**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное автономное образовательное учреждение высшего образования «Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

**Институт среднего профессионального образования**

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 14

по дисциплине: «Основы алгоритмизации и программирования»

**Студент группы № 219/2**

**Клопова Ульяна**

**Вариант 15**

**Преподаватель: Рябова В.В.**

Санкт-Петербург

2025 г.

[**Задача 1.**](../программы/мрак/мрак.sln)

Создайте класс Circle, который будет содержать:

* Переменную radius (радиус).
* Конструктор для инициализации радиуса.
* Метод для вычисления площади круга.
* Метод для вычисления периметра круга.

Напишите программу, которая создает объект класса Circle и выводит его площадь и периметр.

Предусмотрите проверки на введение корректных данных

#include <iostream>

#include <cmath>

using namespace std;

class circle {

private:

double radius;

const double pi = 3.14;

public:

circle(double radius) {

if (radius >= 0) {

this->radius = radius;

}

else {

cout << "Ошибка: Радиус должен быть положительным числом. Радиус по умолчанию = 1.0" << endl;

this->radius = 1.0;

}

}

double kryg() const {

return pi \* pow(radius, 2);

}

double plosh() const{

return 2 \* pi \* radius;

}

void print() const {

cout << "Радиус: " << radius << endl;

cout << "Площадь: " << kryg() << endl;

cout << "Периметр: " << plosh() << endl;

}

};

int main()

{

system("chcp 1251");

double radius;

cout << "Введите радиус круга: ";

cin >> radius;

circle cir(radius);

cir.print();

}

**Результаты запусков:**

**Изображение выглядит как текст, Шрифт, снимок экрана, число

Содержимое, созданное искусственным интеллектом, может быть неверным.**

**Первый запуск**

**Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.**

**Второй запуск**

**Изображение выглядит как текст, Шрифт, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.**

**Третий запуск**

[**Задача 2. Наследование. Указания: ознакомиться с тем, как наследуются классы (слайды 34-37).**](../программы/exhale%20inhale/exhale%20inhale.sln)

Определите базовый класс Animal, который представляет животное и который содержит две приватные переменные: строку для хранения имени животного и целое число для хранения веса животного. Также определите общедоступную функцию print, которая выводит на консоль сообщение с указанием имени и веса объекта Animal.

Также создайте два производных класса Cat (кошка) и Dog (собака), которые наследуются от класса Animal. В функции main создайте несколько объектов типа Cat и Dog и с помощью функции print выведите информацию об этих объектах на консоль.

#include <iostream>

#include <string>

using namespace std;

class animal {

private:

string name;

int weight;

public:

animal(string name, int weight) : name(name), weight(weight) {}

void print() const {

cout << "Имя: " << name << ", Вес: " << weight << " кг" << endl;

}

string getName();

int getWeight();

};

class cat : public animal {

public:

cat(string name, int weight) : animal(name, weight) {}

};

class dog : public animal {

public:

dog(string name, int weight) : animal(name, weight) {}

};

int main() {

system("chcp 1251");

cat cat1("Муська", 3);

cat cat2("Барсик", 5);

cat cat3("Клеопатра", 10);

cat cat4("Фрай", 3);

cat cat5("Бендер", 2);

dog dog1("Рекс", 15);

dog dog2("Белка", 10);

dog dog3("Стрелка", 14);

dog dog4("Асти", 7);

dog dog5("Инокентий", 30);

cout << "Коты:" << endl;

cat1.print();

cat2.print();

cat3.print();

cat4.print();

cat5.print();

cout << "\nСобаки:" << endl;

dog1.print();

dog2.print();

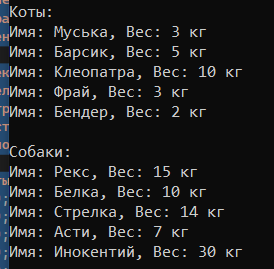
dog3.print();

dog4.print();

dog5.print();

}

**Результат запуска:**

****

**Первый запуск**

[**Задача 3.**](../программы/bad%20apple/bad%20apple.sln)

Сделать класс игрока. Внутри класса должно быть:

-Имя игрока

-Здоровье игрока

-Броня игрока

После этого вывести параметры игрока на экран.

Добавить внутри класса публичные функции:

* Функцию урона (урон не может быть отрицательным), которая уменьшает здоровье по формуле: *Урон - броня*
* Функция, которая будет выдавать, жив ли игрок или нет
* Функция, которая будет выдавать имя игрока.

По наследованию (задача под \*):

Создайте базовый класс Item с приватными полями:

* Имя вещи
* Прочность вещи

В классе Item реализуйте:

* Конструктор для инициализации имени и прочности.
* Метод printInfo(), выводящий информацию о предмете.

Создайте класс Weapon, унаследованный от Item, с дополнительными полями:

* Урон оружия
* Шанс критического урона (в процентах или в диапазоне 0-100)

В классе Weapon реализуйте:

* Конструктор для инициализации всех полей.
* Метод attack(), который возвращает урон: С вероятностью равной максимальному критическому шансу наносит двойной урон, иначе — обычный урон.

