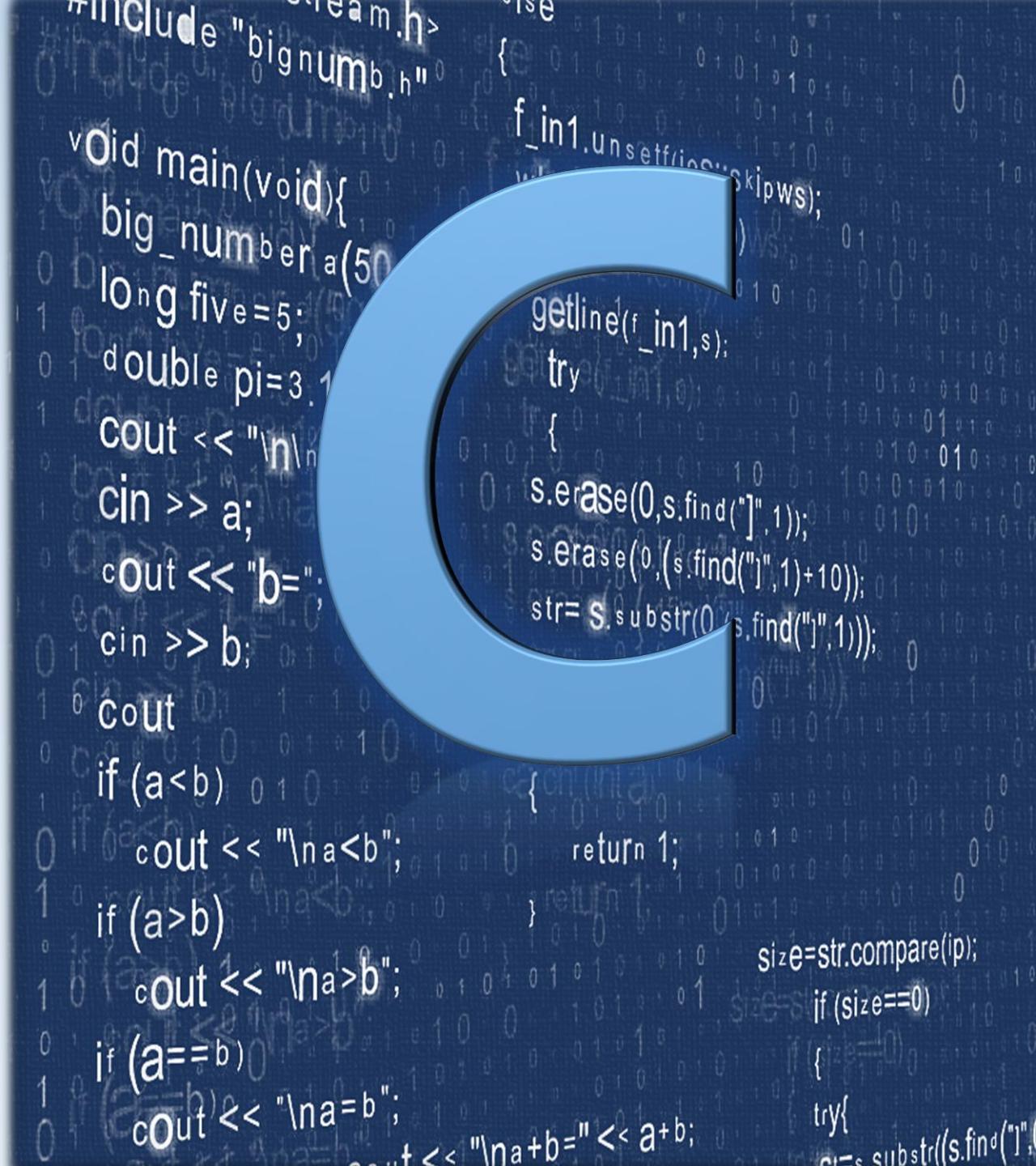


《C语言程序设计》

C语言课程组

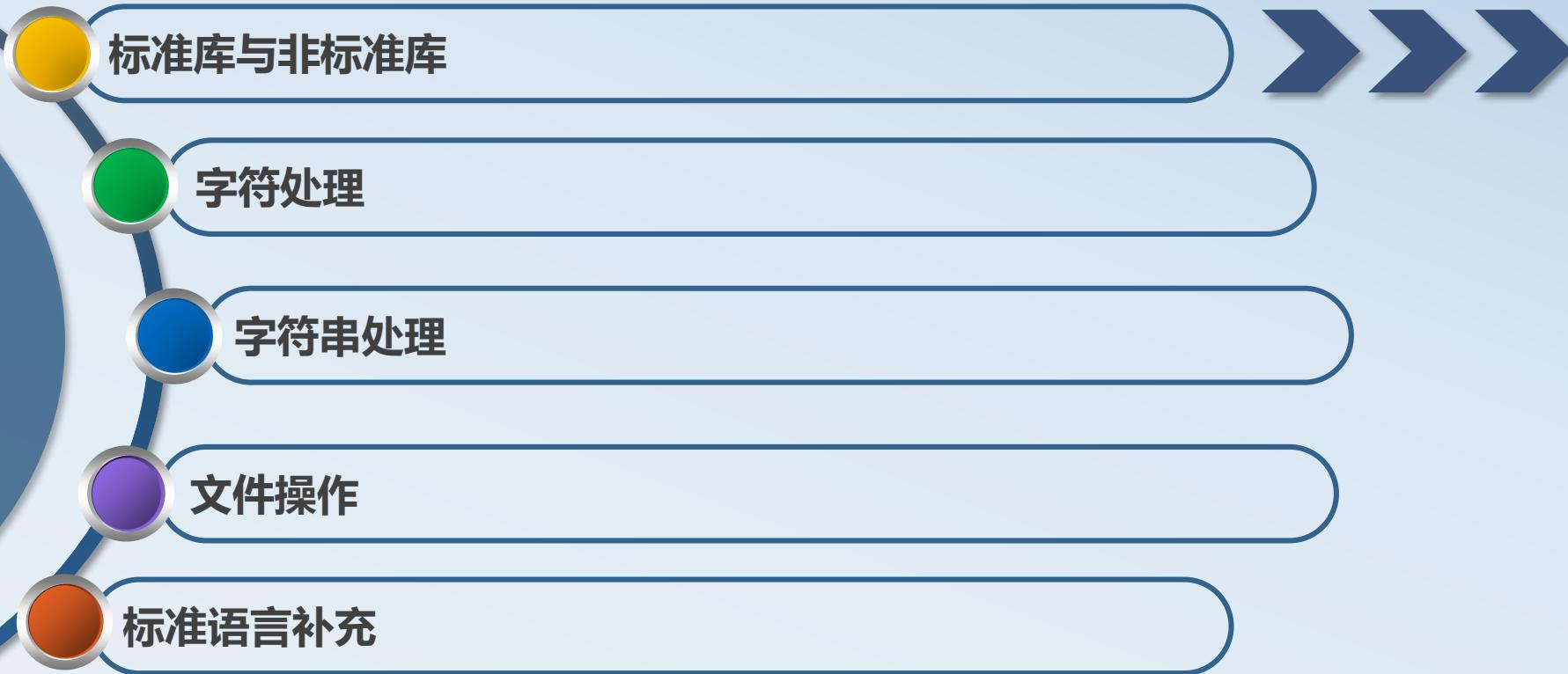


```
#include "stream.h"
#include "bignum.h"
{
    f_in1.unsetf(ios::skipws);
}
getline(f_in1,s);
try
{
    s.erase(0,s.find("]"+1));
    s.erase(0,(s.find("]"+1)+10));
    str=s.substr(0,s.find("]"+1));
    size=str.compare(ip);
    if (size==0)
    {
        try{
            t<<"\n a+b=" << a+b;
        }
    }
}
```

