**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 1**

Тема: Программирование классов на языке С++

Студент: Ваньков Д.А.

Группа: 80-207Б

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2018

1. Постановка задачи.

Необходимо спроектировать и запрограммировать на языке C++ классы фигур (в моем случае классы “Восьмиугольник”, “Шестиугольник”, “Пятиугольник”).

Классы должны удовлетворять следующим правилам:

· Должны иметь общий родительский класс Figure.

· Должны иметь общий виртуальный метод Print, печатающий параметры фигуры и ее тип в стандартный поток вывода cout.

· Должны иметь общий виртуальный метод расчета площади фигуры – Square.

· Должны иметь конструктор, считывающий значения основных параметров фигуры из стандартного потока cin.

· Должны быть расположены в раздельных файлах: отдельно заголовки (.h), отдельно описание методов (.cpp).

Программа должна позволять вводить фигуру каждого типа с клавиатуры, выводить параметры фигур на экран и их площадь.

2. Решения задачи.

Все фигуры объединяет общий класс figure. Необходимо прописать в нем функции, прописанные заданием лабораторной, такие как подсчет площади и печать. Все эти функции раскрываются в файлах своих фигур. Важной частью программы является меню. Используя switch - case строим “диалог” с пользователем. Для каждой фигуры реализованы конструкторы, которые мы и вызываем далее. В конструкторах должна быть написана проверка введенных данных (стороны фигуры не могут быть отрицательными).

Для сборки всех файлов и создание исполняемого файла можно написать *Makefile*.

По итогу, выполнив все шаги, в директории должно оказаться 8 файлов: *Figure.h, Hexagon.cpp, Hexagon.h, Octagon.cpp, Octagon.h, Pentagon.cpp, Pentagon.h, main.cpp, Makefile*

3. Руководство по использованию программы.

Краткий обзор модулей программы:

Figure.h (модуль отвечающий за описание общего родительского класса фигур)

Pentagon.h (заголовочный файл, описывающий класс “Пятиугольник”)

Pentagon.cpp (файл, отвечающий за операции над пятиугольником, содержащий все необходимые функции и переменные, для этого класса)

Octagon.h (заголовочный файл, описывающий класс “Восьмиугольник”)

Octagon.cpp (файл, отвечающий за операции над восьмиугольником, содержащий все необходимые функции и переменные, для этого класса)

Hexagon.h (заголовочный файл, описывающий класс “Шестиугольник”)

Hexagon.cpp (файл, отвечающий за операции над шестиугольником, содержащий все необходимые функции и переменные, для этого класса)

main.cpp (модуль, содержащий интерфейс, позволяющий работать с написанным в других файлах кодом)

Makefile (собирает все файлы для последующей компиляции)

После запуска исполняемого файла, программа предлагает в помощь пользовательское меню работы с кодом, где каждая введенная цифра является командой начала процесса. В примере ниже сначала запускается меню, после чего выбирается цифра в соответствии с выбором работы с фигурой или повторного вызова меню.

Затем, после того как пользователь выбрал нужное ему действие запускается работа с фигурами. Для каждой фигуры пользователь вводит необходимые данные, чтобы ее задать, и затем программа, после выполнения преобразований и подсчета площади этой фигуры, выводит значение стороны и саму площадь на экран.

4. Листинг программы.

***main.cpp***

#include <iostream>

#include <cstdlib>

#include "Pentagon.h"

#include "Hexagon.h"

#include "Octagon.h"

void help(){

std::cout << "Press 0 to get help" << std::endl;

std::cout << "Press 1 to work with Pentagon" << std::endl;

std::cout << "Press 2 to work with Hexagon" << std::endl;

std::cout << "Press 3 to work with Octagon" << std::endl;

std::cout << "Press 4 to exit" << std::endl;

}

int main() {

help();

int k;

Figure \*ptr;

while(std::cin >> k) {

switch(k) {

case 0:

help();

break;

case 1:

ptr = new Pentagon(std::cin);

ptr->Print();

std::cout << "Square = " << ptr->Square() << std::endl;

delete ptr;

break;

case 2:

ptr = new Hexagon(std::cin);

ptr->Print();

std::cout << "Square = " << ptr->Square() << std::endl;

delete ptr;

break;

case 3:

ptr = new Octagon(std::cin);

ptr->Print();

std::cout << "Square = " << ptr->Square() << std::endl;

delete ptr;

break;

case 4:

exit(EXIT\_SUCCESS);

default:

std:: cerr << "Error" << std:: endl;

exit(EXIT\_FAILURE);

}

}

return 0;

}

***Figure.h***

#ifndef OOP\_LAB1\_FIGURE\_H

#define OOP\_LAB1\_FIGURE\_H

class Figure {

public:

virtual double Square() = 0;

virtual void Print() = 0;

virtual ~Figure(){};

};

#endif //OOP\_LAB1\_FIGURE\_H

***Hexagon.h***

#ifndef OOP\_LAB1\_HEXAGON\_H

#define OOP\_LAB1\_HEXAGON\_H

#include <iostream>

#include <cstdlib>

#include "Figure.h"

class Hexagon : public Figure{

public:

Hexagon();

Hexagon(std::istream &is);

Hexagon(size\_t i);

Hexagon(const Hexagon& orig);

double Square() override;

void Print() override;

virtual ~Hexagon();

private:

size\_t side\_a;

