计算机程序设计基础（C++）

实验报告

专业班级 软件工程2205班

学 号 8209220508

姓 名 吴昊宇

**实验报告成绩：**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **实验** | **实验1** | **实验2** | **实验3** | **实验4** | **总评** |
| **成绩** |  |  |  |  |  |

**批阅教师：**

实验五

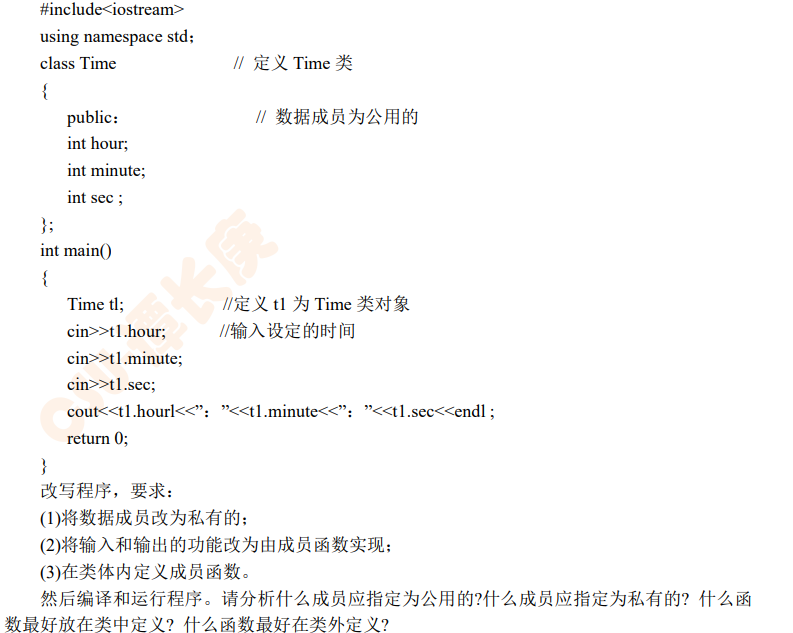
1. 实验目的

1、掌握声明类的方法，类和类的成员的概念以及定义对象的方法。

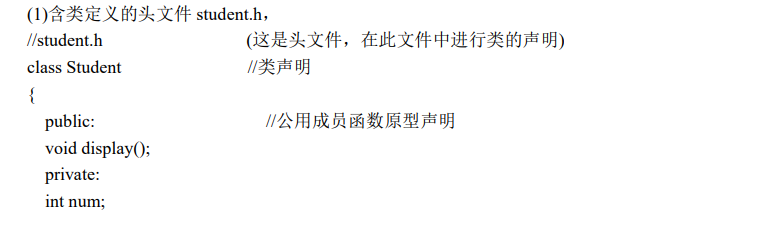
2、初步掌握用类和对象编制基于对象的程序。

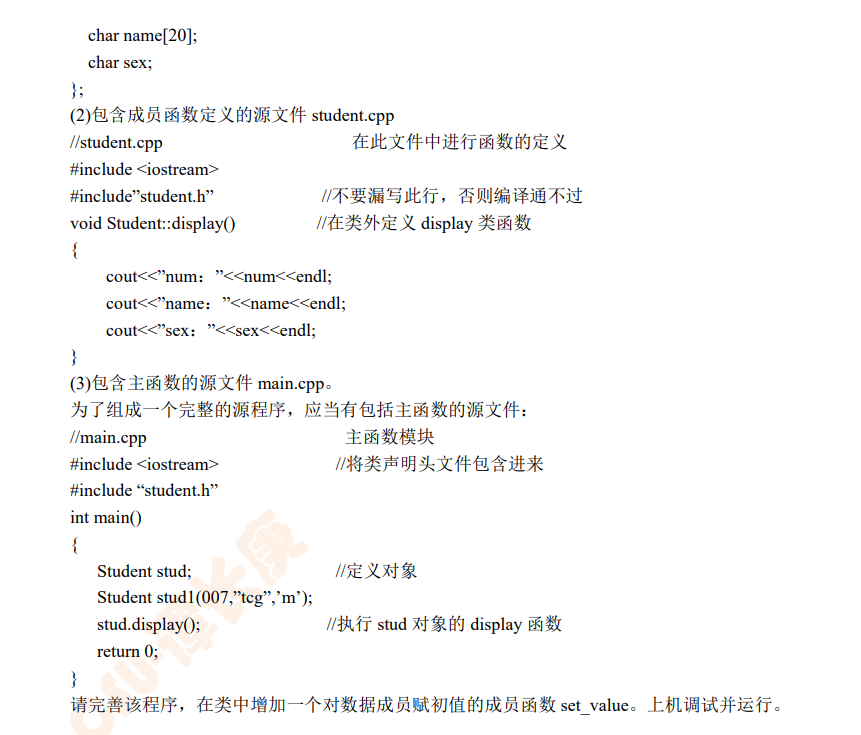
3、学习检查和调试基于对象的程序。

1. **实验内容**
2. 有以下程序：



1. 分别给出如下的 3 个文件：





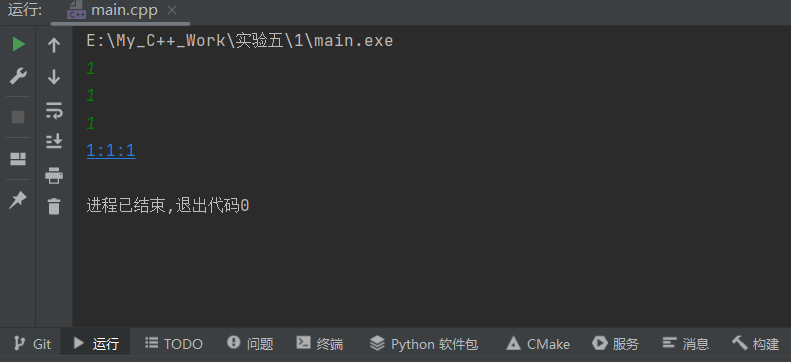
3、需要求 3 个长方柱的体积，请编一个基于对象的程序。数据成员包括 length(长)、width(宽)、 height(高)。要求用成员函数实现以下功能： (1)由键盘分别输入 3 个长方柱的长、宽、高； (2)计算长方柱的体积； (3)输出 3 个长方柱的体积。 请编程序，上机调试并运行。

