If I have seen further, it is by standing on the shoulders of giants.

--- *Isaac Newton*

C 程 序 设 计
C Programming

16

C预处理器和库

理论课程

厦門大學
XIAMEN UNIVERSITY

信息学院
(特色化示范性软件学院)
School of Informatics

黄 炜
博士,副教授
Wei Huang

知识框架

- 预处理
  - 宏定义：#define
  - 文件包含：#include
  - 条件指令：#if #ifdef #ifndef #else #elif #endif
  - 其它指令：#line
- 库函数
  - 标准输入输出库，标准库，字符串库，数学库，内存库
  - 其它库

内容纲要

对程序的翻译处理

- 第一步，字符替换
  - 换将源代码中出现的字符映射到源字符集
  - 三字符组与双字符组

```
#include <stdio.h>
#define A 3 \
*/*ddd*/3

int main(void)
{
    int   啊 = A;
    printf("TWO: ??=\
%d\n", 啊);
    return 0;
}
```

1. 字符替换 →

```
#include <stdio.h>
#define A 3 \
*/*ddd*/3

int main(void)
{
    int   啊 = A;
    printf("TWO: #\
%d\n", 啊);
    return 0;
}
```

对程序的翻译处理

- 第二行，行拼接
  - 将换行转义（反斜杠紧跟换行符）的物理行合并为逻辑行
  - 转义字符前后的字符，尤其下一行未顶格，该字符不丢失

```
#include <stdio.h>
#define A 3 \
*/*ddd*/3

int main(void)
{
    int   啊 = A;
    printf("TWO: #\
    %d\n", 啊);
    return 0;
}
```

2. 行拼接

```
#include <stdio.h>
#define A 3 */*ddd*/3

int main(void)
{
    int   啊 = A;
    printf("TWO: #    %d\n",
    啊);
    return 0;
}
```

对程序的翻译处理

- 第三，单词化
  - 处理注释：用一个空格代替每一条注释
  - 处理空白：用单个空格替代每一个空白字符（除换行符）
  - 得到记号序列：预处理记号序列、空白符序列及注释序列

```c
#include <stdio.h>
#define A 3 */*ddd*/3

int main(void)
{
    int   啊 = A;
    printf("TWO: #     %d\n",
    啊);
    return 0;
}
```

3. 单词化 →

```c
#include <stdio.h>
#define A 3 * 3

int main(void)
{
    int 啊 = A;
    printf("TWO: #     %d\n",
    啊);
    return 0;
}
```

对程序的翻译处理

- 第四，扩展宏和预处理指令处理
  - 格式：自 # 开始，至换行符结束

```
#include <stdio.h>
#define A 3 * 3

int main(void)
{
    int 啊 = A;
    printf("TWO: #     %d\n",
    啊);
    return 0;
}
```

4. 预处理

```
extern int printf (const char
*__restrict __format, ...);

int main(void)
{
    int 啊 = 3 * 3;
    printf("TWO: #     %d\n",
    啊);
    return 0;
}
```

```c
/* preproc.c -- simple preprocessor examples */
#include <stdio.h>
#define TWO 2          /* you can use comments if you like   */
#define OW "Consistency is the last refuge of the unimagina\
tive. - Oscar Wilde" /* a backslash continues a definition */
/* to the next line                              */
#define FOUR  TWO*TWO
#define PX printf("X is %d.\n", x)
#define FMT  "X is %d.\n"
int main(void)
{
    int x = TWO;
    PX;
    x = FOUR;
    printf(FMT, x);
    printf("%s\n", OW);
    printf("TWO: OW\n");
    return 0;
}
```

```c
int printf(const char* restrict, ...);

int main(void)
{
    int x = 2;
    printf("X is %d.\n", x);
    x = 2 * 2;
    printf("X is %d.\n", x);
    printf("%s\n", "Consistency is the
last refuge of the unimaginative. - Oscar
Wilde");
    printf("TWO: OW\n");
    return 0;
}
```

预处理

- 格式：自 # 开始，至换行符结束
  - 翻译处理已完成，斜杠紧跟换行符、注释均不受影响
  - ANSI C允许 # 前后有空格或制表符，但旧版C不允许
- 作用：在编译器处理程序前预扫描源代码，完成操作。

