**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ КЫРГЫЗСКОЙ РЕСПУБЛИКИ**

**КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. И.РАЗЗАКОВА**

**ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

**КАФЕДРА ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ КОМПЬЮТЕРНЫХ СИСТЕМ**

**ОТЧЕТ**

**ПО ДИСЦИПЛИНЕ: ООП**

**ПО ТЕМЕ: Виртуальные методы**

**Лабораторная работа №6**

**ВЫПОЛНИЛ:** студент гр. ПИ(б)-3-18

Гадайбаев М.А.

**ПРОВЕРИЛА:** Каткова С.Н.

**Бишкек 2020**

**ВОПРОСЫ К ЗАЩИТЕ ЛАБОРАТОРНОЙ РАБОТЫ № 6**

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

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

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

При наследовании виртуальный метод может быть переопределен и перегружен. При перегрузке меняется прототип метода, а при переопределении прототип метода остаётся 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. Для того, чтобы производный класс имел свою реализацию производной функции.

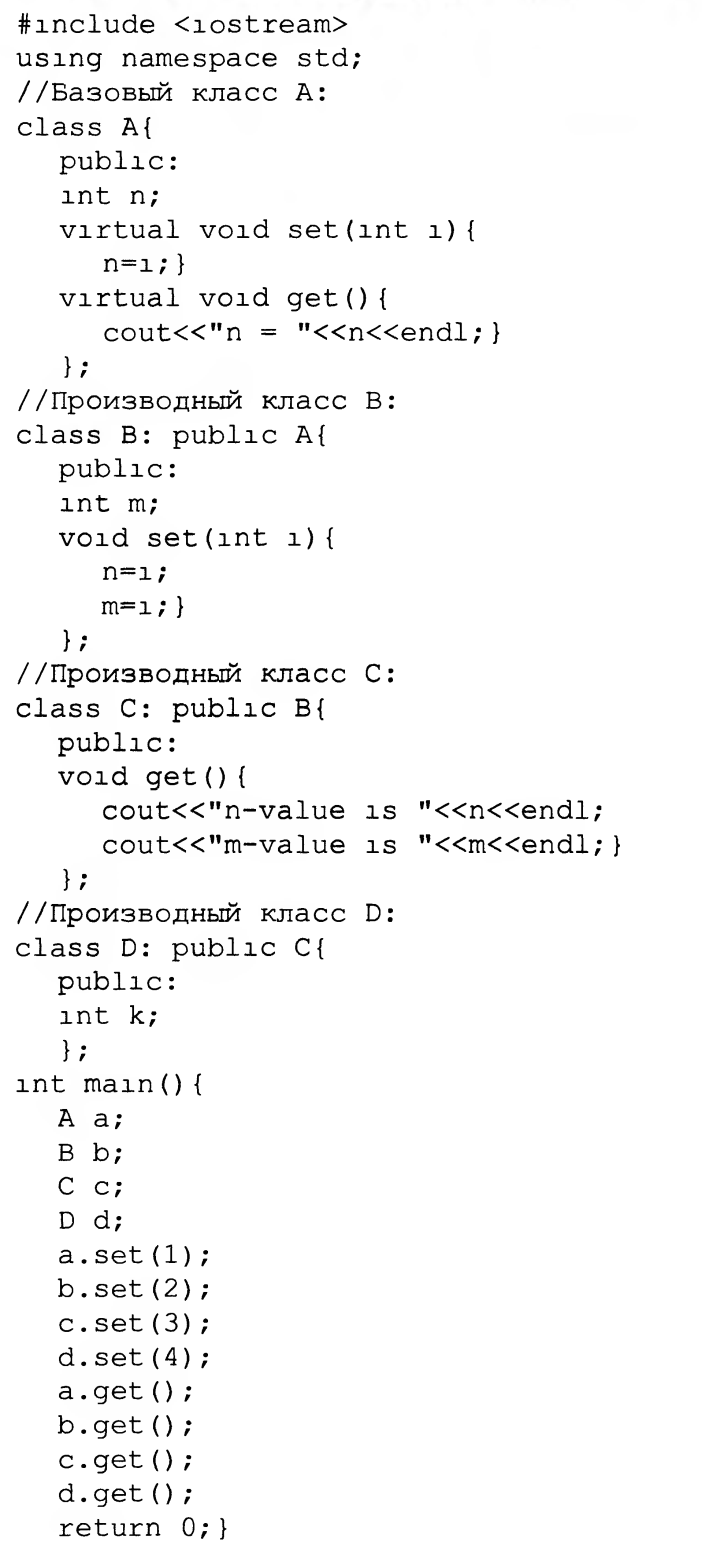
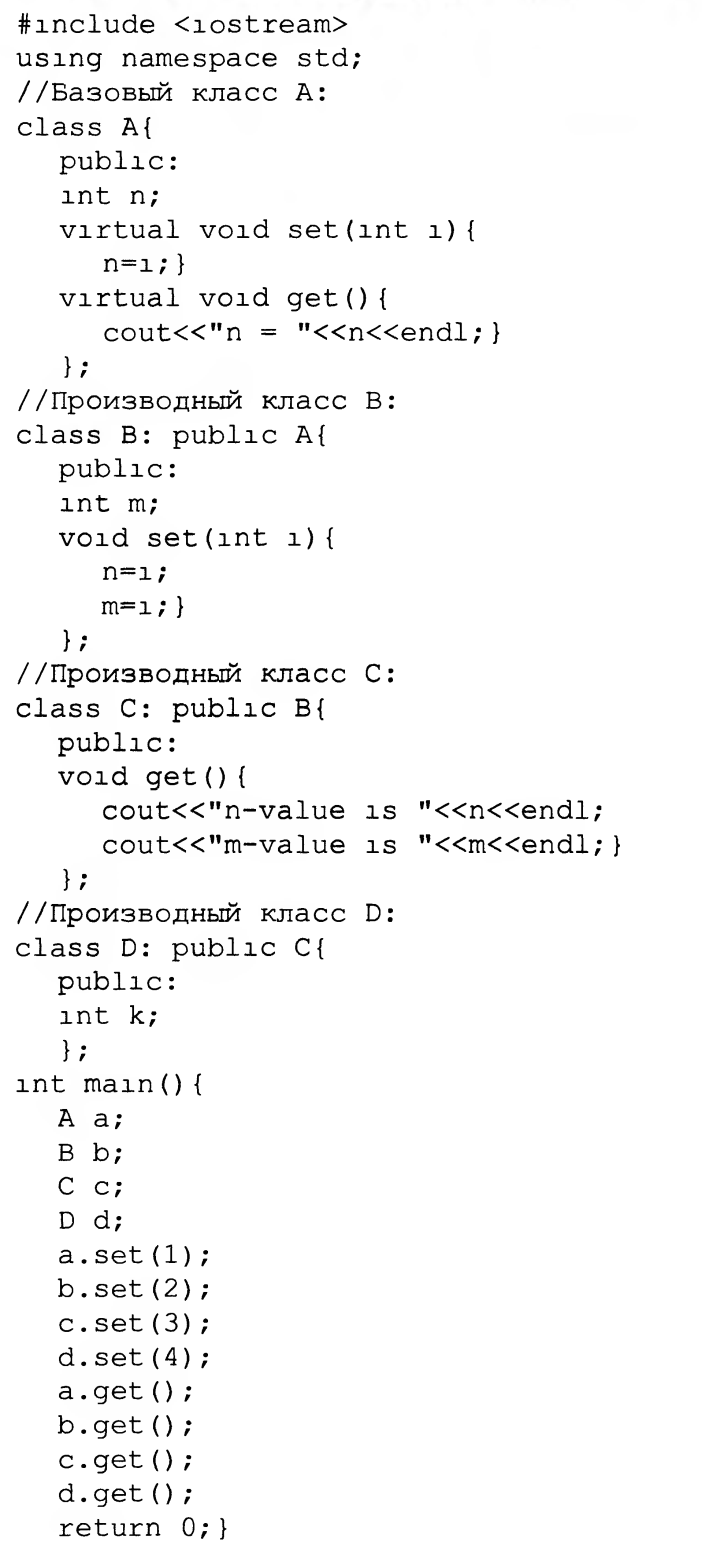
1. Синтаксис виртуальной функции?

