**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**КЫРГЫЗСКОЙ РЕСПУБЛИКИ**

**КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. И.Раззакова**

**ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

Кафедра: **Программное обеспечение компьютерных систем**

Курс «Объектно-ориентированное программирование»

**ОТЧЕТ**

**Лабораторная работа №7**

Выполнил: студент группы ПИ-3-21

Закиров Мустафа

Проверил: Мусабаев Э.Б.

**Бишкек 2024**

# **Задание №1**

**Постановка задачи:**

1. Создать базовый класс Array с полями: массив типа unsigned и поле для хранения количества элементов у текущего объекта массива. Максимально возможный размер массива задается статической константой. Реализовать конструктор инициализации, задающий количество элементов и начальное значение (по умолчанию 0). Реализовать в классе Array виртуальную функцию поэлементного сложения массивов. Реализовать два класса, переопределив виртуальную функцию сложения. Вызывающая программа должна продемонстрировать все варианты вызова виртуальных функций.

**Исходные данные:**

**Исходный код программы на C++:**

#include <iostream>

using namespace std;

class Array {

public:

static const int MAX\_SIZE = 100;

int size;

unsigned int data[MAX\_SIZE];

Array(int size, unsigned int initial = 0) : size(size) {

for (int i = 0; i < size; ++i) {

data[i] = initial;

}

}

virtual void add(const Array& other) {

for (int i = 0; i < size && i < other.size; ++i) {

data[i] += other.data[i];

}

}

void display() const {

for (int i = 0; i < size; ++i) {

std::cout << data[i] << " ";

}

std::cout << std::endl;

}

};

class ArrayA : public Array {

public:

ArrayA(int size, unsigned int initial = 0) : Array(size, initial) {}

void add(const Array& other) override {

for (int i = 0; i < size && i < other.size; ++i) {

data[i] += other.data[i] \* 2;

}

}

};

class ArrayB : public Array {

public:

ArrayB(int size, unsigned int initial = 0) : Array(size, initial) {}

void add(const Array& other) override {

for (int i = 0; i < size && i < other.size; ++i) {

data[i] += other.data[i] \* other.data[i];

}

}

};

int main() {

Array arr(4, 1);

ArrayA arrA(5, 2);

ArrayB arrB(6, 3);

arrA.add(arr);

arrA.display();

arrB.add(arr);

arrB.display();

return 0;

}

**Тесты:**

# **Задание №2**

**Постановка задачи:**

Создать абстрактный базовый класс Figure с виртуальными методами вычисления площади и периметра. Создать производные класса: Rectangle (прямоугольник), Circle (круг), Trapezium (трапеция) со своими функциями площади и периметра. Самостоятельно определить, какие поля необходимы, какие из них можно задать в базовом классе, а какие в производных. Площадь трапеции: *S = (a + b) x h/2.*

**Исходные данные:**

**Исходный код программы на C++:**

#include <iostream>

#include <cmath>

using namespace std;

class Figure {

public:

virtual double area() const = 0;

virtual double perimeter() const = 0;

virtual ~Figure() {}

};

class Rectangle : public Figure {

private:

double width;

double height;

public:

Rectangle(double w, double h) : width(w), height(h) {}

double area() const override {

return width \* height;

}

double perimeter() const override {

return 2 \* (width + height);

}

};

class Circle : public Figure {

private:

double radius;

public:

Circle(double r) : radius(r) {}

double area() const override {

return 3.14159 \* radius \* radius;

}

double perimeter() const override {

return 2 \* 3.14159 \* radius;

}

};

class Trapezium : public Figure {

private:

double a;

double b;

double height;

public:

Trapezium(double sideA, double sideB, double h) : a(sideA), b(sideB), height(h) {}

double area() const override {

return (a + b) \* height / 2;

}

double perimeter() const override {

return a + b + 2 \* hypot((b - a) / 2, height);

}

};

int main() {

setlocale(LC\_ALL, "rus");

Rectangle rect(4, 5);

Circle circle(3);