4、编写程序：定义抽象基类 Shape，由它派生出五个派生类：Circle（圆形）、Square（正 方形）、Rectangle（ 长方形）、Trapezoid （梯形）和 Triangle （三角形），用虚函数分别计算 各种图形的面积，并求出它们的和。要求用基类指针数组。使它的每一个元素指向一个派生类的 对象。 注：主函数中定义如下对象 Circle circle(10.2); Square square(3); Rectangle rectangle(3,4); Trapezoid trapezoid(2.0,4.5,3); Triangle triangle(4,5,6); 15

5、设计一个函数，返回两个向量的交集，函数头如下： template vector intersection(const vector& v1, const vector& v2) 两个向量的交际为它们的共同包含的元素，编写一个测试程序，提示用户输入两个向量，每 个包含 5 个字符串，打印输出它们的交集

1. **实验步骤、算法与结果分析**
2. 程序：
3. #include<iostream>
4. **using** **namespace** std;
5. **class** Time
6. {
7. **private**:
8. **int** hour;
9. **int** minute;
10. **int** sec;
11. **public**:
12. **void** shuru() {
13. cin >> hour >> minute >> sec;
14. }
15. **void** display() {
16. cout << hour << ":" << minute << ":" << sec << endl;
17. }
18. };
19. **int** main()
20. {
21. Time t1;
22. t1.shuru();
23. t1.display();
24. **return** 0;
25. }

