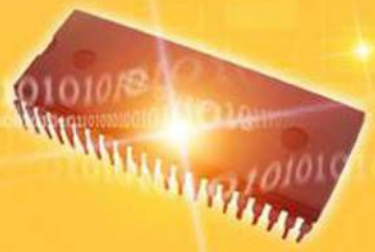
凌阳教育
www.sunplusedu.com
值得信赖的教育品牌

嵌入式系统工程师



Tel: 400-705-9680 , Email: edu@sunplusapp.com , BBS: bbs.sunplusedu.com

结构体、共用体、枚举

Tel: 400-705-9680 , Email: edu@sunplusapp.com , BBS: bbs.sunplusedu.com

大纲

- 结构体定义与应用
- 共用体定义与应用
- 枚举的定义与应用

大纲

- 结构体定义与应用
  - 结构体变量的定义与使用
  - 结构体数组
  - 结构体指针
  - 结构体的内存分配
  - 位段的使用
- 共用体应用
- 枚举应用

结构体变量的定义与使用

- 前面学过一种构造类型——数组:

  描述一组具有**相同类型**数据的有序集合，用于处理大量相同类型的数据运算

- 有时我们需要将**不同类型**的数据组合成一个有机的整体，以便于引用。**如:**

  一个学生有学号/姓名/性别/年龄/地址等属性

```
int num;
char name[20];
char sex;
int age;
int char addr[30];
```

显然单独定义以上变量比较繁琐，数据不便于管理

结构体变量的定义与使用

- 为了解决这个问题

  C语言中给出了另一种构造数据类型——结构体

  如：**一个学生有"学号/姓名/性别"等属性**

```
int num;
char name[20];
char sex;
int age;
int char addr[30];
```

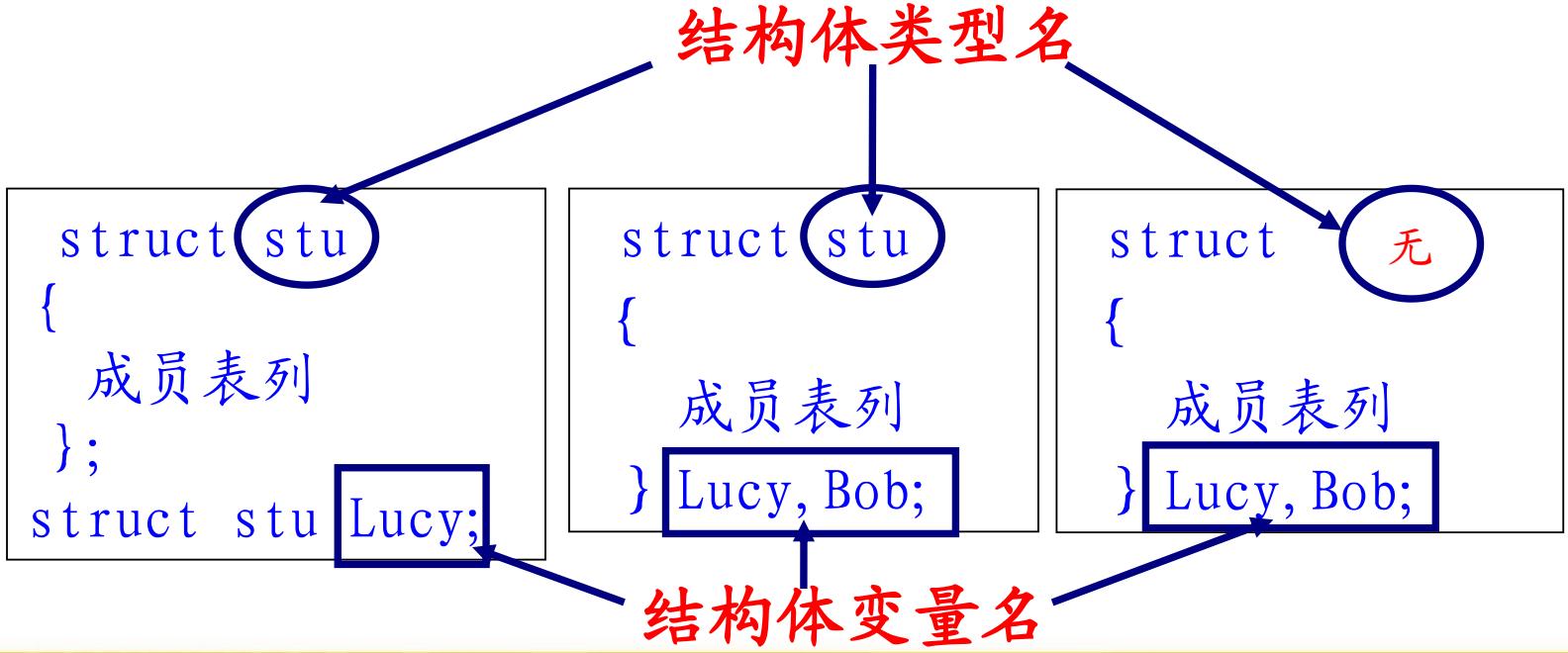
学生信息的一般表示法

```
struct stu
{
    int num;
    char name[20];
    char sex;
    int age;
    int char addr[30];
} student;
```

