Design by yourself

C 程 序 设 计

C Programming

14

结构体与其它数据格式

理论课程

厦門大學

XIAMEN UNIVERSITY

信息学院 黄 煒

(特色化示范性软件学院) 博士,副教授

School of Informatics Wei Huang

知识框架

- 结构体
  - 声明、存储、初始化、访问
  - 结构体与指针：数组、指针、函数调用
- 联合体
- 枚举
- 类型定义

内容纲要

结构体的必要性

- 基本数据类型无法表示复杂的事物
  - 整型、浮点型
  - 数组、指针
- 图书
  - 书名（字符串）
  - 价格（单精度浮点数）
  - 页码（整数）

结构体（Structure）

- 结构体类型
  - 一组物理意义不同，数据类型不同，范畴相关的变量集合
  - 例如：“书”有标题、作者、出版社、价格、总页码等
    - 这些变量物理意义和数据类型不同，但与“书”的范畴相关
- 结构体变量
  - 称：对象（Object）
- 结构体中的变量
  - 称：成员（Member）、属性（Attribute）或字段（Field）

结构体（Structure）

- 修改结构体成员值的相关函数
  - 称：方法（Method）

结构体类型的声明

- 声明结构体类型时必须指明标签名
  - 未指定标签名的两个结构体变量不可相互赋值

声明形式	语句格式	示例

只声明结构体类型	struct <标签名>
	{
	    <成员列表>
	};
		struct book
		{
		    char title[MAXTITL];
		    char author[MAXAUTL];
		    float value;
		};

声明结构体类型和变量	struct <标签名>
	{
	    <成员列表>
	} <变量名列表>;
		struct book
		{
		    char title[MAXTITL];
		    char author[MAXAUTL];
		    float value;
		} mybook;

结构体变量的声明

- 格式

声明形式	语句格式	示例

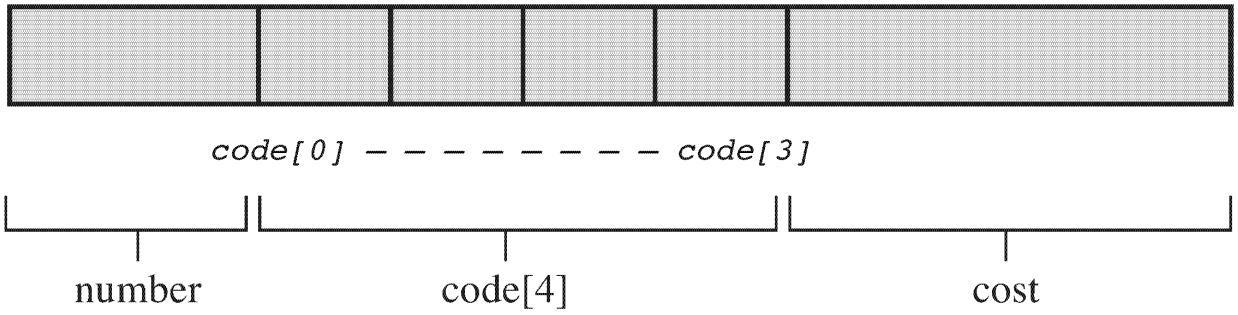
先声明结构体类型，再声明结构体变量	struct <标签名>
	{
	　　<成员列表>
	};
	struct <标签名> <变量名列表>;
		struct book
		{
		　　char title[MAXTITL];
		　　float value;
		};
		struct book mybook;

不声明结构体类型，只声明结构体变量	struct { 　　<成员列表> } <变量名列表>;	struct { 　　char title[MAXTITL]; 　　float value; } mybook;
声明结构体类型和变量	struct <标签名> { 　　<成员列表> } <变量名列表>;	struct book { 　　char title[MAXTITL]; 　　float value; } mybook;

结构体变量的存储

- 结构体变量应先声明后使用，先写后读
- 结构体依其成员顺序在内存中存储
  - 便于在解析格式化的对象时指针类型的强制类型转换
    - 例如：IP报文的头部，文件头

```
struct stuff {
              int number;
              char code[4];
              float cost;
              };
```



```c
/* book.c -- one-book inventory */
#include <stdio.h>
#include <string.h>
char * s_gets(char * st, int n);
#define MAXTITL  41      /* maximum length of title + 1          */
#define MAXAUTL  31      /* maximum length of author's name + 1 */

struct book {            /* structure template: tag is book     */
    char title[MAXTITL];
    char author[MAXAUTL];
    float value;
};                       /* end of structure template           */

int main(void)
{
    struct book library; /* declare library as a book variable*/
    printf("Please enter the book title.\n");
    s_gets(library.title, MAXTITL);/*access to the title portion*/
    printf("Now enter the author.\n");
```

```
    s_gets(library.author, MAXAUTL);
    printf("Now enter the value.\n");
    scanf("%f", &library.value);
    printf("%s by %s: $%.2f\n",library.title, library.author,
library.value);
    printf("%s: \"%s\" ($%.2f)\n", library.author,
           library.title, library.value);
    printf("Done.\n");

    return 0;
}

char * s_gets(char * st, int n)
{
    char * ret_val;
    char * find;

    ret_val = fgets(st, n, stdin);
    if (ret_val)
    {
```

```
        find = strchr(st, '\n');   // look for newline
        if (find)                  // if the address is not NULL,
            *find = '\0';          // place a null character there
        else
            while (getchar() != '\n')
                continue;          // dispose of rest of line
    }

    return ret_val;
}
```

```
Please enter the book title.
C Primer Plus↲
Now enter the author.
Stephen Prata↲
Now enter the value.
60.00↲
C Primer Plus by Stephen Prata: $60.00
Stephen Prata: "C Primer Plus" ($60.00)
Done.
```