本讲教学目标

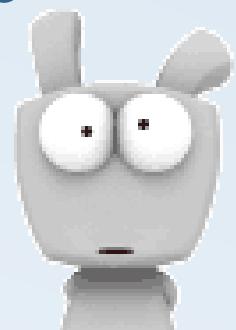
- ◆ 掌握标准库的作用。
- ◆ 了解有哪些标准头文件，各头文件中主要声明什么样的函数。
- ◆ 掌握字符及字符串处理的标准库函数的用法。
- ◆ 掌握内存管理方面的标准库函数的用法。
- ◆ 掌握文件操作的标准库函数，掌握文件与内存间数据“交换”的方法。
- ◆ 掌握标准输入输出函数的使用方法及区别。
- ◆ 了解标准语言补充方面的相关知识。

本讲授课内容



标准库与非标准库

问题：
• C语言还能做什么呢？



标准库与非标准库

- ❖ C语言的标准库：
 - ◆ ".h" 的头文件
 - ◆ 函数库

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main(void)
{
    int x = -9;
    /* printf是标准库函数 */
    printf("x的绝对值是: %d\n", abs(x));
    printf("x绝对值开根号的结果是: %.2f\n", sqrt(abs(x)));

    /* system是标准库函数 */
    system("PAUSE");
    return 0;
}
```

标准库与非标准库

- ❖ C语言的标准头文件：
 - ◆ assert.h : 包含定义 assert 调试宏。
 - ◆ complex.h (C99 新增) : 包含复数算术运算函数
 - ◆ ctype.h (ANSI C) : 包含有关字符分类及转换的函数(如 isalpha 和toascii 等)。
 - ◆ errno.h : 包含通过错误代码报告错误发生条件的宏定义。
 - ◆ fenv.h : 包含各种操作浮点环境的函数及宏。
 - ◆ float.h : 包含有关浮点运算的一些宏和函数原型。
 - ◆ inttype.h (C99) : 包含大量有关printf、 scanf 系列函数使用的宏定义，及与intmax_t类型有关的函数原型。

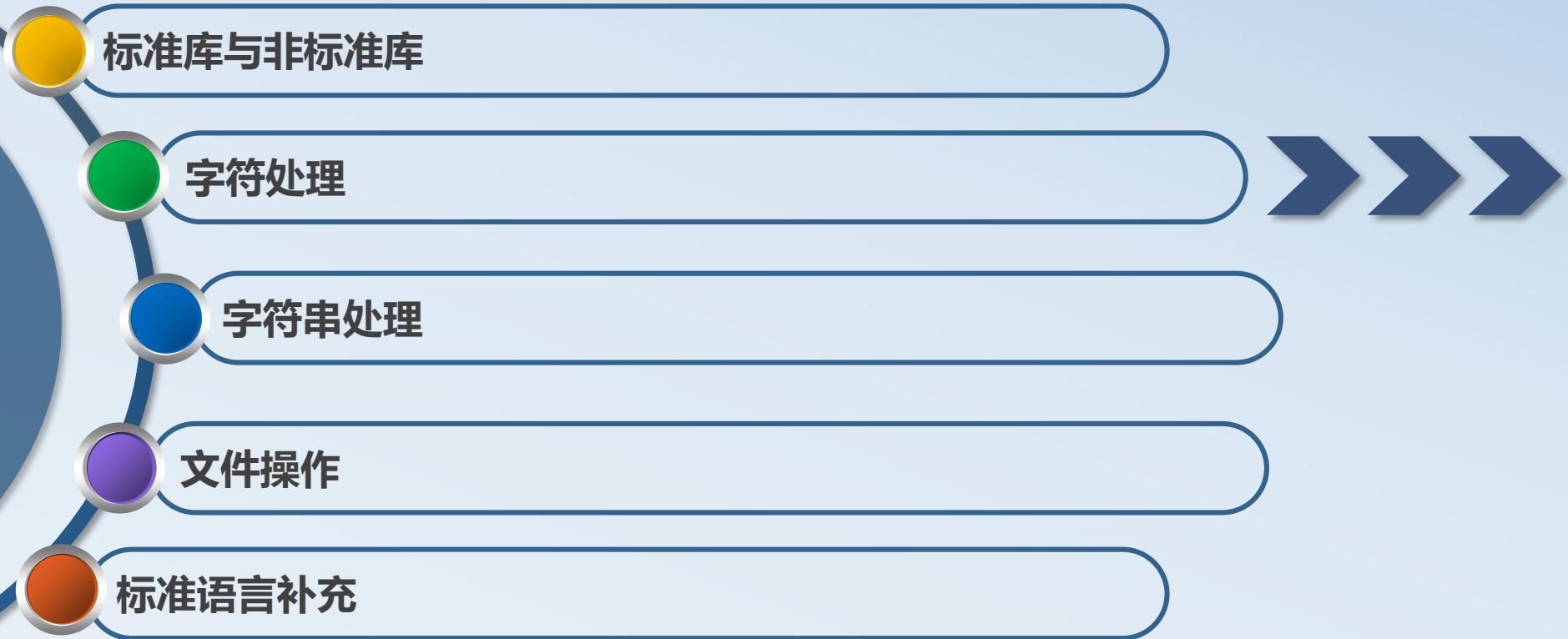
标准库与非标准库

- ◆iso646.h (C95): 包含大量的宏定义以允许程序员方便地使用C语言的位及逻辑操作符。
- ◆limits.h: 包含各环境参数、编译时间限制、数的范围等信息。
- ◆locale.h: 提供两个关键函数: localconv 和setlocale。以及, 定义了一个结构体struct
- ◆lconv。这两个函数及结构体都与区域设置有关。