预处理功能	指令
头文件的包含	#include
宏扩展	#define #undef
条件编译	#if #ifdef #ifndef #else #elif #endif
行控制	#line
其它	#error #warning #pragma

宏定义 #define

- 宏（macro）的分类
  - 类对象宏

```
#define   PX   printf("X is %d.\n", x)
```

预编译指令　宏　替 换 体

  - 类函数宏

```
#define MEAN(X,Y) (((X)+(Y))/2)
```

- 每个#define行（逻辑行）由三个部分组成
  - 宏名：所选择的缩略语
    - 限字母（建议大写字母）、数字、下划线，且不以数字开头
  - 替换体（body）：指令行的其余部分

宏展开

- 宏展开：宏变成最终替换文本的过程
- 规则

规则	替换前	替换后
宏展开	`int x = TWO;`	`int x = 2;`
字符串中的宏不被替换	`printf("TWO: OW\n");`	`printf("TWO: OW\n");`
替换体中表达式不被计算	`int x = SQR_TWO;`	`int x = 2 * 2;`

- 宏定义可以包含其他宏（宏不能包括自己）
  - 部分编译器不支持该功能

```
#define FOUR  TWO * TWO
```

```
#define FOUR  FOUR * FOUR
```

宏的重定义

- 宏重定义：已定义的宏存在不同的宏展开
  - 记号：宏定义主体中单独的词（word），用空白分开

类型	代码	说明
原始定义	`#define X 2 * 2`	这是原始定义
相同定义	`#define X 2 * 2`	和原始定义完全相同是相同预定义
相同定义	`#define X 2   *   2`	预编译阶段，多个空白符替换成单个空格，因此相同。
重定义	`#define X 2*2`	预编译阶段，2 * 2是3个记号，2*2是1个记号
重定义	`#define X 2 * 3`	相同宏名且实体不同的定义

```
a.c:5:0: warning: "X" redefined [enabled by default]
 #define X 2*2
 ^
a.c:4:0: note: this is the location of the previous definition
 #define X 2 * 2
```

宏定义中使用参数

- 类函数宏（function-like macro）

左括号与宏名之间**不能**有空格

$$\underbrace{\texttt{\#define}}_{\text{预编译指令}}\ \underbrace{\texttt{MEAN(}\overset{\text{宏参数}}{\texttt{X,Y}}\texttt{)}}_{\text{宏}}\ \underbrace{\texttt{(((X)+(Y))/2)}}_{\text{替换主体}}$$

- 宏展开是预编译范畴，不是编译器范畴
  - 宏展开是简单替换，不会计算

预定义宏	245/SQR(5+2)的展开	值
#define SQR(X) X*X	245/5+2*5+2	245/5+2*5+2=61
#define SQR(X) (X)*(X)	245/(5+2)*(5+2)	245/(5+2)*(5+2)=245
#define SQR(X) ((X)*(X))	245/((5+2)*(5+2))	245/((5+2)*(5+2))=5

  - 注意对宏参数和替换列表分别加括号避免优先级错误

```
/* mac_arg.c -- macros with arguments */
#include <stdio.h>
#define SQUARE(X) X*X
#define PR(X)   printf("The result is %d.\n", X)

int main(void)
{
    int x = 5;
    int z;

    printf("x = %d\n", x);
    z = SQUARE(x);
    printf("Evaluating SQUARE(x): ");
    PR(z);
    z = SQUARE(2);
    printf("Evaluating SQUARE(2): ");
    PR(z);
    printf("Evaluating SQUARE(x+2): ");
    PR(SQUARE(x + 2));
```

```
z = x*x;
printf("Evaluating SQUARE(x): ");
printf("The result is %d.\n", z);
z = 2*2;
printf("Evaluating SQUARE(2): ");
printf("The result is %d.\n", z);
printf("Evaluating SQUARE(x+2): ");
```

```c
    printf("Evaluating 100/SQUARE(2): ");
    PR(100 / SQUARE(2));
    printf("x is %d.\n", x);
    printf("Evaluating SQUARE(++x): ");
    PR(SQUARE(++x));
    printf("After incrementing, x is %x.\n", x);

    return 0;
}
```

```c
printf("The result is %d.\n", x + 2*x + 2);
printf("Evaluating 100/SQUARE(2): ");
printf("The result is %d.\n", 100 / 2*2);
printf("x is %d.\n", x);
printf("Evaluating SQUARE(++x): ");
printf("The result is %d.\n", ++x*++x);
```

```
x = 5
Evaluating SQUARE(x): The result is 25.
Evaluating SQUARE(2): The result is 4.
Evaluating SQUARE(x+2): The result is 17.
Evaluating 100/SQUARE(2): The result is 100.
x is 5.
Evaluating SQUARE(++x): The result is 49.
After incrementing, x is 7.
```

