|  |  |
| --- | --- |
| **编号:** | **1-2** |

****

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： | 肖威 |
| 学号： | 201911990116 |
| 班级： | 计师本1901 |
| 时间： | 2020年12月08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**

任务二：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）。

任务三：设计一个抽象基类Shapes,在该类内声明了一个纯虚函数display(),以表示图形输出的接口。通过Shapes类公有派生出了2个子类： Rectangle类，Circle类。这两个类分别定义并具体实现了Shapes()函数。然后在主函数内定义抽象类Shapes类型的指针数组Shapes \*p[2]。功能是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()。

1. **实验结果**

任务一：

1. 请解释静态多态的概念和实现方法。

静态多态的概念：同一个类中的同名成员函数，在定义时在形参的个数、顺序、类型方面有所不同，程序在编译时根据实参和形参的匹配情况，即可确定类的对象需要调用哪一个成员函数。

静态多态性实现方法：函数重载法和运算符重载法。

2. 请解释动态多态的概念和实现方法。

动态多态的概念：在 基类和它的公有派生类中，同一个函数名可以对应着不同的实现代码 。需要通过基类的指针或引用指向的是基类对象或派生类对象确定要调用哪个类里的成员函数。动态多态实现方法：借助虚函数和基类指针(或基类引用)实现

