<https://blog.csdn.net/zhanghow/article/details/53463825>

数组（Array），它是一组具有相同类型的数据的集合。但在实际的编程过程中，我们往往还需要一组类型不同的数据，例如对于学生信息登记表，姓名为字符串，学号为整数，年龄为整数，所在的学习小组为字符，成绩为小数，因为数据类型不同，显然不能用一个数组来存放。  
  
在C语言中，可以使用结构体（Struct）来存放一组不同类型的数据。结构体的定义形式为：

struct 结构体名{  
    结构体所包含的变量或数组  
};

结构体是一种集合，它里面包含了多个变量或数组，它们的类型可以相同，也可以不同，每个这样的变量或数组都称为结构体的成员（Member）。请看下面的一个例子：

1. **struct** stu{
2. char \*name; //姓名
3. int num; //学号
4. int age; //年龄
5. char group; //所在学习小组
6. float score; //成绩
7. };

stu 为结构体名，它包含了 5 个成员，分别是 name、num、age、group、score。结构体成员的定义方式与变量和数组的定义方式相同，只是不能初始化。

注意大括号后面的分号;不能少，这是一条完整的语句。

结构体也是一种数据类型，它由程序员自己定义，可以包含多个其他类型的数据。  
  
像 int、float、char 等是由C语言本身提供的数据类型，不能再进行分拆，我们称之为基本数据类型；而结构体可以包含多个基本类型的数据，也可以包含其他的结构体，我们将它称为复杂数据类型或构造数据类型。

## 结构体变量

既然结构体是一种数据类型，那么就可以用它来定义变量。例如：

struct stu stu1, stu2;

定义了两个变量 stu1 和 stu2，它们都是 stu 类型，都由 5 个成员组成。注意关键字struct不能少。  
  
stu 就像一个“模板”，定义出来的变量都具有相同的性质。也可以将结构体比作“图纸”，将结构体变量比作“零件”，根据同一张图纸生产出来的零件的特性都是一样的。  
  
你也可以在定义结构体的同时定义结构体变量：

1. **struct** stu{
2. char \*name; //姓名
3. int num; //学号
4. int age; //年龄
5. char group; //所在学习小组
6. float score; //成绩
7. } stu1, stu2;

将变量放在结构体定义的最后即可。  
  
如果只需要 stu1、stu2 两个变量，后面不需要再使用结构体名定义其他变量，那么在定义时也可以不给出结构体名，如下所示：

1. **struct**{ //没有写 stu
2. char \*name; //姓名
3. int num; //学号
4. int age; //年龄
5. char group; //所在学习小组
6. float score; //成绩
7. } stu1, stu2;

这样做书写简单，但是因为没有结构体名，后面就没法用该结构体定义新的变量。  
  
理论上讲结构体的各个成员在内存中是连续存储的，和数组非常类似，例如上面的结构体变量 stu1、stu2 的内存分布如下图所示，共占用 4+4+4+1+4 = 17 个字节。

http://c.biancheng.net/cpp/uploads/allimg/160809/1-160P91353122A.jpg

但是在编译器的具体实现中，各个成员之间可能会存在缝隙，对于 stu1、stu2，成员变量 group 和 score 之间就存在 3 个字节的空白填充（见下图）。这样算来，stu1、stu2 其实占用了 17 + 3 = 20 个字节。

http://c.biancheng.net/cpp/uploads/allimg/160809/1-160P914122VL.jpg

关于成员变量之间存在“裂缝”的原因，我们将在《[C语言和内存](http://c.biancheng.net/cpp/u/c20/)》专题中的《[C语言内存对齐，提高寻址效率](http://c.biancheng.net/cpp/html/3001.html)》一节中详细讲解。

## 成员的获取和赋值

结构体和数组类似，也是一组数据的集合，整体使用没有太大的意义。数组使用下标[ ]获取单个元素，结构体使用点号.获取单个成员。获取结构体成员的一般格式为：

结构体变量名.成员名;

通过这种方式可以获取成员的值，也可以给成员赋值：

1. #include <stdio.h>
2. int main(){
3. **struct**{
4. char \*name; //姓名
5. int num; //学号
6. int age; //年龄
7. char group; //所在小组
8. float score; //成绩
9. } stu1;
10. //给结构体成员赋值
11. stu1.name = "Tom";
12. stu1.num = 12;
13. stu1.age = 18;
14. stu1.group = 'A';
15. stu1.score = 136.5;
16. //读取结构体成员的值
17. printf("%s的学号是%d，年龄是%d，在%c组，今年的成绩是%.1f！\n", stu1.name, stu1.num, stu1.age, stu1.group, stu1.score);
18. **return** 0;
19. }

运行结果：  
Tom的学号是12，年龄是18，在A组，今年的成绩是136.5！  
  
除了可以对成员进行逐一赋值，也可以在定义时整体赋值，例如：

1. **struct**{
2. char \*name; //姓名
3. int num; //学号
4. int age; //年龄
5. char group; //所在小组
6. float score; //成绩
7. } stu1, stu2 = { "Tom", 12, 18, 'A', 136.5 };

不过整体赋值仅限于定义结构体变量的时候，在使用过程中只能对成员逐一赋值，这和数组的赋值非常类似。  
  
需要注意的是，结构体是一种自定义的数据类型，是创建变量的模板，不占用内存空间；结构体变量才包含了实实在在的数据，需要内存空间来存储