宏定义中的#运算符

- 字符串化运算符：#
  - 字符串化运算符将参数转为字符串

```
#define PSQR(x) printf("Square of " #x " is %d.\n",((x)*(x)))
```

    - 例如：用PSQR(2 + 4)调用宏时，#x 被"2 + 4"替代

- 粘合剂运算符：##
  - 粘合剂运算符将两个语言符号组成一个语言符号
    - 如果该语言符号可以继续宏展开，将继续宏展开

```
#define PRINT_XN(n) printf("x" #n " = %d\n", x ## n);
```

    - 例如：PRINT_XN(1)展开为printf("x" "1" " = %d\n", x1)

```c
/* subst.c -- substitute in string */
#include <stdio.h>
#define PSQR(x) printf("The square of " #x " is %d.\n",((x)*(x)))

int main(void)
{
    int y = 5;

    PSQR(y);
    PSQR(2 + 4);

    return 0;
}
```

```
The square of y is 25.
The square of 2 + 4 is 36.
```

```c
printf("The square of " "y" " is %d.\n",((y)*(y)));
printf("The square of " "2 + 4" " is %d.\n",((2 + 4)*(2 + 4)));
```

```c
// glue.c -- use the ## operator
#include <stdio.h>
#define XNAME(n) x ## n
#define PRINT_XN(n) printf("x" #n " = %d\n", x ## n);

int main(void)
{
    int XNAME(1) = 14;
    int XNAME(2) = 20;
    int x3 = 30;
    PRINT_XN(1);
    PRINT_XN(2);
    PRINT_XN(3);
    return 0;
}
```

一般此替换主体不加末尾分号

```c
int main(void)
{
    int x1 = 14;
    int x2 = 20;
    int x3 = 30;
    printf("x" "1" " = %d\n", x1);;
    printf("x" "2" " = %d\n", x2);;
    printf("x" "3" " = %d\n", x3);;
    return 0;
}
```

```
x1 = 14
x2 = 20
x3 = 30
```

可变参数宏

- 可变参数宏：接受可变数量参数的宏

占位符	作用
...	在宏参数中在参数列表末尾，表示参数数量不确定
__VA_ARGS__	在宏展开中，表示可变长度参量列表
##__VA_ARGS__	在宏展开中，当可变参数为空时，前缀逗号也删除

- 可变参数宏用于展开可变参数函数

项目	内容
宏	`#define PR(FMT, ...) printf(FMT, ##__VA_ARGS__)`
`PR("s");`	`printf("s");`
`PR("%d",3);`	`printf("%d",3);`

```c
// variadic.c -- variadic macros
#include <stdio.h>
#include <math.h>
#define PR(X, ...) printf("Message " #X ": " __VA_ARGS__)

int main(void)
{
    double x = 48;
    double y;

    y = sqrt(x);
    PR(1, "x = %g\n", x);
    PR(2, "x = %.2f, y = %.4f\n", x, y);

    return 0;
}
```

```
Message 1: x = 48
Message 2: x = 48.00, y = 6.9282
```

```c
printf("Message " "1" ": " "x = %g\n", x);
printf("Message " "2" ": " "x = %.2f, y = %.4f\n", x, y);
```