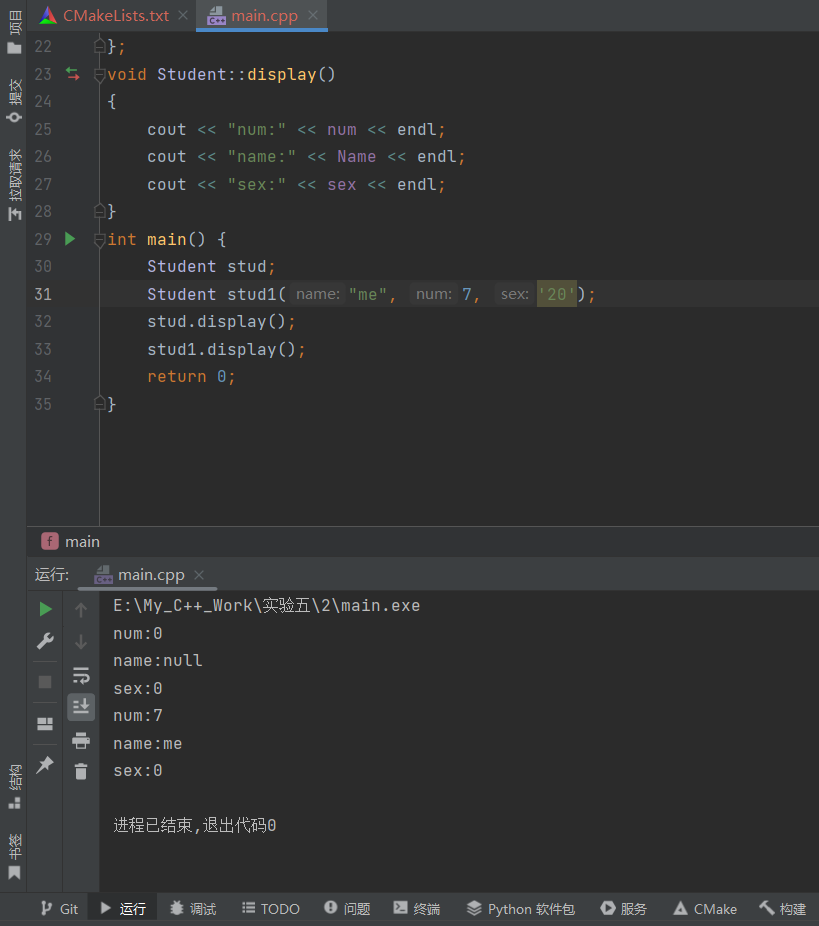
结果：



2、程序：

1. #include <iostream>
2. #include<string>
3. **using** **namespace** std;
4. **class** Student
5. {
6. **private**:
7. **int** num;
8. string Name;
9. **char** sex;
10. **public**:
11. **void** display();
12. **void** set\_value(string name, **int** num, **char** sex) {
13. **this**->num = num;
14. Name = name;
15. **this**->sex = sex;
16. }
17. Student(string name = "null", **int** num = 0, **char** sex = '0') {
18. **this**->num = num;
19. Name = name;
20. **this**->sex = sex;
21. }
22. };
23. **void** Student::display()
24. {
25. cout << "num:" << num << endl;
26. cout << "name:" << Name << endl;
27. cout << "sex:" << sex << endl;
28. }
29. **int** main() {
30. Student stud;
31. Student stud1("me", 7, '20');
32. stud.display();
33. stud1.display();
34. **return** 0;
35. }

结果：



3、程序：

1. #include <iostream>
2. **using** **namespace** std;
3. **class** changfangti {
4. **private**:
5. **int** height;
6. **int** length;
7. **int** width;
8. **public**:
9. **void** set\_value() {
10. cin >> height >> length >> width;
11. }
12. **void** display() {
13. cout << height \* length \* width << endl;;
14. }
15. };
16. **int** main()
17. {
18. cout << "第一个长方体长宽高：";
19. changfangti t1;
20. t1.set\_value();
21. cout << "第二个长方体长宽高：";
22. changfangti t2;
23. t2.set\_value();
24. cout << "第三个长方体长宽高：";
25. changfangti t3;
26. t3.set\_value();
27. cout << "第一个长方体体积为：";
28. t1.display();
29. cout << "第二个长方体体积为：";
30. t2.display();
31. cout << "第三个长方体体积为：";
32. t3.display();
33. }

