

Branch: master hello-world / C++远征之封装篇 / 第四章-对象指针 / 对象成员指针 / 对象成员指针.md Find file Copy path

YHim 111 0323df0 16 days ago

1 contributor

32 lines (22 sloc) 2.31 KB

对象成员指针

对象的指针作为另外一个对象的数据成员。

例子：

一个坐标类

```
class Coordinate
{
public:
    Coordinate(int x, int y);
private:
    int m_iX;
    int m_iY;
};
```



还定义了一个线段（这里用line1来指代）

```
class Line
{
public:
    Line();
    ~Line();
private:
    Coordinate m_coorA;
    Coordinate m_coorB;
};
```



一个起点m_coorA，一个终点m_coorB。现在想把它变成指针，代码更改如下（对象成员指针的定义，这里用line2来指代）：

```
class Line
{
public:
    Line();
    ~Line();
private:
    Coordinate *m_pCoorA;
    Coordinate *m_pCoorB;
};
```



```
Line::Line():m_pCoorA(NULL), m_pCoorB(NULL)
{
}
```

```
Line::Line()
{
    m_pCoorA = NULL;
    m_pCoorB = NULL;
}
```



初始化的时候，与对象成员的初始化方法可以是一样的，使用初始化列表来初始化。因为它是一个指针，所以可以赋值NULL。除了可以使用初始化列表初始化，也可以使用普通的初始化。上图是两种初始化方法。

```
Line::Line()
{
    m_pCoorA = new Coordinate(1, 3);
    m_pCoorB = new Coordinate(5, 6);
}
Line::~~Line()
{
    delete m_pCoorA;
    delete m_pCoorB;
}
```



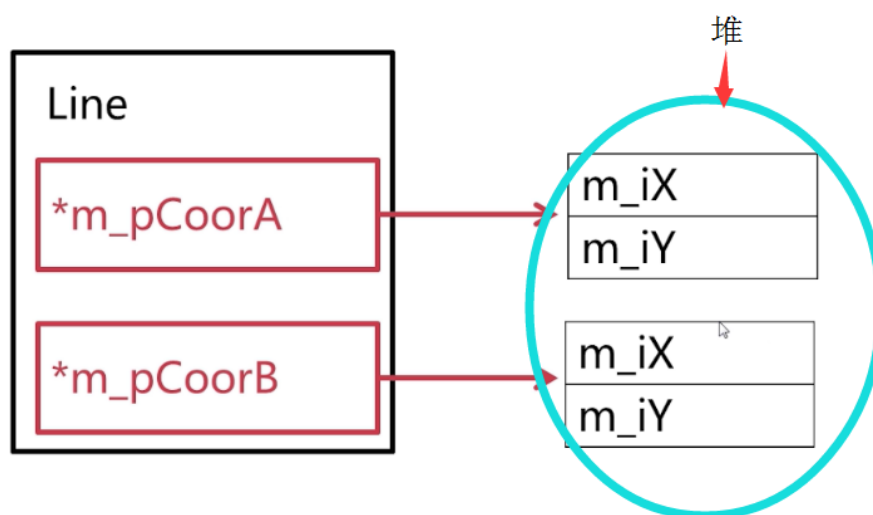
更多的是这种情况。因为是两个指针，指针一定要指向某一个对象才能够进行操作，才有意义。这里指向了两个坐标对象。这样就相当于在构造函数中，从堆中分配了内存。因此，析构函数中就要把这个内存释放掉。这样才能保证内存不被泄漏。此外，作为对象成员和对象成员指针还有另外一个很大的不同。

```
int main(void)
{
    Line line();
    cout << sizeof(line) << endl;    //8
    return 0;
}
```




作为对象成员来说，如果我们使用sizeof line这个对象的话（即sizeof(line)），那么它就应该是里面所有对象的体积的总和。而对象成员指针则不同。一个指针在32位的编译器下面，它只占4个基本内存单元。这里两个指针占8个基本内存单元。而前面的Coordinate类，它有两个数据成员，都是int类型，所以每一个数据成员都占4个基本的内存单元。如果sizeof(line)中的line的定义是line1的话，那么它有两个Coordinate,每个Coordinate里有两个int,就占了16个基本的内存单元。而如果line的定义是line2的话，那么就是两个指针，就占了8个基本的内存单元。

内存当中的对象成员指针



当实例化line这个对象的时候，两个指针就会被定义出来，这两个指针都占4个基本的内存单元。如果我们在构造函数当中，通过new从堆中申请内存，实例化两个Coordinate对象的话，那这两个Coordinate的对象都是在堆中的，而不在line这个对象的内存当中，所以刚刚使用sizeof(line)的时候得到的是8。当销毁line这个对象的时候，先释放掉堆中的内存，然后再释放掉line这个对象。