结构体的意义

- 结构体提供了一个看待问题的视角
  - 将程序划分为数据和流程两个部分
  - 数据按照类别定义为一个个对象
  - 对象具有属性和方法（操作对象的函数）
- 高内聚，低耦合
  - 我们正在面向对象的路上

结构体变量的初始化

- 类似于一般变量和数组

```
int count = 0;
int fibo[7] = {0,1,1,2,3,5,8};
```

- 结构体可以用复合文字初始化

```
struct book library = {
    "The Pious Pirate and the Devious Damsel",
    "Renee Vivotte",
    1.95
};
```

– 注意：只能在声明时初始化，各个初始值的类型要对应。

结构体变量成员的访问

- 结构体就像一个“超级数组”
  - 格式：`[tag].[member]`
  - 示例：`bill.title`
- 结构体指定初始化项目

```
struct book surprise = { .value = 10.99};
struct book gift = { .value = 25.99,
    .author = "James Broadfool",
    .title = "Rue for the Toad"};
```

内容纲要

结构体数组

- 结构体数组
  - 声明语句

```
struct <标签名> <数组名> [<数组大小>];
```

  - 访问表达式

```
<数组名>[<数组下标>].<成员>;
```

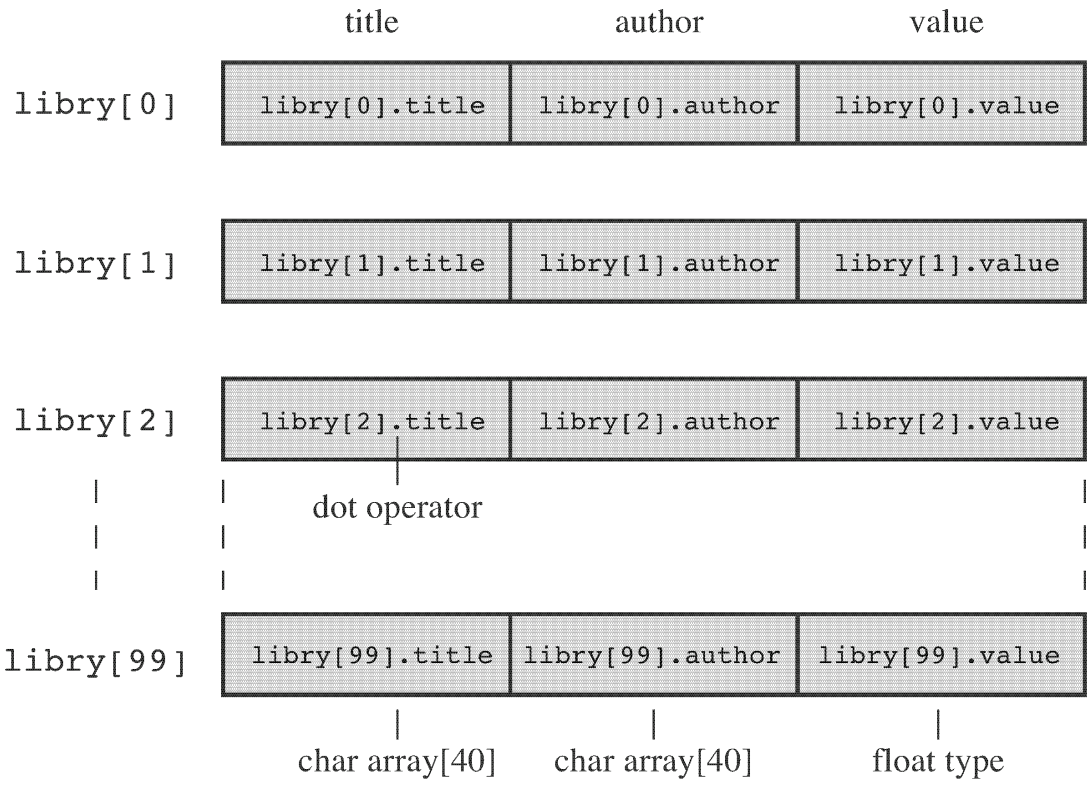
    - 将方括号、圆点视为操作符
    - 下标应直接写在数组名后

- 结构体数组成员的访问

```
library                 // an array of book structures
library[2]              // an array element, hence a book structure
library[2].title        // a char array (the title member of library[2])
library[2].title[4] // a char in the title member array
```

结构体数组

- 内存存储格式



declaration: `struct book libry[MAXBKS]`

```c
/* manybook.c -- multiple book inventory */
#include <stdio.h>
#include <string.h>
char * s_gets(char * st, int n);
#define MAXTITL   40
#define MAXAUTL   40
#define MAXBKS   100                /* maximum number of books  */

struct book {                       /* set up book template     */
    char title[MAXTITL];
    char author[MAXAUTL];
    float value;
};

int main(void)
{
    struct book library[MAXBKS]; /* array of book structures */
    int count = 0;
```

```
    int index;

    printf("Please enter the book title.\n");
    printf("Press [enter] at the start of a line to stop.\n");
    while (count < MAXBKS && s_gets(library[count].title,
MAXTITL) != NULL
           && library[count].title[0] != '\0')
    {
        printf("Now enter the author.\n");
        s_gets(library[count].author, MAXAUTL);
        printf("Now enter the value.\n");
        scanf("%f", &library[count++].value);
        while (getchar() != '\n')
            continue;           /* clear input line          */
        if (count < MAXBKS)
            printf("Enter the next title.\n");
    }
```

```
    if (count > 0)
    {
        printf("Here is the list of your books:\n");
        for (index = 0; index < count; index++)
            printf("%s by %s: $%.2f\n", library[index].title,
                library[index].author, library[index].value);
    }
    else
       printf("No books? Too bad.\n");
    return 0;
}

char * s_gets(char * st, int n)
{
    char * ret_val;
    char * find;
    ret_val = fgets(st, n, stdin);
    if (ret_val)
```

```
    {
        find = strchr(st, '\n');    // look for newline
        if (find)                   // if the address is not NULL,
            *find = '\0';           // place a null character there
        else
            while (getchar() != '\n')
                continue;           // dispose of rest of line
    }
    return ret_val;
}
```

```
Please enter the book title.
Press [enter] at the start of a line to stop.
C Primer Plus↲
Now enter the author.
Stephen Prata↲
Now enter the value.
100↲
Enter the next title.
↲
Here is the list of your books:
C Primer Plus by Stephen Prata: $100.00
```

