程序设计语言与方法(C语言)

第十二章 结构体和共用体

个人信息数据表示例

学号	姓名	性别	出生年	数学	英语	计算机原理	程序设计
100310121	王刚	男	1991	72	83	90	82
100310122	李小明	男	1992	88	92	78	78
100310123	王丽红	女	1991	98	72	89	66
100310124	陈莉莉	女	1992	87	95	78	90
…							

- 正文级别 4
- 正文级别 5

问题

- 如何完成个人信息的表示?
  - 个人信息（包含多个数据项）
    - 姓名
    - 年龄
    - 性别
    - ……
  - 表示
    - 按项逐个表示，使用时会如何?
    - 能否整体表示呢?
    - 如何表示多个人的信息呢?

多个数据
每个数据的类型可能是不一样的!

个人信息数据表示例(续)

- 按项逐个表示个人信息，数组表示多人信息

```
long  studentId[30];          /* 学号 */
char  studentName[30][10];    /* 姓名 */
char  studentSex[30];         /* 性别 */
int   yearOfBirth[30];        /* 出生年 */
int   scoreMath[30];          /* 数学课的成绩 */
int   scoreEnglish[30];       /* 英语课的成绩 */
int   scoreComputer[30];      /* 计算机原理课的成绩 */
int   scoreProgramming[30];   /* 程序设计课的成绩 */
```

学号	姓名	性别	出生年	数学	英语	计算机原理	程序设计
100310121	三 刚	男	1991	72	83	90	82
100310122	李小胜	男	1992	88	92	78	78
100310123	王丽红	女	1991	98	72	88	66
100310124	陈莉莉	女	1992	87	95	78	90
…							

个人信息数据表示例(添加数据)

```
long  studentId[30] = {100310121, 100310122, 100310123, 100310124};
char  studentName[30][10] = {"王刚", "李小明", "王丽红", "陈莉莉"};
char  studentSex[30] = {'M', 'M', 'F', 'F'};
int   yearOfBirth[30] = {1991, 1992, 1991, 1992};
int   scoreMath[30] = {72,88,98,87};
int   scoreEnglish[30] = {83,92,72,95};
int   scoreComputer[30] = {90,78,89,78};
int   scoreProgramming[30] = {82,78,66,90};
```

■ 正文级别 5

学号	姓名	性别	出生年	数学	英语	计算机原理	程序设计
100310121	王 刚	男	1991	72	83	90	82
100310122	李小明	男	1992	88	92	78	78
100310123	王丽红	女	1991	98	72	89	66
100310124	陈莉莉	女	1992	87	95	78	90
…							

个人信息数据表示例(存在问题)

- 内存分配分散，访问效率不高
- 各项数据缺乏关联性（使用不便）
- 对数组赋值易错位（导致数据错误）
- 程序处理时需要特别注意处理每个数据项，以实现全部数据的管理

能否将这些数据统一（或整体）表示呢?

结构体(数据类型)

- 正文级别 1

```
struct student
{
    long  studentID;              /* 学号 */
    char  studentName[10];        /* 姓名 */
    char  studentSex;             /* 性别 */
    int   yearOfBirth;            /* 出生年 */
    int   scoreMath;              /* 数学课的成绩 */
    int   scoreEnglish;           /* 英语课的成绩 */
    int   scoreComputer;          /* 计算机原理课的成绩 */
    int   scoreProgramming;       /* 程序设计课的成绩 */
};
```

数据项

域、字段、成员、项等（field)

与数组有哪些差异?

结构体(变量定义)

- 使用预先定义的结构体类型定义（声明）变量
  - struct student aStu;
  - 类型定义的同时，定义结构体变量
- 直接定义结构体变量（不定义结构体类型）

```
struct  student
{
    long  studentID;
    char  studentName[10];
    char  studentSex;
    int   yearOfBirth;
    int   score[4];
} stu1;
```

```
struct
{
    long  studentID;
    char  studentName[10];
    char  studentSex;
    int   yearOfBirth;
    int   score[4];
} stu1;
```

使用类型定义(类型名称)

- 不预先定义结构体类型
  - typedef struct student Student;

- 预先定义结构体类型
  - typedef struct {
  - ……
  - } STUDENT;

  - typedef struct student {
  - ……
  - } STUDENT;

struct student stu1,stu2;

STUDENT stu1,stu2;

初始化结构体变量

- 正文级别 1

stu1:

