本小节内容

结构体指针
typedef 的使用

1 结构体指针

一个结构体变量的指针就是该变量所占据的内存段的起始地址。可以设置一个指针变量，用它指向一个结构体变量，此时该指针变量的值是结构体变量的起始地址。指针变量也可以用来指向结构体数组中的元素，从而能够通过结构体指针快速访问结构体内的每个成员。

下面来看例 1.1。

【例 1.1】结构体指针的使用。

```
#include <stdio.h>



//结构体指针

struct student{

    int num;

    char name[20];

    char sex;

};



int main()

{

    struct student s={1001,"wangle",'M'};

    struct student sarr[3]={1001,"lilei",'M',1005,"zhangsan",'M',1007,"lili",'F'};

    struct student *p;   //定义结构体指针
```

```
    int num;
    p=&s;
    printf("%d %s %c\n",p->num,p->name,p->sex);
    p=sarr;
    printf("%d %s %c\n",(*p).num,(*p).name,(*p).sex);   //方式一获取成员
    printf("%d %s %c\n",p->num,p->name,p->sex);   //方式二获取成员
    printf("--------------------------------\n");
    p=p+1;
    printf("%d %s %c\n",p->num,p->name,p->sex);
    return 0;
}
```

由例 1.1 可以看到，p 就是一个结构体指针，可以对结构体 s 取地址并赋给 p，这样借助成员选择操作符，就可以通过 p 访问结构体的每个成员，然后进行打印。我们知道数组名中存储的是数据的首地址，所以可以将 sarr 赋给 p，这样就可以通过两种方式访问对应的成员。

使用(*p).num 访问成员为什么要加括号呢？原因是“.”成员选择的优先级高于“*”（即取值）运算符，所以必须加括号，通过*p 得到 sarr[0]，然后获取对应的成员。

2 typedef 的使用

前面定义结构体变量时使用的语句是 struct student s，以这种方式来定义结构体变量有些麻烦，即每次都需要写 struct student。那么有没有简单一些的定义方式呢？答案是肯定的，可以选择使用 typedef 声明新的类型名来代替已有的类型名，请看例 2.1。

【例 2.1】typedef 的使用。

```
#include <stdio.h>

//结构体指针
typedef struct student{
        int num;
        char name[20];
        char sex;
```

```
}stu,*pstu;

typedef int INTEGER;

int main()
{
        stu s={1001,"wangle",'M'};
        pstu p;
        INTEGER i=10;
        p=&s;
        printf("i=%d,p->num=%d\n",i,p->num);
        return 0;
}
```

使用 stu 定义结构体变量和使用 struct student 定义结构体变量是等价的；使用 INTEGER 定义变量 i 和使用 int 定义变量 i 是等价的；pstu 等价于 struct student*，所以 p 是结构体指针变量。