|  |  |
| --- | --- |
| **编号:** | **1-1** |

****

信息科学与工程学院实验报告

《面向对象程序设计》

**Object-Oriented Programming**

|  |  |
| --- | --- |
| 姓名： | 姬彬荃 |
| 学号： | 201911020125 |
| 班级： | 计联培1901 |
| 时间： | 2020年12月8日 |

**《面向对象程序设计》实验报告**

**基本要求：**请围绕实验目的、实验内容、实验过程、实验结果（附图）、实验总结（重点阐述）五个部分进行撰写。若报告中若涉及源代码内容，请在附录部分提供完整源码及GitHub源码托管地址。报告撰写完毕后请提交PDF格式版本到云班课。

1. **实验目的**

* 掌握类的多态概念和本质
* 掌握静态多态的原理和实现方式
* 掌握动态多态的原理和实现方式
* 能够使用类的多态特性设计程序
* 理解抽象类，虚函数和纯虚函数的概念

1. **实验内容**

【**任务1**】论述题

1. 请解释静态多态的概念和实现方法。
2. 请解释动态多态的概念和实现方法。
3. 解释虚函数，纯虚函数及抽象类的概念。

【**任务2**】 程序设计题（静态多态）： 请设计一个整型数组类Array，其数据成员包括：数组大小size，数组指针变量ps，该类对外接口功能：（三构一析）、输出对象内容、修改数组对象的元素、实现数组对象的赋值操作，加法操作（成员函数）、减法操作（友元函数）。

【**任务3**】程序设计题（动态多态）： 请设计一个抽象基类Shapes, 该类内声明了一个纯虚函数display(),以表示图形输出的接口。下面通过Shapes类公有派生出了2个子类： Rectangle类，Circle类。上述这两个类分别定义并具体实现了Shapes()函数。请在主函数内定义抽象类Shapes类型的指针数组Shapes \*p[2]。要求当它其中的每个基类指针指向某个派生类对象时，可以通过该指针访问到派生类各自的同名成员函数display()。通过该函数探索分析虚函数的特点，以及它在多态中的起到的核心作用。

1. **实验过程**

**上机时间**： 2020年12月08日 晚 19:00 -- 20:45

**上机地点**： E315机房 (按组序入座)

**实验环境**： Visual Studio 2019

1. **实验结果**

**【任务一】**

1. 静态多态：也称为编译期间的多态，编译器在编译期间完成的，编译器根据函数实参的类型(可能会进行隐式类型转换)，可推断出要调用那个函数，如果有对应的函数就调用该函数，否则出现编译错误。

实现方法：函数重载（同名不同参）、运算符重载。

2. 动态多态：即运行时的多态，在程序执行期间(非编译期)判断所引用对象的实际类型，根据其实际类型调用相应的方法。

实现方法：借助虚函数和基类指针或引用来实现。

3.虚函数：实质是一个指向函数的指针，在赋值兼容时可以父类指针可以指向子类的同名函数实现动态多态。

纯虚函数：在虚函数声明后不去定义而是将其“赋值”为0，等待其派生类去实现。

抽象类：含有纯虚函数的类称为抽象类，不能定义对象，但是可以定义指针或引用。目的在于提供一个统一接口规范。



1. **实验总结**
2. 对于需要申请动态空间的类，**一定不要忘了写拷贝构造函数！**不然写相关含参或返回函数时出大问题。
3. 动态内存申请一定size一定比预定+1。

