Лабораторная работа №20 «Классы и объекты. Инкапсуляция, конструкторы»

Задание: Поле first – дробное положительное число, катет a прямоугольного треугольника, поле second – дробное положительное число, катет b прямоугольного треугольника. Реализовать метод hipotenuse() – вычисление гипотенузы.

Цель работы: написать программу, которая использует только положительные числа для нахождении гипотенузы.

#include <iostream>

#include <cmath>

#include <ctime>

#include <string>

class Math {

private:

double first\_catheter;

double second\_catheter;

double hipotenuse\_result\_class;

public:

// 1. Конструкторы

Math()

{// Конструктор без параметров

first\_catheter = 0;

second\_catheter = 0;

}

Math(double &first\_catheter, double &second\_catheter)

{// Конструктор с параметрами

this->first\_catheter = first\_catheter;

this->second\_catheter = second\_catheter;

}

Math(const Math& b)

{// Конструктор копирования

first\_catheter = b.first\_catheter;

second\_catheter = b.second\_catheter;

}

// Деструктор

~Math()

{// Деструктор

std::cout << "\nDeleted\n";

}

// 3. Сеттеры

inline void Set\_param(double &first\_catheter, double &second\_catheter)

{// Cеттер для first\_catheter и second\_catheter

this->first\_catheter = first\_catheter;

this->second\_catheter = second\_catheter;

}

inline void Set\_hipotenuse\_result(double& hipotenuse\_result)

{// Cеттер для hipotenuse\_result

this->hipotenuse\_result\_class = hipotenuse\_result;

}

// 4. Геттеры

inline double Get\_param() const

{// Геттер для first\_catheter и second\_catheter

return first\_catheter, second\_catheter;

}

inline double hipotenuse()

{// Метод для нахождения гипотинузы

return sqrt(pow(first\_catheter, 2) + pow(second\_catheter, 2));

}

// 5. Перегрузка операторов =, <<

Math& operator=(const Math& other)

{// Перегрузка оператора присваивания

if (this != &other) { // Проверка на самоприсваивание

first\_catheter = other.first\_catheter;

second\_catheter = other.second\_catheter;

}

return \*this;

}

friend std::ostream& operator<<(std::ostream& out, const Math& other);

virtual void draw() const

{// Функция вывода данных на экран

std::cout << first\_catheter << " " << second\_catheter << std::endl;

}

};

std::ostream& operator<<(std::ostream& out, const Math& other)

{// Перегрухка оператор "<<"

out << other.hipotenuse\_result\_class << std::endl;

return out;

}

class Exam : public Math

{// Дочерний класс от Math

double mark;

std::string FIO;

std::string predmet;

public:

// 1. Конструкторы

Exam()

{// Конструктор без параметров

mark = 0;

FIO = "None";

predmet = "None";

}

Exam(double mark, std::string FIO, std::string predmet)

{// Конструктор c параметрами

this->mark = mark;

this->FIO = FIO;

this->predmet = predmet;

}

// 2. Деструктор

~Exam()

{// Деструктор

std::cout << "\nDeleted\n";

}

void draw() const override

{// Функция вывода данных на экран

std::cout << FIO << " " << predmet << " " << mark << std::endl;

}

};

void drawMath(const Math& obj)

{// Функия подстановки

obj.draw();

}

int main()

{

srand(time(NULL));

// 1. Создание и вывод первого объекта

Math math\_first; // Создание первого объекта вызов конструктора без параметров

std::cout << "Part 1" << std::endl << "Math first: ";

drawMath(math\_first);

// 2. Создание и вывод полей второго объекта типа Math

double first\_catheter = rand() % 10;

double second\_catheter = rand() % 10;

if (first\_catheter < 0 || second\_catheter < 0)

return -1;

std::cout << "First catheter: " << first\_catheter << " " << "Second catheter: " << second\_catheter << std::endl;

Math math\_second(first\_catheter, second\_catheter); //Создание и вызов конструктора с параметрами второго объекта

// 3. Демонстрирование перегрузки оператора =

math\_first = math\_second; // Пример перегрузки оператора присваивания

std::cout << "Math first: ";

drawMath(math\_first);

// 4. Поиск гипотенузы и демонстрация перегрузки оператора <<

auto hipotenuse\_result\_main = math\_second.hipotenuse();

math\_second.Set\_hipotenuse\_result(hipotenuse\_result\_main);

std::cout << "Hipotenuse: " << math\_second;

// 5. Создание 3-го обекта типа Exam и демонстрация подстановки

std::cout << std::endl << std::endl << "Part 2" << std::endl;

double mark = 10.0;

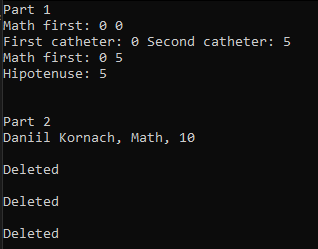
Exam exam(mark ,"Daniil Kornach", "Math");

drawMath(exam);

return 0;

}

Рисунок 1. Результат выполнения



Постановка задачи:

1.Определить пользовательский класс.