 YHim 21

2deddf5 23 seconds ago

1 contributor

148 lines (122 sloc) 2.44 KB

对象成员指针例子

要求如下：

```

/*****
/* 对象成员指针
要求：
    定义两个类：
        坐标类：Coordinate
        数据成员：横坐标m_iX, 纵坐标m_iY
        成员函数：构造函数、析构函数、数据封装函数
        线段类：Line
        数据成员：点A指针 m_pCoorA, 点B指针 m_pCoorB
        成员函数：构造函数、析构函数、信息打印函数
*/
*****/
```

Coordinate.h

```

class Coordinate
{
public:
    Coordinate(int x, int y);
    ~Coordinate();
    int getX();
    int getY();
private:
    int m_iX;
    int m_iY;
};
```

Coordinate.cpp

```

#include <iostream>
#include "Coordinate.h"
using namespace std;

Coordinate::Coordinate(int x, int y)
{
    m_iX = x;
    m_iY = y;
    cout << "Coordinate() " << m_iX << ", " << m_iY << endl;
}

Coordinate::~~Coordinate()
{
    cout << "~Coordinate() " << m_iX << ", " << m_iY << endl;
}

int Coordinate::getX()
{
    return m_iX;
}

int Coordinate::getY()
{
    return m_iY;
}
```

Line.h

```
#include "Coordinate.h"

class Line
{
public:
    Line(int x1, int y1, int x2, int y2);
    ~Line();
    void printInfo();
private:
    Coordinate *m_pCoorA; //坐标类的对象指针, 一个A点, 一个B点。
    Coordinate *m_pCoorB; //它只是一个指针, 而不是一个对象。
};
```

Line.cpp

```
#include <iostream>
#include "Line.h"
using namespace std;

Line::Line(int x1, int y1, int x2, int y2)
{
    //因为数据成员是对象指针, 所以要在构造函数里面去实例化对象。实例化的时候, 需要从堆中去申请内存而实例化这个对象。
    m_pCoorA = new Coordinate(x1, y1); //实例化了两个Coordinate对象
    m_pCoorB = new Coordinate(x2, y2);
    cout << "Line()" << endl;
}

Line::~~Line()
{
    delete m_pCoorA;
    m_pCoorA = NULL;
    delete m_pCoorB;
    m_pCoorB = NULL;
    cout << "~Line()" << endl;
}

void Line::printInfo()
{
    cout << "(" << m_pCoorA->getX() << "," << m_pCoorA->getY() << ")" << endl;
    //因为是指针, 所以通过"-"来访问相应的数据函数。
    cout << "(" << m_pCoorB->getX() << "," << m_pCoorB->getY() << ")" << endl;
}
```

demo.cpp

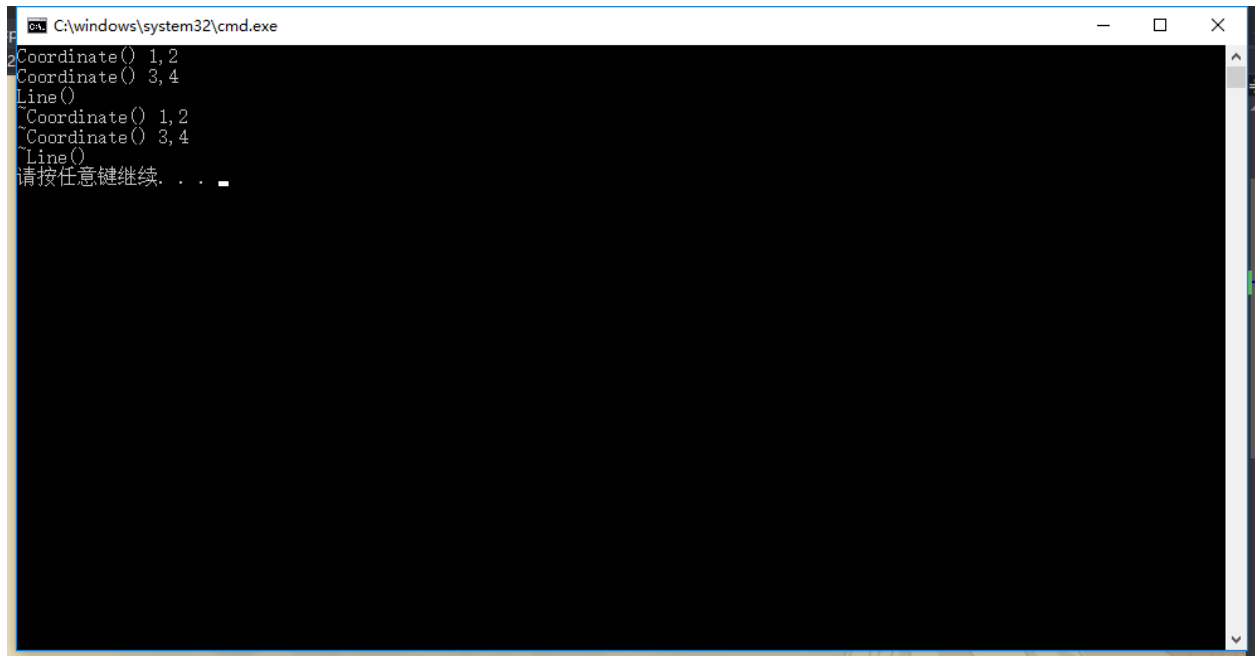
```
#include <iostream>
#include <stdlib.h>
#include "Line.h"
using namespace std;

int main()
{
    Line *p = new Line(1,2,3,4);

    delete p;
    p = NULL;

    system("pause");
    return 0;
}
```