编译器内置宏

- ANSI C标准预定义宏

宏名	含义	实例
__LINE__	当前源代码行号	8
__FILE__	当前源文件名	"sample.c"
__DATE__	当前的编译日期	"Dec 10 2023"
__TIME__	当前编译时间	"21:03:13"
__STDC_VERSION__	C语言标准版本	201710L

- 编译器内置宏
  - 命令行查看内置宏

```
gcc -dM -E - < /dev/null
```

  - 命令行 -D 定义宏

```
gcc -DONLINE_JUDGE
```

```c
// predef.c -- predefined identifiers
#include <stdio.h>
void why_me();
int main()
{
    printf("The file is %s.\n", __FILE__);
    printf("The date is %s.\n", __DATE__);
    printf("The time is %s.\n", __TIME__);
    printf("The version is %ld.\n", __STDC_VERSION__);
    printf("This is line %d.\n", __LINE__);
    printf("This function is %s\n", __func__);
    why_me();
    return 0;
}


void why_me()
{
    printf("This function is %
    printf("This is line %d.\n", __LINE__);
}
```

```
gcc predef.c -o predef -std=c11 ↲
The file is predef.c.
The date is Aug 19 2014.
The time is 22:17:00.
The version is 201112.
This is line 11.
This function is main
This function is why_me
This is line 21.
```

内联函数：inline

扩展内容

- 内联函数
  - 定义：一种程序结构，建议编译器对函数进行内联扩展，将函数体插入并取代每一处调用该函数的地方
  - 目的：节省每次调用函数带来的额外时间开支。
  - 决定权：由编译器自行决定是否内联，不是程序员决定
- 特殊性
  - 调用前应先定义（尤其头文件），不能只声明原型
  - 应针对短小的程序，长的程序调用不花多少时间
  - 不能取地址，不能递归调用

宏、内联和普通函数比较

扩展内容

- 使用宏的考虑
  - 宏产生内联代码，但是简单展开
- 使用内联函数的考虑
  - 宏主体应仔细书写，稍有不慎会产生奇怪现象
- 使用函数的考虑
  - 宏与函数实际上是时间与空间的权衡
  - 调用次数少，执行时间长，使用宏不会更省时
  - 函数只有一个副本，但需要控制权转换

内容纲要

文件包含 #include

- 格式

```
#include 头文件路径
```

文件名格式	说明	示例
尖括号	仅在默认路径中寻找	`#include <stdio.h>`

双引号	在当前路径中寻找，当前路径不存在则寻找默认路径	`#include "my.h"`
		`#include "/usr/my.h"`

- 作用：将指定的文件内容替换文件包含指令
- 包含头文件并不会显著增加程序的大小
  - 无关的部分不会编译进可执行程序
- 多次包含同一个头文件可能产生重定义等错误
  - 需要通过条件编译或#pragma once避免重复包含

文件包含

内容	示例
明显常量	`#define EOF (-1)`

宏函数	`#define getchar()  getc(stdin)`
	`#define putchar(x)  putc(x, stdout)`

函数声明	`FILE* fopen(const char* __restrict _name, const char* __restrict _type);`

结构模板定义	`struct __sFILE {`
	`    unsigned char* _p;`
	`    int _r;`
	`    int _w;`
	`    …`
	`}`

类型定义	`typedef __FILE FILE;`
外部变量	`extern struct _reent* _impure_ptr;`

```c
// names_st.h -- names_st structure header file
// constants
#include <string.h>
#define SLEN 32

// structure declarations
struct names_st
{
    char first[SLEN];
    char last[SLEN];
};

// typedefs
typedef struct names_st names;

// function prototypes
void get_names(names*);
void show_names(const names*);
char* s_gets(char* st, int n);
```

```c
// names_st.c -- define names_st functions
#include <stdio.h>
#include "names_st.h"     // include the header file

// function definitions
void get_names(names* pn)
{
    printf("Please enter your first name: ");
    s_gets(pn->first, SLEN);

    printf("Please enter your last name: ");
    s_gets(pn->last, SLEN);
}

void show_names(const names* pn)
{
    printf("%s %s", pn->first, pn->last);
}
```

```c
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
// useheader.c -- use the names_st structure
#include <stdio.h>
#include "names_st.h"
// remember to link with names_st.c

int main(void)
{
    names candidate;
    get_names(&candidate);
    printf("Let's welcome ");
    show_names(&candidate);
    printf(" to this program!\n");
    return 0;
}
```

```
$ gcc names_st.c useheader.c -o name↲
$ ./name↲
Please enter your first name: Wei↲
Please enter your last name: Huang↲
Let's welcome Wei Huang to this program!
```