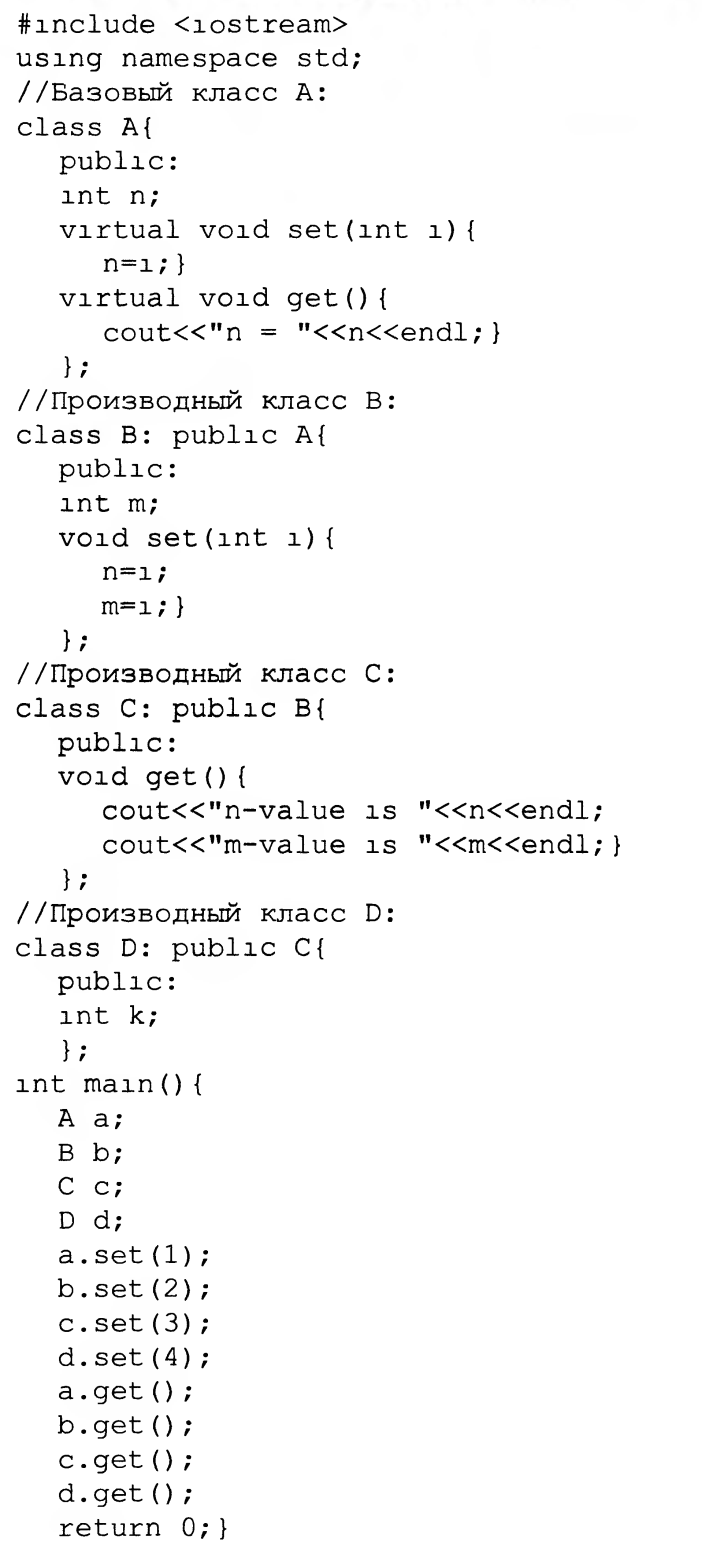
**virtual тип\_результата имя-функции (аргументы);**

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

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

1. Приведите пример иерархической структуры при наследовании?



1. Как использовать виртуальные методы в многоуровневом наследовании?

При многоуровневом наследовании, 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.

**Задание №1.**

**Постановка задачи:**

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.

**Описание входных данных:**

class Array – базовый класс. Поля: array – массив типа unsigned, col – переменная для хранения количества элемента у массива. Методы: set() – виртуальный метод, для вставки значения, sum() – виртуальный метод, для подсчёта суммы элементов, конструктор инициализации (по умолчанию).

class Vector – производный класс. Поля: vect – вектор типа double. С переопределёнными методами: set() – для вставки значения в вектор, sum() – для подсчёта суммы элемента, конструктор инициализации (по умолчанию).

