

第十章 构造数据类型及其应用

本章学习目标：

- ✓ 掌握枚举类型、结构体类型、联合体类型的定义
- ✓ 了解枚举类型、结构体类型、联合体类型及其变量的区别
- ✓ 了解枚举类型、结构体类型、联合体类型的使用背景
- ✓ 掌握枚举成员的编号规则
- ✓ 掌握结构体、联合体在内存中的存储规则
- ✓ 掌握结构体数组的使用。
- ✓ 掌握链表的操作细节。

10.1 实践题

一、链表的构造及遍历

实验目的

1. 理解链表的概念。
2. 掌握链表的创建。
3. 掌握链表的遍历及其元素访问方法。

实验步骤

步骤 1：新建工程，命名 listDemo；

步骤 2：创建一头文件，命名 list.h；在该文件中设计结点结构：

```
#include<stdio.h>

struct Node
{
    int data;
    struct Node * next;
};
```

步骤 3：创建一源文件，命名 list.c；在该文件中编写测试代码：

```
#include "list.h"

int main()
{
```

```

        printf("test\n");
        return 0;
    }

```

步骤 4：在测试代码中创建结点、链接结点之后，链表创建成功。代码如下：

```

struct Node x,y,z,m,n;
struct Node *head;
x.data = 5;
y.data = 4;
z.data = 3;
m.data = 2;
n.data = 1;
x.next = &y;
y.next = &z;
z.next = &m;
m.next = &n;
n.next = 0;//0为结束标识
head = &x; //链表创建成功，head 为头指针，该链表的第一个结点为 x
          ( 值
//为 5 )，然后依次为 y，z，m，n。

```

步骤 4：对链表结点进行输出式访问，代码如下：

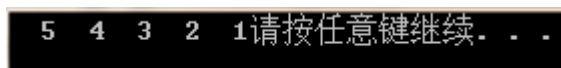
```

while(head!=0)//将链表结点数据输出
{
    printf("%3d",head->data);
    head = head->next;
}

```

实验结果/结论

1. 实验结果



```

5 4 3 2 1请按任意键继续...

```

2. 实验结论

- ✓ 链表的本质就是结点通过指针（链）来维系其前后关系，通过一个指针即可遍历整个链表
- ✓ 头指针保存的是第一元素结点的地址（指向第一个元素结点），其可以作为整个链表的标识，一般在操作链表时会重新定义指针，而不应该直接使用头指针。

- ✓ 该例中的链表结点是在栈区创建的，每个结点都有名字，如果链表结点是在堆区创建，则每个结点是没有名字的，此时，每个结点都是通过其前一个结点来定位，头指针是链表的唯一标识。
- ✓ 本例链表中的结点均是在栈区创建的，读者可以在试着在堆去创建链表：编写一函数，该函数有一个整形参数 n，用以创建 n 个结点的链表，并将第一个结点的地址作为返回值返回。

二、链表结点的插入、删除

实验目的

1. 掌握链表结点的插入过程。
2. 掌握链表结点的删除过程。
3. 理解链表结点（数据元素）之间关系的维系依据。

实验步骤

步骤 1：在实验一代码中，创建一结点：`struct Node o;`

步骤 2：若将该结点的数据赋值为 7，将之插在 4、3 之间，则需要如下代码：

```
o.data = 7;  
o.next = &z;  
y.next = &o;
```

步骤 3：运行测试代码，可以得出输出结果，如图所示：

```
5 4 7 3 2 1请按任意键继续. . .
```

步骤 4：在遍历代码之前加代码：`y.next = &z;`

步骤 5：运行测试代码，可以得出输出结果，如图所示：

```
5 4 3 2 1请按任意键继续. . .
```

实验结果/结论

1. 实验结论

- ✓ 在进行链表结点插入时，若是结点没有名字（在堆区创建的结点），切记指针的修改顺序。
- ✓ 在进行链表结点删除时，若是结点没有名字（在堆区创建的结点），有两点特别要注意，第一，要从头开始找，知道找到被删除结点的前一个结点；第二，要用一个新指针将被删除结点的地址保存起来，这样可以在删除后释放该结点。