Trapezium trapezium(3, 5, 4);

cout << "Площадь прямоугольника: " << rect.area() << endl;

cout << "Периметр прямоугольника: " << rect.perimeter() << endl;

cout << "Площадь круга: " << circle.area() << endl;

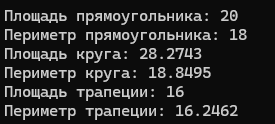
cout << "Периметр круга: " << circle.perimeter() << endl;

cout << "Площадь трапеции: " << trapezium.area() << endl;

cout <<

return 0;

}

**Тесты:**

# **Задание №3**

**Постановка задачи:**

Создать абстрактный базовый класс Currency (валюта) для работы с денежными суммами. Определить виртуальные функции перевода в рубли и вывода на экран. Реализовать производные классы Dollar (доллар) и Euro (евро) со своими функциями перевода и вывода на экран.

**Исходные данные:**

**Исходный код программы на C++:**

#include <iostream>

#include <iomanip>

using namespace std;

class Currency {

public:

virtual double toRuble(double amount) const = 0;

virtual void display(double amount) const = 0;

virtual ~Currency() {}

};

class Dollar : public Currency {

public:

double toRuble(double amount) const override {

return amount \* 90.95;

}

void display(double amount) const override {

cout << "$" << fixed << setprecision(2) << amount;

}

};

class Euro : public Currency {

public:

double toRuble(double amount) const override {

return amount \* 99.52;

}

void display(double amount) const override {

cout << "€" << fixed << setprecision(2) << amount;

}

};

int main() {

setlocale(LC\_ALL, "rus");

Dollar dollar;

Euro euro;

double amountInDollars = 1225;

double amountInEuros = 1500;

cout << "Значение в Долларах: ";

dollar.display(amountInDollars);

cout << " = " << dollar.toRuble(amountInDollars) << " Рублей" << endl;

cout << "Значение в Евро: ";

euro.display(amountInEuros);

cout << " = " << euro.toRuble(amountInEuros) << " Рублей" << endl;

return 0;

}

**Тесты:**

# **Ответы на вопросы:**

1. Что такое виртуальная функция (метод) при наследовании?

Виртуальный метод - это метод, используется в базовом классе, который дает возможность переопределения методов в производном классе при наследовании.

1. Чем отличается переопределение метода от его перегрузки при наследовании?

Виртуальные методы описываются с помощью ключевого слова virtual в базовом классе. При наследовании виртуальный метод может быть переопределен и перегружен. При перегрузке меняется прототип метода. При переопределении прототип метода остается неизменным.

1. Что такое чисто виртуальная функция? Для чего она используется?

**Чисто виртуальной функцией** называется такая виртуальная функция, которая не имеет определения в базовом классе. Чтобы определить чисто виртуальную функцию, после прототипа функции необходимо указать оператор присваивания (знак равенства) и ноль. Для чисто виртуальной функции блок с кодом функции (тело) не указывается.

1. Синтаксис виртуальной функции?

Синтаксис чисто виртуальной функции имеет вид:

virtual тип\_результата имя-функции (аргументы)=0;

1. Как называется класс, содержащий хотя бы одну чисто виртуальную функцию?

Класс, содержащий хотя бы одну чисто виртуальную функцию, называется абстрактным.

1. Приведите пример иерархической структуры при наследовании?

#include <iostream>

#include <string>

using namespace std;

// Базовый класс Animal

class Animal {

protected:

string name;

public:

Animal(const string& name) : name(name) {}

virtual string speak() const = 0; // Чисто виртуальная функция

};

// Дочерний класс Dog, наследуется от Animal

class Dog : public Animal {

public:

Dog(const string& name) : Animal(name) {}

virtual string speak() const override {

return "Woof!";

}

};

int main() {

Dog dog("Buddy");

cout << dog.speak() << endl;

return 0;

}

1. Как использовать виртуальные методы в многоуровневом наследовании?

Виртуальные методы в многоуровневом наследовании

При многоуровневом наследовании, как отмечалось выше, производный класс, в свою очередь, является базовым для следующего класса, и так далее. Образуется иерархическая структура, в вершине которой находится базовый класс, на основе которого по цепочке последовательного наследования создаются новые классы.