实验七 ——虚函数及应用

张林鹏_2021032449

一、实验目的

1. 理解虚函数与运行时 (动态) 多态性之间的关系, 掌握虚函数的定义及应用;
2. 理解纯虚函数与抽象类的概念, 掌握抽象类的定义及应用;
3. 理解虚析构函数的概念及作用.

二、实验内容

程序1: exp_701.cpp

1. 编译运行程序的输出结果是:

```
a=50  b=50
a=10  b=20
a=10  b=20  c=30
```

2. 根据程序的输出结果可知:

   i. 执行 `mp=&mb; mp->show();` 时, 调用的是 Base 类的 `show()`;
   ii. 执行 `mp=&md; mp->show();` 时, 调用的是 Base 类的 `show()`;
   iii. 执行 `((Derived*)mp)->show();` 时, 调用的是 Derived 类的 `shou()`;
   iv. 其中 `((Derived*)mp)` 是将 **Base** 类 `mp` 指针强制转换为 Derived 类的指针.

3. 在基类 `Base` 中的成员函数 `void show()` 改为 `virtual void show()`, 在重新编译运行程序, 输出结果为:

```
a=50  b=50
a=10  b=20
a=10  b=20  c=30
```

4. 当执行 `mp=&md; mp->show();` 时, 调用的是 Derived 类的 `show()`;

   函数 `show()` 成为 虚函数, 实现了 动态 多态性.

程序2: exp_702.cpp

5. 编译运行程序的输出结果为:

```
Base该类无计算
a=10
b=20
```

6. 将`Base`类中的`virtual void show()`改为`virtual void show()=0`, 重新编译程序会出现 编译错误, 其中的`show()`成为 纯 虚函数, 该类成为 抽象 类, 出错的原因是 该类不能被实例化,因为该类中存在纯虚函数.

7. 去掉`main()`函数中的`Base mb;`及`mp=&mb; mp->show();`, 重新编译运行程序的输出结果为:

```
a=10
b=20
```

程序3: exp_703.cpp

- 你分析的程序输出结果是:

```
The parent. version A
The derived 1 info:3 version 1
The derived 2 info:15 version A
```

- 程序运行实际输出结果为:

```
The parent. version A
The derived 1 info:3 version 1
The derived 2 info:15 version A
```

程序4: exp_704.cpp

- 你分析的程序输出结果是:

```
----base1----
----base2----
----derived----
----derived----
```

- 程序运行实际输出结果时:

```
----base1----
----base2----
----derived----
----derived----
```

程序5: exp_705.cpp

- 你分析的程序输出结果是:

```
Triangle with height 10 and base 6 has an area of30
Square with dimension 10 * 6 has an area of 60
Circle with radius 10 has an area of 314.159
```

- 程序实际运行时出结果是:

```
Triangle with height 10 and base 6 has an area of30
Square with dimension 10 * 6 has an area of 60
Circle with radius 10 has an area of 314.159
```

程序6: exp_706.cpp

8. 程序中:
   i. 处应改为: `virtual double area()=0;`;
   ii. 处应改为: `p=&ob1;`;
   iii. 处应改为: `p=&ob2;`;
   iv. 处应改为: `p=&ob3;`;

程序设计实验 (exp_707.cpp)

- exp_707.cpp:

```
#include <bits/stdc++.h>
using namespace std;

class process
{
public:
    virtual void print() = 0;
};
```

```
class person : public process
{
protected:
    string name;
    int age;
    string gender;

public:
    person(string n, int a, string g) : name(n), age(a), gender(g) {}
    void print()
    {
        cout << "name: " << name << endl;
        cout << "age: " << age << endl;
        cout << "gender: " << gender << endl;
    }
};

class student : public person
{
protected:
    string id;
    string major;

public:
    student(string n, int a, string g, string i, string m) : person(n, a,
g), id(i), major(m) {}
    void print()
    {
        person::print();
        cout << "id: " << id << endl;
        cout << "major: " << major << endl;
    }
};

int main()
{
    // 定义 process 指针变量调用 person 对象及 student 对象
    process *p;
    person p1("Tom", 20, "male");
    student s1("Jerry", 18, "male", "20210001", "Computer Science");
    p = &p1;
    p->print();
    cout << endl;
    p = &s1;
    p->print();
    return 0;
}
```