内容纲要

条件编译指令

扩展内容

- **格式**

```
if-part
[ elif-parts ]
[ else-part ]
endif-line
```

成分	说明
if-part	*if-line* 文本
if-line	#if <常量表达式> #ifdef <标识符> #ifndef <标识符>

elif-parts	*elif-line* 文本
	elif-parts elif-line 文本

else-line	#elif <常量表达式>
else-part	*else-line* 文本
else-line	#else
endif-line	#endif

- **预处理器运算符**

```
defined <宏名>
```

  – 判断是否有宏定义，#if defined相当于#ifdef

条件编译指令

扩展内容

- 示例

```
#if defined(CRD)
    credit();
#elif defined(DBT)
    debit();
#else
    printerror();
#endif
```

```
#if DLEVEL > 5
#define SIGNAL  1
#if STACKUSE == 1
#define STACK   200
#else
#define STACK   100
#endif
#else
#define SIGNAL  0
#endif
```

```
#ifndef unix
#define WIN32
#include <windows.h>
#include <winsock.h>
#else
#dundef WIN32
#define closesocket close
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#endif
```

其它指令

扩展内容

- 指令列表

名称	格式	示例
取消定义	#undef <宏名>	#undef WIN32
指定行号	#line <行号> [<文件名>]	#line 9 "a.c"
编译警告	#warning <消息>	#warning Warning!
编译错误	#error <消息>	#error OpenCV 4 required

- 编译指示：#pragma
  - 每个编译器都有自己的编译指示集
  - C99提供了_Pragma预处理器运算符

```
#pragma once
#pragma comment ( lib, "wpcap.lib" )
```

```c
/* test.c -- simple macro definitions */
#include <stdio.h>

int main(void)
{
#line 9 "a.c"
    printf("%s, %s, %s, %s\n", __FILE__, __DATE__, __TIME__, __func__);
    printf("%d\n", __LINE__);
    return 0;
}
```

```
a.c, Feb  1 2024, 17:34:13, main
10
```

```c
/* ifdef.c -- uses conditional compilation */
#include <stdio.h>
#define JUST_CHECKING
#define LIMIT 4
int main(void)
{
    int i;
    int total = 0;
    for (i = 1; i <= LIMIT; i++)
    {
        total += 2 * i * i + 1;
#ifdef JUST_CHECKING
        printf("i=%d, running total = %d\n", i, total);
#endif
    }
    printf("Grand total = %d\n", total);
    return 0;
}
```

```
i=1, running total = 3
i=2, running total = 12
i=3, running total = 31
i=4, running total = 64
Grand total = 64
```

```c
// names.h --revised with include protection
#ifndef NAMES_H_
#define NAMES_H_
// constants
#define SLEN 32
// structure declarations
struct names_st
{
    char first[SLEN];
    char last[SLEN];
};
// typedefs
typedef struct names_st names;
// function prototypes
void get_names(names *);
void show_names(const names *);
char * s_gets(char * st, int n);
#endif
```

```c
// doubincl.c -- include header twice
#include <stdio.h>
#include "names.h"
#include "names.h"   // accidental second inclusion

int main()
{
    names winner = {"Less", "Ismoor"};
    printf("The winner is %s %s.\n", winner.first,
           winner.last);
    return 0;
}
```

```
The winner is Less Ismoor.
```

内容纲要

C库简介

- 发展史
  - 最初的C没有官方库
  - UNIX C成为事实标准，ANSI认为需要官方库
  - 随着应用范围扩大，该会重定义了库
- 系统使用不同的方法搜索C库的函数
  - 自动访问：常见的库函数自动可用，只需包含头文件
  - 文件包含：定义为宏的函数，通过包含文件而展开
  - 库包含：不常用的函数库，必须显式指定库名

