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**

**Отчёт по лабораторной работе № 2**

**по учебной дисциплине «Реализация классов»**

**Тема: «Структура консольного приложения в C#»**

Выполнил(а) студент(ка)

специальности 09.02.03

Программирование в компьютерных

системах

IV курса группы 419/7

Косовский Семен

Андреевич

Преподаватель

Молькова Лолита Юрьевна

Санкт-Петербург,

2024

Цель работы:

1. структуры класса, механизм создания и использования, описание членовданных класса и методов доступа к ним.

Задание:

1. Создайте консольный проект. В теле функции main будет выполняться

демонстрация работы объекта.

2. Добавьте в проект новый класс и назовите этот класс Worker. В класс добавьте два

общедоступных поля: имя и возраст и одно скрытое поле: вес:

class Worker {

public:

int age;

char\* name;

private:

float weight;

};

3. В теле функции main создайте объект класса Worker:

Worker \*wrk1 = new Worker ();

wrk1->age = 34;

wrk1->name = “Иванов”;

Добавьте оператор(функцию) вывода на экран созданного объекта.

4. Запустите программу на выполнение.

5. Попробуйте записать значение в поле weight. Почему данные не записались?

6. Для записи и чтения данных из скрытых полей используют методы. Добавим во

внутрь класса Worker новый метод (действие) который будет отвечать за еду, если

человек чего-то там съест, то его вес должен будет увеличиться на количество

съеденного.

в структуру класса:

public:

…

void eat (float how\_much);

после описания класса, но до функции main:

void Worker::eat (float how\_much){

weight = weight + how\_much;

}

7. Если поле вес скрытое, то мы в него не только писать не можем, но и читать тоже

не можем. Для чтения данных из скрытого поля необходимо использовать еще

один метод:

в структуру класса:

public:

…

float get\_weight();

после описания класса, но до функции main:

float Worker::get\_weight(){

return weight;

}

8. Почему в последних двух функциях после слова public идут различные слова? Что

они обозначают и на что влияют?

9. Теперь эти два метода надо использовать в нашей программе. Заставьте рабочего

съесть 2, а затем 3 кг пищи. Проверьте его вес.

wrk1->eat(2);

wrk1->eat(3);

float ves;

ves = wrk1->get\_weight();

Отобразите результат на экран.

10. Запустите программу на выполнение. Проверьте работоспособность. Добавьте

комментарии.

11. Усовершенствуйте метод eat таким образом, что если рабочий за раз съедает более

чем 10 кг, то его возраст увеличивается на год, а вес увеличивается только на

половину от съеденного.

12. Попросите рабочего съесть 15 кг и посмотрите на результат работы программы.

13. Измените программу так, что бы имя рабочего и его первоначальный возраст

вводились с клавиатуры и вносились в соответствующие переменные.

14. Запустите программу. Проверьте ее работоспособность.

15. Добавьте рабочему еще одно скрытое поле, которое будет отвечать за настроение и

будет иметь первоначальное значение равное 10.

16. Добавьте три метода: гулять (метод должен увеличивать настроение на 1),

танцевать (метод должен увеличивать настроение на 2) и работать (метод должен

уменьшать настроение на 2).

17. Дополните основную программу так, что бы рабочий после еды два раза погулял и

три раза потанцевал.

18. Добавьте в класс функцию, которая будет возвращать текущее настроение

пользователя.

19. Добавьте в основную программу метод работать 9 раз (можно в цикле) и выведите

настроение пользователя на экран.

20. Настроение получилось отрицательным? – ужасно. Измените метод работать таким

образом, что бы настроение никогда не было меньше нуля (т.е. если настроение

было 1 и человек поработал, то оно должно стать не меньше 0).

21. Проверьте заново работоспособность программы.

**Индивидуальное задание:**

Вариант 8.  
 Реализовать пользовательский класс в соответствии с вариантом задания.

При реализации классов поля должны быть скрытыми.

Определить метод установки свойств (при недопустимых аргументах функции

возвращать «false» и выдавать текст ошибки на экран).

Определить метод чтения свойств.

Написать демонстрационную программу, в которой показать использование

объектов созданного класса.