三、 结构体数组使用

实验目的

1. 掌握结构体数组的定义方法
2. 掌握结构体数组的使用
3. 理解结构体数组与普通数组的相同本质

实验步骤

问题：有十个学生信息存储在计算机中，找出考试成绩最高的学生将其信息输出出来，
假设学生信息包括整型的学号、字符数组型姓名、浮点型成绩。

步骤 1：新建工程，命名 structArrDemo；

步骤 2：创建一头文件，命名 structArr.h；在该文件中设计结点结构：

```
#include <stdio.h>
```

```
struct student
```

```
{
```

```
    int NO;
```

```
    char name[10];
```

```
    float score;
```

```
};
```

步骤 3：创建一源文件，命名 structArr.c；在该文件中编写测试代码：

```
#include "structArr.h"
```

```
int main()
```

```
{
```

```
    printf("test\n");
```

```
    return 0;
```

```
}
```

步骤 4：在测试代码中创建结构体数组并附初值，然后查找分数最高的学生。代码如下：

```
struct student stuArr[10]={1,"赵一",93},{1,"钱二",52},
```

```
{1,"孙三",86},{1,"李四",78},{1,"周五",93},{1,"郑六",39},
```

```
{1,"冯七",88},{1,"陈八",99},{1,"楚九",77},{1,"魏十",69}};
```

```
int i = 0,index=0;
```

```

while(i<10)

{

    if(stuArr[index].score<stuArr[i].score)

        index = i;

    i++;

}

printf("学号 : %5d\\n姓名 : %s\\n分数 : %f\\n",
stuArr[index].NO,stuArr[index].name,stuArr[index].score);

```

实验结果/结论

1. 实验结果

✓

```

学号:      1
姓名: 陈八
分数: 99.000000
请按任意键继续. . .

```

2. 实验结论

- ✓ 结构体数组与普通数组没有本质区别，仅仅是数组的每个元素是结构体而已，当然操作数组元素就相当于操作结构体变量。
- ✓ 重做实验：对这十个学生信息按成绩进行排序（从低到高）。

10.2 理论题

A 类

一、选择题

1. 下列不属于自定义数据类型的是（ ）。
 - A. 结构体
 - B. 整型
 - C. 联合
 - D. 枚举
2. 下列关于枚举的叙述中不正确的是（ ）。
 - A. 枚举变量只能取对应枚举类型的枚举元素表中的元素

- B. 可以在定义枚举类型时对枚举元素进行初始化
 - C. 枚举元素表中的元素有先后次序,可以进行比较
 - D. 枚举元素的值可以是整数或字符串
3. 以下关于 typedef 的叙述正确的是 ()。
- A. 用 typedef 既定义各种类型名,也可以用来定义变量
 - B. 用 typedef 可以增加新类型
 - C. 用 typedef 只是将已存在的类型用一个新的名字来代表
 - D. typedef 与 #define 的作用是相同的
4. 下列关于结构体叙述不正确的是 ()。
- A. 结构体成员可以是本身结构体类型
 - B. 结构体类型的成员名可以与程序中的变量名相同
 - C. 结构体类型的成员可以是一个结构体变量
 - D. 可以单独使用结构体变量中的成员,它的作用相当于普通变量
5. 下列可以出现在结构体成员列表中的是 ()。
- A. 本身结构体类型变量
 - B. 本身结构体指针类型变量
 - C. 函数
 - D. 静态变量
6. 对于结构体类型变量在程序执行期间描述正确的是 ()。
- A. 所有成员一直驻留在内存中
 - B. 只有一个成员驻留在内存中
 - C. 部分成员驻留在内存中
 - D. 没有成员驻留在内存中
7. 在 C 语言中,需要把一些属于不同类型的数据作为一个整体来处理时常用 ()。
- A. 基本数据类型
 - B. 数组
 - C. 指针
 - D. 结构体类型
8. 在声明一个联合体变量时,系统分配给它的存储空间是 ()。
- A. 该联合体中第一个成员所需存储空间补齐后大小
 - B. 该联合体中占用最大存储空间的成员所需存储空间补齐后大小
 - C. 该联合体中最后一个成员所需的存储空间补齐后大小
 - D. 该联合体中所有成员所需存储空间的总和补齐后大小
9. 在声明一个结构体变量时,系统分配给它的存储空间是 ()。
- A. 该结构体变量中第一个成员所需存储空间补齐后大小
 - B. 该结构体变量中最后一个成员所需存储空间补齐后大小
 - C. 该结构体变量中占用最大存储空间的成员所需存储空间补齐后大小
 - D. 该结构体变量中所有成员所需存储空间的总和补齐后大小
10. 以下对 C 语言中联合体类型数据的叙述正确的是 ()。
- A. 可以对联合体变量名直接赋值
 - B. 一个联合体变量中不能同时存放其所有成员
 - C. 一个联合体变量中可以同时存放其所有成员
 - D. 联合体类型定义中不能出现结构体类型的成员