库函数分类

- 分类（不完整）

类别	头文件	类别	头文件
断言	assert.h	字符分类	ctype.h
浮点数	float.h	缓冲相关读写	io.h
数学	math.h	内存	mem.h
多线程	pthread.h	正则表达式	regex.h
转移进程	setjmp.h	标准输入输出	stdio.h
标准库	stdlib.h	信号	signal.h
字符串	string.h	时间	time.h
宽字符	wchar.h		

数学库

- 头文件：math.h
- 编译指令中使用-lm连接数学库

```c
/* rect_pol.c -- converts rectangular coordinates to polar */
#include <stdio.h>
#include <math.h>
#define RAD_TO_DEG (180/(4 * atan(1)))
typedef struct polar_v {
    double magnitude;
    double angle;
} Polar_V;
typedef struct rect_v {
    double x;
    double y;
} Rect_V;
Polar_V rect_to_polar(Rect_V);
int main(void) {
    Rect_V input;
    Polar_V result;
    puts("Enter x and y coordinates; enter q to quit:");
    while (scanf("%lf %lf", &input.x, &input.y) == 2) {
        result = rect_to_polar(input);
        printf("magnitude = %0.2f, angle = %0.2f\n",
               result.magnitude, result.angle);
    }
    puts("Bye.");
    return 0;
}
Polar_V rect_to_polar(Rect_V rv) {
    Polar_V pv;
    pv.magnitude = sqrt(rv.x * rv.x + rv.y * rv.y);
    if (pv.magnitude == 0)
        pv.angle = 0.0;
    else
        pv.angle = RAD_TO_DEG * atan2(rv.y, rv.x);
    return pv;
}
```

```
Enter x and y coordinates; enter q to quit:
23 43↲
magnitude = 48.76, angle = 61.86
65 9 ↲
magnitude = 65.62, angle = 7.88
q ↲
Bye.
```

通用工具库

扩展内容

- exit()和atexit()函数
- qsort()函数
- 请自习

```c
//  generic.c  -- defining generic macros
#include <stdio.h>
#include <math.h>
#define RAD_TO_DEG (180/(4 * atanl(1)))

// generic square root function
#define SQRT(X) _Generic((X),\
    long double: sqrtl, \
    default: sqrt, \
    float: sqrtf)(X)

// generic sine function, angle in degrees
#define SIN(X) _Generic((X),\
long double: sinl((X)/RAD_TO_DEG),\
default:     sin((X)/RAD_TO_DEG),\
float:       sinf((X)/RAD_TO_DEG)\
)
```

```c
int main(void)
{
    float x = 45.0f;
    double xx = 45.0;
    long double xxx =45.0L;
    long double y = SQRT(x);
    long double yy= SQRT(xx);
    long double yyy = SQRT(xxx);
    printf("%.17Lf\n", y);   // matches float
    printf("%.17Lf\n", yy);  // matches default
    printf("%.17Lf\n", yyy); // matches long double
    int i = 45;
    yy = SQRT(i);            // matches default
    printf("%.17Lf\n", yy);
    yyy= SIN(xxx);           // matches long double
    printf("%.17Lf\n", yyy);
    return 0;
}
```

```c
/* byebye.c -- atexit() example */
#include <stdio.h>
#include <stdlib.h>
void sign_off(void);
void too_bad(void);
int main(void)
{
    int n;
    atexit(sign_off);    /* register the sign_off() function */
    puts("Enter an integer:");
    if (scanf("%d",&n) != 1)
    {
        puts("That's no integer!");
        atexit(too_bad); /* register the too_bad()  function */
        exit(EXIT_FAILURE);
    }
    printf("%d is %s.\n", n,  (n % 2 == 0)? "even" : "odd");
    return 0;
}
```

```c
void sign_off(void)
{
    puts("Thus terminates another magnificent program from");
    puts("SeeSaw Software!");
}

void too_bad(void)
{
    puts("SeeSaw Software extends its heartfelt condolences");
    puts("to you upon the failure of your program.");
}
```

```
Enter an integer:
56↲
56 is even.
Thus terminates another magnificent program from
SeeSaw Software!
```

```c
/* qsorter.c -- using qsort to sort groups of numbers */
#include <stdio.h>
#include <stdlib.h>
#define NUM 40
void fillarray(double ar[], int n);
void showarray(const double ar[], int n);
int mycomp(const void * p1, const void * p2);

int main(void)
{
    double vals[NUM];
    fillarray(vals, NUM);
    puts("Random list:");
    showarray(vals, NUM);
    qsort(vals, NUM, sizeof(double), mycomp);
    puts("\nSorted list:");
    showarray(vals, NUM);
    return 0;
}
```

```c
void fillarray(double ar[], int n)
{
    int index;
    for( index = 0; index < n; index++)
        ar[index] = (double)rand()/((double) rand() + 0.1);
}

void showarray(const double ar[], int n)
{
    int index;
    for( index = 0; index < n; index++)
    {
        printf("%9.4f ", ar[index]);
        if (index % 6 == 5)
            putchar('\n');
    }
    if (index % 6 != 0)
        putchar('\n');
}
```

