C++结构体

什么是结构体?

让我们先从一个案例讲起:

> SDUT某实验室集训队的选拔又要开始了，集训队需要根据参加者的分数排名来决定谁能够进入集训队。先要求你写一个自动排名的程序，根据每一个参与选拔的同学的分数进行排名，集训队需要根据大家的分数排名来决定谁能够进入集训队。 你能完成这个程序么?

在这个案例中,我们想要存储集训队同学的数据,很显然我们需要存储学生姓名name,以及分数score,而且有很多学生,那么我们需要一个数组.但是数组只能存储一种类型的元素,就是某个单一类型元素的序列,如果我们想要同时存储string类型和int类型的数据,是字符串数组和整型数组都无法完成的

为了解决上面的任务,C语言提供了结构体这个构造数据类型

结构体的定义

> 结构体是一种用户自定义的数据类型，用于将不同类型的数据组合在一起

也就是说,我们可以将不同类型的数据类型封装起来,组成一个新的数据类型

结构体的声明

在C和C++中,我们都使用struct关键字来声明结构体

```
struct type_name {
member_type1 member_name1;
member_type2 member_name2;
member_type3 member_name3;
.
.
} object_names;
```

type_name就是自定义结构体的名称,类似于int和float

member_type1是结构体的成员数据类型,可以是基本数据类型,也可以是结构体

object_names是一个结构体的实例,类似于int a;其中a就是用int声明的一个实例

基于一开始的问题,我们可以声明一个类型为stu的结构体,里面存放着int类型的分数和string类型的姓名

```
struct stu{
    string name;
    int score;
};
```

创建结构体实例

我们可以在声明结构体的同时创建结构体实例

```
struct stu{
    string name;
    int score;
}student1, student2;
```

我们也可以在主函数当中创建结构体实例

```
int main(){
    struct stu student3; //C语言语法
    stu student4; //C++语法
}
```

访问结构体成员变量

我们可以通过.运算符来访问结构体当中的成员变量

```
int main(){
    stu student;
    student.name;
    student.score;
}
```

结构体成员变量的赋值

整体赋值

```
stu student={"wujinhao", 60};
```

在这条语句中,我们创建了一个结构体实例student并且为其初始化,name初始化为"wujinhao",score初始化为60

单独赋值

我们也可以通过.运算符访问成员变量单独为某一个成员变量赋值

```
stu student;
student.name="wujinhao";
student.score=10;
```

当然我们也可以结合输入函数来为结构体成员赋值

```
stu student;
scanf("%s",&student.name);
cin>>student.name;

scanf("%d", &student.score);
cin>>student.score;

cout<<student.name<<" "<<student.score; //输出
```

结构体数组

现在,我们学会了如何声明一个结构体类型,以及如何创建一个结构体实例,但是我们还不能满足,因为我们的人物是要存储一系列学生信息,一个结构体实例肯定无法完成这样的要求,那么我们就可以使用结构体数组

和普通的数组一样,我们需要先声明

```
stu students[N];
// int a[N];
```