**Ход работы:**

На рисунке 1 изображена вся работы программы

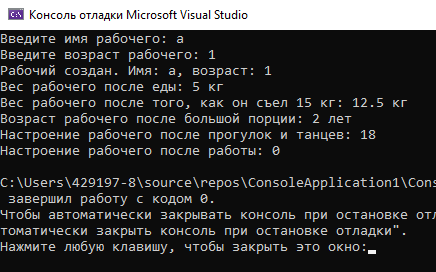


Рисунок 1 – Вся работы программы

Код:

#include <iostream>

#include <string> // Для работы с std::string

using namespace std;

// Определение класса Worker

class Worker {

public:

int age;

string name; // Изменение на std::string для корректной работы со строками

// Конструктор

Worker() {

age = 0;

weight = 0.0;

mood = 10; // Настроение по умолчанию

}

void setName(const string& workerName) {

name = workerName;

}

// Метод для увеличения веса после еды

void eat(float how\_much);

// Метод для получения текущего веса

float get\_weight();

// Методы для изменения настроения

void walk();

void dance();

void work();

// Метод для получения текущего настроения

int get\_mood();

private:

float weight;

int mood; // Скрытое поле настроение

};

// Реализация метода eat

void Worker::eat(float how\_much) {

if (how\_much > 10) {

age++; // Увеличиваем возраст, если за раз съедает более 10 кг

weight += how\_much / 2; // Увеличиваем вес на половину

}

else {

weight += how\_much;

}

}

// Реализация метода get\_weight

float Worker::get\_weight() {

return weight;

}

// Реализация метода walk

void Worker::walk() {

mood += 1; // Гулять увеличивает настроение на 1

}

// Реализация метода dance

void Worker::dance() {

mood += 2; // Танцевать увеличивает настроение на 2

}

// Реализация метода work

void Worker::work() {

if (mood > 0) {

mood -= 2; // Работать уменьшает настроение на 2

if (mood < 0) mood = 0; // Настроение не может быть меньше 0

}

}

// Реализация метода get\_mood

int Worker::get\_mood() {

return mood;

}

// Главная функция

int main() {

setlocale(LC\_ALL, "Russian");

// Создание объекта класса Worker

Worker\* wrk1 = new Worker();

// Ввод имени и возраста с клавиатуры

cout << "Введите имя рабочего: ";

string nw;

cin >> nw; // Вводим имя с использованием std::string

wrk1->setName(nw);

cout << "Введите возраст рабочего: ";

cin >> wrk1->age;

// Демонстрация записи данных

cout << "Рабочий создан. Имя: " << wrk1->name << ", возраст: " << wrk1->age << endl;

// Рабочий съедает 2 и 3 кг еды

wrk1->eat(2);

wrk1->eat(3);

// Получение текущего веса

float ves = wrk1->get\_weight();

cout << "Вес рабочего после еды: " << ves << " кг" << endl;

// Рабочий съедает 15 кг еды

wrk1->eat(15);

ves = wrk1->get\_weight();

cout << "Вес рабочего после того, как он съел 15 кг: " << ves << " кг" << endl;

cout << "Возраст рабочего после большой порции: " << wrk1->age << " лет" << endl;

// Рабочий дважды гуляет и трижды танцует

wrk1->walk();

wrk1->walk();

wrk1->dance();

wrk1->dance();

wrk1->dance();

// Вывод текущего настроения

int mood = wrk1->get\_mood();

cout << "Настроение рабочего после прогулок и танцев: " << mood << endl;

// Рабочий работает 9 раз

for (int i = 0; i < 9; i++) {

wrk1->work();

}

// Вывод настроения после работы

mood = wrk1->get\_mood();

cout << "Настроение рабочего после работы: " << mood << endl;

// Освобождение памяти

delete wrk1;

return 0;

}

UML диаграмма:

| Worker |  |
| --- | --- |
| Название метода | Описание |
| Worker() | Конструктор класса Worker, инициализирует возраст, вес и настроение. |
| ~Worker() | Деструктор класса Worker, освобождает память. |
| void setName(const string&) | Устанавливает имя рабочего. |
| string getName() | Возвращает имя рабочего. |
| void eat(float how\_much) | Увеличивает вес рабочего в зависимости от количества съеденной еды. |
| float get\_weight() | Возвращает текущий вес рабочего. |
| void walk() | Увеличивает настроение рабочего на 1 единицу. |
| void dance() | Увеличивает настроение рабочего на 2 единицы. |
| void work() | Уменьшает настроение рабочего на 2 единицы, но не позволяет ему опуститься ниже 0. |
| int get\_mood() | Возвращает текущее настроение рабочего. |

2.2 Индивидуальное задание (50%)

Реализовать пользовательский класс в соответствии с вариантом задания.

При реализации классов поля должны быть скрытыми.

Определить метод установки свойств (при недопустимых аргументах функции

возвращать «false» и выдавать текст ошибки на экран).

Определить метод чтения свойств.