结果：



4、程序：

1. #include<iostream>
2. #include<cmath>
3. **using** std::cin;
4. **using** std::cout;
5. **using** std::endl;
7. **class** Shape
8. {
9. **public**:
10. **virtual** **double** caculate() = 0;
12. };
13. //构造抽象基类Shape,并定义纯虚函数caculate()计算其面积;

16. **class** Circle : **public** Shape
17. {
18. **public**:
19. Circle(**double** R)
20. {
21. **this**->radius = R;
22. }
23. //构造函数给私有空间内的成员赋初值;
25. **public**:
26. **virtual** **double** caculate()
27. {
28. **return** 3.41 \* pow(**this**->radius, 2);
29. }
31. **private**:
32. **double** radius;
33. };
34. //子类Circle
36. **class** Square : **public** Shape
37. {
38. **public**:
39. Square(**double** L)
40. {
41. **this**->length = L;
42. }
44. **public**:
45. **virtual** **double** caculate()
46. {
47. **return** pow(**this**->length, 2);
48. }
50. **private**:
51. **double** length;
52. };
53. //子类Square
55. **class** Rectangle :**public** Shape
56. {
57. **public**:
58. Rectangle(**double** L, **double** W)
59. {
60. **this**->length = L;
61. **this**->width = W;
62. }
64. **public**:
65. **virtual** **double** caculate()
66. {
67. **return** **this**->length \* **this**->width;
68. }
70. **private**:
71. **double** length;
72. **double** width;
73. };
74. //子类Rectangle
76. **class** Trapezoid : **public** Shape
77. {
78. **public**:
79. Trapezoid(**double** up\_L, **double** down\_L, **double** H)
80. {
81. **this**->up\_length = up\_L;
82. **this**->down\_length = down\_L;
83. **this**->height = H;
84. }
86. **public**:
87. **virtual** **double** caculate()
88. {
89. **return** (**this**->up\_length + **this**->down\_length) \* **this**->height / 2;
90. }
92. **private**:
93. **double** up\_length;
94. **double** down\_length;
95. **double** height;
96. };
97. //子类Trapezoid
99. **class** Triangle : **public** Shape
100. {
101. **public**:
102. Triangle(**double** L1, **double** L2, **double** L3)
103. {
104. **this**->length1 = L1;
105. **this**->length2 = L2;
106. **this**->length3 = L3;
107. }
108. **public**:
109. **virtual** **double** caculate()
110. {
111. **double** p = (**this**->length1 + **this**->length2 + **this**->length3) / 2;
112. **return** sqrt(p \* (p - length1) \* (p - length2) \* (p - length3));
113. //海伦公式求面积
114. }
116. **private**:
117. **double** length1;
118. **double** length2;
119. **double** length3;
120. };
121. //子类Triangle
122. //
123. //以上代码均是利用继承,重写了纯虚函数caculate(),计算其各自的面积
125. **int** main()
126. {
127. Shape\* array[5];
128. //定义Shape的基类指针数组
130. Circle circle(10.2);
131. Square square(3);
132. Rectangle rectangle(3, 4);
133. Trapezoid trapezoid(2.0, 4.5, 3);
134. Triangle triangle(4, 5, 6);
135. //定义派生类对象
137. array[0] = &circle;
138. array[1] = □
139. array[2] = &rectangle;
140. array[3] = &trapezoid;
141. array[4] = ▵
142. //使其中的每一个元素指针指向一个派生类的对象
144. **double** sum = 0;
145. **for** (**int** i = 0; i < 5; i++)
146. {
147. sum += array[i]->caculate();
148. **if** (i == 0) {
149. cout << "圆形面积是:" << array[i]->caculate() << endl;
150. }
151. **if** (i == 1) {
152. cout << "正方形面积是:" << array[i]->caculate() << endl;
153. }
154. **if** (i == 2) {
155. cout << "长方形面积是:" << array[i]->caculate() << endl;
156. }
157. **if** (i == 3) {
158. cout << "梯形面积是:" << array[i]->caculate() << endl;
159. }
160. **if** (i == 4) {
161. cout << "三角形形面积是:" << array[i]->caculate() << endl;
162. }
164. }
165. //for循环求它们的面积之和;
167. cout << "五个图形的总面积是:" << sum << endl;
169. **return** 0;
170. }

