

25 Constructors in C++

1. 构造函数

```
class Entity
{
public:
    float X, Y;

    void Print()
    {
        std::cout << X << ", " << Y << std::endl;
    }
};

int main()
{
    Entity e;
    e.Print();
    std::cin.get();
}
```

C:\Dev\HelloWorld\bin\Win32\Debug\HelloWorld.exe
-1.07374e+08, -1.07374e+08

原因：我们实例化Entity类并且分配内存的时候，但实际上还没有对内存进行初始化

```
std::cout << e.X << std::endl; // C(编译报错):uninitialized local variable 'e' used
```

一般做法：

```
void Init()  
{  
    X = 0.0f;  
    Y = 0.0f;  
}  
  
void Print()  
{  
    std::cout <<  
}  
  
int main()  
  
    Entity e;  
    e.Init();  
    std::cout << e.X  
    e.Print();  
    std::cin.get();
```

Constructor

构造函数是一种特殊类型的方法，主要就是用在这里，每当你创建一个对象的时候就会被调用

```

class Entity
{
public:
    float X, Y;

    Entity()    // 构造函数
    {
        X = 0.0f;
        Y = 0.0f;
    }

    void Print()
    {
        std::cout << X << "," << Y << std::endl;
    }
};

int main()
{
    Entity e;
    std::cout << e.X << std::endl;    // 0
    e.Print();    // 0,0
    std::cin.get();
};

```

默认构造函数

default construction 是默认就有的，大约就像空函数体那样，什么都没做

```

Entity()    // default construction 类似
{
}

```

没有初始化变量。

C++中必须手动初始化**所有的基本类型**，不然它们会被设置为之前留存在内存中的值

2. 函数重载

可以写很多个同名构造函数，但是提供不同的参数

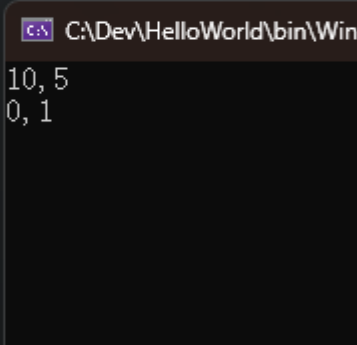
```
Entity()
{
    X = 0.0f;
    Y = 1.0f;
}

Entity(float x, float y)
{
    X = x;
    Y = y;
}

void Print()
{
    std::cout << X << ", " << Y << std::endl;
}

};

int main()
{
    Entity e(x: 10.0f, y: 5.0f);
    e.Print();
    Entity e1;
    e1.Print();
}
```



```
C:\Dev\HelloWorld\bin\Win
10, 5
0, 1
```

用`new`关键字创建对选哪个实例的时候也会调用构造函数。
我只想让别人像`Log::Write`这样使用这个类，而不希望别人创建实例。

a. 设置private隐藏构造函数

```
class Log
{
private:
    Log(){}
public:
    static void Write()
    {
    }
};

int main()
{
    Log::Write();
    Log l;
}
```

```

class Log
{
private:
    Log() {}
public:
    static void Write()
    {
    }
};

int main()
{
    Log::Write();
    Log l;
    Ent class Log
    e.F 'Log' does not name a value
}

```

或者使用

b. delete

C++

```
Log() = delete;
```

删除默认构造函数