Написать демонстрационную программу, в которой показать использование

объектов созданного класса.   
Вариант 8.   
**Ход работы:**

На рисунке 1 изображена валидация ввода количество треугольников

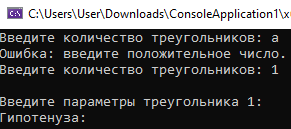


Рисунок 1 – Валидация ввода количество треугольников

На рисунке 2 изображена валидация ввода 2 свойств класса

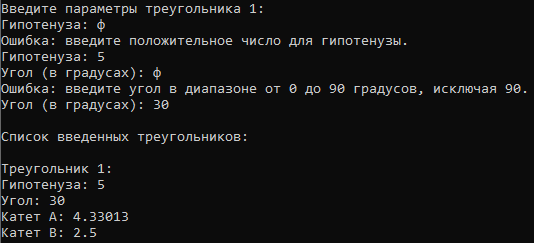


Рисунок 2 – валидация ввода 2 свойств класса

На рисунке 3 изображена валидация ввода выбора треугольника

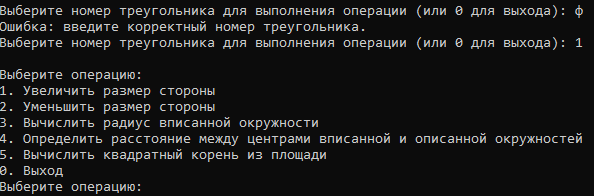


Рисунок 3 – валидация ввода выбора треугольника

На рисунке 4 изображена валидация ввода выбора метода

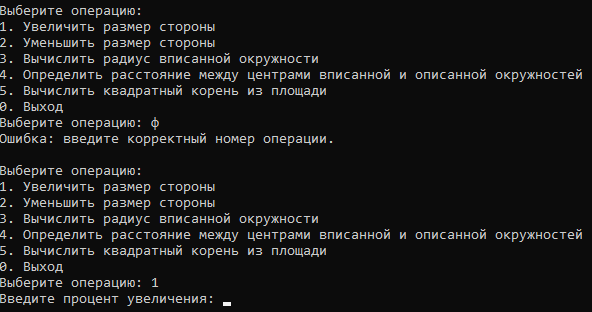


Рисунок 4 – валидация ввода выбора метода

На рисунке 5 изображена работа метода увеличения стороны

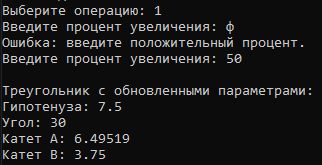


Рисунок 5 – работа метода увеличения стороны

На рисунке 6 изображена работа метода уменьшения стороны

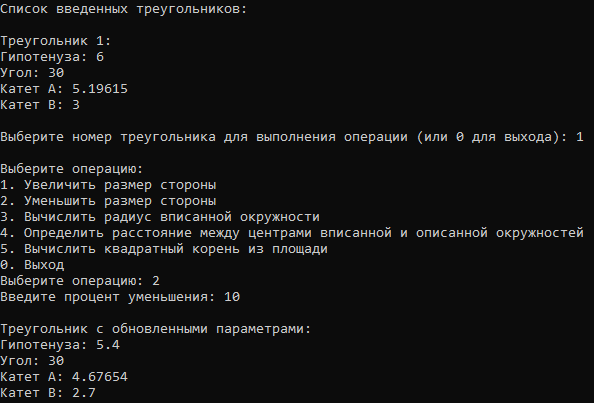


Рисунок 6 – работа метода уменьшения стороны

На рисунке 7 изображена работа метода расчета радиуса вписанной окружности

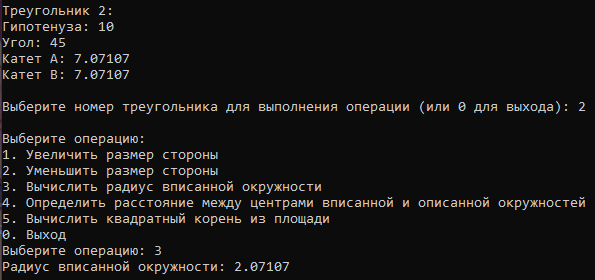


Рисунок 7 – работа метода расчета радиуса вписанной окружности

На рисунке 8 изображена работа метода расчета расстояния между центрами вписанной и описанной окружностей