Определяю пользовательский класс Math при помощи зарезервированного слова class. В нем поля first и second имеют спецификатор доступа private, все остальные методы класса имеют спецификатор доступа public.

2.Определить в классе следующие конструкторы: без параметров, с параметрами, копирования.

Math()

{// Конструктор без параметров

first\_catheter = 0;

second\_catheter = 0;

}

Math(double &first\_catheter, double &second\_catheter)

{// Конструктор с параметрами

this->first\_catheter = first\_catheter;

this->second\_catheter = second\_catheter;

}

Math(const Math& b)

{// Конструктор копирования

first\_catheter = b.first\_catheter;

second\_catheter = b.second\_catheter;

}

3.Определить в классе деструктор.

~Math()

{// Деструктор

std::cout << "\nDeleted\n";

}

4. Определить в классе компоненты-функции для просмотра и установки полей данных (селекторы и модификаторы).

inline void Set\_param(double &first\_catheter, double &second\_catheter)

{// Cеттер для first\_catheter и second\_catheter

this->first\_catheter = first\_catheter;

this->second\_catheter = second\_catheter;

}

inline void Set\_hipotenuse\_result(double& hipotenuse\_result)

{// Cеттер для hipotenuse\_result

this->hipotenuse\_result\_class = hipotenuse\_result;

}

5. Перегрузить операцию присваивания.

Math& operator=(const Math& other)

{// Перегрузка оператора присваивания

if (this != &other) { // Проверка на самоприсваивание

first\_catheter = other.first\_catheter;

second\_catheter = other.second\_catheter;

}

return \*this;

}

6. Перегрузить операции ввода и вывода объектов с помощью потоков.

В программе отсутствует перегрузка операторов ввода так как в ней нет надобности.

friend std::ostream& operator<<(std::ostream& out, const Math& other);

std::ostream& operator<<(std::ostream& out, const Math& other)

{// Перегрухка оператор "<<"

out << other.hipotenuse\_result\_class << std::endl;

return out;

}

7.Определить производный класс.

class Exam : public Math

{// Дочерний класс от Math

double mark;

std::string FIO;

std::string predmet;

public:

// 1. Конструкторы

Exam()

{// Конструктор без параметров

mark = 0;

FIO = "None";

predmet = "None";

}

Exam(double mark, std::string FIO, std::string predmet)

{// Конструктор c параметрами

this->mark = mark;

this->FIO = FIO;

this->predmet = predmet;

}

//// 2. Деструктор

//~Exam()

//{// Деструктор

// std::cout << "\nDeleted\n";

//}

void draw() const override

{// Функция вывода данных на экран

std::cout << FIO << ", " << predmet << ", " << mark << std::endl;

}

};

8.Написать программу, в которой продемонстрировать создание объектов и работу всех перегруженных операций.

В главной функции main() создаем объект math\_first класса Math, а с помощью конструктора без параметров устанавливаем значения для созданного объекта. Затем создаем объект math\_second класса Math, а с помощью конструктора с параметрами устанавливаем значения для созданного объекта. Перегружаем “=” math\_first = math\_second. Затем находим гипотенузу auto hipotenuse\_result\_main = math\_second.hipotenuse();

9.Реализовать функции, получающие и возвращающие объект базового класса. Продемонстрировать принцип подстановки.

Для метода подстановки в базовом классе используется

virtual void draw() const

{// Функция вывода данных на экран

std::cout << first\_catheter << " " << second\_catheter << std::endl;

}

В дочернем классе используется

void draw() const override

{// Функция вывода данных на экран

std::cout << FIO << ", " << predmet << ", " << mark << std::endl;

}

Эти обе функции реализуются через метод drawMath

void drawMath(const Math& obj)

{// Функия подстановки

obj.draw();

}

Содержание отчета

1.Постановка задачи (общая и конкретного варианта)

Описано в «Постановки задачи»

2. Описание класса

Описано в «Постановки задачи»

3.Объяснение результатов работы программы.

Программа на вход принимает два значения катетов треугольника, далее идет проверка этих чисел на положительность. Если границы соблюдены, то программа переходит к вычислению гипотенузы треугольника, если нет – программа прекращает свою работу.

Вывод по работе:

В ходе выполнения работы были получены навыки по созданию классов, а также знания о спецификаторах доступа, полях и методов, селекторах и модификаторах. Основная проблема заключалась в проверки введенных данных на положительность, а именно в дальнейшей работе программы, если оказалось, что границы не соблюдены. Решение нашлось в функции return -1, которая полностью прекращает работу программы.