- ◆math.h: 包含数学运算函数原型, 大多数函数涉及到浮点类型的数。
- ◆setjmp.h: 主要包含两个函数: setjmp、longjmp用于实现“非本地跳转”。主要用于进行异常处理
- ◆singal.h: 包含程序在执行过程中如何处理信号的函数原型。

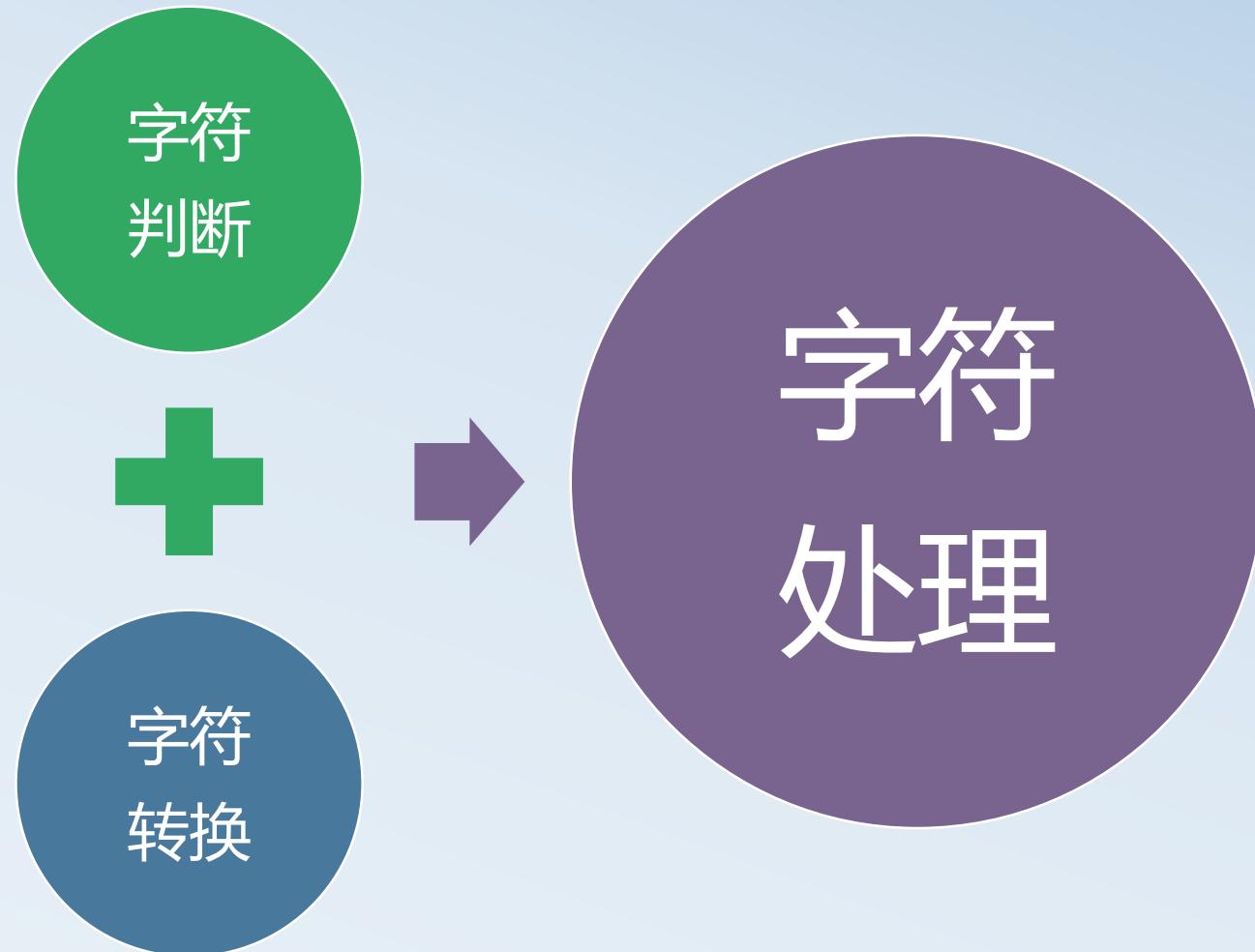
标准库与非标准库

- ◆ stdarg.h: 包含与可变参数有关的宏及函数定义。这使得C语言的函数形参个数可以是不确定的。
- ◆ stdbool.h (C99): 包含4个宏: bool、true、false、_bool_true_false_are_defined。
- ◆ stddef.h: 包含宏: NULL、offsetof 及类型: ptrdiff_t、wchar_t 及size_t。
- ◆ stdint.h (C99): 包含精确宽度的整型类型，及每一种整型类型的最大、最小值的宏定义。
- ◆ stdio.h: 包含针对各种输入输出操作的宏定义、字面值、函数原型及类型定义。

本讲授课内容



字符处理



字符处理

ctype.h 中声明的判断类的部分标准库函数

函数名	功能
isalnum	判断形参 c 是否是字母或数字字符，即：是否是 0~9、A~Z、a~z 中的一个
isalpha	判断形参 c 是否是 A~Z、a~z 中的一个
isctrl	判断形参 c 是否是“控制字符”
isdigit	判断形参 c 是否是 10 个十进制数字之一
isodigit	判断形参 c 是否是 8 个 8 进制数字之一
isxdigit	判断形参 c 是否是 0~9、A~F、a~f 中的一个
isgraph	判断形参 c 是否是图形字符的代码（除空格以外的任何打印字符）
isprint	判断形参 c 是否是可打印字符
ispunct	判断形参 c 是否是标点符号
islower	判断形参 c 是否是 26 个小写字母之一
isupper	判断形参 c 是否是 26 个大写字母之一
isblank	判断形参 c 是否是：标准空白符、空格、't'中的一个
isspace	判断形参 c 是否是：'t'、'r'、'n'、'v'、'f'和空格中的一个

ctype.h 中声明的转换类的部分标准库函数

函数名	功能
tolower	将形参 c 转换成小写字母，并返回
toupper	将形参 c 转换成大小字母，并返回

字符处理

❖例 编写一个对字符串进行加密的函数，形参为待加密的字符串及存放加密后的字符串。加密的规则为：对于英文字母则大写变小写、小写变大写；对非英文字符则保持不变

字符处理