学生信息的结构体表示法

结构体变量的定义与使用

- 定义结构体变量的方式:
  - ① 先声明结构体类型再定义变量名
  - ② 在声明类型的同时定义变量
  - ③ 直接定义结构体类型变量（无类型名）



结构体变量的定义与使用

- 定义结构体变量的方式：
  - 结构体类型名：

    指定了一个**结构体类型**，它相当于一个**模型**，但其中并无具体数据，系统对之也**不分配**实际内存单元

  - 结构体变量名：

    实际分配空间—为了能在程序中使用结构类型的数据，应当定义结构体类型的变量，并在其中存放具体的数据

结构体变量的定义与使用

- 实际我们比较推荐以下写法

```
typedef struct student
{
    int  num;
    char name[20];
    char sex;
}STU;                    //使用typedef重定义一个结构体类型名
STU Lucy,Bob;     //使用STU去定义相应的对象
```

类型与变量分开定义，当一个结构体类型有多个文件需要使用时，可以将类型定义放在一个.h文件，需要使用的文件包含相应的头文件即可

结构体变量的定义与使用

注意事项:

① 结构体变量的成员引用必须**单独**使用:

结构体变量名.成员名

如: Lucy.num = 101;

scanf("%c",&Lucy.sex);

printf("%s",Lucy.name);

② 允许具有相同类型的结构变量可以相互赋值

多用于结构体成员整体操作(排序等)

Bob = Lucy;

③ 结构体可以定义的时候进行初始化

STU Lucy={12,"hello",'m'};

④ 允许嵌套定义结构体变量,成员引用多级引用

Lucy.birthday.month = 12;

结构体变量的定义与使用

01.struct_var.c

```
1  #include <stdio.h>
2  typedef struct stu
3  {
4      int num;
5      char name[100];
6      char sex;
7  }STU;
8  int main(void)
9  {
10     STU boy1,boy2,girl3 = {104,"LiLi",'F'};
11     printf("input boy1 num, name, sex and score\n");
12     scanf("%d %s %c",&boy2.num,boy2.name,&boy2.sex);
13     boy1 = boy2;
14     printf("boy1,Number=%d,Name=%s,Sex=%c\n",boy1.num,boy1.name,boy1.sex);
15     printf("girl3,Number=%d,Name=%s,Sex=%c\n",girl3.num,girl3.name,girl3.sex);
16     return 0;
17 }
```

大纲

- 结构体定义与应用
  - 结构体变量的定义与使用
  - 结构体数组
  - 结构体指针
  - 结构体的内存分配
  - 位段的使用
- 共用体应用
- 枚举应用

结构体数组

- 结构体数组
  - 一个结构体变量中可以存放一组数据：
    如一个学生的学号、姓名、成绩等数据
  - 如果有10个学生的数据，定义10个结构体变量很不方便，这时候我们可以使用**结构体数组**
  - 结构体数组与以前介绍过的数值型数组不同之处：
    每个数组元素都是一个结构体类型的数据
    它们都分别包括各个成员（分量）项