Class List – производный класс. Поля: spisok – список типа float. С переопределёнными методами: set() – для вставки значения в список, sum() – для подсчёта суммы элемента, конструктор инициализации (по умолчанию)

**Функция main() Рис.1.**

начало

Array array;

Array.set();

Array.get();

Array.sum()

Vector vector;

vector.set();

vector.get();

vector.sum()

List spisok;

spisok.set();

spisok.get();

spisok.sum()

конец

Рис.1.

**Класс Array Рис.2.**

начало

unsigned array[];

int col;

virtual void sum()

virtual void set() virtual void get()

конец

Рис.2.

**Метод set() для Array Рис.3.**

начало

int i=0; i<SIZE; i++

сin>>arr[i];

конец

Рис.3.

**Метод get() для Array Рис.4.**

начало

int i=0; i<SIZE; i++

cout<<arr[i];

конец

Рис.4.

**Метод sum для Array Рис.5.**

начало

int sum;

int i=0;i<SIZE;i++

sum+=arr[i];

конец

Рис.5.

**Класс Vector Рис.6.**

начало

Vector<double>vect

void sum()

void set() void get()

конец

Рис.6.

**Метод set() для Vector Рис.7.**

начало

double a;

int i;

i<SIZE;

cin>>a;

vect.push\_back(a); i++;

конец

Рис.7.

**Метод get() для Vector Рис.8.**

начало

int i=0; i<SIZE; i++

cout<<vect[i];

Рис.8.

конец

**Метод sum для Vector Рис.9.**

начало

double sum;

int i=0;i<SIZE;i++

sum+=vect[i];

конец

Рис.9.

**Класс List Рис.10.**

начало

Vector<float>spisok

void sum()

void set() void get()

конец

Рис.10.

**Метод set() для List Рис.11.**

начало

float a;

int i;

i<SIZE;

cin>>a;

spisok.push\_back(a); i++;

конец

Рис.11.

**Метод get() для List Рис.12.**

начало

auto i = spisok.begin(); i != lst.end(); i++

cout<< \*i;

Рис.12.

конец

**Метод sum для List Рис.13.**

начало

float sum;

auto i = spisok.begin(); i != spisok.end(); i++

sum+= \*i;

конец

Рис.13.

**Код:**

#include<iostream>

#include<vector>

#include<list>

using namespace std;

const int SIZE = 5; //константный размер массива 5

class Array { // базовый класс Array

unsigned array[SIZE]; // массив

int kol;

public:

Array() { //конструктор по умолчанию

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

array[i] = 0;

}

kol = 0;

}

virtual void set() { // виртуальный метод set, используется для инициализации

cout << "Введите значение для массива: ";

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

cin >> array[i];

}

}

virtual void get() { // виртуальный метод get, используется для вывода данных на консоль

cout << "Массив: ";

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

cout << array[i] << " ";

}

}

virtual void sum() { // виртуальный метод sum, используется для вычисления суммы

int sum = 0;

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

sum += array[i];

}

cout << "\nСумма массива: " << sum << endl;

}

};

class Vector :protected Array { //производный класс Vector, механизм наследования protected

vector<double>vect; //объявление вектора

public:

void set() { //переопределение метода set

cout << "Введите значение для вектора: ";

double a;

int i = 0;

while (i < SIZE) {

cin >> a;

vect.push\_back(a);

i++;

}

}

void get() { //переопределение метода get

cout << "Вектор: ";

for (int i = 0; i < SIZE; i++) cout << vect[i] << " ";

}

void sum() { //переопределение метода sum

double sum = 0;

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

sum += vect[i];

}

cout << "\nСумма вектора: " << sum << endl;

}

};

class List :protected Array { //производный класс List, механизм наследования protected

list<float>spisok; //объявление списка

public:

List() { //конструктор по умолчанию

for (auto i = spisok.begin(); i != spisok.end(); i++) {

\*i = 0;