100310121	王刚	M	1991	72	83	90	82

- 正文级别 3

struct student stu1={100310121, “王刚”, ‘M’, 1991, 72, 83, 90, 82};

- 正文级别 4

STUDENT stu1={100310121, “王刚”, ‘M’, 1991, 72, 83, 90, 82};

- 正文级别 5

操作(访问)结构体变量

- “.”运算符
  - 结构体变量名.域名
  - stu1.studentID
- 赋值
  - 只能按项赋值
    - stu1.studentID = 100310121;
- 取值
  - 只能按项取值（变量定义时除外）
    - printf(“%s”,stu1.studendName);

结构体变量赋值和运算

- 赋值
  - 试一试
  - 试一试包含指针变量的情况
- 运算
  - 结构体变量没有可执行的任何运算功能

结构体的内存块大小

- 结构体内存块大小
  - 各项数据的内存块大小的总和
- sizeof同样可用

结构体数组

- struct student Students[10];
- Students[1].studentID = 100310121;
- printf(“%d”, students[1].studentID);

结构体指针

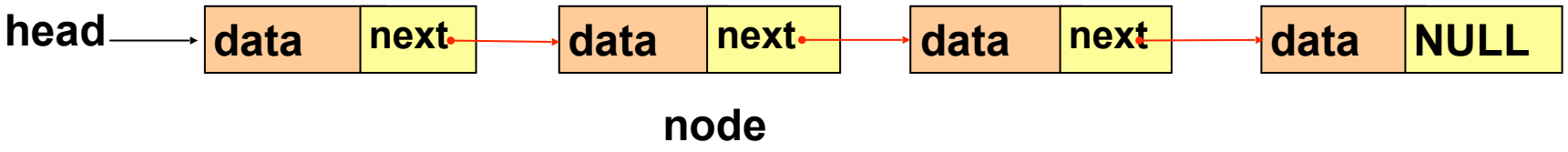
- 指向结构体变量的指针变量
- 试一试

动态数据结构——链表

- 存储个数不确定的数据。
  - 数据在哪里?
  - 下一个数据在哪里?
  - 节点（数据和一下个数据的表示方式）
- 单向链表
- 双向链表
- 循环链表

单向链表

- 线性表的链式存储结构
- struct link {
- int data;
- struct link *next;
- }



单向链表(操作)

- 空链表
- 创建节点
  - 申请节点的存储空间
- 计算链表的长度
  - 一次遍历求链表的长度
- 插入一个节点
- 删除一个节点
  - 释放节点的存储空间
- 搜索一个节点

向函数传递结构体变量

- 试一试
  - 传值
  - 传地址（指针）

返回结构体变量

- 试一试
  - 返回值
  - 返回指针

嵌套结构体

- 课程成绩
  - 数据表示
    - 课程名称
    - 成绩
  - 结构体类型定义
    - struct score {
      - char name[48];
      - float score;
    - };

- 出生日期
  - 数据表示
    - 年
    - 月
    - 日
  - 结构体类型定义
    - struct Date {
      - short year;
      - short month
      - short day ;
    - };

嵌套结构体(续)

- 重新定义学生成绩类型
  - struct student {
  - long studentID;
    - ……
    - struct Date Birthday;
    - struct score Scores[4];
  - };

实验

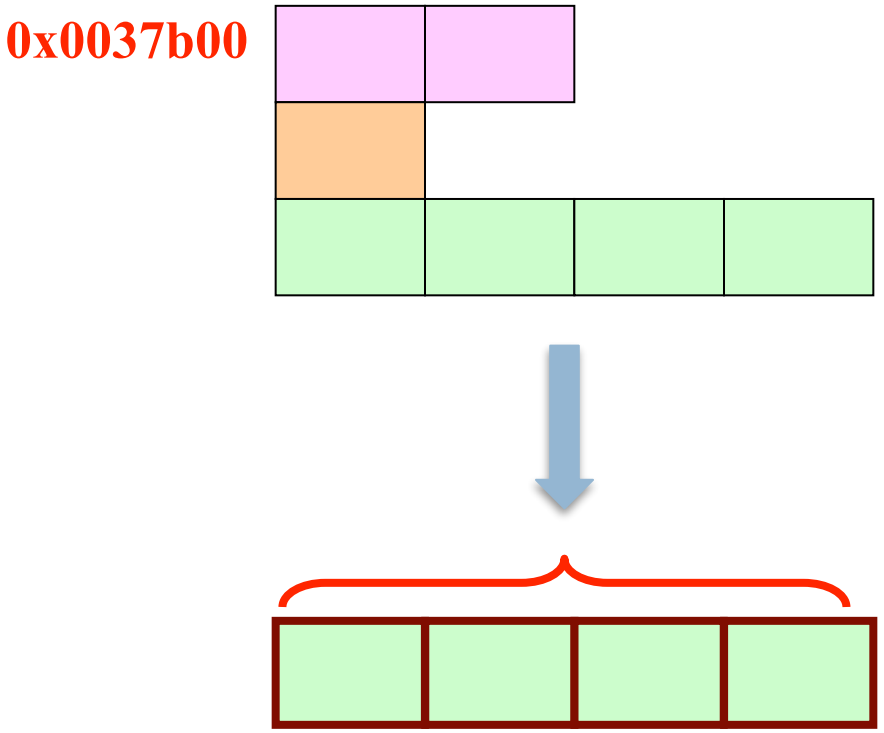
- 从键盘输入不定个数的学生的程序设计课程的成绩
  - 1. 求成绩最高和最低的学生的姓名和成绩；
  - 2. 删除所有不及格的学生的信息。

