* 三种结构体初始化方法

1.利用结构体自带的默认构造函数  
2.利用带参数的构造函数  
3.利用默认无参的构造函数

**要点**：  
在建立结构体数组时,如果只写了带参数的构造函数将会出现数组无法初始化的错误！！！各位同学要牢记呀！  
下面是一个比较安全的带构造的结构体示例

struct node{

int data;

string str;

char x;

//注意构造函数最后这里没有分号哦！

node():x(), str(), data(){} //无参数的构造函数数组初始化时调用

node(int a, string b, char c) :data(a), str(b), x(c){}//有参构造

}N[10];

下面我们分别使用默认构造和有参构造，以及自己手动写的初始化函数进行会结构体赋值  
并观察结果  
测试代码如下:

#include <iostream>

#include <string>

using namespace std;

struct node{

int data;

string str;

char x;

//自己写的初始化函数

void init(int a, string b, char c){

this->data = a;

this->str = b;

this->x = c;

}

node():x(), str(), data(){}

node(int a, string b, char c) :x(c), str(b), data(a){}

}N[10];

int main(){

N[0] = { 1,"hello",'c' };

N[1] = { 2,"c++",'d' }; //无参默认结构体构造体函数

N[2].init(3, "java", 'e'); //自定义初始化函数的调用

N[3] = node(4, "python", 'f'); //有参数结构体构造函数

N[4] = { 5,"python3",'p' };

//现在我们开始打印观察是否已经存入

for (int i = 0; i < 5; i++){

cout << N[i].data << " " << N[i].str << " " << N[i].x << endl;

}

system("pause");

return 0;

}

**输出结果**

1 hello c

2 c++ d

3 java e

4 python f

5 python3 p

发现与预设的一样结果证明三种赋值方法都起了作用,好了就写到这里