}

}

void set() { //переопределение метода set

cout << "Введите значение для списка: ";

float del;

int i = 0;

while (i < SIZE) {

cin >> del;

spisok.push\_back(del);

i++;

}

}

void get() { //переопределение метода get()

cout << "Список: ";

for (auto i = spisok.begin(); i != spisok.end(); i++) {

cout << \*i << " ";

}

}

void sum() { //переопределение метода sum()

float sum = 0;

for (auto i = spisok.begin(); i != spisok.end(); i++) {

sum += \*i;

}

cout << "\nСумма списка: " << sum << endl;

}

};

int main() {

setlocale(LC\_ALL, "rus");

//Массив

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_МАССИВ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

Array array;

array.set();

cout << "Вывод массива:\n";

array.get();

array.sum();

cout << endl << endl;

//Вектор

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ВЕКТОР\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

Vector vect;

vect.set();

cout << "Вывод вектора:\n";

vect.get();

vect.sum();

cout << endl << endl;

//Список

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_СПИСОК\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

List spisok;

spisok.set();

cout << "Вывод списка:\n";

spisok.get();

spisok.sum();

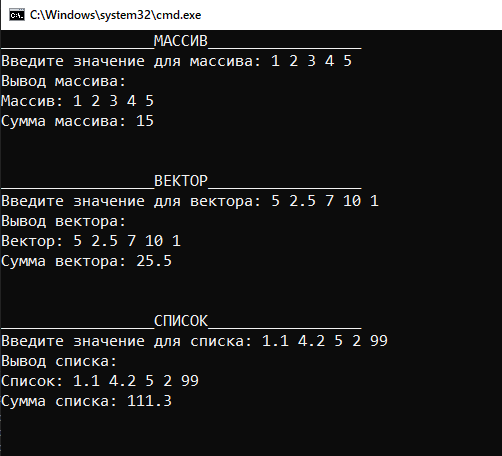
cout << endl << endl;

system("pause>null");

return 0;

}

**Результат:**

****

**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*

**Описание входных данных:**

Class Figure-Базовый класс, с переменными length, width, методы: area() - чисто виртуальный метод, для вычисления площади, perimetr() - чисто виртуальный метод, для вычисления периметра.

Class Rectangle - Производный класс. С переопределенными методами: area()-для вычисления площади, perimetr()-для вычисления периметра, set()-для вставки значения, get()-для получения значения.

Class Circle - Производный класс. Поля: radius – переменная типа double. С переопределенными методами: area()-для вычисления площади, perimetr()-для вычисления периметра, set()-для вставки значения, get()-для получения значения.

Class Trapezium-Производный класс от Figure. Поля: a, b, c, d – переменные типа double. С переопределенными методами: area()-для вычисления площади, perimetr()-для вычисления периметра, set()-для вставки значения, get()-для получения значения.

Rectangle rect - объект класса Rectangle. Переменные объекта: double length\_rect, width\_rect;

Circle circle - объект класса Circle. Переменные объекта: double length\_circle, width\_circle;

Trapezium trap - объект класса Trapezium. Переменные объекта: double length\_trap, width\_trap;

**Класс Figure Рис.1.**

начало

length,width

area()=0,perimetr()=0

конец

Рис.1.

**Класс Rectangle Рис.2.**

начало

Area(),perimetr(),set()

конец

Рис.1.

**Класс Circle Рис.3.**

начало

Radius,Area(),perimetr(),set()

конец

Рис.3.

**Класс Trapezium Рис.4.**

начало

a,b,c,d,Area(),perimetr(),set()

конец

Рис.4.

**Метод set() для Rectangle Рис.5.**

начало

This->length=length;

This->width=width;

конец

Рис.5.

**Метод set() для Circle Рис.6.**

начало

This->length=length;

This->radius=radius;

конец

Рис.6.

**Метод set() для Trapezium Рис.7.**

начало

This->a=a;This->b=b;

This->c=c;This->d=d;

This->length=length;

конец

Рис.7.

print()

2

break

Add()

choose

**Метод area() для Rectangle Рис.8.**

начало

area=length\*width

area

Return area

конец

Рис.8.

**Метод area() для Circle Рис.9.**

начало

area=pi\*width2

area

Return area

конец

Рис.9.

