 YHim 111

0323df0 14 days ago

1 contributor

22 lines (15 sloc) 1.22 KB

对象指针

就是一个指针，它指向一个对象。

图1-例子:

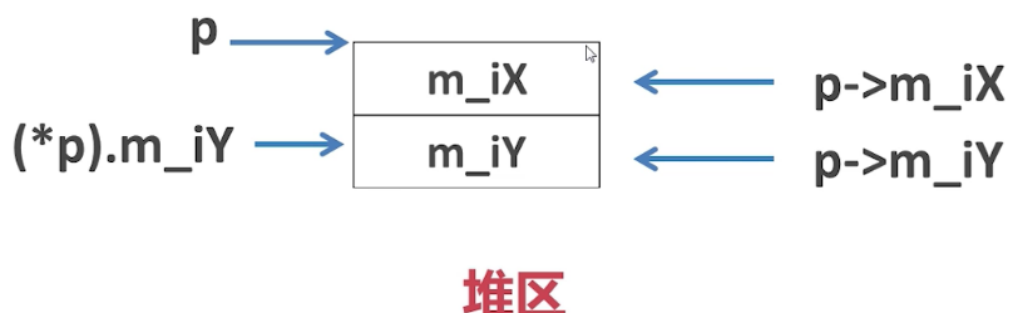
```
class Coordinate
{
public:
    int m_iX;
    int m_iY;
};
```



图2：



Coordinate *p = new Coordinate;




上图是指针p与Coordinate这个对象在内存中的相关位置以及它们的对应关系。当我们实例化完以后，就在堆中分配出了一块这样的空间。此时，m_iX的地址与p保存的地址是一致的（即p指向这个对象中的第一个元素m_iX）。通过p->m_iX来访问m_iX这个元素。第二个元素m_iY可以通过p->m_iY来访问，也可以通过(*p).m_iY来访问。*p就使得指针p变成个对象，这样就可以通过"."来访问数据成员。

使用的具体案例：

```
int main(void)
{
    Coordinate *p = new Coordinate;
    p->m_iX = 10;  //( *p).m_iX = 10;
    p->m_iY = 20;  //( *p).m_iY = 20;
    delete p;
    p = NULL;
    return 0;
}
```



使用new去实例化，它就会分配内存并调用构造函数。与C语言相比，new与C语言当中的malloc有着本质的区别。如果使用new，就会自动调用相关对象的构造函数（这里是Coordinate对象）。而使用malloc，则是单纯的分配内存，不会去调用相关对象的构造函数。例子中，m_iX和m_iY就可以通过图中的两种方式访问。使用完成后，要使用delete删除掉在堆中的对象，也就释放掉了相关的内存，保证内存不会泄漏。

 YHim 111	0323df0 14 days ago
1 contributor	

105 lines (88 sloc) 2.21 KB

对象指针例子

用此例子来说明对象指针的定义和使用方法。 要求如下：

```

/*****
/*
    要求：
        定义Coordinate类
        数据成员：m_iX和m_iY
        声明对象指针，并通过指针操控对象

        计算两个点，横、纵坐标的和
*/
*****/
```

Coordinate.h

```

class Coordinate
{
public://同样的访问限定符可以出现多次
    Coordinate();//构造函数
    ~Coordinate();//析构函数
public:
    int m_iX;
    int m_iY;
};
```

Coordinate.cpp

```

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

Coordinate::Coordinate()
{
    cout << "Coordinate()" << endl;
}

Coordinate::~~Coordinate()
{
    cout << "~Coordinate()" << endl;
}
```

demo.cpp

```

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

//两个对象代表两个坐标点
int main()
{
    Coordinate *p1 = NULL;
    p1 = new Coordinate;//因为定义Coordinate构造函数时是一个默认的
```

```

//构造函数，所以也可以
//写成p1 = new Coordinate();
Coordinate *p2 = new Coordinate;//两种定义对象的方式由p1、p2
//分别展示。同样的，这里也可以
//new Coordinate();

p1->m_iX = 10;//比较正统的访问方式
p1->m_iY = 20;
(*p2).m_iX = 30;
(*p2).m_iY = 40;
cout << p1->m_iX + (*p2).m_iX << endl;//计算两个坐标的x轴的
//和，这里p1、p2使用了
//两种不同的访问方法。

cout << p1->m_iY + (*p2).m_iY << endl;
delete p1;
p1 = NULL;
delete p2;
p2 = NULL;

system("pause");
return 0;
}

```

运行结果如下：

```

C:\windows\system32\cmd.exe
Coordinate()
Coordinate()
40
60
Coordinate()
Coordinate()
请按任意键继续. . .