结构体嵌套

- 函数可以嵌套，数组可以多重，结构体也可以嵌套

```
struct names {                          // first structure
    char first[LEN];
    char last[LEN];
};
struct guy {                            // second structure
    struct names handle;                // nested structure
    char favfood[LEN];
    char job[LEN];
    float income;
};
```

- 嵌套结构体成员的访问

```
fellow.handle.first
```

```c
// friend.c -- example of a nested structure
#include <stdio.h>
#define LEN 20
const char * msgs[5] =
{
    "    Thank you for the wonderful evening, ",
    "You certainly prove that a ",
    "is a special kind of guy. We must get together",
    "over a delicious ",
    " and have a few laughs"
};
struct names {                          // first structure
    char first[LEN];
    char last[LEN];
};
struct guy {                            // second structure
    struct names handle;                // nested structure
    char favfood[LEN];
```

```
    char job[LEN];
    float income;
};
int main(void)
{
    struct guy fellow = {   // initialize a variable
        {"Ewen", "Villard"}, "grilled salmon", "personality
coach", 68112.00
    };
    printf("Dear %s, \n\n", fellow.handle.first);
    printf("%s%s.\n", msgs[0], fellow.handle.first);
    printf("%s%s\n", msgs[1], fellow.job);
    printf("%s\n", msgs[2]);
    printf("%s%s%s", msgs[3], fellow.favfood, msgs[4]);
    if (fellow.income > 150000.0)
        puts("!!");
    else if (fellow.income > 75000.0)
        puts("!");
```

```
    else
        puts(".");
    printf("\n%40s%s\n", " ", "See you soon,");
    printf("%40s%s\n", " ", "Shalala");
    return 0;
}
```

```
Dear Ewen,

    Thank you for the wonderful evening, Ewen.
You certainly prove that a personality coach
is a special kind of guy. We must get together
over a delicious grilled salmon and have a few laughs.

                                        See you soon,
                                        Shalala
```

指向结构体的指针

- 结构体指针的声明、查址、取址与一般类型一致

```
struct guy * him;
```

```
him = &barney;
him = &fellow[0];
```

- 指针成员运算符 -> <指针名>-><成员名>
  - 以下三种写法等价（应使用第一种写法）

```
him->income
```

```
(*him).income
```

```
him[0].income
```

```c
/* friends.c -- uses pointer to a structure */
#include <stdio.h>
#define LEN 20
struct names {
    char first[LEN];
    char last[LEN];
};
struct guy {
    struct names handle;
    char favfood[LEN];
    char job[LEN];
    float income;
};

int main(void)
{
    struct guy fellow[2] = {
        {{ "Ewen", "Villard"},"grilled salmon","personality coach",
            68112.00
```

```
        },
        {{"Rodney", "Swillbelly"}, "tripe", "tabloid editor",
            432400.00
        }
    };
    struct guy * him;     /* here is a pointer to a structure */
    printf("address #1: %p #2: %p\n", &fellow[0], &fellow[1]);
    him = &fellow[0];     /* tell the pointer where to point  */
    printf("pointer #1: %p #2: %p\n", him, him + 1);
    printf("him->income is $%.2f: (*him).income is $%.2f\n",
           him->income, (*him).income);
    him++;                /* point to the next structure      */
    printf("him->favfood is %s:  him->handle.last is %s\n",
           him->favfood, him->handle.last);
    return 0;
}
```

```
address #1: 00E1FC9C #2: 00E1FCF0
pointer #1: 00E1FC9C #2: 00E1FCF0
him->income is $68112.00: (*him).income is $68112.00
him->favfood is tripe:  him->handle.last is Swillbelly
```

运算符的优先级

序号	符号	说明	序号	符号	说明
1 →	后缀++ --	后缀增减量	3	* / %	算术运算：乘除模
	( )	函数调用	4	+ -	算术运算：加减
	[ ]	数组下标	6	< > <= >=	关系运算符：大小
	.	结构体联合体成员	7	== !=	关系运算符：相等
	->	结构/联合指针成员			
	(type){list}	复合文字			
2 ←	前缀++ --	前缀增减量	11	&&	逻辑运算符：与
	+ -	正负号	12	||	逻辑运算符：或
	!	逻辑运算符：非	13	?:	三元条件运算符
	(type)	强制类型转换	14 ←	=	赋值
	*	间接寻址		+= -= *= /= %=	自增自减自乘自除自模
	&	取址			
	sizeof	存储空间	15	,	逗号表达式