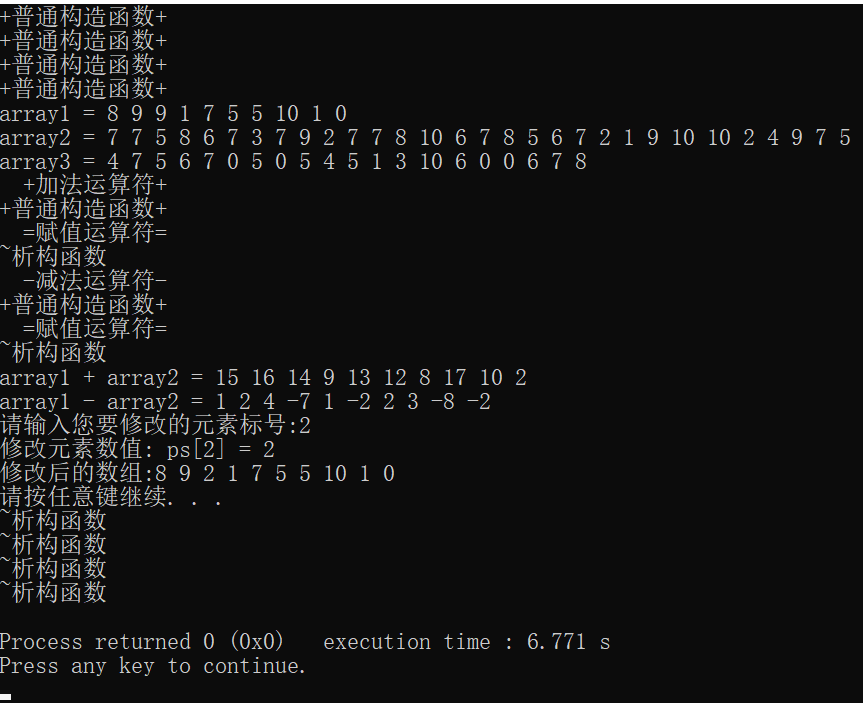
3. 解释虚函数，纯虚函数及抽象类的概念

在类的定义中，前面有virtual 关键字修饰的成员函数就是虚函数

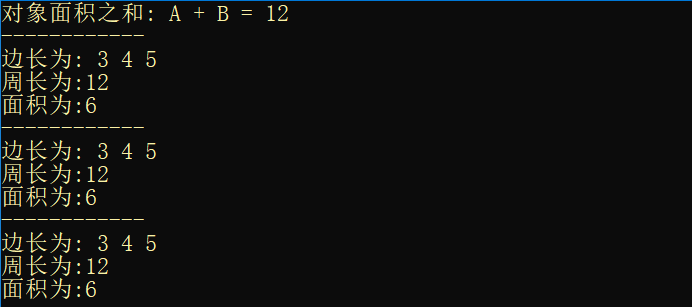
纯虚函数是一种只有函数声明没有函数体的虚函数

只要包含1个或多个纯虚函数的类，那么该类就是抽象类

任务二：



任务三：



1. **实验总结**

通过建立父类子类理解了类的三种不同关系（组合，依赖，继承），掌握复合类构造函数、析构函数的定义方法与使用方法，熟练掌握了类继承的定义方式（单继承，多继承），理解三种不同继承方式间的区别（公有，私有，保护），掌握派生类同名覆盖原理及相应同名冲突解决方法，掌握赋值兼容性基本原理（左公 = 右派），熟练掌握复杂类的设计方法（三构一析+普函）收获很多

* **附录：程序源码（建议基于Highlight软件导入）**

任务二

1 **#include <iostream>**

2 **#include <cstdlib>**

3

4

5 **using namespace** std**;**

6

7 /\*----------------------------------\*

8 Main Function

9 \*-----------------------------------\*/

10 class MyArray

11 **{**

12

13 private**:**

14 **int** **\***ps**;**

15 **int** size**;**

16

17 public**:**

18

19 MyArray**(int** \_size**=20);**

20 MyArray**(const** MyArray **&**r**);**

21 **~**MyArray**();**

22

23 **void** show**();**

24 MyArray**&** modifyArray**();**

25

26 MyArray**&** operator**=(** **const** MyArray **&**r**);**

27 MyArray operator**+(** **const** MyArray **&**r**);**

28

29 friend MyArray operator**-(**MyArray **&**r1**,** MyArray **&**r2**);**

30

31 **};**

32 MyArray**::**MyArray**(int** \_size**)**

33 **{**

34 cout **<<** "+普通构造函数+" **<<** endl**;**

35

36 **this->**size **=** \_size**;**

37

38 **this->**ps **=** **new** **int[**\_size**];**

39

40 //初始化为[0,10]内的随机整数

41 **for** **(int** i **=** **0;** i **<** \_size**;** i**++)**

42 **{**

43 **this->**ps**[**i**] =** rand**() %** **11;**

44

45 **}**

46

47

48 **}**

49

50 //拷贝构造函数(老对象初始化当前新对象)

