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: Севастьянов В. С.
Преподаватель: Поповкин А. В.
Группа: 08-207

Вариант: 19
Дата:
Оценка:
Подпись:

Москва, 2017

Лабораторная работа №1

1 Цель работы

- 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.

2 Задача

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)

Фигура: прямоугольник, ромб, трапеция

3 Описание

Абстракция данных — Абстрагирование означает выделение значимой информации и исключение из рассмотрения незначимой. В ООП рассматривают лишь абстракцию данных (нередко называя её просто «абстракцией»), подразумевая набор значимых характеристик объекта, доступный остальной программе.

Инкапсуляция — свойство системы, позволяющее объединить данные и методы, работающие с ними, в классе. Одни языки (например, C++, Java или Ruby) отождествляют инкапсуляцию с сокрытием, но другие (Smalltalk, Eiffel, OCaml) различают эти понятия.

Наследование — свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствующейся функциональностью. Класс, от которого производится наследование, называется базовым, родительским или суперклассом. Новый класс — потомком, наследником, дочерним или производным классом.

Полиморфизм подтипов (в ООП называемый просто «полиморфизмом») — свойство системы, позволяющее использовать объекты с одинаковым интерфейсом без информации о типе и внутренней структуре объекта. Другой вид полиморфизма — параметрический — в ООП называют обобщённым программированием.

Класс — универсальный, комплексный тип данных, состоящий из тематически единого набора «полей» (переменных более элементарных типов) и «методов» (функций для работы с этими полями), то есть он является моделью информационной сущности с внутренним и внешним интерфейсами для оперирования своим содержимым (значениями полей). В частности, в классах широко используются специальные блоки из одного или чаще двух спаренных методов, отвечающих за элементарные операции с определенным полем (интерфейс присваивания и считывания значения), которые имитируют непосредственный доступ к полю. Эти блоки называются «свойствами» и почти совпадают по конкретному имени со своим полем (например, имя поля может начинаться со строчной, а имя свойства — с заглавной буквы). Другим проявлением интерфейсной природы класса является то, что при копировании соответствующей переменной через присваивание, копируется только интерфейс, но не сами данные, то есть класс — ссылочный тип данных. Переменная-объект, относящаяся к заданному классом типу, называется экземпляром этого класса. При этом в некоторых исполняющих системах класс также может представляться некоторым объектом при выполнении программы посредством динамической идентификации типа данных. Обычно классы разрабатывают таким образом, чтобы обеспечить отвечающие природе объ-

екта и решаемой задаче целостность данных объекта, а также удобный и простой интерфейс. В свою очередь, целостность предметной области объектов и их интерфейсов, а также удобство их проектирования, обеспечивается наследованием.

Объект — сущность в адресном пространстве вычислительной системы, появляющаяся при создании экземпляра класса (например, после запуска результатов компиляции и связывания исходного кода на выполнение).

4 Исходный код

```
1
2  class Figure {
3  public:
4    virtual double Square() = 0;
5    virtual void Print() = 0;
6    virtual ~Figure() {};
7  };
8
9  class Rectangle : public Figure
10 {
11 public:
12   Rectangle();
13   Rectangle(double side_a, double side_b);
14   Rectangle(std::istream &is);
15   Rectangle(const Rectangle& orig);
16
17   double Square() override;
18   void Print() override;
19
20   virtual ~Rectangle();
21 private:
22   double side_a;
23   double side_b;
24 };
25
26 class Rhombus : public Figure
27 {
28 public:
29   Rhombus();
30   Rhombus(double side_a, double side_b);
31   Rhombus(std::istream &is);
32   Rhombus(const Rhombus& orig);
33
34   double Square() override;
35   void Print() override;
```

```
36 |
37 |   virtual ~Rhombus();
38 | private:
39 |   double side_a;
40 |   double side_b;
41 | };
42 |
43 | class Trapeze : public Figure
44 | {
45 | public:
46 |   Trapeze();
47 |   Trapeze(double side_a, double side_b, double side_c, double side_d);
48 |   Trapeze(std::istream &is);
49 |   Trapeze(const Trapeze& orig);
50 |
51 |   double Square() override;
52 |   void Print() override;
53 |
54 |   virtual ~Trapeze();
55 | private:
56 |   double side_a;
57 |   double side_b;
58 |   double side_c;
59 |   double side_d;
60 | };
```

5 Выводы

В ходе реализации классов фигур, я ознакомился с базовыми понятиями ООП, такими, как класс, объект, инкапсуляция, абстрактный тип данных, наследование, полиморфизм, а так же познакомился с синтаксисом C++, который все же отличается от языков, в которых мы работали ранее.