结构体和共用体

//**********

允许用户自定义数据类型

出现了构造数据类型　　也称为复合数据类型

抽象数据类型　　是指这样的一种数据类型

其不再是单纯的一组数值，而是增加了更多的操作

结构体的定义和处理

为什么要使用结构体：

内存分配不集中。

内容的相关性质很差，赋值的时候容易发生错位

结构显得很零散，不便于进行管理和处理

//**********

//结构体是将不同类型的数据成员统一到一起的结构

//共用体下的各种的情况都是对立的

//仅仅只是声明了一种数据类型，定义了数据的组织形式，并未声明数据结构的变量

//先声明结构体模板，再定义结构体变量

```
struct node//结构体标签，定义结构体的必须内容
{



};//结束的标志

struct node node1;
```

//在声明的同时进行变量的声明过程

```
struct node//结构体标签，定义结构体的必须内容
{



}node1;//结束的标志，同时在此处声明了一个变量
```

//**
>>>>>>typedef 用于声明一个别名，通常使用大写的字母来完成这个内容。但也不是强制性的
例如typedef int INTER;
那么INTER就可以代替int的功能

例如

>>>>>>>>>>>>>重要
//声明一个结构体node，并定义一个结构体变量node1
struct node//结构体标签，定义结构体的必须内容
{

}node1;//结束的标志，同时在此处声明了一个变量

//声明一个结构体node，同时在此处将node起一个别名NODE
//也可以称作是一个结构体模板
typedef struct node//结构体标签，定义结构体的必须内容
{

}NODE;//结束的标志，同时在此处将node起一个别名NODE

>>>>使用的方法
>>>>typedef只是为一种已经存在的类型定义一个新的名字而已，并没有定义一种新的数据类型
struct node node1,node2;
以及
NODE node1,node2;

结构体的初始化操作:可以将初值置于花括号之内进行初始化。
如果内容有数组的话:数组也可以套一个花括号即可

例如：

NODE stu1 ={"wangzewei",12,1550422,{1,2,3,4}};
>>>注意：字符数组的处理上需要" ",字符的处理上需要'　　',数组的处理上需要加一个括号{};
>>>结构体可以嵌套另一个结构体，在声明的时候就可以使用

typedef struct node//结构体标签，定义结构体的必须内容
{
DATA birthday;//嵌套使用了另一个结构体的内容

}NODE;//结束的标志，同时在此处将node起一个别名NODE

//**********
结构体变量的引用过程
>>结构体名称.成员名称
如果存在结构体的嵌套的时候，就必须以
级联
的方式访问结构体成员，>>>就是通过成员选择运算符逐级查找到最底层的成员调用
例如：stu1.birthday.year = 1919;

C语言运行对具有相同结构体类型的变量进行整体的赋值操作和处理。
对常用的数据使用赋值运算符赋值即可，对字符串类型的结构体的成员使用字符串函数进行赋值即可。

###放在全部的函数之外的声明的内容称为：全局变量，所以任何一个函数都可以使用
###放在函数之内的声明的内容称为：局部变量，只能在该函数体内进行使用，离开这个函数，声明失效。
打印地址的方式####　　%p &shuju　　sizeof(s)
结构体变量的地址是结构体变量的所占的内存空间的首地址，
结构体成员的地址和其的结构体的设计和变量的类型

>>>不要自己进行猜测，建议使用sizeof进行处理
>>>字节数据是各个数据之和或大于这个数据：与结构体有关，还与计算机本身有关
because：为特定的数据引入了特殊的内存对齐操作。

不同的系统中的内存对齐操作的方式又有所不同，
为了满足处理器的对齐要求，可能会在较小的成员的后面进行补位。
为了补成最大的那个类型的格式，所以要进行补位

//**********
结构体数组的定义处理
NODE node[20];

初始化与初始化单个的数组的形式没有多少区别，
只是将其转化为数组的形式进行输入即可

//**********
共用体
共用体>>>>也称为联合
是将不同类型的数据组织在一起共同占有同一段内存的一种构造数据类型。
共用体和结构体很相似，只是关键字不同而已

共用体类型所占的内存空间取决于其成员中占内存空间最多的那个成员变量

//**********
结构体和共用体的差别，即其只能有一个值被赋值
//**********
在每一瞬间起作用的成员就是最后一次被赋值的成员
不能为共用体成员同时进行初始化处理，只能对一个成员进行初始化处理
同时共用体无法进行比较，也不能作为函数的参数
也就是其只有一个值可以使用
//**********
枚举类型
枚举数据类型的例子：
enum res{no,yes,none};
一般情况下的第一个为0，之后的依次进行递增

定义方式和变量的声明方式与结构体和共用体相似。
也可以定义枚举数组，也可以准确的定义每一个枚举常量的值
例如

enum res{no = -1,yes = 0,none = 1};
如果明确的声明某一个的值是多少，那么其的后面的值会依次
enum res{no = 1,yes,none};
输出的格式的时候使用
pintf("%d",answer)
beause::这个枚举的可以出现的类型是一个整形值，而不能作为字符串来进行输出。
代码结构
*/
//***
	//枚举类型
	//typedef enum ansuer2{no,yes,none}Enmm;:::::注意，已经出现的枚举项，不允许在后面出现在下一个枚举里面出现
	//	typedef enum ansuer1{ok = 1,ojbk = 2,xjbf = 3}Enum2;::::::在这种情况下按照声明的格式来进行确认其的值
	//typedef enum ansuer1{ok,ojbk = 2,xjbf}Enum2;::::::在这种情况下其的ok = 0,而xjbf的值变为了3
	//这将意味着在标定的数据之前的数据并非是本身的值减1，而是从0，开始依次进行上升。
	//从声明之后的数据向后的话，其的值是依次增加的