在函数中访问结构体信息

- 传递结构体成员

```
void showinfo (const char * info);
```

  - 优点：函数与结构体无关，适用范围广

- 传递结构体地址

```
void showinfo (const struct namect * p);
```

  - 优点：在传递大型结构体时，指针传递只需赋值4或8字节
  - 为了防止修改可加const

- 传递结构体整体

```
void showinfo (struct namect info);
```

  - 缺点：传递大型结构体需要占用更多的内存空间
  - 仅用于较小的结构体

```c
/* funds1.c -- passing structure members as arguments */
#include <stdio.h>
#define FUNDLEN 50
struct funds {
    char   bank[FUNDLEN];
    double bankfund;
    char   save[FUNDLEN];
    double savefund;
};
double sum(double, double);

int main(void)
{
    struct funds stan = {
        "Garlic-Melon Bank",4032.27,"Lucky's Savings and
Loan",8543.94
    };
```

```
    printf("Stan has a total of $%.2f.\n", sum(stan.bankfund,
stan.savefund));

    return 0;
}

/* adds two double numbers */
double sum(double x, double y)
{
    return(x + y);
}
```

```
Stan has a total of $12576.21.
```

```c
/* funds2.c -- passing a pointer to a structure */
#include <stdio.h>
#define FUNDLEN 50
struct funds {
    char   bank[FUNDLEN];
    double bankfund;
    char   save[FUNDLEN];
    double savefund;
};
double sum(const struct funds *);  /* argument is a pointer */

int main(void)
{
    struct funds stan = {
        "Garlic-Melon Bank",4032.27,"Lucky's Savings and
Loan",8543.94
    };
```

```
    printf("Stan has a total of $%.2f.\n", sum(&stan));

    return 0;
}

/* adds two double numbers */
double sum(const struct funds * money)
{
    return(money->bankfund + money->savefund);
}
```

```
Stan has a total of $12576.21.
```

```c
/* funds2.c -- passing a pointer to a structure */
#include <stdio.h>
#define FUNDLEN 50
struct funds {
    char   bank[FUNDLEN];
    double bankfund;
    char   save[FUNDLEN];
    double savefund;
};
double sum(struct funds moolah);  /* argument is a structure */

int main(void)
{
    struct funds stan = {
        "Garlic-Melon Bank",4032.27,"Lucky's Savings and
Loan",8543.94
    };
```

```
    printf("Stan has a total of $%.2f.\n", sum(stan));

    return 0;
}

/* adds two double numbers */
double sum(struct funds moolah)
{
    return(moolah.bankfund + moolah.savefund);
}
```

```
Stan has a total of $12576.21.
```

结构体的其它特性

- 结构体可以赋值给另一个结构体
  - 可用于初始化
- 结构体可以作为参数返回
- 结构体和一般的整型实数型没有什么区别

```c
/* names1.c -- uses pointers to a structure */
#include <stdio.h>
#include <string.h>

#define NLEN 30
struct namect {
    char fname[NLEN];
    char lname[NLEN];
    int letters;
};
void getinfo(struct namect *);
void makeinfo(struct namect *);
void showinfo(const struct namect *);
char * s_gets(char * st, int n);

int main(void)
{
    struct namect person;
```

```
    getinfo(&person);
    makeinfo(&person);
    showinfo(&person);
    return 0;
}

void getinfo (struct namect * pst)
{
    printf("Please enter your first name.\n");
    s_gets(pst->fname, NLEN);
    printf("Please enter your last name.\n");
    s_gets(pst->lname, NLEN);
}

void makeinfo (struct namect * pst)
{
    pst->letters = strlen(pst->fname) + strlen(pst->lname);
}
```

```
void showinfo (const struct namect * pst)
{
    printf("%s %s, your name contains %d letters.\n",
           pst->fname, pst->lname, pst->letters);
}

char * s_gets(char * st, int n)
{
    char * ret_val;
    char * find;
    ret_val = fgets(st, n, stdin);
    if (ret_val)
    {
        find = strchr(st, '\n');   // look for newline
        if (find)                  // if the address is not NULL,
            *find = '\0';          // place a null character there
        else
```

```
            while (getchar() != '\n')
                continue;                   // dispose of rest of line
    }
    return ret_val;
}
```

```
Please enter your first name.
Wei↲
Please enter your last name.
Huang↲
Wei Huang, your name contains 8 letters.
```

```c
/* names2.c -- passes and returns structures */
#include <stdio.h>
#include <string.h>

#define NLEN 30
struct namect {
    char fname[NLEN];
    char lname[NLEN];
    int letters;
};

struct namect getinfo(void);
struct namect makeinfo(struct namect);
void showinfo(struct namect);
char * s_gets(char * st, int n);

int main(void)
{
    struct namect person;
```

```
    person = getinfo();
    person = makeinfo(person);
    showinfo(person);
    return 0;
}

struct namect getinfo(void)
{
    struct namect temp;
    printf("Please enter your first name.\n");
    s_gets(temp.fname, NLEN);
    printf("Please enter your last name.\n");
    s_gets(temp.lname, NLEN);

    return temp;
}

struct namect makeinfo(struct namect info)
{
```

```
    info.letters = strlen(info.fname) + strlen(info.lname);

    return info;
}

void showinfo(struct namect info)
{
    printf("%s %s, your name contains %d letters.\n",
           info.fname, info.lname, info.letters);
}

char * s_gets(char * st, int n)
{
    char * ret_val;
    char * find;

    ret_val = fgets(st, n, stdin);
    if (ret_val)
```

```
    {
        find = strchr(st, '\n');    // look for newline
        if (find)                   // if the address is not NULL,
            *find = '\0';           // place a null character there
        else
            while (getchar() != '\n')
                continue;           // dispose of rest of line
    }
    return ret_val;
}
```

```
Please enter your first name.
Wei↲
Please enter your last name.
Huang↲
Wei Huang, your name contains 8 letters.
```

结构体成员使用数组还是指针

- 结构体成员是数组
  - 占用内存较大，但方便读写文件或传输
- 结构体成员是指针
  - 需要程序员确保空间的新建和释放
  - 交给别人使用时，容易误用

```
#define LEN 20
struct names {
    char first[LEN];
    char last[LEN];
};
```

```
struct names {
    char *first;
    char *last;
};
```