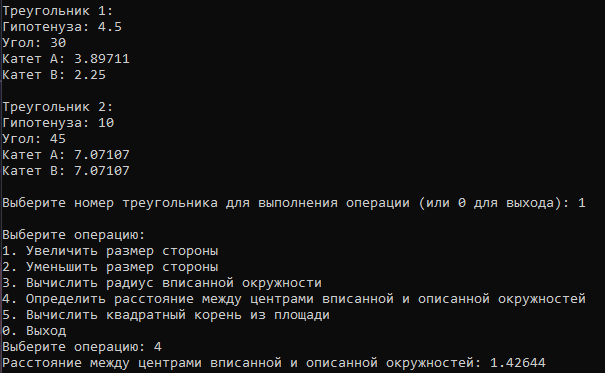


Рисунок 8 – работа метода расчета расстояния между центрами вписанной и описанной окружностей

На рисунке 9 изображена работа метода расчета квадратного корня из площади треугольника

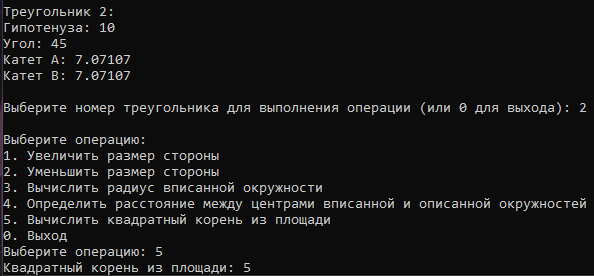


Рисунок 9 – работа метода расчета квадратного корня из площади треугольника

Код:

Заголовочный файл Triangle.h  
#ifndef TRIANGLE\_H

#define TRIANGLE\_H

#include <cmath>

#include <iostream>

using namespace std;

class Triangle {

public:

Triangle(double hypotenuse, double angle);

bool setSides(double hypotenuse, double angle);

void increaseSide(double percent);

void decreaseSide(double percent);

double calculateInradius() const;

double distanceBetweenInradiusAndCircumradius() const;

double calculateSquareRootOfArea() const;

void print() const;

private:

double hypotenuse;

double angle; // angle in degrees

double side\_a;

double side\_b;

void updateSides();

double toRadians(double degrees) const;

double calculateCircumradius() const;

double calculateArea() const;

};

#endif // TRIANGLE\_H

Файл реализации методов класса Triangle.cpp  
#include "Triangle.h"

#include <cmath> // Для функций cos, sin, sqrt и определения числа PI

#include <iostream>

using namespace std;

#ifndef M\_PI

#define M\_PI 3.14159265358979323846 // Определение PI, если оно не определено

#endif

Triangle::Triangle(double hypotenuse, double angle) : hypotenuse(hypotenuse), angle(angle) {

if (!setSides(hypotenuse, angle)) {

cerr << "Ошибка: неверные начальные параметры." << endl;

}

}

bool Triangle::setSides(double hypotenuse, double angle) {

if (hypotenuse <= 0 || angle <= 0 || angle >= 90) {

cout << "Ошибка: недопустимые значения гипотенузы или угла." << endl;

return false;

}

this->hypotenuse = hypotenuse;

this->angle = angle;

updateSides();

return true;

}

void Triangle::updateSides() {

double angleRad = toRadians(angle);

side\_a = hypotenuse \* cos(angleRad);

side\_b = hypotenuse \* sin(angleRad);

}

double Triangle::toRadians(double degrees) const {

return degrees \* M\_PI / 180.0;

}

double Triangle::calculateInradius() const {

return (side\_a + side\_b - hypotenuse) / 2.0;

}

double Triangle::calculateCircumradius() const {

return hypotenuse / 2.0;

}

double Triangle::distanceBetweenInradiusAndCircumradius() const {

return fabs(calculateCircumradius() - calculateInradius());

}

double Triangle::calculateArea() const {

return 0.5 \* side\_a \* side\_b;

}

double Triangle::calculateSquareRootOfArea() const {

return sqrt(calculateArea());

}

void Triangle::increaseSide(double percent) {

side\_a \*= (1 + percent / 100);

side\_b \*= (1 + percent / 100);

hypotenuse = sqrt(side\_a \* side\_a + side\_b \* side\_b);

}

void Triangle::decreaseSide(double percent) {

side\_a \*= (1 - percent / 100);

side\_b \*= (1 - percent / 100);

hypotenuse = sqrt(side\_a \* side\_a + side\_b \* side\_b);

}

void Triangle::print() const {

cout << "Гипотенуза: " << hypotenuse << endl;

cout << "Угол: " << angle << endl;

cout << "Катет A: " << side\_a << endl;

cout << "Катет B: " << side\_b << endl;

}  
3. Основной файл main.cpp  
#include <iostream>

#include <vector>

#include "Triangle.h"

using namespace std;

void printMenu() {

cout << "\nВыберите операцию:" << endl;

cout << "1. Увеличить размер стороны" << endl;

cout << "2. Уменьшить размер стороны" << endl;

cout << "3. Вычислить радиус вписанной окружности" << endl;

cout << "4. Определить расстояние между центрами вписанной и описанной окружностей" << endl;

cout << "5. Вычислить квадратный корень из площади" << endl;

cout << "0. Выход" << endl;