```

此外，对象指针还可以指向栈中的一块地址。快捷键：按住ctrl+k+c就会将选中的代码注释掉。

将demo.cpp的代码更改如下：

```

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

int main()
{
    Coordinate p1;//从栈中实例化一个对象
    Coordinate *p2 = &p1;//p2是一个指针，这样就可以通过p2来操作
    //p1的数据成员和成员函数了。

    p2->m_iX = 10;//也可以通过(*p2).m_iX来访问
    p2->m_iY = 20;

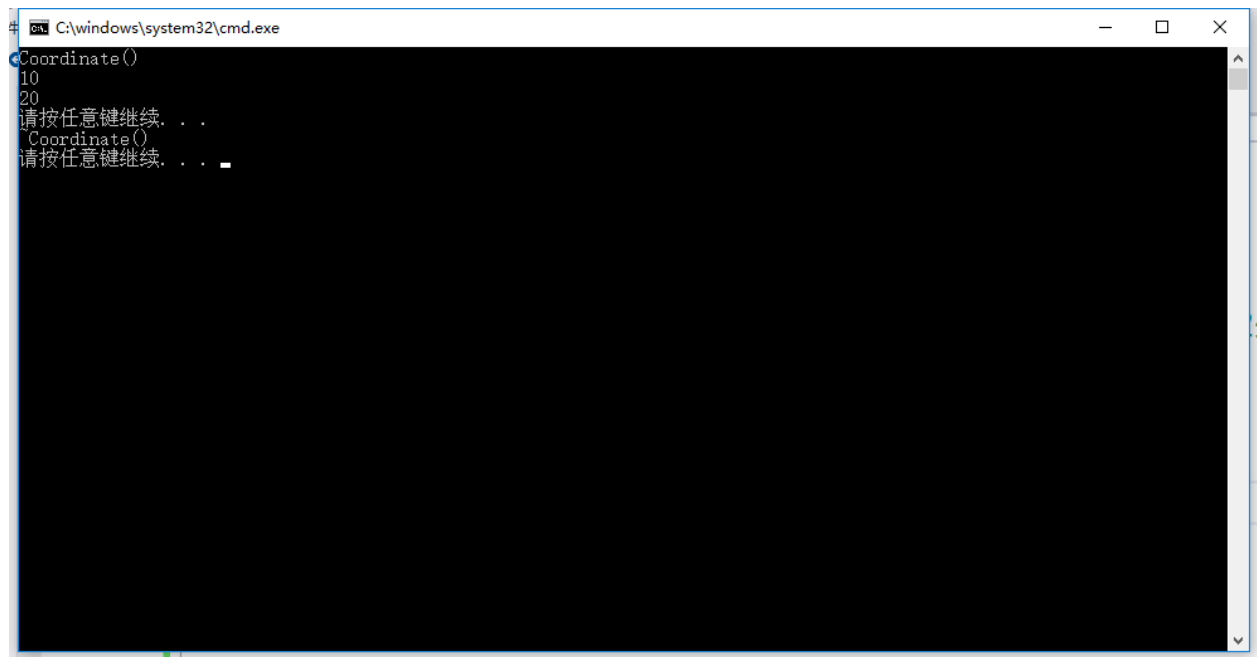
    cout << p1.m_iX << endl;
    cout << p1.m_iY << endl;

    system("pause");
}

```

```
    return 0;  
}
```

运行结果如下：



```
C:\windows\system32\cmd.exe  
Coordinate()  
10  
20  
请按任意键继续. . .  
Coordinate()  
请按任意键继续. . .
```