在结构体成员中使用数组还是指针

- 如果在结构体使用了指针，初始化略麻烦

```
struct namect {
    char * fname; // using pointers instead of arrays
    char * lname;
    int letters;
};
void getinfo(struct namect * pst)
{
    char temp[SLEN];
    printf("Please enter your first name.\n");
    s_gets(temp, SLEN);
    // allocate memory to hold name
    pst->fname = (char *)malloc(strlen(temp) + 1);
    strcpy(pst->fname, temp);          // copy name to allocated memory
    printf("Please enter your last name.\n");
    s_gets(temp, SLEN);
    pst->lname = (char *)malloc(strlen(temp) + 1);
    strcpy(pst->lname, temp);
}
```

```c
// names3.c -- use pointers and malloc()
#include <stdio.h>
#include <string.h>   // for strcpy(), strlen()
#include <stdlib.h>   // for malloc(), free()
#define SLEN 81
struct namect {
    char * fname;  // using pointers
    char * lname;
    int letters;
};

void getinfo(struct namect *);         // allocates memory
void makeinfo(struct namect *);
void showinfo(const struct namect *);
void cleanup(struct namect *);         // free memory when done
char * s_gets(char * st, int n);
```

```c
int main(void)
{
    struct namect person;
    getinfo(&person);
    makeinfo(&person);
    showinfo(&person);
    cleanup(&person);
    return 0;
}

void getinfo (struct namect * pst)
{
    char temp[SLEN];
    printf("Please enter your first name.\n");
    s_gets(temp, SLEN);
    // allocate memory to hold name
    pst->fname = (char *) malloc(strlen(temp) + 1);
```

```
    // copy name to allocated memory
    strcpy(pst->fname, temp);
    printf("Please enter your last name.\n");
    s_gets(temp, SLEN);
    pst->lname = (char *) malloc(strlen(temp) + 1);
    strcpy(pst->lname, temp);
}

void makeinfo (struct namect * pst)
{
    pst->letters = strlen(pst->fname) +
    strlen(pst->lname);
}

void showinfo (const struct namect * pst)
{
    printf("%s %s, your name contains %d letters.\n",
           pst->fname, pst->lname, pst->letters);
}
```

```c
void cleanup(struct namect * pst)
{
    free(pst->fname);
    free(pst->lname);
}

char * s_gets(char * st, int n)
{
    char * ret_val;
    char * find;

    ret_val = fgets(st, n, stdin);
    if (ret_val)
    {
        find = strchr(st, '\n');   // look for newline
        if (find)                  // if the address is not NULL,
            *find = '\0';          // place a null character there
```

```
        else
            while (getchar() != '\n')
                continue;           // dispose of rest of line
    }
    return ret_val;
}
```

```
Please enter your first name.
Wei↲
Please enter your last name.
Huang↲
Wei Huang, your name contains 8 letters.
```

结构体复合文字

- 数组有复合文字，结构体同样也有

```
(struct book) {"Mr. Bouncy's Nice Hat",
               "Fred Winsome",
               5.99};
```

- 伸缩型数组成员
  - 一定是数组的最后一个成员，并且有其他成员

```
struct flex
{
    size_t count;
    double average;
    double scores[];   // flexible array member
};
```

```
pf1 = malloc(sizeof(struct flex) + n * sizeof(double));
```

```c
/* complit.c -- compound literals */
#include <stdio.h>
#define MAXTITL   41
#define MAXAUTL   31

struct book {           // structure template: tag is book
    char title[MAXTITL];
    char author[MAXAUTL];
    float value;
};

int main(void)
{
    struct book readfirst;
    int score;

    printf("Enter test score: ");
    scanf("%d",&score);
```

```
    if(score >= 84)
        readfirst = (struct book) {"Crime and Punishment",
            "Fyodor Dostoyevsky",
            11.25};
    else
        readfirst = (struct book) {"Mr. Bouncy's Nice Hat",
            "Fred Winsome",
            5.99};
    printf("Your assigned reading:\n");
    printf("%s by %s: $%.2f\n",readfirst.title,
           readfirst.author, readfirst.value);

    return 0;
}
```

```
Enter test score: 87↲
Your assigned reading:
Crime and Punishment by Fyodor Dostoyevsky: $11.25
```

```c
// flexmemb.c -- flexible array member (C99 feature)
#include <stdio.h>
#include <stdlib.h>

struct flex
{
    size_t count;
    double average;
    double scores[];   // flexible array member
};
void showFlex(const struct flex * p);

int main(void)
{
    struct flex * pf1, *pf2;
    int n = 5;
    int i;
    int tot = 0;
```

```
// allocate space for structure plus array
pf1 = malloc(sizeof(struct flex) + n * sizeof(double));
pf1->count = n;
for (i = 0; i < n; i++)
{
    pf1->scores[i] = 20.0 - i;
    tot += pf1->scores[i];
}
pf1->average = tot / n;
showFlex(pf1);
n = 9;
tot = 0;
pf2 = malloc(sizeof(struct flex) + n * sizeof(double));
pf2->count = n;
for (i = 0; i < n; i++)
{
    pf2->scores[i] = 20.0 - i/2.0;
    tot += pf2->scores[i];
}
```

```
    pf2->average = tot / n;
    showFlex(pf2);
    free(pf1);
    free(pf2);
    return 0;
}
```

```
Scores : 20 19 18 17 16
Average: 18
Scores : 20 19.5 19 18.5 18 17.5 17 16.5 16
Average: 17
```

```
void showFlex(const struct flex * p)
{
    int i;
    printf("Scores : ");
    for (i = 0; i < p->count; i++)
        printf("%g ", p->scores[i]);
    printf("\nAverage: %g\n", p->average);
}
```

