1.关于c语言的结构体:

首先我们为什么要用到结构体，我们都已经学了很多int char …等类型还学到了同类型元素构成的数组，以及取上述类型的指针，在一些小应用可以灵活使用，然而，在我们实际应用中，每一种变量进行一次声明，再结合起来显然是不太实际的，类如一位学生的信息管理，他可能有，姓名（char），学号（int）成绩（float）等多种数据。如果把这些数据分别单独定义，就会特别松散、复杂，难以规划，因此我们需要把一些相关的变量组合起来，以一个整体形式对对象进行描述，这就是结构体的好处。

2首先我们要了解一些小知识

2.1**只有结构体变量才分配地址，而结构体的定义是不分配空间的。**
2.2**结构体中各成员的定义和之前的变量定义一样，但在定义时也不分配空间。**
2.3***结构体变量的声明需要在主函数之上或者主函数中声明，如果在主函数之下则会报错***
2.4**c语言中的结构体不能直接进行强制转换，只有结构体指针才能进行强制转换**
2.5相同类型的成员是可以定义在同一类型下的
列如

```
1
2  struct Student
3  {
4      int number,age; //int型学号和年龄
5      char name[20],sex;//char类型姓名和性别
6      float score;
7  };
```

最后的分号不要忘了 有的编译器会自动加上，因此有的同学就会不注意。

3关于结构体变量的定义和引用

在编译时，结构体的定义并不分配存储空间，对结构体变量才按其数据结构分配相应的存储空间

```
1
2   struct Book
3   {
4      char title[20];//一个字符串表
5
6  示的titile 题目
7      char author[20];//一个字符串表示的author作者
8      float value;//价格表示
9   };//这里只是声明 结构体的定义
10 struct Book book1,book2;//结构体变量的定义 分配空间
11
12 book1.value;//引用结构体变量
```

定义结构体变量以后，系统就会为其分配内存单元，比如book1和book2在内存中占**44**个字节（20+20+4）具体的长度你可以在你的编译器中使用sizeof关键字分别求出来。

1结构体整体空间是占用空间最大的成员（的类型）所占字节数的整数倍。

2.结构体的每个成员相对结构体首地址的偏移量(offset)都是最大基本类型成员字节大小的整数倍，如果不是编译器会自动补齐，

1.14.1，空结构体多大？

结构体所占的内存大小是其成员所占内存之和（关于结构体的内存对齐，请参考预处理那章）。这点很容易理解，但是下面的这种情况呢？

```
struct student
{
}stu;
```

sizeof(stu)的值是多少呢？在 Visual C++ 6.0 上测试一下。

很遗憾，不是 0，而是 1。为什么呢？你想想，如果我们把 struct student 看成一个模子的话，你能造出一个没有任何容积的模子吗？显然不行。编译器也是如此认为。编译器认为任何一种数据类型都有其大小，用它来定义一个变量能够分配确定大小的空间。既然如此，编译器就理所当然的认为任何一个结构体都是有大小的，哪怕这个结构体为空。那万一结构体真的为空，它的大小为什么值比较合适呢？假设结构体内只有一个 char 型的数据成员，那其大小为 1byte（这里先不考虑内存对齐的情况）.也就是说非空结构体类型数据最少需要占一个字节的空间，而空结构体类型数据总不能比最小的非空结构体类型数据所占的空间大吧。这就麻烦了，空结构体的大小既不能为 0，也不能大于 1，怎么办？定义为 0.5 个 byte？但是内存地址的最小单位是 1 个 byte，0.5 个 byte 怎么处理？解决这个问题的最好办法就是折中，编译器理所当然的认为你构造一个结构体数据类型是用来打包一些数据成员的，而最小的数据成员需要 1 个 byte，编译器为每个结构体类型数据至少预留 1 个 byte 的空间。所以，空结构体的大小就定位 1 个 byte。

14.6 指向结构的指针

喜欢使用指针的人一定很高兴能使用指向结构的指针。至少有 4 个理由可以解释为何要使用指向结构的指针。第一，就像指向数组的指针比数组本身更容易操控（如，排序问题）一样，指向结构的指针通常比结构本身更容易操控。第二，在一些早期的C实现中，结构不能作为参数传递给函数，但是可以传递指向结构的指针。第三，即使能传递一个结构，传递指针通常更有效率。第四，一些用于表示数据的结构中包含指向其他结构的指针。

14.6.1 声明和初始化结构指针

声明结构指针很简单：

```
struct guy * him;
```

首先是关键字 struct，其次是结构标记 guy，然后是一个星号（*），其后跟着指针名。这个语法和其他指针声明一样。

该声明并未创建一个新的结构，但是指针him现在可以指向任意现有的guy类型的结构。例如，如果barney是一个guy类型的结构，可以这样写：

```
him = &barney;
```

和数组不同的是，结构名并不是结构的地址，因此要在结构名前面加上&运算符。

14.7 向函数传递结构的信息

函数的参数把值传递给函数。每个值都是一个数字——可能是int类型、float类型，可能是ASCII字符码，或者是一个地址。然而，一个结构比一个单独的值复杂，所以难怪以前的C实现不允许把结构作为参数传递给函数。当前的实现已经移除了这个限制，ANSI　C允许把结构作为参数使用。所以程序员可以选择是传递结构本身，还是传递指向结构的指针。如果你只关心结构中的某一部分，也可以把结构的成员作为参数。我们接下来将分析这3种传递方式，首先介绍以结构成员作为参数的情况。