};

#endif //OOP\_LAB1\_HEXAGON\_H

***Hexagon.cpp***

#include "Hexagon.h"

#include <istream>

#include <cmath>

Hexagon::Hexagon() :Hexagon(0) {

}

Hexagon::Hexagon(size\_t i) : side\_a(i){

std::cout << "Hexagon created: " << side\_a << std::endl;

}

Hexagon::Hexagon(std::istream &is) {

int ptr;

std::cout << "Insert side a:" << std::endl;

is >> ptr;

while (ptr < 0) {

std:: cout << "Please, insert correct data" << std:: endl;

is >> ptr;

}

side\_a = ptr;

std::cout << "Hexagon created: " << side\_a << std::endl;

}

Hexagon::Hexagon(const Hexagon &orig) {

std::cout << "Hexagon copy created" << std::endl;

side\_a = orig.side\_a;

}

double Hexagon::Square() {

return 1.5 \* pow(side\_a,2) \* sqrt(3);

}

void Hexagon::Print() {

std::cout << "Side: a = " << side\_a << std::endl;

}

Hexagon::~Hexagon() {

std::cout<<"Hexagon deleted" << std :: endl;

}

***Pentagon.h***

#ifndef OOP\_LAB1\_PENTAGON\_H

#define OOP\_LAB1\_PENTAGON\_H

#include <iostream>

#include <cstdlib>

#include "Figure.h"

class Pentagon : public Figure{

public:

Pentagon();

Pentagon(std::istream &is);

Pentagon(size\_t i);

Pentagon(const Pentagon& orig);

double Square() override;

void Print() override;

virtual ~Pentagon();

private:

size\_t side\_a;

};

#endif //OOP\_LAB1\_PENTAGON\_H

***Pentagon.cpp***

#include "Pentagon.h"

#include <istream>

#include <cmath>

Pentagon::Pentagon() :Pentagon(0) {

}

Pentagon::Pentagon(size\_t i) : side\_a(i){

std::cout << "Pentagon created: " << side\_a << std::endl;

}

Pentagon::Pentagon(std::istream &is) {

int ptr;

std::cout << "Insert side a:" << std::endl;

is >> ptr;

while (ptr < 0) {

std:: cout << "Please, insert correct data" << std:: endl;

is >> ptr;

}

side\_a = ptr;

std::cout << "Pentagon created: " << side\_a << std::endl;

}

Pentagon::Pentagon(const Pentagon &orig) {

std::cout << "Pentagon copy created" << std::endl;

side\_a = orig.side\_a;

}

double Pentagon::Square() {

return pow(side\_a,2) \* 0.25 \* sqrt(25 + 10 \* sqrt(5));

}

void Pentagon::Print() {

std::cout << "Side: a = " << side\_a << std::endl;

}

Pentagon::~Pentagon() {

std::cout<<"Pentagon deleted" << std :: endl;

}

***Octagon.h***

#ifndef OOP\_LAB1\_OCTAGON\_H

#define OOP\_LAB1\_OCTAGON\_H

#include <iostream>

#include <cstdlib>

#include "Figure.h"

class Octagon : public Figure{

public:

Octagon();

Octagon(std::istream &is);

Octagon(size\_t i);

Octagon(const Octagon& orig);

double Square() override;

void Print() override;

virtual ~Octagon();

private:

size\_t side\_a;

};

#endif //OOP\_LAB1\_OCTAGON\_H

***Octagon.cpp***

#include "Octagon.h"

#include <istream>

#include <cmath>

Octagon::Octagon() :Octagon(0) {

}

Octagon::Octagon(size\_t i) : side\_a(i){

std::cout << "Octagon created: " << side\_a << std::endl;

}

Octagon::Octagon(std::istream &is) {

int ptr;

std::cout << "Insert side a:" << std::endl;

is >> ptr;

while (ptr < 0) {

std:: cout << "Please, insert correct data" << std:: endl;

is >> ptr;

}

side\_a = ptr;

std::cout << "Octagon created: " << side\_a << std::endl;

}

Octagon::Octagon(const Octagon &orig) {

std::cout << "Octagon copy created" << std::endl;

side\_a = orig.side\_a;

}

double Octagon::Square() {

return 2 \* pow(side\_a,2) \* (1 + sqrt(2));

}

void Octagon::Print() {

std::cout << "Side: a = " << side\_a << std::endl;

}

Octagon::~Octagon() {

std::cout<<"Octagon deleted" << std :: endl;

}

5. Выводы.

Главными целями данной лабораторной работы были: программирование классов на языке С++ и понимание основной концепции ООП.

ООП - принцип программирования, основой которого является представление программы как совокупность объектов, которые в свою очередь являются частью определенных классов.

Были изучены принципы программирования классов на С++, что является отправной точкой в написании объектно-ориентированного кода. В C++ удобно работать с объектами. Можно создавать процедуры с одним и тем же именем, который при переопределении для различных данных будут работать по-разному. Благодаря всему этому мы можем избежать частого повторения кода и пользоваться заранее заготовленными программами.

СПИСОК ЛИТЕРАТУРЫ

Хоар, 1966, “Совместное использование кода”

Бьерн Страуструп — “Язык программирования С++” (Издание 3 специальное)