结构体数组

- 定义结构体数组与定义结构体变量是同样的方法;
- 以下以直接定义结构体数组为例:

```
struct student
{
    int  num;
   char name[10];
   int age;
}edu[2];
```

结构体数组的引用:

```
edu[0].num = 101;
strcpy(edu[0].name, "Lucy" );
edu[0].age = 24;
```

	num	name	age
edu[0]	101	Lucy	24
edu[1]	102	Bob	23

结构体数组

练习：定义一个结构体数组，求学生平均成绩

02.struct_avr.c

```
typedef struct student
{
    int   num;
    char  name[20];
    float score;
}STU;
STU edu[3]={
            {101,"Lucy",78},
            {102,"Bob",59.5},
            {103,"Tom",85},
          };
```

大纲

- 结构体定义与应用
  - 结构体变量的定义与使用
  - 结构体数组
  - 结构体指针
  - 结构体的内存分配
  - 位段的使用
- 共用体应用
- 枚举应用

结构体指针

- 结构体指针:
  - 指向结构体变量首地址的指针
  - 通过结构体指针即可访问该结构体变量

- 结构指针变量定义:

```
struct   结构体名     *结构体指针变量名
struct   student      *p = &Lucy;
```

结构体指针

- 利用结构体指针变量，就能很方便地访问结构体变量的各个成员
- 以下3种形式等价：

  ① Lucy.num = 101;

  ② (*p).num = 101;

  ③ p->num = 101;

  注意：“->” 称为指向运算符

- 结构体指针主要用于结构体变量传参以及后面的链表中

结构体指针

03.struct_point.c

```
1  #include <stdio.h>
2  struct student
3  {
4    int    num;
5    char   name[20];
6    float score;
7  }Lucy = {101,"Lucy",78},*p = NULL;
8
9  int main( )
10 {
11   p = &Lucy;
12   printf("num1=%d,  num2=%d\n",Lucy.num,p->num);
13   printf("name1=%s,name2=%s\n",Lucy.name,p->name);
14   printf("score1=%f,score2=%f\n",Lucy.score,(*p).score);
15   return 0;
16 }
```

大纲

- 结构体定义与应用
  - 结构体变量的定义与使用
  - 结构体数组
  - 结构体指针
  - 结构体的内存分配
  - 位段的使用
- 共用体应用
- 枚举应用

结构体的内存分配

例:

```
struct data {
    char c;
    int i;
};
struct data  stu;
printf("%d\n",sizeof(a)); // 5?    8?
```

以上程序会输出什么结果?

结构体的内存分配

- 假设变量stu存放在内存中的起始地址为0x00
  那么c的起始地址为0x00、i的起始地址为0x01
  变量stu共占用了5个字节
  对变量c访问：CPU只需要一个读周期
  对变量i访问：
  1. 首先CPU用一个读周期，从0x00处读取了4个字节(32位架构)，然后将0x01-0x03的3个字节暂存
  2. 再花一个读周期读取了从0x04-0x07的4字节数据，将0x04这个字节与刚刚暂存的3个字节进行拼接从而读取到成员变量i的值。
- **读取一个成员变量i，CPU却花费了2个读周期。**

结构体的内存分配

- 如果数据成员i的起始地址被放在了**0x04处**

  读取c成员，花费周期为1

  读取i所花费的周期也变成了1

  引入字节对齐可以避免读取效率的下降

  同时也浪费了3个字节的空间(0x01-0x03)。

- 结构体内部成员对齐是为了实现用**空间换取时间**

结构体的内存分配

- 结构体对齐，默认对齐原则：

1. 数据类型对齐值：

char型数据自身对齐值为1

short为2，int、float为4，double为8(windows)

解释：

char变量只要有一个空余的字节即可存放

short要求首地址能被2整除

int、float、double同理

2. 结构体的对齐值：

其成员中自身对齐值最大的那个值。

解释：

结构体最终对齐按照数据成员中最长的类型的整数倍

结构体的内存分配

分析以下结构体所占空间大小

04.sizeof_struct.c

```
struct student1
{
    char a;
    int b;
    short c;
}boy1;
```