```
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include <ctype.h>
#define MAX_LENGTH 1000
char * encrypt(const char *src, char *dest)
{
    char c = *src;
    assert(src && dest);
    do
    {
        c = *src++;
        if(isalpha(c) && isupper(c))
            *dest++ = tolower(c);
        elseif(isalpha(c) && islower(c))
            *dest++ = toupper(c);
        else
            *dest++ = c;
    }while('\0' != c);
    return dest;
}
```

字符处理

```
int main(void)
{
    char src[MAX_LENGTH] = "abCDef14G";
    char dest[MAX_LENGTH];
    encrypt(src, dest);
    printf("%s加密后: %s\n",src,dest);
    system("PAUSE");
    return 0;
}
```

本讲授课内容



字符串处理

- ❖ 字符串指： **字符串字面值或以'\\0'为最后一个元素的字符数组。**
- ❖ 例 观察下面的字符串。
 - ❖ 字符串字面值：“abcd”。编译器会自动在字面值“abcd”最后加上'\\0'。
 - ❖ 字符数组： `char str1[] = "abcd";`， 编译器会在字面值“abcd”最后加上'\\0'。
 - ❖ 字符数组： `char str2[] = { 'a', 'b', 'c', 'd', '\\0' };`， 这种初始化方式要自己置'\\0'。

字符串处理

❖ 字符串处理函数大多数在string.h文件中声明

string.h 中声明的用于进行字符串处理的部分标准库函数

函数原型	功能简介
char *strcat(char *dest, const char *src)	将 src 的内容添加到 dest 末尾
char *strncat(char *dest, const char *src, size_t n)	将 src 中前 n 个字符添加到 dest 末尾
int strcmp(const char *s1, const char *s2);	比较 s1 和 s2 的大小
int strncmp(const char *s1, const char *s2, size_t n);	比较 s1 和 s2 中前 n 个字符的大小
char *strcpy(char *dest, const char *src);	将 src 的内容复制到 dest 中，覆盖 dest 原内容
char *strncpy(char *dest, const char *src, size_t n);	将 src 的前 n 个字符复制到 dest 中，覆盖 dest 原内容
size_t strlen(const char *s);	返回 s 中，'\0'之前的字符数
char *strchr(const char *s, int c);	返回指向 s 中第一次出现的 c 的指针
char * strrchr(const char *s, int c);	返回指向 s 中最后一次出现字符 c 的指针
size_t strspn(const char *s, const char *set);	搜索 s 中第一个不包含在 set 中的字符
size_t strcspn(const char *s, const char *set);	搜索 s 中第一个在 set 中的字符
char * strtok(char *str, const char *set);	将 str 分解为由 set 中的字符分隔的若干个子串
char * strstr(const char *src, const char *sub);	找到 src 中第一次出现 sub 的位置

stdio.h 中声明的用于进行字符串处理的部分标准库函数

函数原型	功能简介
char* gets(char* s)	从键盘上接收字符串到 s 中
int puts(char* s)	将 s 中内容输出到屏幕上

字符串处理

❖例 编一个函数，从给定字符串（由形式参数给出）中去掉标点符号（由形式参数给出）。

```
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include <string.h>
#include <memory.h>
#define MAX_LENGTH 1000
char * erase(char *src, const char *set)
{
    char *word;
    char *tmpDest =(char *) malloc(strlen(src) + 1);
    tmpDest[0] = '\0';
    assert(src);
    if(!set)
        return src;
    word = strtok(src, set);
```

字符串处理

```
while(NULL != word)
{
    strcat(tmpDest, word);
    word = strtok(NULL, set);
}
strcpy(src, tmpDest);
free(tmpDest);
return src;
}
int main(void)
{
    char src[MAX_LENGTH] = "This is an example, given by Wenbin.";
    puts(src);
    erase(src, ",.");
    puts(src);
    return 0;
}
```

字符串处理

❖例 有5个国家的名称，存放在字符数组中，请编写函数按从大到小的顺序排列并输出它们。

