данные: ФИО, пол, год рождения, год поступления, номер зачетки, средний балл.
Создать массив из 3 студентов и установить их личные данные с помощью сеттеров. Еще одного студента создать отдельно в куче и установить его данные в конструкторе с параметрами. В главной функции проимитировать три сессии – случайным образом сформировать по 4 новые оценки и пересчитать в отдельном методе средний балл. Вывести результаты студентов, отсортировав их в порядке убывания среднего балла.

Вариант 2

Класс МАГАЗИН.
Данные: название, адрес, год основания, номер, суммарная прибыль.
Создать два объекта-магазина в куче и один в стеке. Данные первых двух заполнить с помощью сеттеров, а третий проинициализировать с помощью конструктора с параметрами. В главной функции проимитировать отдельно продажи за сентябрь, октябрь и ноябрь. Вывести все магазины отдельно в двух рейтинговых списках (сначала один, затем второй): 1) отсортировать в порядке убывания общей прибыли за 3 месяца; 2) отсортировать в порядке убывания среднего прироста прибыли за 3 месяца.

Вариант 3

Класс МУЗЫКАНТ.
Данные: имя, фамилия, пол, год рождения, инструмент, рейтинг.
Создать трех музыкантов в стеке и одного в куче. Данные первых трех заполнить с помощью сеттеров, а четвертый проинициализировать с помощью конструктора с параметрами. В главной функции проимитировать три концерта и голосование зрителей по их результатам (нарастить рейтинг каждого музыканта на случайное число из диапазона 1..5). Вывести список музыкантов в порядке убывания суммарного рейтинга, но клавишники должны идти впереди, независимо от рейтинга ☺.

Вариант 4

Класс ТЕЛЕФОН.
Данные: модель, номер телефона, последний набранный номер, остаток на счету.
Создать два телефона в куче. Установить данные первого телефона с помощью сеттеров, второго – в конструкторе с параметрами. В главной функции проимитировать десять звонков – позвонить по нескольку раз на 3 номера, в том числе на номер другого телефона в программе. За каждую минуту разговора снимается 0,5 грн. Вывести всю информацию о телефонах после проведенных звонков.

Вариант 5

Класс КОНДИЦИОНЕР.
Данные: фирма, модель, вес, температура, режим, год выпуска.
Создать 2 кондиционера в куче и проинициализировать их с помощью конструкторов с параметрами. Еще один кондиционер создать отдельно в стеке и установить его данные с помощью сеттеров. В главной функции проимитировать настройку кондиционеров персоналом помещений – установить каждому режим; если выбран режим охлаждения, то установить также температуру. Проделать эту процедуру три раза. Вывести информацию об использовании кондиционеров – режим, в котором на данный момент работает техника, среднее изменение температуры за весь период настройки.

Вариант 6

Класс АВТОМОБИЛЬ.
Данные: фирма, модель, номер, цена, год выпуска, пробег.
Создать массив из 3 автомобилей и установить их данные с помощью сеттеров. Еще один автомобиль создать отдельно в куче и установить его данные в конструкторе с параметрами. В главной функции проимитировать три дальние поездки на всех машинах – случайным образом сформировать расстояния и нарастить суммарный пробег. Вывести все автомобили в порядке убывания пробега, но машины с годом выпуска от 2011 вывести первыми.

Вариант 7

Класс СТАДИОН.
Данные: адрес, футбольный клуб, количество секторов, вместимость, посещаемость.
Создать 2 стадиона в куче и один в стеке. Установить данные первых двух с помощью сеттеров, данные третьего установить в конструкторе с параметрами. В главной функции проимитировать четыре матча – случайным образом сформировать количество пришедших посетителей и просчитать процент заполнения каждого стадиона. Вывести информацию о стадионах, отсортировав их в порядке убывания среднего процента заполнения за четыре матча.

Вариант 8

Класс ФОТОГРАФ.
Данные: имя, фамилия, пол, год рождения, год начала деятельности, рейтинг.
Создать 2 фотографов в куче и одного в стеке. Данные первых двух заполнить с помощью сеттеров, а третий проинициализировать с помощью конструктора с параметрами. В главной функции проимитировать три фотосессии и голосование зрителей по их результатам (нарастить рейтинг каждого фотографа на случайное число из диапазона 0.0..1.0; если количество проголосовавших людей меньше 10, то не наращивать рейтинг). Вывести список фотографов в порядке убывания суммарного рейтинга.

Вариант 9

Класс САМОЛЕТ.
Данные: модель, авиалинии, год выпуска, вместимость, количество пассажиров.
Создать массив из 3 самолетов и установить их данные с помощью сеттеров. Еще один самолет создать отдельно в куче и установить его данные в конструкторе с параметрами. В главной функции проимитировать три рейса – случайным образом сформировать фактическое количество пассажиров и просчитать процент заполнения каждого самолета. Вывести информацию о самолетах, отсортировав их в порядке убывания среднего процента заполнения за три рейса.

