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: М. В. Спиридонов
Преподаватель:
Группа: М8О-206Б

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

```
1  //figure.h
2  #ifndef FIGURE_H
3  #define FIGURE_H
4  class Figure
5  {
6    public:
7      virtual double Square() = 0;
8      virtual void Print() = 0;
9      virtual ~Figure(){};
10 };
11 #endif // !FIGURE_H
12
13 //rectangle.h
14 #ifndef RECTANGLE_H
15 #define RECTANGLE_H
16 #include <iostream>
17 #include "figure.h"
18 class Rectangle : public Figure
19 {
20 public:
21   Rectangle();
22   Rectangle(std::istream &is);
23   Rectangle(size_t sideA, size_t sideB);
24   Rectangle(const Rectangle &orig);
25
26   double Square() override;
27   void Print() override;
28
29   virtual ~Rectangle();
30
31 private:
32   size_t _sideA;
33   size_t _sideB;
34 };
35 #endif // !RECTANGLE_H
36
37 //rectangle.cpp
38 #include <iostream>
39 #include <cmath>
40 #include "rectangle.h"
41 Rectangle::Rectangle() : Rectangle(0, 0)
42 {
43 }
44
45 Rectangle::Rectangle(size_t sideA, size_t sideB) : _sideA(sideA), _sideB(sideB)
46 {
47 #ifndef NDEBUG
```

```
48      std::cout << "Rectangle created(" << _sideA << _sideB << " )" << std::endl;
49  #endif
50  }
51  Rectangle::Rectangle(std::istream &is)
52  {
53      bool ret = true;
54      int sideA;
55      int sideB;
56      while (ret)
57      {
58          is >> sideA;
59          is >> sideB;
60          if (sideA > 0 && sideB > 0)
61          {
62              ret = false;
63              _sideA = sideA;
64              _sideB = sideB;
65          }
66          else
67          {
68              std::cout << "Invalid values, please input one again" << std::endl;
69          }
70      }
71  #ifndef NDEBUG
72      std::cout << "Rectangle created(" << _sideA << _sideB << " )" << std::endl;
73  #endif
74  }
75  double Rectangle::Square()
76  {
77      size_t res = 0.0;
78      res = _sideA * _sideB;
79      return (double)res;
80  }
81  void Rectangle::Print()
82  {
83      std::cout << "Figure = Rectangle. Side: "
84                << " " << _sideA << " " << _sideB
85                << std::endl;
86  }
87  Rectangle::~Rectangle()
88  {
89  #ifndef NDEBUG
90      std::cout << "Rectangle destructed" << std::endl;
91  #endif
92  }
93  //triangle.h
94  #ifndef TRIANGLE_H
95  #define TRIANGLE_H
96  #include <iostream>
```

```
97  #include "figure.h"
98  class Triangle : public Figure {
99   public:
100   Triangle();
101   Triangle(std::istream& is);
102   Triangle(size_t i, size_t j, size_t k);
103   Triangle(const Triangle& orig);
104
105   double Square() override;
106   void Print() override;
107
108   virtual ~Triangle();
109
110  private:
111   size_t _sideA;
112   size_t _sideB;
113   size_t _sideC;
114 };
115 #endif // !TRIANGLE_H
116 //triangle.cpp
117 #include <iostream>
118 #include <cmath>
119 #include "triangle.h"
120 Triangle::Triangle() : Triangle(0, 0, 0)
121 {
122 }
123
124 Triangle::Triangle(size_t i, size_t j, size_t k) : _sideA(i), _sideB(j), _sideC(k)
125 {
126 #ifndef NDEBUG
127     std::cout << "T created(" << _sideA + ", " << _sideB + ", " << _sideC + ")" << std
        ::endl;
128 #endif
129 }
130 Triangle::Triangle(std::istream &is)
131 {
132     bool ret = true;
133     int sideA;
134     int sideB;
135     int sideC;
136     while (ret)
137     {
138         is >> sideA;
139         is >> sideB;
140         is >> sideC;
141         if (sideA > 0 && sideB > 0 && sideC > 0)
142         {
143             ret = false;
144             _sideA = sideA;
```

```
145             _sideB = sideB;
146             _sideC = sideC;
147         }
148         else
149         {
150             std::cout << "Invalid value, please input one again" << std::endl;
151         }
152     }
153 #ifndef NDEBUG
154     std::cout << "T created(" << _sideA << ", " << _sideB << ", " << _sideC << ")" <<
        std::endl;
155 #endif
156 }
157 double Triangle::Square()
158 {
159     double res = 0.0;
160     double perimetr = (double)(_sideA + _sideB + _sideC);
161     double p = perimetr / 2.;
162     res = sqrt(p * (p - _sideA) * (p - _sideB) * (p - _sideC));
163     return res;
164 }
165 void Triangle::Print()
166 {
167     std::cout << "Figure = Triangle. Sides:" << std::endl;
168     std::cout << "A=" << _sideA << " "
169               << "B=" << _sideB << " "
170               << "C=" << _sideC << " "
171               << std::endl;
172 }
173 Triangle::~Triangle()
174 {
175 #ifndef NDEBUG
176     std::cout << "T destructed" << std::endl;
177 #endif
178 }
179 //main.cpp
180 #include "triangle.h"
181 #include "square.h"
182 #include "rectangle.h"
183
184 int main(int argc, char **argv)
185 {
186     //triangle
187     std::cout << "Enter Triangle Sides: " << std::endl;
188     Figure *ptr = new Triangle(std::cin);
189     ptr->Print();
190     std::cout << "Square = " << ptr->Square() << std::endl;
191     delete ptr;
192
```

```
193     //square
194     std::cout << "Enter Square Sides: " << std::endl;
195     Figure *psq = new TSquare(std::cin);
196     psq->Print();
197     std::cout << "Square = " << psq->Square() << std::endl;
198     delete psq;
199 
200     //rectangle
201     std::cout << "Enter Rectangle Sides: " << std::endl;
202     Figure *prec = new Rectangle(std::cin);
203     prec->Print();
204     std::cout << "Square = " << prec->Square() << std::endl;
205     delete prec;
206 
207 #ifndef NDEBUG
208     int deb;
209     std::cin >> deb;
210 #endif // !DEBUG
211 
212     return 0;
213 }
214 //CMakeLists.txt
215 cmake_minimum_required(VERSION 3.0)
216 project(prog)
217 set(CMAKE_CXX_STANDARD 11)
218 set(CMAKE_C_FLAGS_DEBUG "-DNDEBUG")
219 set(CMAKE_INSTALL_PREFIX ${CMAKE_CURRENT_SOURCE_DIR}/../)
220 include_directories(${CMAKE_CURRENT_BINARY_DIR} ${CMAKE_CURRENT_SOURCE_DIR})
221 file(GLOB CPPS "*.cpp")
222 add_definitions(-Wall -O2)
223 add_executable(${PROJECT_NAME} ${CPPS})
224 target_link_libraries(${PROJECT_NAME})
```

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 Triangle, Square и Rectangle от класса Figure, полиморфизм в виде переопределении методов Print и Square. Удобно использовать объекты, не задумываясь о внутренней реализации.