4.5 虚析构函数_物联网 / 嵌入式工程师 - 慕课网

> 慕课网慕课教程 4.5 虚析构函数涵盖海量编程基础技术教程，以图文图表的形式，把晦涩难懂的编程专业用语，以通俗易懂的方式呈现给用户。

5. 虚析构函数

一、分析代码

```
#include <iostream>

using namespace std;

class A{
public:
    A(){
        cout << "A()" << endl;
        aptr = new char[100];
    }

    ~A(){
        cout << "~A()" << endl;
        delete[] aptr;
    }

private:
    char *aptr;
};

class B:public A{

public:
    B(){
        cout << "B()" << endl;
        bptr = new char[100];
    }

    ~B(){
        cout << "~B()" << endl;
        delete[] bptr;
    }
```

```
private:
     char *bptr;
};

void free_function(A *p)
{
        delete p;
}



int main(void)
{
        A *aobject = new A;

        free_function(aobject);

        cout << "--------------------------" << endl;

        B *bobject = new B;

        free_function(bobject);



        return 0;
}
```

程序运行结果如下:

```
A()
~A()
--------------------------
A()
B()
~A()
```

二、将基类的析构函数变成虚函数

```
#include <iostream>

using namespace std;

class A{
        public:
```

```
        A(){
                cout << "A()" << endl;
                aptr = new char[100];
        }

        ~A(){
                cout << "~A()" << endl;
                delete[] aptr;
        }

private:
        char *aptr;
};

class B:public A{

public:
        B(){
                cout << "B()" << endl;
                bptr = new char[100];
        }

        ~B(){
                cout << "~B()" << endl;
                delete[] bptr;
        }

private:
        char *bptr;

};

void free_function(A *p)
{
        delete p;
}



int main(void)
{
        A *aobject = new A;

        free_function(aobject);
```

```
    cout << "--------------------------" << endl;

    B *bobject = new B;

    free_function(bobject);


    return 0;
}
```

程序运行结果如下:

```
A()
~A()
--------------------------
A()
B()
~B()
~A()
```

三、分析

1. 基类析构函数不是虚函数时 free_function 实现

```
_Z13free_functionP1A:
    .fnstart
.LFB983:
    @ args = 0, pretend = 0, frame = 8
    @ frame_needed = 1, uses_anonymous_args = 0
    push    {r4, r7, lr}
    .save {r4, r7, lr}
    .pad #12
    sub sp, sp, #12
    .setfp r7, sp, #0
    add r7, sp, #0
    str r0, [r7, #4]
    ldr r4, [r7, #4]
    cmp r4, #0
    beq .L13
    mov r0, r4
    bl  _ZN1AD1Ev
    mov r0, r4
    bl  _ZdlPv
.L13:
    adds    r7, r7, #12
    mov sp, r7
    @ sp needed
    pop {r4, r7, pc}
```

直接调用A类的析构函数

2. 基类析构函数是虚函数时 free_function 实现

```
_Z13free_functionP1A:
    .fnstart
.LFB985:
    @ args = 0, pretend = 0, frame = 8
    @ frame_needed = 1, uses_anonymous_args = 0
    push    {r7, lr}
    .save {r7, lr}
    .pad #8
    sub sp, sp, #8
    .setfp r7, sp, #0
    add r7, sp, #0
    str r0, [r7, #4]
    ldr r3, [r7, #4]
    cmp r3, #0
    beq .L36
    ldr r3, [r7, #4]
    ldr r3, [r3]
    adds    r3, r3, #4
    ldr r3, [r3]
    ldr r0, [r7, #4]
    blx r3
.L36:
    adds    r7, r7, #8
    mov sp, r7
    @ sp needed
    pop {r7, pc}
    .fnend
    .size   _Z13free_functionP1A, .-_Z13free_functionP1A
    .section    .rodata
```

动态获取要调用的函数

3.gdb 调试打印 B 类对象

```
(gdb) p *bobject
$1 = (B) {
  <A> = {
    _vptr.A = 0x8048ca8 <vtable for B+8>,
    aptr = 0x804b018 ""
  },
  members of B:
  bptr = 0x804b080 ""
}
(gdb) p /a *8048ca8@3
Invalid number "8048ca8".
(gdb) p /a *0x8048ca8
$2 = 0x8048b32 <B::~B()>
```

B的析构函数自动成为虚函数，出现在虚函数表中

四、总结

- 将基类的析构函数声明为虚函数，则派生类的析构函数自动为虚函数
- 基类的指针或引用指向派生类对象的时候，可以通过删除基类指针调用派生类析构函数和基类析构函数，防止资源的未释放
- 建议将基类的析构函数声明为虚函数，防止在多态场合造成资源未释放问题

全文完