#include <iostream>

#include <string>

#include <cstdlib>

#include <ctime>

using namespace std;

class player {

private:

string name;

int health;

int armor;

public:

player(string name, int health, int armor) : name(name), health(health), armor(armor) {}

void print() const {

cout << "Имя: " << name << endl;

cout << "Здоровье: " << health << endl;

cout << "Броня: " << armor << endl;

}

void damage(int damage) {

int adamage = damage - armor;

if (adamage > 0) {

health -= adamage;

if (health < 0) {

health = 0;

}

}

}

bool isalive() const {

return health > 0;

}

string getName() const {

return name;

}

};

class item {

private:

string name;

int durability;

public:

item(string name, int durability) : name(name), durability(durability) {}

void printi() const {

cout << "Имя: " << name << endl;

cout << "Прочность: " << durability << endl;

}

string getName();

int getDurability();

void set(int durability) {

this->durability = durability;

}

};

class weapon : public item {

private:

int damage;

int criticalChance;

public:

weapon(string name, int durability, int damage, int criticalChance) : item(name, durability), damage(damage), criticalChance(criticalChance) {

if (criticalChance < 0 || criticalChance > 100) {

this->criticalChance = 0;

}

}

int attack() const {

int num = rand() % 100;

if (num < criticalChance) {

cout << "Крит!" << endl;

return 2 \* damage;

}

else {

return damage;

}

}

void printInfo() const {

item::printi();

cout << "Урон: " << damage << endl;

cout << "Шанс критического урона: " << criticalChance << "%" << endl;

}

};

int main()

{

system("chcp 1251");

player player1("Геральт", 100, 20);

player1.print();

player1.damage(30);

cout << "Жив ли игрок? " << (player1.isalive() ? "Да" : "Нет") << endl;

cout << "Имя игрока: " << player1.getName() << endl;

cout << endl;

srand(time(0));

weapon sword("Меч Истины", 100, 20, 10);

sword.printInfo();

int damage = sword.attack();

cout << "Нанесенный урон: " << damage << endl;

}

**Результат запуска:**

**Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.**

**Первый запуск**

**Дополнительное задание:**

Создайте объект игрока и несколько предметов (оружие и обычные вещи). Выведите параметры игрока, примените к нему урон, а также продемонстрируйте использование оружия для атаки

#include <iostream>

#include <string>

#include <cstdlib>

#include <ctime>

#include <vector>

using namespace std;

class player {

private:

string name;

int health;

int armor;

public:

player(string name, int health, int armor) : name(name), health(health), armor(armor) {}

void print() const {

cout << "Имя: " << name << endl;

cout << "Здоровье: " << health << endl;

cout << "Броня: " << armor << endl;

}

void damage(int damagea) {

int adamage = damagea - armor;

if (adamage > 0) {

health -= adamage;

if (health < 0) {

health = 0;

}

}

}

bool isalive() const {

return health > 0;

}

string getName() const {

return name;

}

int getHealth()

{

return health;

}

void setHealth(int newHealth) {

health = newHealth;

}

};

class item {

private:

string name;

int durability;

public:

item(string name, int durability) : name(name), durability(durability) {}

void print() const {

cout << "Имя: " << name << endl;

cout << "Прочность: " << durability << endl;

}

string getName() const {

return name;

}

int getDurability() const {

return durability;

}

void setDurability(int newDurability) {

durability = newDurability;

}

};

class weapon : public item {

private:

int damage;

int criticalChance;

public:

weapon(string name, int durability, int damage, int criticalChance) : item(name, durability), damage(damage), criticalChance(criticalChance) {

if (criticalChance < 0 || criticalChance > 100) {

this->criticalChance = 0;

}

}

int attack() const {

int num = rand() % 100;

if (num < criticalChance) {

cout << "Крит!" << endl;

return 2 \* damage;

}

else {

return damage;

}

}

void print() const {

item::print();

cout << "Урон: " << damage << endl;

cout << "Шанс критического урона: " << criticalChance << "%" << endl;

}

};

int main()

{

system("chcp 1251");

player player1("Геральт", 100, 20);

cout << "Информация об игроке:" << endl;

player1.print();

weapon sword("Меч Истины", 100, 50, 10);

item potion("Зелье здоровья", 5);

item shield("Щит", 80);

cout << "\nИнформация о предметах:" << endl;

sword.print();

potion.print();

shield.print();

cout << "\nАтакуем мечом:" << endl;

int damage = sword.attack();

cout << "Нанесенный урон: " << damage << endl;

cout << "\nНаносим урон игроку:" << endl;

player1.damage(damage);

cout << "Здоровье игрока после атаки: " << player1.getHealth() << endl;

cout << "\nИспользуем зелье:" << endl;

int currentHealth = player1.getHealth();

player1.setHealth(currentHealth + 30);

cout << "Здоровье игрока после использования зелья: " << player1.getHealth() << endl;

cout << "\nИтог:" << endl;

cout << "Жив ли игрок? " << (player1.isalive() ? "Да" : "Нет") << endl;

cout << "Имя игрока: " << player1.getName() << endl;

}

**Результаты запуска:**

**Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.**

**Первый запуск**