}

int main() {  
setlocale(LC\_ALL, "rus");

vector<Triangle> triangles;

int numTriangles;

cout << "Введите количество треугольников: ";

cin >> numTriangles;

while (cin.fail() || numTriangles <= 0) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Ошибка: введите положительное число." << endl;

cout << "Введите количество треугольников: ";

cin >> numTriangles;

}

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

double hypotenuse, angle;

cout << "\nВведите параметры треугольника " << (i + 1) << ":\n";

cout << "Гипотенуза: ";

cin >> hypotenuse;

while (cin.fail() || hypotenuse <= 0) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Ошибка: введите положительное число для гипотенузы." << endl;

cout << "Гипотенуза: ";

cin >> hypotenuse;

}

cout << "Угол (в градусах): ";

cin >> angle;

while (cin.fail() || angle <= 0 || angle >= 90) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Ошибка: введите угол в диапазоне от 0 до 90 градусов, исключая 90." << endl;

cout << "Угол (в градусах): ";

cin >> angle;

}

Triangle triangle(hypotenuse, angle);

triangles.push\_back(triangle);

}

int choice;

while (true) {

cout << "\nСписок введенных треугольников:" << endl;

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

cout << "\nТреугольник " << (i + 1) << ":" << endl;

triangles[i].print();

}

cout << "\nВыберите номер треугольника для выполнения операции (или 0 для выхода): ";

int index;

cin >> index;

while (cin.fail() || index < 0 || index > triangles.size()) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Ошибка: введите корректный номер треугольника." << endl;

cout << "Выберите номер треугольника для выполнения операции (или 0 для выхода): ";

cin >> index;

}

if (index == 0) break;

Triangle &selectedTriangle = triangles[index - 1];

printMenu();

cout << "Выберите операцию: ";

cin >> choice;

while (cin.fail() || choice < 0 || choice > 5) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Ошибка: введите корректный номер операции." << endl;

printMenu();

cout << "Выберите операцию: ";

cin >> choice;

}

switch (choice) {

case 1: {

double percent;

cout << "Введите процент увеличения: ";

cin >> percent;

while (cin.fail() || percent < 0) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Ошибка: введите положительный процент." << endl;

cout << "Введите процент увеличения: ";

cin >> percent;

}

selectedTriangle.increaseSide(percent);

break;

}

case 2: {

double percent;

cout << "Введите процент уменьшения: ";

cin >> percent;

while (cin.fail() || percent < 0) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Ошибка: введите положительный процент." << endl;

cout << "Введите процент уменьшения: ";

cin >> percent;

}

selectedTriangle.decreaseSide(percent);

break;

}

case 3:

cout << "Радиус вписанной окружности: " << selectedTriangle.calculateInradius() << endl;

break;

case 4:

cout << "Расстояние между центрами вписанной и описанной окружностей: "

<< selectedTriangle.distanceBetweenInradiusAndCircumradius() << endl;

break;

case 5:

cout << "Квадратный корень из площади: " << selectedTriangle.calculateSquareRootOfArea() << endl;

break;

default:

cout << "Неверный выбор!" << endl;

break;

}

cout << "\nТреугольник с обновленными параметрами:" << endl;

selectedTriangle.print();

}

return 0;

}

UML диаграмма:

| **Triangle** |  |
| --- | --- |
| **Название метода** | **Описание** |
| Triangle(double hypotenuse, double angle) | Конструктор класса Triangle. |
| bool setSides(double hypotenuse, double angle) | Устанавливает стороны треугольника. |
| void increaseSide(double percent) | Увеличивает размер стороны на заданный процент. |
| void decreaseSide(double percent) | Уменьшает размер стороны на заданный процент. |
| double calculateInradius() const | Вычисляет радиус вписанной окружности. |
| double distanceBetweenInradiusAndCircumradius() const | Определяет расстояние между центрами вписанной и описанной окружностей. |
| double calculateSquareRootOfArea() const | Вычисляет квадратный корень из площади треугольника. |
| void print() const | Выводит информацию о треугольнике. |
| void updateSides() | Обновляет значения сторон треугольника на основе гипотенузы и угла. |
| double toRadians(double degrees) const | Преобразует угол из градусов в радианы. |
| double calculateCircumradius() const | Вычисляет радиус описанной окружности. |
| double calculateArea() const | Вычисляет площадь треугольника. |