18 CLASSES in C++

Object Oriented Programming(面向对象编程)，只是你在编写代码时采用的一种编程风格，Java、C#等一些其他语言本质上也是OOP的，使用这两种该语言来说最好不要使用其他编码风格（除非你非这样不可），但总的来说这些语言只适合面向对象编程。

而C++不仅仅支持面向对象编程（C++支持面向过程、基于对象、面向对象、泛型编程四种）。
C语言不支持面向对象，因为OOP需要诸如*class, object*(类和对象)这样的概念。

1. Class 类

简而言之，类是一种将数据和函数组织在一起的方式。

如果你要做一个游戏，游戏人物有很多属性，包括所在位置，人物状态等等。
如果有多个人物，你将不得不重复定义属性。
你可以用数组来取代这些，但是重点是这还是让一堆无组织的变量充斥在代码中，显然不是一个好主意。
而且如果你要设置一个函数来移动人物角色，我们将需要三个整数来作为函数的参数，如下：

```C++
void Move(int x,int y,int speed) //坐标和速度
```

这样会产生很多代码，而且难以维护

因此可以用*class*来简化，设置一个Player类，一次性包含所有想要的数据，最终作为一个整体

由*class*类型制成的变量叫做*object*，新创建对象叫*instance*(实例），新创建对象的过程叫实例化。

2. Class的访问

```C++
class Player
{
    int x, y;
    int speed;
};


int main()
{
    Player player;
    player.x = 5;    // 编译报错
    std::cin.get();
}
```

默认情况下，类中的成员的访问控制都是*private*(私有)的，意味着只有类内部的*functions*才能访问这些变量。
我们想在main函数中访问这些变量，所以我们要做的是定义为*public*(公有)的。

```C++
class Player
{
public:
    int x, y;
    int speed;
};


int main()
{
    Player player;
    player.x = 5;  // 编译成功
    std::cin.get();
}
```

3. method （方法）

*class*内的函数称作*method*(方法)

一般定义Move()

```C++
class Player
{
public:
    int x, y;
    int speed;
};


void Move(Player& player, int xa, int ya)
{
    player.x += xa * player.speed;
    player.y += ya * player.speed;

}

int main()
{
    Player player;
    Move(player,1,-1)
    std::cin.get();
}
```

Move()转为method

```C++
class Player
{
public:
    int x, y;                   // 变量1
    int speed;                  // 变量2
    void Move(int xa, int ya)  // 变量3    //去除了player参数
    {
        x += xa * speed;
        y += ya * speed;
    }
};

int main()
{
    Player player;
    player.Move( 1, -1);  // 去除player参数
    std::cin.get();
}
```

代码简洁了很多，可读性更强，更好维护

4. 总结

本质上将*class*就是能使我们能对变量进行组织，变成一个类型，还为这些变量添加了函数。
我们有了数据和操作这些数据的函数。

用类能做的事不用类也行，这就是C语言存在还挺好用的原因。只是让程序员更舒服的*syntax sugar*