结构体的内存分配

- 指定对齐原则:

  使用#pragma pack改变默认对其原则

  - 格式:

  #pragma pack (value)时的指定对齐值value。

  - 注意:

    1. value只能是: 1 2 4 8等
    2. 指定对齐值与数据类型对齐值相比取较小值

    如: 如果指定对齐值:

    设为1: 则short、int、float等均为1

    设为2: 则char仍为1, short为2, int 变为2

结构体的内存分配

分析以下结构体所占空间大小

04.sizeof_struct.c

```
#pragma pack (1)
struct student2
{
    char a;
    int c;
    short b;
}boy2;
```

大纲

- 结构体定义与应用
  - 结构体变量的定义与使用
  - 结构体数组
  - 结构体指针
  - 结构体的内存分配
  - 位段的使用
- 共用体应用
- 枚举应用

位段的使用

- **位段**
  - 信息在计算机的存取长度一般以字节为单位。
  - 有时存储一个信息**不必用一个或多个字节**

  例如:

  1. "真"或"假": 用0或1表示，只需一位即可。

  2. 在计算机用于过程控制、参数检测或数据通信领域时，控制信息往往只占一个字节中的一个或几个二进制位
- 怎样向一个字节的一个或几个二进制位赋值和改变它的值呢?

  (1) 利用前面讲过的位运算符: << >> & | ~ ^

  (2) 结构体中定义位段

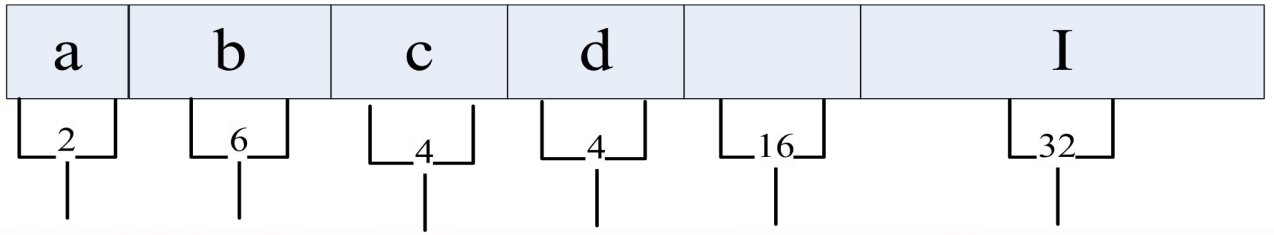
  利用位段可以减少存储空间并简化位操作

位段的使用

- C语言允许在一个结构体中以位为单位来指定其成员所占内存长度，以位为单位的成员称为“位段”或称“位域”
-

```
struct  packed_data{
    unsigned int a:2;
    unsigned int b:6;
    unsigned int c:4;
    unsigned int d:4;
    unsigned int i;
  } data;
```

其中a,b,c,d分别占2位，6位，4位，4位，i为整型，占4个字节



位段的使用

- **关于位段的定义与引用有几点重要说明:**

  ①对于位段成员的引用如下:

  data.a =2

  赋值时，不要超出位段定义的范围;

  如段成员a定义为2位，最大值为3,即（11）2

  所以data a =5，就会取5的低两位进行赋值

  ② 位段**成员**的类型必须指定为整形或字符型

  ③一个位段必须存放在一个存储单元中，不能跨两个单元

  第一个单元空间不能容纳下一个位段，则该空间不用，而从下一个单元起存放该位段

  ④位段的长度不能大于存储单元的长度

位段的使用

⑤ 如一个段要从另一个存储单元开始，可以定义：

```
unsigned a:1;
unsigned b:2;
unsigned  :0;
unsigned c:3; (另一个单元)
```