运行结果如下：



```
C:\windows\system32\cmd.exe
Coordinate() 1,2
Coordinate() 3,4
Line()
Coordinate() 1,2
Coordinate() 3,4
Line()
请按任意键继续. . .
```

将demo.cpp更改如下：

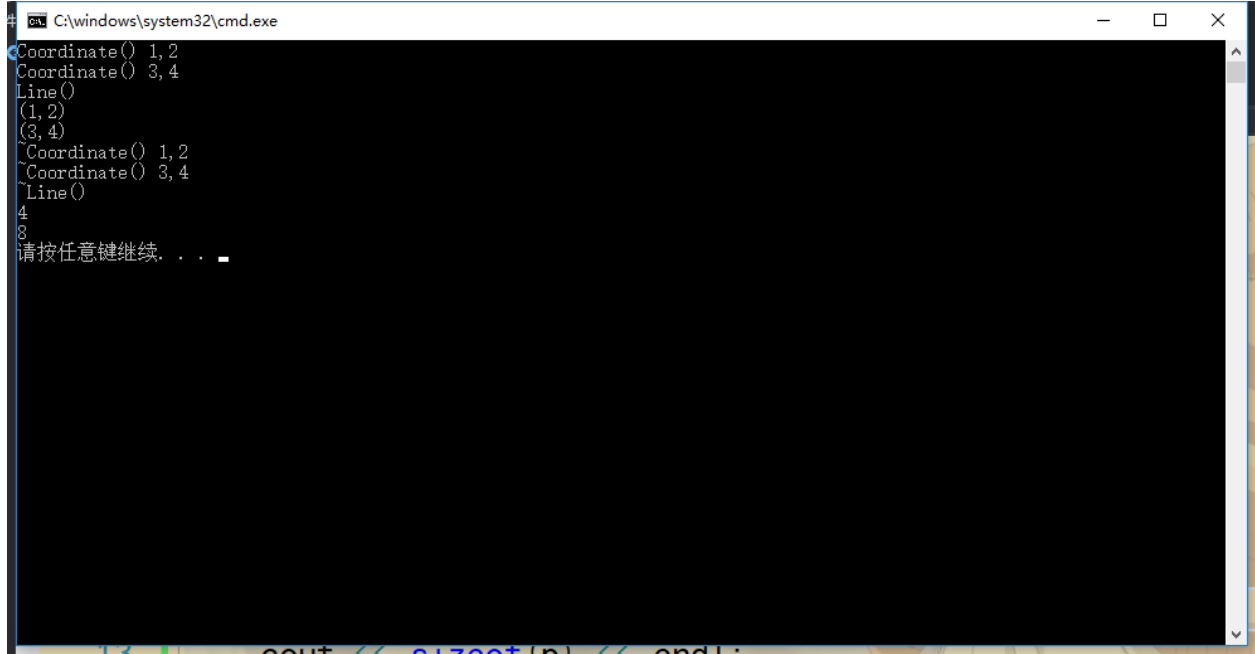
```
#include <iostream>
#include <stdlib.h>
#include "Line.h"
using namespace std;

int main()
{
    Line *p = new Line(1,2,3,4);
    p->printInfo();
    delete p;
    p = NULL;

    cout << sizeof(p) << endl; // 指针p的大小
    cout << sizeof(Line) << endl; // Line对象的大小

    system("pause");
    return 0;
}
```

运行结果如下：



```
C:\windows\system32\cmd.exe
Coordinate() 1,2
Coordinate() 3,4
Line()
(1,2)
(3,4)
~Coordinate() 1,2
~Coordinate() 3,4
~Line()
4
8
请按任意键继续. . .
```

即指针p所占的大小是4个基本的内存单元。(32位的编译环境) 64位操作系统, 指针占8字节。