```
#include <stdio.h>    #include <stdlib.h>    #include <assert.h>
#include <string.h>    #include <memory.h>
#define MAX_LENGTH 100
void sortCountry(char (*country)[MAX_LENGTH], int countryCount)
{
    int i = 0;
    int j = 0;
    char *c = (char *)malloc(MAX_LENGTH + 1);
    assert(country);
    if(0 >= countryCount || MAX_LENGTH <= countryCount)
    {
        printf("国家数超过了处理的范围");
        return;
    }
```

字符串处理

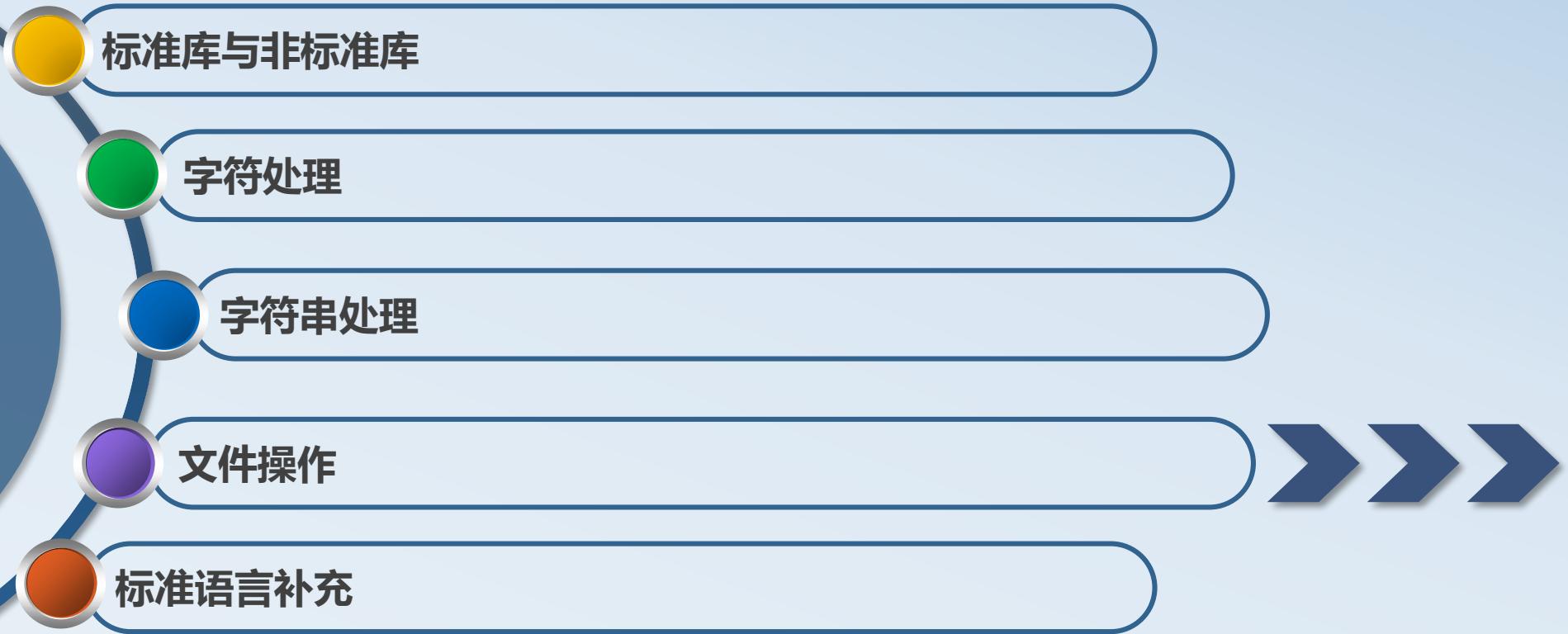
```
for(i = 0; i < countryCount - 1; i++)
{
    for(j = 0; j < countryCount - 1 - i; j++)
    {
        if(strcmp(country[j], country[j+1]) > 0)
        {
            strcpy(c, country[j]);
            strcpy(country[j], country[j+1]);
            strcpy(country[j+1], c);
        }
    }
    free(c);
}
```

字符串处理

```
int main(void)
{
    char country[][MAX_LENGTH]={"Papua New Guinea",
        "Albania", "Colombia", "Kyrgyzstan", "China"};
    int countryCount = 5;
    int i = 0;
    /*排序*/
    sortCountry(&country[0], countryCount);
    for(i = 0; i < countryCount; i++)
        puts(country[i]);

    return 0;
}
```

本讲授课内容



文件操作

- ❖ 所谓“文件”一般是指存储在外部介质上数据的集合。
- ❖ 基于数据流的概念，C语言提供了丰富的输入/输出函数
 - ◆ 文本流
 - ◆ 二进制流

文件操作

❖ 文件的打开与关闭函数： **fopen** 生成的 FILE 类型的对象才是有意义的.

函数原型	描述	
<code>FILE *fopen(const char * restrict filename, const char * restrict mode);</code>	找开文件, filename 代表文件名, mode 代表打开方式	
<code>int fclose(FILE * restrict stream);</code>	关闭文件, stream 代表要关闭的文件	
适用文件类型	方式	意义
<hr/>		
文本文件 (文件流)	r	打开现有文件以便读取
	w	若指定文件存在, 清空原内容后从头写; 否则生成新文件写。
	a	若指定文件存在, 从原内容后写; 否则生成新文件写。
	r+	打开现有文件, 从文件开头位置开始读/写
	w+	若指定文件存在, 清空原内容后从头读/写; 否则生成新文件读/写。
	a+	若指定文件存在, 在原内容后写或从开头处读; 否则生成新文件读/写。
<hr/>		
二进制文件 (二进制流)	rb	打开现有文件以便读取
	wb	若指定文件存在, 清空原内容后从头写; 否则生成新文件写。
	ab	若指定文件存在, 从原内容后写; 否则生成新文件写。
	rb+	打开现有文件, 从文件开头位置开始读/写
	wb+	若指定文件存在, 清空原内容后从头读/写; 否则生成新文件读/写。
	ab+	若指定文件存在, 在原内容后写或从开头处读; 否则生成新文件读/写。

文件操作

❖例 设有FILE *fp;, 且c 盘根目录下存在a.txt 文件,
文件内容为 “This is an example.”

<code>fp = fopen("c:\\a.txt", "r");</code>	<code>fp = fopen("c:\\a.txt", "w");</code>	<code>fp = fopen("c:\\a.txt", "a");</code>
<code>fp = fopen("c:\\b.txt", "r+");</code>	<code>fp = fopen("c:\\a.txt", "r+");</code>	<code>fp = fopen("c:\\a.txt", "a+");</code>

文件操作

- ❖ 标准库提供了定位函数以帮助用户实现对文件的随机定位读写.
- ❖ C语言中的文件定位函数:

函数原型	功能简介
<pre>fseek(FILE *restrict stream, long int offset, int wherefrom);</pre>	从 wherefrom 处将指针移动偏移量 offset
<pre>long int ftell(FILE *restrict stream);</pre>	获取文件的当前读写位置
<pre>void rewind(FILE *restrict stream);</pre>	文件内部的位置指针移到文件首
<pre>int fgetpos(FILE *restrict stream, fpos_t *pos);</pre>	对宽字符文件的指针进行操作, 介绍略
<pre>int fsetpos(File *restrict stream, const fpos_t *pos);</pre>	对宽字符文件的指针进行移动, 介绍略

取值	含义
SEEK_SET	文件首
SEEK_CUR	当前位置
SEEK_END	文件末尾

文件操作

❖例 请在c盘根目录下建立一个名为a.txt 的文本文件，其内容设置为： This ia an example, given by Wenbin..