由于用了长度为0的位段，其作用是使下一个位段从下一个存储单元开始存放

将a、b存储在一个存储单元中，c另存在下一个单元

⑥ 可以定义无意义位段，如：

```
unsigned a:  1;
unsigned   :  2;
unsigned b:  3;
```

位段的使用

05.struct_bit.c

```
#include <stdio.h>
typedef struct
{
   unsigned int a:1;
   unsigned int b:2;
   unsigned int c:3;
} M;
```

```
int main()
{
    M k;
    k.a = 1;
    k.b = 1;
    k.c = 4;
    return 0;
}
```

大纲

- 结构体定义与应用
  - 结构体变量的定义与使用
  - 结构体数组
  - 结构体指针
  - 结构体的内存分配
  - 位段的使用
- 共用体应用
- 枚举应用

共用体应用

- 共用体
  - 在进行某些算法的时候，需要使几种不同类型的变量存到同一段内存单元中，几个变量所使用空间相互重叠
  - 这种几个不同的变量共同占用一段内存的结构
    在C语言中，被称作“共用体”类型结构
  - 共用体所有成员占有同一段地址空间

共用体应用

- 共用体的定义:
  - 与结构体非常类似，有三种方式，我们推荐下面这种

```
typedef union data{
   short int i;
   char  ch;
   float  f;
}DATA;
DATA  a,b;
```


```
a.i(引用共用体变量中的整型变量i)
a.ch(引用共用体变量中的字符变量ch)
a.f(引用共用体变量中的实型变量f)
```

共用体应用

- 共用体特点

① 同一内存段可以用来存放几种不同类型的成员，但每一瞬时只有一种起作用

② 共用体变量中起作用的成员是最后一次存放的成员，在存入一个新的成员后原有的成员的值会被覆盖

③ 共用体变量的地址和它的各成员的地址都是同一地址

④ 共用体变量的初始化

union data a={123}; 初始化共用体为第一个成员

共用体应用

- 共

```
1  #include<stdio.h>
2  //共用体
3  union xx
4  {
5      long int x;
6      char y;
7      int z;
8  }a={10};
9
10 int main()
11 {
12     a.x=5;
13     a.y=6;
14     a.z=15;
15     printf("%d\n",a.x+a.y);
16     return 0;
17 }
```

06.union.c

大纲

- 结构体定义与应用
  - 结构体变量的定义与使用
  - 结构体数组
  - 结构体指针
  - 结构体的内存分配
  - 位段的使用
- 共用体应用
- 枚举应用

枚举应用

- 枚举

  将变量的值一一列举出来，变量的值只限于列举出来的值的范围内

- 枚举类型定义:

```
enum   枚举名
{
    枚举值表
};
```

在枚举值表中应列出所有可用值,也称为枚举元素
枚举变量仅能取枚举值所列元素

枚举应用

- 例子:
- **定义枚举类型 week**

```
enum week                    //枚举类型
{
    mon, tue, wed, thu, fri, sat,sun
};
enum week  workday、weekday;//枚举变量
workday与weekday只能取sun....sat中的一个
```

结果:

```
workday = mon;   //正确
weekday = tue;   //正确
workday = abc;   //错误，枚举值中没有abc
```

枚举应用

- **注意:**

① 枚举值是常量,不能在程序中用赋值语句再对它赋值

例如: sun=5; mon=2; sun=mon; 都是错误的.

② 枚举元素本身由系统定义了一个表示序号的数值

从0开始顺序定义为0, 1, 2...

如在week中, sun值为0, mon值为1, ...,sat值为6

③ 可以改变枚举值的默认值: 如

```
enum week                    //枚举类型
{
    mon=3, tue, wed, thu, fri, sat,sun
};
mon=3   tue=4,以此类推
```

➢ 例

07.enum.c

```c
#include <stdio.h>
enum weekday
{
    sun=2,mon,tue,wed,thu,fri,sat
} a,b,c;
enum bool
{
    flase,true
} bl;
int main()
{
    a=sun;
    b=mon;
    c=tue;
    printf("%d,%d,%d",a,b,c);
    bl=true;
    if( bl == 1)
        printf("bl为真\n");
    return 0;
}
```

凌阳教育
www.sunplusedu.com

值得信赖的教育品牌

Tel: 400-705-9680 , Email: edu@sunplusapp.com , BBS: bbs.sunplusedu.com