使用结构体数组的函数

- 跟整型数组作为函数参量时一样
  - 实际上还是指针
  - 说明这里并非只有一个元素
  - 如果不改变其元素值应加上const

```c
/* funds4.c -- passing an array of structures to a function */
#include <stdio.h>
#define FUNDLEN 50
#define N 2
struct funds {
    char   bank[FUNDLEN];
    double bankfund;
    char   save[FUNDLEN];
    double savefund;
};
double sum(const struct funds money[], int n);

int main(void)
{
    struct funds jones[N] = {
        {
            "Garlic-Melon Bank",
            4032.27,
```

```
            "Lucky's Savings and Loan",
            8543.94
        },
        {
            "Honest Jack's Bank",
            3620.88,
            "Party Time Savings",
            3802.91
        }
    };
    printf("The Joneses have a total of $%.2f.\n", sum(jones,N));

    return 0;
}

double sum(const struct funds money[], int n)
{
    double total;
```

```
    int i;

    for (i = 0, total = 0; i < n; i++)
        total += money[i].bankfund + money[i].savefund;

    return(total);
}
```

```
The Joneses have a total of $20000.00.
```

将结构体内容保存到文件中

- 定义

```
#define MAXTITL 40
#define MAXAUTL 40
struct book {
    char title[MAXTITL];
    char author[MAXAUTL];
    float value;
};
```

- 明文保存：占用空间，读写不便

```
fprintf(pbooks, "%s %s %.2f\n", cpp.title, cpp.author, cpp.value);
```

- 有效率的用法：成员有指针的时候不能这么用

```
fwrite(&primer, sizeof (struct book), 1, pbooks);
```

```c
/* booksave.c -- saves structure contents in a file */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAXTITL  40
#define MAXAUTL  40
#define MAXBKS   10                 /* maximum number of books */
char * s_gets(char * st, int n);
struct book {                       /* set up book template    */
    char title[MAXTITL];
    char author[MAXAUTL];
    float value;
};
int main(void) {
    struct book library[MAXBKS]; /* array of structures     */
    int count = 0;
    int index, filecount;
    FILE * pbooks;
    int size = sizeof (struct book);
```

```
    if ((pbooks = fopen("book.dat", "a+b")) == NULL) {
        fputs("Can't open book.dat file\n",stderr);
        exit(1);
    }
    rewind(pbooks);                /* go to start of file      */
    while (count < MAXBKS && fread(&library[count], size, 1,
pbooks) == 1) {
        if (count == 0)
            puts("Current contents of book.dat:");
        printf("%s by %s: $%.2f\n",library[count].title,
               library[count].author, library[count].value);
        count++;
    }
    filecount = count;
    if (count == MAXBKS) {
        fputs("The book.dat file is full.", stderr);
        exit(2);
    }
```

```
puts("Please add new book titles.");
puts("Press [enter] at the start of a line to stop.");
while (count < MAXBKS && s_gets(library[count].title, MAXTITL)!=NULL
       && library[count].title[0] != '\0') {
    puts("Now enter the author.");
    s_gets(library[count].author, MAXAUTL);
    puts("Now enter the value.");
    scanf("%f", &library[count++].value);
    while (getchar() != '\n')
        continue;                       /* clear input line  */
    if (count < MAXBKS)
        puts("Enter the next title.");
}
if (count > 0) {
    puts("Here is the list of your books:");
    for (index = 0; index < count; index++)
        printf("%s by %s: $%.2f\n",library[index].title,
            library[index].author, library[index].value);
```

```
        fwrite(&library[filecount], size, count - filecount, pbooks);
    }
    else
       puts("No books? Too bad.\n");
    puts("Bye.\n");
    fclose(pbooks);
    return 0;
}

char * s_gets(char * st, int n) {
    char * ret_val;
    char * find;
    ret_val = fgets(st, n, stdin);
    if (ret_val) {
        find = strchr(st, '\n');   // look for newline
        if (find)                  // if the address is not NULL,
            *find = '\0';          // place a null character there
        else
```

```
            while (getchar() != '\n')
                continue;      // dispose of rest of line
    }
    return ret_val;
}
```

```
Please add new book titles.
Press [enter] at the start of a line to stop.
C Primer Plus↲
Now enter the author.
Stephen Prata↲
Now enter the value.
60↲
Enter the next title.

Here is the list of your books:
C Primer Plus by Stephen Prata: $60.00
Bye.
```

程序讨论

- book.dat的内容

```
00000000h:  43 20 50 72 69 6D 65 72 20 50 6C 75 73 00 00 00   C Primer Plus...
00000010h:  02 00 2C 00 B8 84 57 00 DC 00 00 00 18 9E 56 00   ..,.??W.?....?V.
00000020h:  00 00 00 00 00 00 00 00 53 74 65 70 68 65 6E 20   ........Stephen
00000030h:  50 72 61 74 61 00 00 53 0C FA 46 00 45 66 76 77   Prata..S.?F.Efvw
00000040h:  E0 1D 12 B0 FE FF FF FF 64 F7 46 00 D3 06 72 77   ?..?????d?F.?.rw
00000050h:  00 00 70 42                                       ..pB
```

- 程序要点
  - 打开文件模式：a+b
  - 选用fread()和fwrite()
  - 保证指针处于文件开始rewind()
  - 文件空间还是有点费

结构体能做什么

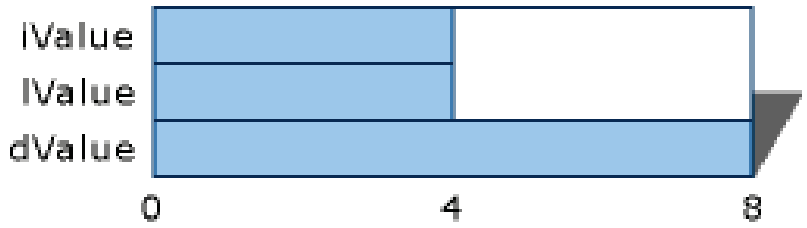
- 重要应用：创建新的数据形式
- 一些已有的形式
  - 链表（第17章将讲解）
  - 队列
  - 二叉树
  - 堆
  - 哈希表
  - 图