Вариант 10

Класс РЕСТОРАН.
Данные: название, адрес, год основания, рейтинг, количество посетителей за месяц.
Создать два ресторана в куче и один в стеке. Данные первых двух заполнить с помощью конструктора с параметрами, а третий – с помощью сеттеров. В главной функции проимитировать отдельно посещение и голосование посетителей за март, апрель и май. Вывести все рестораны отдельно в двух рейтинговых списках (сначала один, затем второй): 1) отсортировать в порядке убывания суммарного количества посетителей за 3 месяца; 2) отсортировать в порядке убывания среднего рейтинга за 3 месяца.

Приложение В. Фрагменты кода для варианта 0.

Задание 3.

Код заголовочного файла класса РОБОТА (robot.h).

```
#pragma once

#include <string>

class Robot
{
public:
	Robot();                                    // конструктор
	Robot(int x, int y);                        // конструктор с параметрами
	~Robot();                                   // деструктор

	void setName(const std::string& name);   // сеттеры
	void setPos(int x, int y);
	int getSpeed() const;                       // геттеры
	double getCharge() const;

	void moveLeft();           // переместиться влево
	void moveRight();          // переместиться вправо

	void accelerate();         // ускориться
	void slowDown();           // замедлиться
	                           // сериализация / десериализация:
	void serialize() const;
	void serialize(const std::string& filename) const;
	void deserialize();
	void deserialize(const std::string& filename);

	void printInfo() const;           // вывод информации о себе

private:
	std::string name_;         // имя робота
	int xpos_, ypos_;          // координаты робота
	double charge_;            // уровень заряда батареи
	int speed_;                // скорость (на сколько передвигается за один шаг)
};
```

Код файла реализации класса РОБОТА (robot.cpp):

```
#include "Robot.h"
#include <iostream>


Robot::Robot() : xpos_(10), ypos_(10)
{
	speed_ = 0;
	charge_ = 100.0;
}

Robot::Robot(int x, int y): xpos_(x), ypos_(y)
{
	speed_ = 0;
	charge_ = 100.0;
}

Robot::~Robot()
{
}
```

```cpp
void Robot::setName(const std::string& name)
{
	name_ = name;
}

void Robot::setPos(int x, int y)
{
	xpos_ = x;
	ypos_ = y;
}

double Robot::getCharge() const
{
	return charge_;
}

int Robot::getSpeed() const
{
	return speed_;
}

void Robot::moveLeft()
{
	xpos_ -= speed_;
	charge_ -= speed_ * 3;
}

void Robot::moveRight()
{
	xpos_ += speed_;
	charge_ -= speed_ * 3;
}

void Robot::accelerate()
{
	speed_++;
}

void Robot::slowDown()
{
	if (speed_ > 1)
		speed_--;
}

void Robot::printInfo() const
{
	std::cout << name_ << " : ";
	std::cout << "( " << xpos_ << "," << ypos_ << " ) ";
	std::cout << "CHARGE: " << charge_ << std::endl;
}

void Robot::serialize() const
{...}                              // здесь код для сохранения данных класса в файл...

void Robot::serialize(const std::string& filename) const
{...}

void Robot::deserialize()
{...}

void Robot::deserialize(const std::string& filename)
{...}
```

В главной функции можно написать, например, так:

```
Robot r1(300, 400), r2, r3;       // создаем трех роботов
r2.setPos(150, 200);

r1.setName("R2D2");
r2.setName("C3PO");
r3.setName("Sheldon");

std::cout << "Before:" << std::endl;
r1.printInfo();
r2.printInfo();
r3.printInfo();

// ---------------------------------- имитация перемещений и ускорений роботов
r1.accelerate();
r1.moveRight();
r2.moveRight();
r3.moveRight();

r1.accelerate();
r2.accelerate();
r1.moveRight();
r2.moveRight();
r3.moveRight();

r1.accelerate();
r1.moveRight();
r2.moveRight();
r3.moveRight();

r2.accelerate();
r2.accelerate();
r1.moveRight();
r2.moveRight();
r3.moveRight();

r3.accelerate();
r1.moveRight();
r2.moveRight();
r3.moveRight();
// ---------------------------------------------------------------------------

std::cout << "\nAfter:" << std::endl;
r1.printInfo();
r2.printInfo();
r3.printInfo();

r1.serialize("D:\\Robots\\1.txt");
r2.serialize("D:\\Robots\\2.txt");
r3.serialize("D:\\Robots\\3.txt");
```

Результат работы программы:

```
Before:
R2D2 : ( 300,400 ) CHARGE: 100
C3PO : ( 150,200 ) CHARGE: 100
Sheldon : ( 10,10 ) CHARGE: 100

After:
R2D2 : ( 312,400 ) CHARGE: 64
C3PO : ( 158,200 ) CHARGE: 76
Sheldon : ( 11,10 ) CHARGE: 97
```