* **附录：程序源码（建议基于Highlight软件导入）**



1. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
2. \*\* 功能 : 第六章实验（类的多态性）
3. \*\* 作者 : 2019Jibinquan
4. \*\* 创建 : 2020-12-8 / 19:09
5. \*\* 版权 : 431263064@qq.com
6. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
8. #include <iostream>
9. #include <cstdio>
10. #include <cmath>
11. #include <algorithm>
12. #include <cstring>
13. #include <queue>
14. #include <stack>
15. #include <vector>
16. #include <map>
17. #include <set>
18. #define ios ios::sync\_with\_stdio(false);cin.tie(0);cout.tie(0);
19. #define debug(a) cout << #a << " " << a << endl
20. **using** **namespace** std;
21. **typedef** **long** **long** ll;
22. **const** **double** pi = acos(-1);
23. **const** **double** eps = 1e-8;
24. **const** **int** inf = 0x3f3f3f3f;
25. **const** **int** maxn = 100007;//1e5+7
26. **const** ll mod = 1000000007;//1e9+7
28. /\*----------------------------------\*
29. Class Definition
30. \*-----------------------------------\*/
32. **class** Array {
33. **private**:
34. **int** size;
35. **int**\* ps;
36. **public**:
37. Array() {
38. size = 0;
39. ps = NULL;
40. }
41. Array(**int** s,**int** a[]) {
42. size = s;
43. ps = **new** **int**[size+1];//必须多申请一个空间，不然谜之内存泄漏
44. **for** (**int** i = 0; i <= s; i++) {
45. ps[i] = a[i];
46. }
47. }
48. Array(**const** Array &arr){//两个下午一场空，不写深拷见祖宗
49. size = arr.size;
50. ps = **new** **int**[size + 1];
51. **for** (**int** i = 0; i <= size; i++) {
52. ps[i] = arr.ps[i];
53. }
54. }
55. ~Array()
56. {
57. **delete**[]ps;
58. }
59. **int** length() {
60. **return** size;
61. }
62. **void** print() **const**{
63. **for** (**int** i = 0; i < size; i++) {
64. cout << ps[i] << " \n"[i == size-1];
65. }
66. }
67. **int**& operator[](**const** **int** pos) {
68. **return** ps[pos];
69. }
70. Array& operator = (**const** Array &arr) {
71. **if** (&arr == **this**) {
72. **return** \***this**;
73. }
74. **if** (ps != NULL) {
75. **delete** []ps;
76. }
77. size = arr.size;
78. **if** (size > 0) {
79. **this**->ps = **new** **int**[size+1];
80. }
81. **for** (**int** i = 0; i < size; i++) {
82. **this**->ps[i] = arr.ps[i];
83. }
84. **return** (\***this**);
85. }
86. Array operator + (**const** Array& arr) {
87. Array tmp;
88. tmp.size = size + arr.size;
89. **if** (tmp.size > 0) {
90. tmp.ps = **new** **int**[tmp.size+1];
91. }
92. **int** tp = 0;
93. **for** (**int** i = 0; i < size;i++) {
94. tmp.ps[tp++] = ps[i];
95. }
96. **for** (**int** i = 0; i < arr.size; i++) {
97. tmp.ps[tp++] = arr.ps[i];
98. }
99. **return** tmp;
100. }
101. **friend** Array operator - (Array&, Array&);
102. };
104. Array operator - (Array& ar1, Array& ar2) {
105. Array tmp;
106. tmp.ps = **new** **int**[ar1.size+1];
107. **int** tp = 0;
108. set<**int**>se;
109. queue<**int**>que;
110. **for** (**int** i = 0; i < ar1.size; i++) {
111. que.push(ar1.ps[i]);
112. }
113. **for** (**int** i = 0; i < ar2.size; i++) {
114. se.insert(ar2.ps[i]);
115. }
116. **while** (!que.empty()) {
117. **int** nw = que.front();
118. que.pop();
119. **if** (!se.count(nw)) {
120. tmp.ps[tp++] = nw;
121. }
122. }
123. tmp.size = tp;
124. **while** (tp < ar1.size) {
125. tmp.ps[tp++] = 0;//剩下的用0填满，不然还是内存谜之泄露
126. }
127. **return** tmp;
128. }
130. **class** Shapes {
131. **public**:
132. **virtual** **void** dispaly() = 0;
133. };
135. **class** Rectangle : **public** Shapes
136. {
137. **public**:
138. **void** dispaly() {
139. cout << "这是一个矩形" << endl;
140. }
141. };
143. **class** Cricle : **public** Shapes
144. {
145. **public**:
146. **void** dispaly() {
147. cout << "这是一个圆形" << endl;
148. }
149. };
151. **void** task2()
152. {
153. **int** a[] = { 1,2,3,4,5 };
154. **int** b[] = { 3,4,5,6,7 };
155. **int** c[] = { 8,9,10,11,12 };
156. Array ar1(5, a), ar2(5, b), ar3(5, a);
157. cout << endl << "重载 [] ： " << endl;
158. **for** (**int** i = 0; i < ar3.length(); i++) {
159. cout << ar3[i] << " \n"[i == ar3.length() - 1];
160. }
161. cout << endl << "重载 + ： " << endl;
162. ar3 = ar1 + ar2;
163. ar3.print();
164. cout << endl << "重载 - ： " << endl;
165. ar3 = ar1 - ar2;
166. ar3.print();
167. **return**;
168. }
170. **void** task3()
171. {
172. Shapes\* p[2+1];
173. Rectangle rec;
174. Cricle cri;
175. p[1] = &rec;
176. p[2] = &cri;
177. p[1]->dispaly();
178. p[2]->dispaly();
179. cout << endl;
180. **return**;
181. }
183. /\*----------------------------------\*
184. Main Function
185. \*-----------------------------------\*/
187. **int** main()
188. {
189. cout << "任务二：" << endl;
190. task2();
191. cout << endl << "任务三：" << endl;
192. task3();
193. system("pause");
195. **return** 0;
196. }