**Метод area() для Trapezium Рис.10.**

начало

area=(a+b)/2\*length

area

Return area

конец

Рис.9.

**Метод perimeter() для Rectangle Рис.10.**

начало

perimetr=(length+width)\*2

perimetr

Return perimetr

конец

Рис.10.

**Метод perimeter() для Circle Рис.11.**

начало

perimetr=2\*pi\*radius

perimetr

Return perimetr

конец

Рис.11.

**Метод perimeter() для Trapezium Рис.12.**

начало

Perimetr=a+b+c+d;

perimetr

Return perimetr

конец

Рис.12.

**Код:**

#include<iostream>

using namespace std;

const double pi = 3.1415;

class Figure { //Базовый абстрактный класс

protected:

double length;

double width;

virtual double perimetr() = 0; //чисто виртуальный метод

virtual double area() = 0; //чисто виртуальный метод

};

class Rectangle :private Figure {

public:

double perimetr() {//переопределение метода

double perim = (length + width) \* 2;

cout << "Периметр прямоугольника: " << perim << endl;

return perim;

}

double area() {//переопределение метода

double area = length \* width;

cout << "Площадь прямоугольника: " << area << endl;

return area;

}

void set(double length, double width) {

this->width = width;

this->length = length;

}

void get() {

cout << "Длина: " << length << endl;

cout << "Ширина: " << width << endl;

}

};

class Circle :private Figure {

double radius;

public:

void set(double radius, double length) {

this->radius = radius;

this->length = length;

}

double perimetr() {//переопределение метода

length = 2 \* pi \* radius;

cout << "Периметр круга: " << length << endl;

return length;

}

double area() {//переопределение метода

double area = pi \* pow(radius, 2);

cout << "Площадь круга: " << area << endl;

return area;

}

};

class Trapezium :private Figure {

double a;

double b;

double c;

double d;

public:

void set(double a, double b, double c, double d, double length) {

this->a = a;

this->b = b;

this->c = c;

this->d = d;

this->length = length;

}

double area() {//переопределение метода

double area;

area = (a + b) / 2 \* length;

cout << "Площадь трапеции: " << area << endl;

return area;

}

double perimetr() {//переопределение метода

double perm = a + b + c + d;

cout << "Периметр прямоугольника: " << perm << endl;

return perm;

}

};

int main() {

setlocale(LC\_ALL, "rus");

int choose;

while (true) {

cout << "Выберите фигуру:\n1 - Прямоугольник\n2 - Круг\n3 - Трапеция\n0 - Выход\n";

cout << "Выбор пользователя: ";

cin >> choose;

if (choose == 1) {

Rectangle rect;

double length\_rect, width\_rect;

cout << "Введите длину прямоугольника: ";

cin >> length\_rect;

cout << "Введите ширину прямоугольника: ";

cin >> width\_rect;

if (length\_rect <= 0 || width\_rect <= 0 || cin.fail()) {

cout << "\nНекорректный ввод\n";

cin.clear();

}

else {

rect.set(length\_rect, width\_rect);

rect.perimetr();

rect.area();

}

}

else if (choose == 2) {

Circle circle;

double radius\_circle, length\_circle;

cout << "Введите радиус круга: ";

cin >> radius\_circle;

cout << "Введите длину круга: ";

cin >> length\_circle;

if (radius\_circle <= 0 || length\_circle <= 0) {

cout << "Некорректный ввод\n";

cin.clear();

}

else {

circle.set(radius\_circle, length\_circle);

circle.perimetr();

circle.area();

}

}

else if (choose == 3) {

Trapezium trap;

double a, b, c, d, length;

cout << "Введите сторону а: ";

cin >> a;

cout << "Введите сторону b: ";

cin >> b;

cout << "Введите сторону c: ";

cin >> c;

cout << "Введите сторону d: ";

cin >> d;

cout << "Введите длину трапеции:";

cin >> length;

if (a <= 0 || b <= 0 || c <= 0 || d <= 0 || length <= 0) {

cout << "Некорректный ввод\n";

cin.clear();

}

else {

trap.set(a, b, c, d, length);

trap.perimetr();

trap.area();

}

}

else if (choose == 0) {

exit(0);

}

else cout << "Ошибка\n";

system("pause>null");

system("cls");