内容纲要

联合体

- 联合体（union）是一个能在同一个存储空间（但不同时）存储不同类型数据的数据类型。

```
union NumericType
{
    int         iValue;
    long        lValue;
    double      dValue;
};
```



联合体的使用

- 声明

```
union hold fit; // union variable of hold type
union hold save[10]; // array of 10 union variables
union hold * pu; // pointer to a variable of hold type
```

- 初始化

```
union hold valA;
valA.letter = 'R';
union hold valB = valA; // initialize one union to another
union hold valC = {88}; // initialize digit member of union
union hold valD = {.bigfl = 118.2}; // designated initializer
```

- 一般用法

```
fit.digit = 23; // 23 is stored in fit; 2 bytes used
fit.bigfl = 2.0; // 23 cleared, 2.0 stored; 8 bytes used
fit.letter = 'h'; // 2.0 cleared, h stored; 1 byte used
```

联合体的使用

- 指针用法

```
pu = &fit;
x = pu->digit; // same as x = fit.digit
```

- 错误用法：应避免使用非初始化的内存区域

```
fit.letter = 'A';
flnum = 3.02 * fit.bigfl; // ERROR ERROR ERROR
```

- 结构体可以和共同体搭配使用

```
union data {
    struct owner owncar;
    struct leasecompany
leasecar;
};
```

```
struct car_data {
    char make[15];
    int status; /* 0 owned, 1 leased */
    union data ownerinfo;
    ...
};
```

匿名联合体（C11）

扩展内容

- 可以直接使用flits.owncar.socsecurity

```
struct owner {
    char socsecurity[12];
    ...
};

struct leasecompany {
    char name[40];
    char headquarters[40];
    ...
};
```

```
struct car_data {
    char make[15];
    int status; /* 0 = owned, 1 =
leased */
    union {
        struct owner owncar;
        struct leasecompany leasecar;
    };
    ...
};
```

内容纲要

枚举

- 自定义一种类型并且自定义其值

```
enum spectrum {red, orange, yellow, green, blue, violet};
enum spectrum color;
```

- 使用方法

```
color = blue;
if (color == yellow)
    ...;
for (color = red; color <= violet; color++)
    ...;
```

- 枚举常量
  - 如果有赋值则用赋值；如果无赋值，则为上一个值加一；首个元素默认为0

```
enum kids {nippy, slats, skippy, nina, liz};
enum feline {cat, lynx = 10, puma, tiger};
```

```c
/* enum.c -- uses enumerated values */
#include <stdio.h>
#include <string.h>     // for strcmp(), strchr()
#include <stdbool.h>    // C99 feature
char * s_gets(char * st, int n);
enum spectrum {red, orange, yellow, green, blue, violet};
const char * colors[] = {"red", "orange", "yellow",
    "green", "blue", "violet"};
#define LEN 30

int main(void)
{
    char choice[LEN];
    enum spectrum color;
    bool color_is_found = false;
    puts("Enter a color (empty line to quit):");
    while (s_gets(choice, LEN) != NULL && choice[0] != '\0')
    {
        for (color = red; color <= violet; color++)
```

```
    {
        if (strcmp(choice, colors[color]) == 0)
        {
            color_is_found = true;
            break;
        }
    }
    if (color_is_found)
        switch(color)
    {
        case red    : puts("Roses are red.");
            break;
        case orange : puts("Poppies are orange.");
            break;
        case yellow : puts("Sunflowers are yellow.");
            break;
        case green  : puts("Grass is green.");
            break;
        case blue   : puts("Bluebells are blue.");
```

```
                break;
            case violet : puts("Violets are violet.");
                break;
        }
        else
            printf("I don't know about the color %s.\n", choice);
        color_is_found = false;
        puts("Next color, please (empty line to quit):");
    }
    puts("Goodbye!");
    return 0;
}

char * s_gets(char * st, int n)
{
    char * ret_val;
    char * find;
    ret_val = fgets(st, n, stdin);
```

```
    if (ret_val)
    {
        find = strchr(st, '\n');   // look for newline
        if (find)                  // if the address is not NULL,
            *find = '\0';          // place a null character there
        else
            while (getchar() != '\n')
                continue;          // dispose of rest of line
    }
    return ret_val;
}
```

```
Enter a color (empty line to quit):
Red↲
I don't know about the color Red.
Next color, please (empty line to quit):
blue↲
Bluebells are blue.
Next color, please (empty line to quit):
pink↲
I don't know about the color pink.
Next color, please (empty line to quit):

Goodbye!
```

共享命名空间

- C使用命名空间识别一个名字的程序部分
- 作用域
- 结构体、联合体、枚举标记共享一个命名空间
  - 与普通变量不冲突

```
struct rect { double x; double y; };
int rect; // not a conflict in C
```

内容纲要

2	结构体与指针
3	联合体
4	枚举
5	类型定义
6	函数与指针

typedef简介

- 自定义类型名
  - 仅限针对类型，而非对值
- typedef和#define的区别
  - typedef解释在编译器，#define的解释在处理器

```
typedef unsigned char BYTE;
BYTE x, y[10], * z;
```