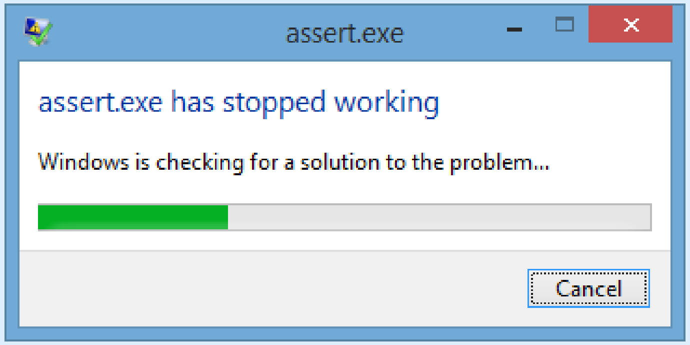
```c
/* sort by increasing value */
int mycomp(const void * p1, const void * p2)
{
    /* need to use pointers to double to access values   */
    const double * a1 = (const double *) p1;
    const double * a2 = (const double *) p2;

    if (*a1 < *a2)
        return -1;
    else if (*a1==*a2)
        return 0;
    else
        return 1;
}
```

```
Random list:
    0.0022    0.2390    1.2191    0.3910    1.1021    0.2027
    1.3835   20.2830    0.2508    0.8880    2.2179   25.4866
    0.0236    0.9308    0.9911    0.2507    1.2802    0.0939
    0.9760    1.7217    1.2054    1.0326    3.7892    1.9635
    4.1137    0.9241    0.9971    1.5582    0.8955   35.3798
    4.0579   12.0460    0.0096    1.0109    0.8506    1.1529
    2.3614    1.5876    0.4825    6.8749

Sorted list:
    0.0022    0.0096    0.0236    0.0939    0.2027    0.2390
    0.2507    0.2508    0.3910    0.4825    0.8506    0.8880
    0.8955    0.9241    0.9308    0.9760    0.9911    0.9971
    1.0109    1.0326    1.1021    1.1529    1.2054    1.2191
    1.2802    1.3835    1.5582    1.5876    1.7217    1.9635
    2.2179    2.3614    3.7892    4.0579    4.1137    6.8749
   12.0460   20.2830   25.4866   35.3798
```

```c
/* assert.c -- use assert() */
#include <stdio.h>
#include <math.h>
#include <assert.h>
int main()
{
    double x, y, z;
    puts("Enter a pair of numbers (0 0 to quit): ");
    while (scanf("%lf%lf", &x, &y) == 2
           && (x != 0 || y != 0))
    {
        z = x * x - y * y;  /* should be + */
        assert(z >= 0);
        printf("answer is %f\n", sqrt(z));
        puts("Next pair of numbers: ");
    }
    puts("Done");
    return 0;
}
```