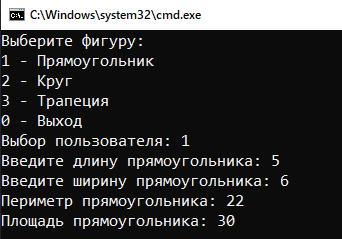
}

return 0;

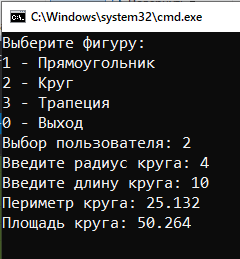
}

**Результат:**

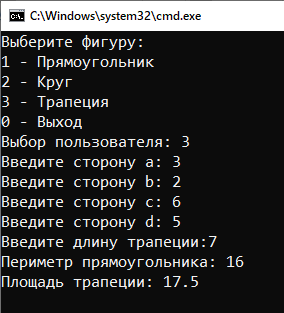
1. **Для прямоугольника**

****

1. **круга**



1. **Для трапеции**

****

**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

**Описание входных данных:**

Class Currency - Базовый класс, с переменной value, методы: perevod () - чисто виртуальный метод, для перевода валюты, print() - чисто виртуальный метод, для вывода валюты.

Class Dollar - Производный класс. С переопределенными методами: perevod()-для перевода валюты, print()-для вывода валюты.

Class Euro - Производный класс. С переопределенными методами: perevod()-для перевода валюты, print() - для вывода валюты.

Dollar dollar - объект класса Dollar. Переменные объекта:value;

Euro euro-объект класса Euro.Переменные объекта:value;

**Класс Currency Рис.1.**

начало

Value;virtual perevod()=0;virtual print()=0

конец

Рис.1.

**Класс Dollar Рис.2.**

начало

money;perevod();print();set();get()

конец

Рис.2.

**Класс Euro Рис.3.**

начало

money;perevod();print();set();get()

конец

Рис.3.

**Метод perevod Рис.4.**

начало

Value=dollar\*79.90

Value=euro\*87.65

конец

Рис.4.

**Метод print Рис.5.**

начало

cout << value<<end

конец

Рис.5.

**Код:**

#include<iostream>

using namespace std;

class Currency {//Абстрактный класс

protected:

double value;

virtual void perevod() = 0;//чисто виртуальный метод

virtual void print() = 0;//чисто виртуальный метод

};

class Dollar :protected Currency {//ПРоизводный от класса Currency

double dollar;

public:

void perevod() {//переопределение метода

value = dollar \* 79.90;

}

void set(double value) {

this->dollar = value;

}

double get() {

return this->dollar;

}

void print() {//переопределение метода

cout << value << " рублей" << endl;

}

};

class Euro :protected Currency {//Производный класс от Currency

double euro;

public:

void perevod() {//переопределение метода

value = euro \* 87.65;

}

void set(double value) {

this->euro = value;

}

double get() {

return this->euro;

}

void print() {//переопределение метода

cout << value << " рублей" << endl;

}

};

int main() {

setlocale(LC\_ALL, "rus");

Dollar dollar;

Euro euro;

while (true) {

int choose;

cout << "Выберите валюту для перевода:\n";

cout << "1 - Доллар\n2 - Евро\n0 - Выход\n";

cout << "Ответ пользователя: ";

cin >> choose;

if (choose == 0) {

exit(0);

}

else if (choose > 2) {

cout << "Ошибка\n";

break;

}

double value;

cout << "Введите валюту: ";

cin >> value;

if (value <= 0 || cin.fail()) {

cout << "Ошибка, некорректный ввод данных\n";

exit(0);

}

else {

if (choose == 1) {

dollar.set(value);

dollar.get();

dollar.perevod();

dollar.print();

}

else if (choose == 2) {

euro.set(value);

euro.get();

euro.perevod();

euro.print();

}

}

system("pause>null");

system("cls");

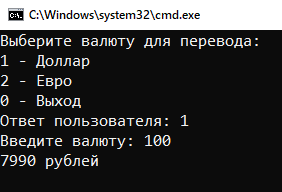
}

return 0;

}

**Результат:**

1. **Для доллара**

****

1. **Для евро**

