结构体和共同体

2016年06月29日 14:08:55 yesNow_xiao 阅读数：3266

结构体和共同体

一、结构体的定义

1、结构体的应用环境

结构体适用用是将不同类型的数据成员组合到一起，适用于关系紧密和逻辑相关的数据进行处理；
与共同体相比较而言，共同体虽然也能表示不同类型数据的数据集合，但是其数据成员的情形是互斥的，每一时刻只有一个数据成员起作用，例如一个况，未婚、已婚、离婚，这三个状态在某一时期只能存在一种情况。

2、结构体变量的定义

- 结构体模板的定义

```
struct 结构体变量名
{
    数据类型 成员名称;
    ......
    ......
};
```

```
struct student
{
    char name[10];
    long studentID;
    char studentSex;
    int score[4];
    ......
};
```

- 结构体变量的定义方法
(1)先声明模板再定以结构体变量
结构体模板上面我们已定义；

```
struct student stu;
```

(2)再声明结构体模板的同时定义结构体变量

```
struct student
{
    char name[10];
    long studentID;
    char studentSex;
    int score[4];
    ......
}stu;       //只需在模板结尾处定义即可；
```

- 用typedef来定义数据类型
typedef相当于给结构体变量起了一个别名，这样我们就不用每定义一个写一大串模板声明了。
下面我同样介绍他的两种用法：
(1)先声明模板再定义别名：

```
typedef struct student STUDENT;
```

(2)在声明模板的时候定义别名：

```
typedef struct student
{
    char name[10];
    long studentID;
    char studentSex;
    int score[4];
    ......
}STUDENT;
```

这样再给起了一个别名之后，我们就可以愉快的使用他来定义结构体变量了

```
struct student stu1, stu2;
STUDENT stu1, stu2;      //我是不是比他要帅好多啊！
```

- 结构体变量的初始化
  就拿上面的模板来说事：

```
STUDENT stu1 = {"fengxiao", 110112119, m, {99, 99, 99, 99}....};
```

- 嵌套的结构体
  所谓结构体嵌套就是在一个结构体里有包裹了一个结构体，其实就是袋鼠妈妈定理；
  先定义一个结构体：

```
typedef struct date
{
    int years;
    int month;
    int day;
}DATE;
```

下面我们来做袋鼠妈妈：

```
typedef struct baby
{
    char name[10];
    char sex;
    DATE birthday;      //这就是所谓的嵌套结构体，是不是so easy!;
}BABY;
```

- 结构体变量的引用
  访问结构体变量成员时，必须使用*成员选择运算符*（也叫做圆点运算符），格式：

> 结构体变量名.成员名称

当遇到嵌套结构体时，我们圆点运算符逐级访问他们

```
stu1.brithday.years = 2016;
stu1.brithday.month = 06;
....
```

- 结构体变量的赋值操作
  c语言允许相同结构体类型变量之间的整体赋值操作；

```
stu1 = stu2;  //这是同类型变量之间的赋值操作；
```

结构体成员也可以进行使用赋值运算符来进行赋值，但是对于字符数组类型的结构体变量进行赋值时，必须用字符串处理函数来进行赋值：

```
stu1.brithday,years = stu2.brithday.years;
```

```
strcpy(stu1.name,stu2.name);
```

- 结构体所占内存的字节数
  下面我用一段代码来说明：

```
#include "stdio.h"
typedef struct sample
{
    char m1;
    int m2;
    char m3;
}SAMPLE;

int main()
{
    SAMPLE s = {'a', 2, 'c'};
    printf("bytes = %d\n", sizeof(s) );
    return 0;
}
```

结果是：bytes = 12；
这是为什么呢？
我们下面来捋一捋，一个字符占1个字节，一个数字占4个字节，但是为什么是12个字节呢（1+4+1 = ?),这里呢引入了内存对齐的需求，未满足处理器可能会在较小的成员后面添加补位，所以会将两个字符的字节数扩展到4个，在其后面添加3个补位，所以4*3 = 12。