```
#include <stdio.h>  #include <stdlib.h>  #include <memory.h>
#include <string.h>  #include <assert.h>
#define MAX_LENGTH 1000
int main(void)
{
    FILE * fp = fopen("c:\\a.txt", "r");
    long int pos;
    char * str = (char *)malloc(MAX_LENGTH);
    assert(str);
    errno = 0;
    if(!fp)
    {
        fprintf(stderr,
                "open c:\\a.txt failed. Error code: %s\\n", strerror(errno));
        return -1;
    }
```

文件操作

```
fscanf(fp, "%s", str);
puts(str);
fseek(fp, 8L, SEEK_SET);    pos = ftell(fp);
if(-1L == pos && errno <= 0)
{
    fprintf(stderr,
"返回c:\a.txt的指针当前读取位置失败. Error code: %s\n", strerror(errno));
    return -1;
}
printf("指针当前位置为: %ld\n", pos);
fscanf(fp, "%s", str);    puts(str);
fseek(fp, 20L, SEEK_SET);
fscanf(fp, "%s", str);    puts(str);
fseek(fp, pos, SEEK_SET);
fscanf(fp, "%s", str);    puts(str);
fclose(fp); /*关闭文件*/
free(str);
return 0;
}
```

文件操作

❖ 文件的读取函数：

函数原型	功能简介
int fgetc(FILE *stream);	读取下一个字符作为返回值
int getc(FILE *stream);	比 fgetc 高效，功能相同
int ungetc(int c, FILE *stream);	将字符 c “推回” 指定的输入流
char *fgets(char *s, int n, FILE *stream);	从输入流中最多读取 n-1 个字符返回
int fscanf(FILE *restrict stream, const char *restrict format, ...);	从文件中读取数据，用法与 scanf 类似，尤其是格式说明

文件操作

❖例 c盘根目录下有一名为a.txt 的文件，用于存放学生成绩，部分内容如下：

◆学号	姓名	数学	英语	语文
◆9801	张飞	98	89	90
◆9802	关羽	99	87	92

❖请编写程序读取该文件中的内容到一个结构体数组中并打印。

```
typedef struct
{
    char stuNo[STU_NO_LENGTH];           /*学号*/
    char name[NAME_LENGTH];              /*姓名*/
    float mathScore;                    /*数学成绩*/
    float engScore;                     /*英语成绩*/
    float literaScore;                 /*语文成绩*/
}STUDENT_SCORE;
```

文件操作

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define HEAD_OF_FORM "学号 姓名 数学英语语文\n"
#define STUDENT_COUNT 1000
#define STU_NO_LENGTH 10
#define NAME_LENGTH 20

typedef struct{
    char stuNo[STU_NO_LENGTH];
    char name[NAME_LENGTH];
    float mathScore;
    float engScore;
    float literaScore;
} STUDENT_SCORE;
```

文件操作

```
int main(void)
{
    int i = 0, j = 0;
    char *fileName = "c:\\a.txt";
    STUDENT_SCORE stuScores[STUDENT_COUNT];
    FILE *fp = fopen(fileName, "r");
    errno = 0;
    if(!fp){
        fprintf(stderr, "open failed. Error code: %s\n", strerror(errno));
        return -1;
    }
    fseek(fp, sizeof(HEAD_OF_FORM), SEEK_SET);
    i = 0;
    while(!feof(fp) && i < STUDENT_COUNT){
        fscanf(fp, "%s %s %f %f %f\n", stuScores[i].stuNo, stuScores[i].name,
&(stuScores[i].mathScore),&(stuScores[i].engScore),&(stuScores[i].literaScore));
        i++;
    }
}
```

文件操作

```
printf("从文件中读出的数据如下: \n");
for(j = 0; j < i; j++)
{
    printf("%s %s %.2f %.2f %.2f\n", stuScores[j].stuNo,
           stuScores[j].name,stuScores[j].mathScore,
           stuScores[j].engScore, stuScores[j].literaScore);
}
fclose(fp);
system("PAUSE");
return 0;
}
```

文件操作

❖ 文件的写入函数：

函数原型	功能简介
<code>int fputc(int c, FILE *stream);</code>	在输出流的当前位置写入字符 c
<code>int putc(int c, FILE *stream);</code>	与 <code>fputc</code> 功能相同
<code>char *fputs(const char *s, int n, FILE *stream);</code>	将 s 中字符（不包括'\0'）写入到输出流中
<code>int fprintf(FILE *restrict stream, const char *restrict format, ...);</code>	向输出流中输出数据，与 <code>printf</code> 用法类似，尤其是转换说明

文件操作

❖例 将通讯录中的数据写入到一个文件中，
通讯录结构体如下：

```
typedef struct{
    char name[NAME_LENGTH];
    char address[ADDRESS_LENGTH];
    char mobileNo[MOBILE_LENGTH];
    char email[EMAIL_LENGTH];
}PERSON;
```