结果：

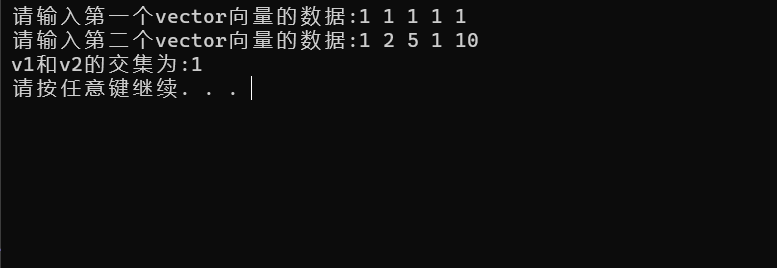


5、程序：

1. #include<iostream>
2. **using** **namespace** std;
3. #include<vector>
5. **template**<**typename** T>
6. vector<T> intersection(**const** vector<T>& v1, **const** vector<T>& v2)
7. {
8. vector<T>v;                               //这个向量用于储存v1和v2的交集
9. **bool** is = **false**;                          //用于判断两个向量是否有交集
10. **for** (**int** i = 0; i < v1.size(); i++)
11. {
12. **for** (**int** j = 0; j < v2.size(); j++)
13. {
14. **if** (v1.at(i) == v2.at(j))
15. {
16. is = **true**;
18. **if** (v.size() != 0)                  //判断v1,v2的这个交集是否已存在于v中
19. {
20. **for** (**int** z = 0; z < v.size(); z++)
21. {
22. **if** (v1.at(i) == v.at(z))
23. {
24. **break**;
25. }
26. **else**
27. {
28. v.push\_back(v1.at(i));
29. }
31. }
32. }
33. **else**
34. {
35. v.push\_back(v1.at(i));
36. }
37. }
38. }
39. }
41. **if** (is == **false**)
42. {
43. cout << "这两个向量没有交集!" << endl;
44. **return** v;
45. }
46. **else**
47. {
48. **return** v;
49. }
51. }
53. **void** printStringVector(vector<string>& v)
54. {
55. **for** (vector<string>::iterator it = v.begin(); it != v.end(); it++)
56. {
57. cout << (\*it) << " ";
58. }
59. cout << endl;
60. }
61. //函数printStringVector来打印string类型的vector向量,利用了迭代器

64. **int** main()
65. {
66. cout << "请输入第一个vector向量的数据:";
67. vector<string>v1;
68. **for** (**int** i = 0; i < 5; i++)
69. {
70. string complex;
71. cin >> complex;
72. v1.push\_back(complex);
73. }
75. cout << "请输入第二个vector向量的数据:";
76. vector<string>v2;
77. **for** (**int** j = 0; j < 5; j++)
78. {
79. string complex;
80. cin >> complex;
81. v2.push\_back(complex);
82. }
84. vector<string>v = intersection(v1, v2);                //定义向量v储存v1和v2的交集
85. cout << "v1和v2的交集为:";
86. printStringVector(v);
88. system("pause");
89. **return** 0;
90. }

结果：



1. **遇到的问题与解决办法**

主要是不知道一些函数，导致有些操作无法完成，最后通过查阅资料得到方法才解决了问题。没有理清楚类成员之间的关系，导致写代码的时候没有思路。

1. **体会**

通过这次实验，我认识到了会看错误信息的重要性，这可以帮助我们准确地找到错误，并找到解决方法。

要清楚类成员之间的关系，注意成员的可访问性，清楚类之间的调用顺序，才能完成一个程序的编写。