51 MyArray**::**MyArray**(const** MyArray **&** r**)**

52 **{**

53 cout **<<** "+拷贝构造函数+" **<<** endl**;**

54

55 **if** **(**r**.**ps**)**

56 **{**

57 //为新对象分配空间

58 **this->**ps **=** **new** **int[**r**.**size**];**

59

60

61 //老对象数据初始化新对象

62 **this->**size **=** r**.**size**;**

63

64 **for** **(int** i **=** **0;** i **<** **this->**size**;** i**++)**

65 **{**

66 **this->**ps**[**i**] =** r**.**ps**[**i**];**

67 **}**

68

69 **}**

70 **else**

71 **{**

72 cout **<<** "右值对象为空,无法进行赋值" **<<** endl**;**

73 **}**

74

75

76 **}**

77 //析构函数

78 MyArray**::~**MyArray**()**

79 **{**

80 **if** **(this->**ps**)**

81 **{**

82 **delete[]** **this->**ps**;**

83 **}**

84

85 cout **<<** "~析构函数" **<<** endl**;**

86 **}**

87 //---------------------普通函数----------------

88

89 **void** MyArray**::**show**()**

90 **{**

91 **if** **(this->**ps**)**

92 **{**

93 **for** **(int** i **=** **0;** i **<** **this->**size**;** i**++)**

94 **{**

95 cout **<<** **this->**ps**[**i**] <<** " "**;**

96 **}**

97

98 cout **<<** endl**;**

99 **}**

100

101

102 **}**

103

104 MyArray **&** MyArray**::**modifyArray**()**

105 **{**

106 **if** **(**ps**)**

107 **{**

108 cout **<<** "请输入您要修改的元素标号:"**;**

109 **int** id**;**

110 cin **>>** id**;**

111

112 **if** **(**id **>=** **0** **&&** id **<** size**)**

113 **{**

114 cout **<<** "修改元素数值: ps[" **<<** id **<<** "] = "**;**

115 cin **>>** ps**[**id**];**

116 **}**

117 **else**

118 cout **<<** "您输入的元素标号超出数组范围!" **<<** endl**;**

119 **}**

120 **else**

121 **{**

122 cout **<<** "数组无内存空间，无法修改元素!" **<<** endl**;**

123 **}**

124 **return** **\*this;**

125

126 **}**

127

128 //重载赋值=运算符:成员函数形式:二目运算符(老对象 = 老对象)

129 MyArray**&** MyArray**::**operator**=(** **const** MyArray **&** r**)**

130 **{**

131 cout **<<** " =赋值运算符= " **<<** endl**;**

132

133 //1. 清空当前老对象的内存空间,重新分配新空间134 **if** **(this->**ps**)**

135 **{**

136 **delete[]** ps**;**

137 **}**

138

139 **if** **(**r**.**ps**)**

140 **{**

141 **this->**ps **=** **new** **int[**r**.**size**];**

142

143 **}**

144

145 //2. 数据对位赋值拷贝(批量对位赋值)

146 **this->**size **=** r**.**size**;**

147

148 **for** **(int** i **=** **0;** i **<** **this->**size**;** i**++)**

149 **{**

150 **this->**ps**[**i**] =** r**.**ps**[**i**];**

151 **}**

152

153 **return** **(\*this);**

154

155 **}**

156

157 //重载加法+运算符(成员函数形式):二目运算符,可以少提供一个操作数,第一操作数由\*this对象担当

158 MyArray MyArray**::**operator**+(** **const** MyArray **&**r**)**

159 **{**

160 cout **<<** " +加法运算符+ " **<<** endl**;**

161

162 //执行加法操作的两个数组对象大小不一定相同,所以以小数组尺寸加法为准

163 **int** minsize **= (this->**size **<=** r**.**size**) ?** **this->**size **:** r**.**size**;**

164

165

166 //1. 创建临时数组类对象temp

167 MyArray temp**(**minsize**);**

168

169 //2. 批量加法操作

170 **for** **(int** i **=** **0;** i **<** minsize**;** i**++)**

171 **{**

172 temp**.**ps**[**i**] =** **this->**ps**[**i**] +** r**.**ps**[**i**];**

173 **}**

174

175 //3. 返回对象类型(不要返回对象引用,因为temp对象很快会被释放)

176 **return** temp**;**

177

178 **}**

179

180

181 //重载减法运算符(友元函数形式):二目运算符,几目运算符友元函数需要提供几个操作数

182 MyArray operator**-(**MyArray **&** r1**,** MyArray **&** r2**)**

183 **{**

184

185 cout **<<** " -减法运算符- " **<<** endl**;**

186

187 //执行加法操作的两个数组对象大小不一定相同,所以以小数组加法为准

188 **int** minsize **=** r1**.**size **<=** r2**.**size **?** r1**.**size **:** r2**.**size**;**

189

190 //1. 创建临时数组类对象temp

191 MyArray temp**(**minsize**);**

192

193 //2. 批量减法操作

194 **for** **(int** i **=** **0;** i **<** minsize**;** i**++)**

195 **{**

196 temp**.**ps**[**i**] =** r1**.**ps**[**i**] -** r2**.**ps**[**i**];**

197 **}**

198

199 //3. 返回对象类型(不要返回对象引用,因为temp对象很快会被释放)