- 类型定义与使用

```
#define BYTE unsigned char
```

typedef简介

- 类型定义与使用

```
typedef char * STRING;
STRING name, sign;
```

  - 含义为

```
char * name, * sign;
```

  - 如果使用的是#define

```
#define STRING char *
STRING name, sign;
```

  - 含义为

```
char * name, sign;
```

typedef简介

- 自定义结构体

```
typedef struct complex {
    float real;
    float imag;
} COMPLEX;
```

```
typedef struct {
    float real;
    float imag;
} COMPLEX;
```

  - 此时结构体的名字可以忽略

- 自定义过于复杂的类型

```
typedef char (* FRPTC ()) [5];
```

声明中的修饰符

- C允许同时使用多个修饰符
- 优先级为括号高于星号

修饰符	含义
*	表示一个指针
()	表示一个函数
[]	表示一个数组

```
int board[8][8]; // an array of arrays of int
int ** ptr; // a pointer to a pointer to int
int * risks[10]; // a 10-element array of pointers to int
int (* rusks)[10]; // a pointer to an array of 10 ints
int * oof[3][4]; // a 3 x 4 array of pointers to int
int (* uuf)[3][4]; // a pointer to a 3 x 4 array of ints
int (* uof[3])[4]; // a 3-element array of pointers to 4-
element arrays of int
```

声明中的修饰符

- 修饰符的优先级
  - 数组修饰符[]优先级与函数()相同，但高于指针*
  - 数组修饰符[]优先级与函数()都是从左至右结合
  - 如果破坏上述规则，则应加上小括号()

```
char * fump(int);        // function returning pointer to char
char (* frump)(int);     // pointer to a function that returns type char
char (* flump[3])(int); // array of 3 pointers to func.s that return type char
```

```
typdef int arr5[5];
typedef arr5 * p_arr5;
typedef p_arr5 arrp10[10];
arr5 togs; // togs an array of 5 int
p_arr5 p2; // p2 a pointer to an array of 5 int
arrp10 ap; // ap an array of 10 pointers to array-of-5-int
```

内容纲要

指向函数的指针

- 函数的名称是函数所在代码段的首地址（类似数组）
- 作用：函数指针可以作为另一个函数的参数
- 定义与赋值

```
void ToUpper(char *);
void ToLower(char *);
int round(double);
void (*pf)(char *);
pf = ToUpper; // valid, ToUpper is address of the function
pf = ToLower; // valid, ToLower is address of the function
pf = round; // invalid, round is the wrong type of function
pf = ToLower(); // invalid, ToLower() is not an address
```

使用指向函数的指针

- 使用指针指向的函数甚至指针本身来使用函数指针
  - 因为函数名本身就是一个指针

```
void ToUpper(char *);
void ToLower(char *);
void (*pf)(char *);
char mis[] = "Nina Metier";
pf = ToUpper;
(*pf)(mis); // apply ToUpper to mis (syntax 1)
pf = ToLower;
pf(mis); // apply ToLower to mis (syntax 2)
```

```c
// func_ptr.c -- uses function pointers
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#define LEN 81
char * s_gets(char * st, int n);
char showmenu(void);
void eatline(void);     // read through end of line
void show(void (* fp)(char *), char * str);
void ToUpper(char *);   // convert string to uppercase
void ToLower(char *);   // convert string to uppercase
void Transpose(char *); // transpose cases
void Dummy(char *);     // leave string unaltered

int main(void) {
    char line[LEN];
    char copy[LEN];
    char choice;
    void (*pfun)(char *); // points a function having a
                          // char * argument and no return value
```

```
    puts("Enter a string (empty line to quit):");
    while (s_gets(line, LEN) != NULL && line[0] != '\0') {
        while ((choice = showmenu()) != 'n') {
            switch (choice) { // switch sets pointer
                case 'u' : pfun = ToUpper;   break;
                case 'l' : pfun = ToLower;   break;
                case 't' : pfun = Transpose; break;
                case 'o' : pfun = Dummy;     break;
            }
            strcpy(copy, line);// make copy for show()
            show(pfun, copy);  // use selected function
        }
        puts("Enter a string (empty line to quit):");
    }
    puts("Bye!");
    return 0;
}
char showmenu(void) {
    char ans;
    puts("Enter menu choice:");
```

```
    puts("u) uppercase       l) lowercase");
    puts("t) transposed case o) original case");
    puts("n) next string");
    ans = getchar();    // get response
    ans = tolower(ans); // convert to lowercase
    eatline();          // dispose of rest of line
    while (strchr("ulton", ans) == NULL) {
        puts("Please enter a u, l, t, o, or n:");
        ans = tolower(getchar());
        eatline();
    }
    return ans;
}
void eatline(void) {
    while (getchar() != '\n')
        continue;
}
void Dummy(char * str) {
    // leaves string unchanged
}
```

```
void ToUpper(char * str) {
    while (*str) {
        *str = toupper(*str);
        str++;
    }
}
void ToLower(char * str) {
    while (*str) {
        *str = tolower(*str);
        str++;
    }
}
void Transpose(char * str) {
    while (*str) {
        if (islower(*str))
            *str = toupper(*str);
        else if (isupper(*str))
            *str = tolower(*str);
        str++;
    }
}
```

```c
void show(void (* fp)(char *), char * str) {
    (*fp)(str); // apply chosen function to str
    puts(str);  // display result
}
char * s_gets(char * st, int n) {
    char * ret_val;
    char * find;
    ret_val = fgets(st, n, stdin);
    if (ret_val) {
        find = strchr(st, '\n');   // look for newline
        if (find)                  // if the address is not NULL,
            *find = '\0';          // place a null character there
        else
            while (getchar() != '\n')
                continue;          // dispose of rest of line
    }

    return ret_val;
}
```

C 程 序 设 计

C Programming

14

谢谢观看

理论课程

厦門大學

XIAMEN UNIVERSITY

信息学院 黄 炜

(特色化示范性软件学院) 博士,副教授

School of Informatics Wei Huang