	//printf("%d\n",e1);
	//printf("%d\n",e2);
	//printf("%d\n",sizeof(e1));
	//printf("%d\n",sizeof(e2));
	//printf("%d\n",sizeof(int));
	//printf("%d\n",sizeof(Enum));
	//printf("%d\n",sizeof(Enum2));
	//这代表着着enum的值也是以最大的作为其的结构的总的内存容量,与共用体十分的相似
	//并且仅允许附一个值，与共用体也是十分的相似

//***
/*
#include<stdio.h>
int main()

```
{
        typedef enum ansuer{no,yes,none}Enum;
        //typedef enum ansuer2{no,yes,none}Enmm;:::::注意，已经出现的枚举项，不允许在后面出现在下一个枚举里面出现
        //      typedef enum ansuer1{ok = 1,ojbk = 2,xjbf = 3}Enum2;::::::在这种情况下按照声明的格式来进行确认其的值
        typedef enum ansuer1{ok,ojbk = 2,xjbf}Enum2;//::::::在这种情况下其的ok = 0,而xjbf的值变为了3
        //这将意味着在标定的数据之前的数据并非是本身的值减1，而是从0，开始依次进行上升。
        //从声明之后的数据向后的话，其的值是依次增加的
        Enum e1;
        Enum2 e2;
        e1 = no;
        e2 = xjbf;
        if(e1 == yes)
        {
        printf("ojbk\n");
        }
        else
        {
        printf("no\n");
        }

        printf("%d\n",e1);
        printf("%d\n",e2);
        printf("%d\n",sizeof(e1));
        printf("%d\n",sizeof(e2));
        printf("%d\n",sizeof(int));
        printf("%d\n",sizeof(Enum));
        printf("%d\n",sizeof(Enum2));
}
*/

//*************************************************************************
```

```
        //共用体
        //Sampe e1;
        //e1.a = 7;
        //e1.c = 'a';

        //只有e1.x才可以输出
        //其只能包含一个数据，在该段代码中e1.任何东西都是  ==  ‘a’
        //其的大小与整个结构体中所占最大的内存的成员有关，与其他的无关

        //可以对一个共用体变量直接通过另一个共用体变量进行赋值的处理
        //e2 = e1;
        //printf("%d\n",e1);
        //printf("%d\n",sizeof(e1));
        //printf("%d\n",sizeof(int));
        //printf("%d\n",sizeof(Sampe));

//*************************************************************************

/*
#include<stdio.h>
int main()
{

        typedef union sampe
        {
                int a;
                char c;
        }Sampe;

        Sampe e1;
        e1.a = 7;
        e1.c = 'a';
        if(e1.a == 7)
        {
                printf("ojbk1\n");//这句话没有用
```

```
        }
        if(e1.c == 'a')
        {
                printf("ojbk2\n");
        }
        else
        {
                printf("no\n");
        }

        //只有e1.x才可以输出
        //其只能包含一个数据，在该段代码中e1.任何东西都是  ==  ‘a’
        //其的大小与整个结构体中所占最大的内存的成员有关，与其他的无关

        printf("%d\n",e1);
        printf("%d\n",sizeof(e1));
        printf("%d\n",sizeof(int));
        printf("%d\n",sizeof(Sampe));
}*/

//*************************************************************************
        //结构体
        //可以对一个结构体变量直接通过另一个结构体变量进行赋值的处理
        //e2 = e1;
        //结构体和共用体都不可以进行直接对体的比较的过程处理
        //只能对结构体变量的成员变量进行判断是否相等的过程

        //printf("%d\n",e1);不可以用这种方法进行输出的处理
        //printf("%d\n",sizeof(e1));//==printf("%d\n",sizeof(Sampe));//8,相当于两个int之和
        //printf("%d\n",sizeof(int));//4,int占有4位
        //printf("%d\n",sizeof(char));//1,char只占有1位
        //printf("%d\n",sizeof(Sampe));//8,相当于两个int之和

//*************************************************************************
```

```
/*
#include<stdio.h>
int main()
{

        typedef struct sampe
        {
                int a;
                char c;
        }Sampe;

        Sampe e1,e2;
        e1.a = 7;
        e1.c = 'a';

        //可以对一个结构体变量直接通过另一个结构体变量进行赋值的处理
        e2 = e1;
        //结构体和共用体都不可以进行直接对体的比较的过程处理
        //只能对结构体变量的成员变量进行判断是否相等的过程

        //if (e1 == e2)
        //{
        //      printf("e1==e2\n");
        //}
        //else
        //{
        //      printf("e1!=e2\n");
        //}

        if(e1.a == 7)
        {
                printf("ojbk1\n");//此处这句话被执行
        }
        if(e1.c == 'a')
        {
```

```
            printf("ojbk2\n");
        }
        else
        {
            printf("no\n");
        }

        //printf("%d\n",e1);不可以用这种方法进行输出的处理
        printf("%d\n",sizeof(e1));//==printf("%d\n",sizeof(Sampe));//8,相当于两个int之和
        printf("%d\n",sizeof(int));//4,int占有4位
        printf("%d\n",sizeof(char));//1,char只占有1位
        printf("%d\n",sizeof(Sampe));//8,相当于两个int之和
}*/
```