文件操作

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define HEAD_OF_FORM "姓名 地址 电话号码 电子邮件"
#define NAME_LENGTH 20
#define ADDRESS_LENGTH 200
#define MOBILE_LENGTH 12
#define EMAIL_LENGTH 50
#define PERSON_COUNT 100
typedef struct{
    char name[NAME_LENGTH];
    char address[ADDRESS_LENGTH];
    char mobileNo[MOBILE_LENGTH];
    char email[EMAIL_LENGTH];
}PERSON;
```

文件操作

```
int main(void){
    int i = 0;
    char *fileName = "c:\\contact.txt"; /*写入的文件名*/
    PERSON persons[PERSON_COUNT] = {
        {"zhaoyun", "zy's address", "13666666666", "zy@163.com"},  

        {"guanyu", "gy's address", "13888888888", "gy@163.com"},  

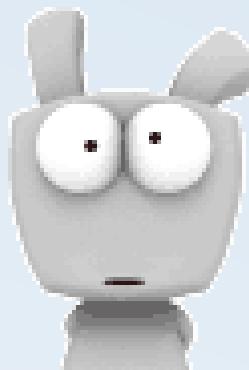
        {"machao", "mc's address", "13777777777", "mc@263.com"}};
    int personCount = 3;
    FILE *fp = fopen(fileName, "w");
    errno = 0;
    if(!fp){
        fprintf(stderr, "Creat/Open failed. Error code: %s\n",
                strerror(errno));
        return -1;
    }
```

文件操作

```
fprintf(fp, "%s\n", HEAD_OF_FORM);
for(i = 0; i < personCount; i++)
{
    fprintf(fp, "%-10s %-20s %-13s %-20s\n",
            persons[i].name, persons[i].address,
            persons[i].mobileNo, persons[i].email);
}
fclose(fp);
system("PAUSE");
return 0;
}
```

文件操作

- ❖ C提供了五种标准输入/输出数据：
 - ❖ stdin:代表键盘
 - ❖ stdout:代表屏幕
 - ❖ stderr:代表屏幕
 - ❖ stdoux:代表串行口
 - ❖ stdprn:代表打印机



问题:

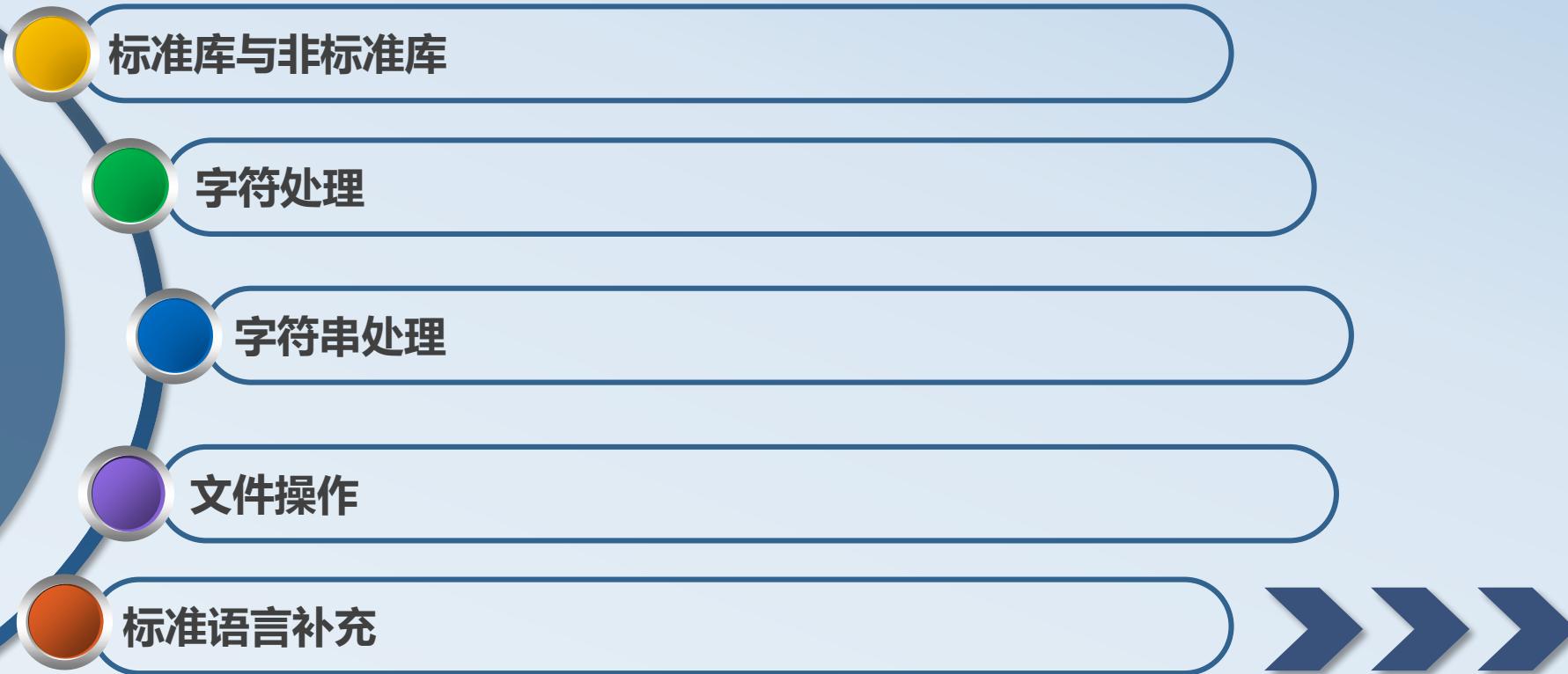
• stderr是什么文件啊?

文件操作

❖例 请阅读如下代码，并体会其中fprintf 与printf、fscanf 与 scanf 在使用上的相同之处。