> 由此可见，系统为结构体变量分配的内存大小，不仅与定义的结构体类型有关，还与计算机本身有关。

二、结构体数组的定义和初始化

1、结构体数组的定义：

```
STUDENT stu[30];
```

2、结构体数组的初始化

```
STUDENT stu[30] = {{"fengxiao", 110112112, 'm', {99. 99, 99, 99}},
                   {"dongxiaohui", 110112113, 'm', {60, 60, 60, 60}},
                   {"mawei", 112112114, 'm', {59, 59, 59, 59}},
                   ......
                  };
```

三、结构体指针的定义和初始化

1、指向结构体变量的指针

```
STUDENT *p = &stu;
```

接下来我来介绍另一个访问结构体变量成员的运算符，叫做指向运算符，也叫箭头运算符；
语法：指向结构体的指针变量->成员名

```
p->studentID = 110112112;
```

2、指向结构体数组的指针

指向结构体数组的指针其实和指向一般数组的指针原理是一样的

```
STUDENT *pa = stu;
等价于
STUDENT *pa = stu[0];
```

其实也还能写成：

```
STUDENT *pa;
pa = stu;
```

四、在函数中传递结构体

1、用结构提的单个成员作为函数参数，向函数传递结构体的单个成员；

其实这种方法和一般的普通类型的变量做函数参数一毛一样，都是传值调用，对结构体原来的成员变量不会有变化。

2、用结构体变量来作为函数参数，向函数传递结构体完整结构；

用结构体变量做函数的实参，其实就是将结构体的完整结构和内容给被调函数，这种方法也是传值调用不会对原来的结构体造成影响。

3、用结构体指针和数组做函数参数，向函数传递结构体地址；

因为是传值调用，所以在函数中对形参的修改，会影响到实参结构体的内容。
其实在这里我要为大家添料：
结构体除了做函数形参外，还可以作为函数的返回值:

```
#include "stdio.h"
typedef struct date
{
    int year;
    int month;
    int day;
}DATE;

DATE func(DATE stu)      //结构体作为函数的返回值;
{
    stu.year = 2016;
    stu.month = 6;
    stu.day = 29;
    return stu;
}

int main()
{
    DATE p;
    p.year = 1996;
    p.month = 9;
    p.day = 21;
    printf("Before function call:%d/%2d/%2d\n", p.year, p.month, p.day);
    p = func(p);
    printf("After function call:%d/%2d/%2d\n", p.year, p.month, p.day);
}
```

结果：
Before function call:1996/ 9/21
After function call:2016/ 6/29

五、共用体

共用体，也叫联合，是将不同类型的数据组合到一起共同占用同一段内存的构造数据类型，定义方法与结构体类似，关键字是union；

1、共用体所占内存字节数的计算

```
#include "stdio.h"

union sample
{
    short i;
    char ch;
    float f;;
};

typedef  union sample SAMPLE;

int main()
{
    printf("bytes=%d\n", sizeof(SAMPLE));
```

```
15      return 0;
16  }
```

运算结果是：bytes=4
为啥会是4个呢，我们再来看一下，short占2个字节，char占一个字节，float占4个字节。我们前面不是说共用体是将数据组合放到一块内存上面的吗有足够大的内存来占据内存空间最大的那块内存，所以共用体所占内存空间的大小取决于其成员中占内存空间最大的那个成员变量。
由于同一块内存在每一瞬间只能存放一种数据类型的成员，也就是说同一时刻就只有一各成员是有意义的。每一瞬间起作用的成员就是最后一次被赋值
不能为共用体所有成员进行初始化，只能对第一个成员进行初始化，还有就是共用体不能做比较操作，也不能作为函数参数。
与结构体相似，可以通过成员选择运算符和指向运算符来访问共用体成员变量：

```
1  SAMPLE num;
2  num.i = 20;
```

```
1  union maritalState
2  {
3      int single;     //单身；
4      int married;    //已婚；
5      int divorce;    //离婚；
6  };
7
8
9  struct person
10 {
11     char name[10];
12     char sex;
13     int age;
14     union maritalState marital;     //婚姻状况；
15     int marryFlag;      //婚姻状况标记；
16 };
```

z这里我来简要说明一下marryFlag,这其实是一个标记项，所以我们可以这样定义：
marryFlag为1时，表明未婚；
marryFlag为2时，表明已婚；
....


想对作者说点什么


寂草堂： 对初学者很有帮助，感谢  (5个月前 #1楼)

上一页 1 下一页