二、综合题

1. 定义一个结构体 STUDENT，表示一个学生的信息，其中包括学生的姓名（字符数组）、学号（short 型）、数学成绩（double 型）、英语成绩（double 型）、总成绩（double 型）。编写一个程序，实现输入 10 个学生姓名学号信息、数学英语成绩，并自动计算每个学生总成绩的功能。
2. 利用题目 2 所定义的结构体，写一函数实现对长度为 len 的结构体数组按总成绩进行排序（降序）的功能。
3. 利用题目 2 所定义的结构体，写一函数实现对头指针为 head 的链表进行结点统计的功能。

B 类

一、选择题

1. 有如下定义，则联合体变量 u1 在内存中占用的字节数为（ ）。

```
union data
{
    int a;
    char b;
    char c[5];
} u1;
```

- A. 5
- B. 8
- C. 10
- D. 12

2. 已知如下结构体类型,则下列结构体变量声明错误的是（ ）。

```
typedef struct student
{
    char acName[20];
    int nAge;
} Student ;
```

- A. Student s1;
 - B. struct studentn s1 = {0};
 - C. student s1 = {"zhangsang", 21};
 - D. Student s1 = {"zhangsang"} ;
3. 以下枚举类型的定义中正确的是（ ）。
 - A. enum a{one = 9, two = -1, three};
 - B. enum a = {one, two, three};
 - C. enum a = {"one", "two", "three"};
 - D. enum a{"one", "two", "three"};
 4. 如下程序的输出结果为（ ）。

```
struct point
{
```

```

    int x ;
int y ;
int z ;
} ;
struct point p1[] = {0,1,2,3,4,5} ;
printf("%d\n", p1[0].x + p1[1].y) ;

```

- A. 3
- B. 4
- C. 5
- D. 6

5. 有如下结构体定义，那么如果要对 zhangsan 的出生年份进行赋值的话正确的赋值语句是（ ）。

```

struct date { int month; int day; int year;};
struct person
{
char name[20];
struct date birthday;
} zhangsan;

```

- A. birthday.year = 1995 ;
- B. person.birthday.year = 1995 ;
- C. zhangsan.year = 1995 ;
- D. zhangsan.birthday.year = 1995 ;

6. 有如下联合体定义，那么关于联合体变量初始化正确的是（ ）。

```

union data
{
char *p;
int n;
};

```

- A. union data u1= "zhangfei";
- B. union data u1 = {"zhangfei"};
- C. union data u1 = 15;
- D. union data u1 = {15};

7. 对如下程序叙述错误的是（ ）。

```

struct student
{
char acName[20];
int nAge;
} s1;

```

- A. struct 是结构体类型的关键字
- B. struct student 是用户定义的结构体类型

- C. s1 是用户定义的结构类型名
D. acName 和 nAge 都是结构成员名

8. 有如下程序，则以下对结构体变量中成员的引用方式不正确的是（ ）。

```
struct book
{
    char acName[20];
    int nPageCount;
} b1, *pB2;
pB2 = &b1;
```

- A. b1.acName
B. pB2→nPageCount
C. (*pB2).nPageCount
D. *p.acName

9. 有如下程序，则此程序的输出结果是（ ）。

```
enum team
{
    a,
    b = 4,
    c = a - b,
    d = c + 10
};
printf("%d,%d,%d,%d\n", a, b, c, d);
```