assert.exe has stopped working

Windows is checking for a solution to the problem...

```
Enter a pair of numbers (0 0 to quit):
32↲
39↲
Assertion failed: z >= 0, file assert.c, line 14
```

```
Enter a pair of numbers (0 0 to quit):
6 3↲
answer is 5.196152
Next pair of numbers:
0 0↲
Done
```

```c
//  statasrt.c
#include <stdio.h>
#include <limits.h>
_Static_assert(CHAR_BIT == 16, "16-bit char falsely assumed");
int main(void)
{
    puts("char is 16 bits.");
    return 0;
}
```

```
gcc statasrt.c↲
statasrt.c:4:1: error: static assertion failed: "16-bit
char falsely assumed"
 _Static_assert(CHAR_BIT == 16, "16-bit char falsely
assumed");
```

```c
// mems.c -- using memcpy() and memmove()
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define SIZE 10
void show_array(const int ar[], int n);
// remove following if C11 _Static_assert not supported
_Static_assert(sizeof(double) == 2 * sizeof(int), "double not
twice int size");

int main()
{
    int values[SIZE] = {1,2,3,4,5,6,7,8,9,10};
    int target[SIZE];
    double curious[SIZE / 2] = {2.0, 2.0e5, 2.0e10, 2.0e20,
5.0e30};

    puts("memcpy() used:");
    puts("values (original data): ");
    show_array(values, SIZE);
    memcpy(target, values, SIZE * sizeof(int));
    puts("target (copy of values):");
    show_array(target, SIZE);
    puts("\nUsing memmove() with overlapping ranges:");
    memmove(values + 2, values, 5 * sizeof(int));
    puts("values -- elements 0-5 copied to 2-7:");
    show_array(values, SIZE);

    puts("\nUsing memcpy() to copy double to int:");
    memcpy(target, curious, (SIZE / 2) * sizeof(double));
    puts("target -- 5 doubles into 10 int positions:");
    show_array(target, SIZE/2);
    show_array(target + 5, SIZE/2);

    return 0;
}
```

```c
void show_array(const int ar[], int n)
{
    int i;

    for (i = 0; i < n; i++)
        printf("%d ", ar[i]);
    putchar('\n');
}
```

```
dev@ubuntu:~/Desktop$ gcc mems.c↲
dev@ubuntu:~/Desktop$ ./a.out↲
memcpy() used:
values (original data):
1 2 3 4 5 6 7 8 9 10
target (copy of values):
1 2 3 4 5 6 7 8 9 10

Using memmove() with overlapping ranges:
values -- elements 0-5 copied to 2-7:
1 2 1 2 3 4 5 8 9 10

Using memcpy() to copy double to int:
target -- 5 doubles into 10 int positions:
0 1073741824 0 1091070464 536870912
1108516959 2025163840 1143320349 -2012696540 1179618799
```

```c
//varargs.c -- use variable number of arguments
#include <stdio.h>
#include <stdarg.h>
double sum(int, ...);

int main(void)
{
    double s,t;

    s = sum(3, 1.1, 2.5, 13.3);
    t = sum(6, 1.1, 2.1, 13.1, 4.1, 5.1, 6.1);
    printf("return value for "
           "sum(3, 1.1, 2.5, 13.3):                %g\n", s);
    printf("return value for "
           "sum(6, 1.1, 2.1, 13.1, 4.1, 5.1, 6.1): %g\n", t);

    return 0;
}
```

```c
double sum(int lim,...)
{
    va_list ap;           // declare object to hold arguments
    double tot = 0;
    int i;

    va_start(ap, lim);    // initialize ap to argument list
    for (i = 0; i < lim; i++)
        tot += va_arg(ap, double); // access each item in
argument list
    va_end(ap);                    // clean up

    return tot;
}
```

```
return value for sum(3, 1.1, 2.5, 13.3):                16.9
return value for sum(6, 1.1, 2.1, 13.1, 4.1, 5.1, 6.1): 31.6
```

库文件归纳说明

扩展内容

- 具体有哪些常用的库函数，很难总结归纳
- 多阅读，出现多的即为常用

C 程 序 设 计
C Programming

16

谢谢观看

理论课程

厦門大學
XIAMEN UNIVERSITY

信 息 学 院
(特色化示范性软件学院)
School of Informatics

黄 炜
博士, 副教授
Wei Huang