Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №1 по курсу «Объектно-ориентированное программирование»

Студент: С. М. Бокоч
Преподаватель:
Группа: М8О-204Б
Дата:
Оценка:
Подпись:

Москва, 2017

Лабораторная работа №1

Задача: Цель работы Целью лабораторной работы является:

- Программирование классов на языке C++
- Управление памятью в языке C++
- Изучение базовых понятий ООП.
- Знакомство с классами в C++.
- Знакомство с перегрузкой операторов.
- Знакомство с дружественными функциями.
- Знакомство с операциями ввода-вывода из стандартных библиотек.

Задание. Необходимо спроектировать и запрограммировать на языке C++ классы фигур, согласно вариантов задания. 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
- Должный иметь общий виртуальный метод расчета площади фигуры –Square
- Должны иметь конструктор, считывающий значения основных параметров фигуры из стандартного потока cin
- Должны быть расположенны в раздельных файлах: отдельно заголовки (.h), отдельно описание методов (.cpp)

Нельзя использовать:

- Стандартные контейнеры std.
- Шаблоны (template).
- Различные варианты умных указателей (shared_ptr, weak_ptr).

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

1 Теория

Основная идея ООП объект. Объект есть сущность, одновременно содержащая данные и поведение. Они являются строительными блоками объектно-ориентированных программ. Та или иная программа, которая задействует объектно-ориентированную технологию, по сути является набором объектов. Перед тем как создать объект C++, необходимо определить его общую форму, используя ключевое слово class. Класс определяет новый пользовательский тип данных, который соединяет в себе код и данные. Классы в программировании состоят из свойств (атрибутов) и методов. Свойства

В ООП существует 3 основных принципа построение классов:

Инкапсуляция – это свойство, позволяющее объединить в классе и данные, и методы, работающие с ним и скрыть детали реализации от пользователя. Наследование – это свойство, позволяющее создать новый класс-потомок на основе уже существующего, при этом все характеристики класса-родителя присваиваются классу-потомку. Полиморфизм – свойство классов, позволяющее использовать объекты классов с одинаковым интерфейсом без информации о типе и внутренней структуре объекта.

Виртуальная функция – это функция-член, которая, как предполагается, будет переопределена в производных классах. При обращении к объекту производного класса, используя указатель или ссылку на базовый класс, можно вызвать виртуальные функции объекта и выполнить переопределенную в производном классе версию функции. Виртуальные функции гарантируют, что выбрана верная версия функции для объекта, независимо от выражения, используемого для вызова функции.

2 Листинг

```
//Rectangle.h
#ifndef RECTANGLE_H
#define RECTANGLE_H
#include <iostream>
#include "Figure.h"
class Rectangle : public Figure
{
public:
  Rectangle();
  Rectangle(std::istream& is);
    Rectangle(size_t a, size_t b);
    Rectangle(const Rectangle& object1); // Copy orig for new object

    void Print() const override;
  double Area() const override;
    ~Rectangle();

private:
  size_t sideA;
  size_t sideB;
};
//Rectangle.cpp
#include "Rectangle.h"

Rectangle::Rectangle() : Rectangle(0, 0) {}

Rectangle::Rectangle(size_t a, size_t b) : sideA(a), sideB(b){
    std::cout << "Rectangle created with sides " << a << " and " << b << std::endl;
}
Rectangle::Rectangle(std::istream& is)
{
  std::cout << "Enter side A: ";
  is >> sideA;
  std::cout << "Enter side B: ";
  is >> sideB;
}

void Rectangle::Print() const
{
  std::cout << "Figure type: rectangle" << std::endl;
  std::cout << "Side A size: " << sideA << std::endl;
  std::cout << "Side B size: " << sideB << std::endl;
}

double Rectangle::Area() const {
  return sideA * sideB;
}
```

```
48
49 Rectangle::~Rectangle() {
50     std::cout << "Rectangle deleted" << std::endl;
51 }
52 //main.cpp
53 #include "Square.h"
54 #include "Rectangle.h"
55 #include "Trapezoid.h"
56 void TestFigure(Figure* figure);
57 int main()
58 {
59   TestFigure(new Square(std::cin));
60   //TestFigure(new Rectangle(std::cin));
61   //TestFigure(new Trapezoid(std::cin));
62   return 0;
63 }
64 void TestFigure(Figure* figure)
65 {
66   figure->Print();
67   std::cout << "Area: " << figure->Area() << std::endl;
68   delete figure;
69 }
```

3 Выводы

Я приобрел навыки проектирования классов и работы с ними. Фундаментальные концепции ООП инкапсуляция, наследование и полиморфизм также отражены в моей работе. Инкапсуляция в виде разделения интерфейса (печать параметров фигуры и подсчет площади) и реализации (параметры фигуры), наследование в виде производных классов Trapezoid, Square и Rectangle от класса Figure, полиморфизм в виде переопределении методов Print и Area. Удобно использовать объекты, не задумываясь о внутренней реализации.