- A. 0,4,5,15
B. 0,4,0,10
C. 0,4,-4,6
D. 0,4,4,14

10. 有如下程序，则此程序的输出结果是（ ）。

```
#include <stdio.h>
int main()
{
    union
    {
        short int i[2];
        long k;
        char c[4];
    } r, *s = &r;
    s->i[0] = 0x39;
    s->i[1] = 0x38;
    printf("%lx\n", s->k);
    return 0;
```

}

- A. 390038
- B. 380039
- C. 3939
- D. 3838

二、综合题

1. 编写一程序能够输入若干本图书的信息并按出版年月排序后进行输出，每本图书包含书名（booktitle）、作者（author）、出版年月（date）、出版社（publishunit）信息。请用一个结构体数组存放图书信息，图书结构体类型已给出，请编写完成程序。

```
struct Data {
    int year;
    int month;
    int day;
};

struct book
{
    char booktitle[50];    // 书名
    char author[20];       // 作者
    struct Data data;      // 出版日期
    char publishunit[100]; // 出版社
};
```

2. 已知一个带有表头结点的单链表，结点结构如下，假设该链表只给出了头指针 list。在不改变链表的前提下，请设计一个尽可能高效的函数，实现“查找链表中倒数第 k 个位置上的结点（k 为正整数）。若查找成功，算法输出该结点的 data 值，并返回 1；否则，只返回 0”的功能。

```
struct Node {
    int data;
    struct Node *next;
};
```

3. 已知链表结构如 2 题。试写一函数实现“将偶数元素放在前面，奇数元素放在后面”的功能。

本章答案

A 类

一、选择题

1-5: BDCAB 6-10: ADBDB

二、综合题

B 类

一、选择题

1-5: BCABD 6-10: BCDCA

二、综合题

1.

```
#include <stdio.h>
#include <stdlib.h>

struct Data {
    int year;
    int month;
    int day;
};

struct book
{
    char booktitle[50];    // 书名
    char author[20];       // 作者
    struct Data data;      // 出版日期
    char publishunit[100]; // 出版社
};

int main(int argc, char ** argv)
{
    int i, j, n;           // 循环变量
    struct book * library; // 存放图书信息的内存地址指针
    struct book temp;      // 图书临时变量

    printf("请输入要处理的图书数量: \n");
    scanf("%d", &n);
    library = (struct book *)malloc(n * sizeof(struct book));

    for(i=0; i<n; ++i)
    {
        printf("请输入第%d 本书的信息: \n", i + 1);
        printf("书名: ");
        fflush(stdin);
        scanf("%s", &library[i].booktitle);
```

```

        printf("作者: ");
        fflush(stdin);
        scanf("%s", &library[i].author);
        printf("出版年月: ");
        fflush(stdin);
        scanf("%d.%d.%d",
                &library[i].data.year,
                &library[i].data.month,
                &library[i].data.day);
        printf("出版社: ");
        fflush(stdin);
        scanf("%s", &library[i].publishunit);
    }

    for(i=0; i<n-1; ++i)
    {
        for(j=i+1; j<n; ++j)
        {
            if((library[i].data.year < library[j].data.year)
                || (library[i].data.year == library[j].data.year
                    && library[i].data.month < library[j].data.month)
                || (library[i].data.year == library[j].data.year
                    && library[i].data.month == library[j].data.month
                    && library[i].data.day < library[j].data.day))
            {
                temp = library[i];
                library[i] = library[j];
                library[j] = temp;
            }
        }
    }

    printf("\n 排序后的图书信息: ");
    for(i=0; i<n; ++i)
    {
        printf("\n 书名:   %s\n 作者:   %s\n 出版年月:   %d.%d.%d\n 出
        版社:   %s\n",
                library[i].booktitle,
                library[i].author,
                library[i].data.year,
                library[i].data.month,
                library[i].data.day,
                library[i].publishunit);
    }

```

```

        free(library);
        return 0;
    }
2.
int LocateElement(struct Node *list, int k)
{
    struct Node * P1;
    P1 = list->next;
    P = list;
    i = 1;
    while(P1)
    {
        p1 = p1->next;
        i++;
        if(i > k)    //如果 i>k , 则 P 也往后移动
            p = p->next;
    }
    if(p == list)    //说明链表就没有 k 个节点
        return 0;
    else
    {
        printf("%d\n",p->data);
        return 1;
    }
}

```