200 **return** temp**;**

201 **}**

202 **int** main**()**

203 **{**

204 //1. 生成数组类的三个对象

205 MyArray array1**(10);**

206 MyArray array2**(30);**

207 MyArray array3**(20);**

208 MyArray array4**;**

209

210

211 //2. 输出数组类的三个对象

212 cout **<<** "array1 = "**;**

213 array1**.**show**();**

214 cout **<<** "array2 = "**;**

215 array2**.**show**();**

216 cout **<<** "array3 = "**;**

217 array3**.**show**();**

218

219 //3. 调用运算符函数实现对象加减法

220 array3 **=** array1 **+** array2**;**

221 array4 **=** array1 **-** array2**;**

222

223 cout **<<** "array1 + array2 = "**;**

224 array3**.**show**();**

225 cout **<<** "array1 - array2 = "**;**

226 array4**.**show**();**

227

228 //4.调用普通成员函数修改对象内的数据

229 array1**.**modifyArray**();**

230

231 cout **<<** "修改后的数组:"**;**

232 array1**.**show**();**

233

234 system**(**"pause"**);**

235

236 **return** **0;**

237 **}**

238

任务三：

1 **#include <iostream>**

2 **#include<cmath>**

3 **using namespace** std**;**

4 //−−−−−−−−−−−−−−−−−− 抽象类 Base−−−−−−−−−−−−−−−−−−−

5 class Shapes

6 **{**

7 public**:**

8 virtual **void** display**() =** **0;**// 纯虚函数

9 **};**

10 //−−−−−−−−−−−−−−−−− 具体类 Rectangle−−−−−−−−−−−−−−

11 class Rectangles**:**public Shapes // 公有继承自基类

12 **{**

13 private**:**

14 **double** a**,** b**;**// 长和宽

15 public**:**

16 Rectangles**(double** \_a**,** **double** \_b**)**

17 **{**

18 a **=** \_a**;**

19 b **=** \_b**;**

20 **}**

21 **~**Rectangles**() {}**

22

23 **void** display**()**// 虚函数 ( 和基类的 display 的函数原型相同 )

24 **{**

25 cout **<<** "矩形边长为:" **<<** a **<<** b **<<** endl**;**

26 cout **<<** " 矩形面积为:" **<<** a **<<**b **<<** endl**;**

27 **}**

28 **};**

29 //−−−−−−−−−−−−−−−−− 具体类 Circle−−−−−−−−−−−−−−

30 class Circles**:** public Shapes //公有继承自基类

31 **{**

32 private**:**

33 **double** r**;**// 半径

34

35 public**:**

36 Circles**(double** \_r**)**

37 **{**

38 r **=** \_r**;**

39 **}**

40 **~**Circles**()**

41 **{**

42

43 **}**

44 **void** display**()** // 虚函数 ( 和基类的 display 函数原型相同 )

45 **{**

46 cout << " −−−−−−−−−−−−−−−−−− " << endl;

47 cout << " 圆的半径为 : " <<r << endl;

48 cout << " 圆的面积为 : " <<3.14\*r\*r << endl;

49 cout << " −−−−−−−−−−−−−−−−−− " << endl;

50 **}**

51 **};**

52 //∗−−−−−−−−−−−−−−−−−− 主函数 −−−−−−−−−−−−−−−−−−−−−∗/

53

54 **int** main**()**

55 **{**

56 Shapes **\***p**[2];** // 定义指针数组 , 数组内存放 2 个指针变量

57

58 Rectangles rectangle**(3,** **4);**// 矩形对象

59 Circles circle**(2);** // 圆对象

60

61 p**[0] = &**rectangle**;**// 这里指哪 , 下面 p[0]−>display() 就打哪 : 打矩形函数

62 p**[1] = &**circle**;** // 这里指哪 , 下面 p[1]−>display() 就打哪 : 打圆形函数

63

64 p**[0]->**display**();**

65 p**[1]->**display**();**

66

67 **return** **0;**

68 **}**