共用体

- 联合 union
- 将不同类型的数据组织在一起共同占用同一段内存的一种数据类型
- 共用体类型所占内存空间的大小取决于其成员中占内存空间最多的那个成员变量。
- 同一个数据，按不同的类型来访问

共用体示例

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



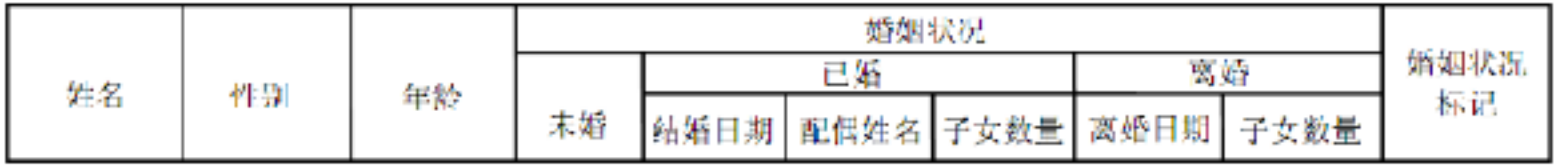
- union sample s;
- union sample s = 23;

共用体使用注意事项

- 同一内存单元在每一瞬时只能存放其中一种类型的成员
- 起作用的成员是最后一次存放的成员，不能作为函数参数
- 不能进行比较操作，只能对第一个成员初始化

共用体示例2

- 正文级别 1




```
24      struct person                          /* 定义职工个人信息结构体类型 */
25      {
26          char name[20];                     /* 姓名 */
27          char sex;                          /* 性别 */
28          int  age;                          /* 年龄 */
29          union maritalState marital;        /* 婚姻状况 */
30          int marryFlag;                     /* 婚姻状况标记 */
31      };
```

```
18      union maritalState                     /* 定义婚姻状况共用体类型 */
19      {
20          int single;                        /* 未婚 */
21          struct marriedState married;       /* 已婚 */
22          struct divorceState divorce;       /* 离婚 */
23      };
```

共用体示例2(续)

- 正文级别 1

姓名	性别	年龄	婚姻状况						婚姻状况标记
			未婚	已婚			离婚		
				结婚日期	配偶姓名	子女数量	离婚日期	子女数量	

```
7       struct marriedState                 /* 定义已婚结构体类型 */
8       {
9           struct date marryDay;           /* 结婚日期 */
10          char spouseName[20];            /* 配偶姓名 */
11          int  child;                     /* 子女数量 */
12      };
```

- 正文级

```
13      struct divorceState                 /* 定义离婚结构体类型 */
14      {
15          struct date divorceDay;         /* 离婚日期 */
16          int  child;                     /* 子女数量 */
17      };
```

```
18      union maritalState                  /* 定义婚姻状况共用体类型 */
19      {
20          int single;                     /* 未婚 */
21          struct marriedState married;    /* 已婚 */
22          struct divorceState divorce;    /* 离婚 */
23      };
```

枚举类型(ENUM)

- 描述的是一组整型值的集合
- 用于当某些量仅由有限个数据值组成时

- enum weeks {SUN, MON, TUE, WED, THU, FRI, SAT};
- enum weeks today;
- enum response {no, yes, none};
- enum response answer;

- today = TUE;
- answer = yes;
-
- enum response {no = –1, yes = 1, none = 0};