```
#include <stdio.h>
#include <string.h>
#define MAX_LENGTH 1000
int main(void){
    char str1[MAX_LENGTH] = "Hello world!";
    char str2[MAX_LENGTH];
    fprintf(stdout, "由fprintf输出的: %s\n", str1);
    printf("由printf输出的: %s\n", str1);
    printf("请输入一个字符串(将用fscanf接收到内存): ");
    fscanf(stdin, "%s", str2);
    printf("你刚才输入的字符串为: %s\n", str2);
    printf("请输入一个字符串(将用scanf接收到内存): ");
    scanf("%s", str2);
    printf("你刚才输入的字符串为: %s\n", str2);
    return 0;
}
```

本讲授课内容



标准语言补充

- ❖ “标准语言补充” 的标准库对应的头文件有：
 - ◆ float.h
 - ◆ iso646.h
 - ◆ limits.h
 - ◆ stdarg.h
 - ◆ stdbool.h
 - ◆ stddef.h
 - ◆ stdint.h

标准语言补充

- ❖ iso646.h 中的内容大致有：
 - ◆ 定义了某些运算符的助记符

```
/*头文件：iso646.h*/  
#define and      &&  
#define and_eq   &=  
#define bitand   &  
#define bitor    |  
#define compl   ~  
#define not     !  
#define not_eq  !=  
#define or      ||  
#define or_eq   |=  
#define xor     ^  
#define xor_eq  ^=
```

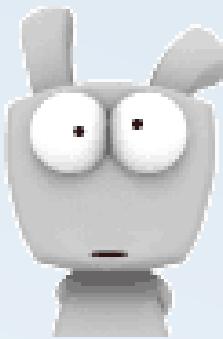
标准语言补充

❖例 阅读下面的程序，体会“助记符”。

```
#include <stdio.h>
#include <stdlib.h>
#include <iso646.h>
int main(void)
{
    int a = 3, b = 4, c = 0;
    if(a and b)
        printf("a && b 的值为: %d\n", a and b);
    c = a bitor b;
    printf("a bitor b 的值为: %d\n", c);
    return 0;
}
```

标准语言补充

- ❖ stdarg.h 中包含的函数为编程人员提供了访问可变参数表的方式



问题:

•什么叫函数的
可变参数啊?

标准语言补充

❖例 请编写一个可变参数的函数，能对多个整数求和。

```
#include <stdio.h>
#include <stdarg.h>
void sum(char *string, int num_args, ...)
{
    int s = 0;
    va_list ap;
    int i;
    va_start(ap, num_args);
    for(i = 0; i < num_args; i++)
        s += va_arg(ap, int);
    printf(string, s);
    va_end(ap);
    return;
}
```

标准语言补充

```
int main(void)
{
    sum("The sum of 10 + 15 is %d.\n", 2, 10, 15);
    sum("The sum of 10 + 15 + 13 is %d.\n", 3, 10, 15, 13);

    return 0;
}
```

标准语言补充

❖ stdarg.h 中包含的类型和宏：

- ◆ va_list：该类型用于声明局部状态变量（ap），用于遍历可变参数列表。
- ◆ va_start：它有两个参数：ap、可变参数的个数。该宏用于对ap 进行初始化。在标准C 语言中，其主要作用是将ap中的内部指针指向第一个可变参数
- ◆ va_arg：它有两个参数：ap、通过ap 的内部指针获取的当前的可变参数的类型。这个宏首先将ap 的内部指针指向下一可变参数，此后返回前一可变参数的值
- ◆ va_end：这个宏有一个参数：ap。在用va_arg 读取所有参数之后调用，用于对ap进行必要的整理操作

标准语言补充

- ❖ ANSI C 的格式中的可变参数函数，基本形式：
 - ◆ 返回值类型函数名(形式参数1 类型形式参数1, ...);
- ❖ 注意：
 - ◆ 至少需要一个普通的形式参数，可变参数就是通过 '...' 来说明的，这三个点是函数原型的一部分，不能省略。
 - ◆ 可变参数只能位于形式参数列表的最后。

标准语言补充

- ❖ stdbool.h 中定义了 bool 型变量的值：
 - ◆ false (0)
 - ◆ true (1)

/*头文件： stdbool.h*/

#define bool	_Bool	/*_Bool 在 C99 新增，用于表示布尔值*/
#define false	0	
#define true	1	
#define __bool_true_false_are_defined	1	

标准语言补充

❖例 阅读下面的程序，体会bool型变量的使用。

```
#include <stdio.h>
#include <stdbool.h>
int main(void)
{
    bool x = true;
    bool y;
    if(x)
        printf("x = %d\n", x);
    y = !x;
    if(!y)
        printf("y = %d\n", y);
    y = 3;
    return 0;
}
```

标准语言补充

- ❖ stddef.h中定义了几种常见的宏：
 - ◆ NULL，该宏代表空指针的字面值
 - ◆ ptrdiff_t，是两个指针相减结果的类型
 - ◆ size_t，是sizeof运算返回值的类型
 - ◆ offsetof，是一个函数式宏：
 - 参数：结构体类型；结构体类型的成员名称
 - 返回值：该结构体成员相对结构体对象的存储首地址而言的偏移量
- ❖ 注意：
 - ◆ 以上的所有宏用已有的变量类型通过typedef的来的

标准语言补充

❖例 阅读下面的程序，体会NULL、ptrdiff_t、size_t、offsetof 的使用

```
#include <stdio.h> #include <stdlib.h> #include <stddef.h>
struct Test{
    char x;
    int y;
};
int main(void){
    int a[] = {1, 2, 3, 4};
    int *p = NULL, *q = NULL;
    ptrdiff_t num = 0;
    size_t st = 0;
    p = &a[1];      q = &a[3];
    num = q - p;
    st = sizeof(num);
    printf("sizeof(num) = %d, num = %d\n", st, num);
    printf("offsetof(struct Test, y) = %d\n", offsetof(struct Test, y));
    return 0;
}
```

标准语言补充

❖ stdint.h中定义了五类数据类型：

- ◆ 准确长度类型
- ◆ 最小长度类型
- ◆ 快速长度类型
- ◆ 指针长度类型
- ◆ 最大长度类型

标准语言补充

❖准确长度类型：

- ◆ int8_t、int16_t、int32_t、int64_t
- ◆ uint8_t、uint16_t、uint32_t、uint64_t

/*头文件：某实现 stdint.h 中关于准确长度类型定义的部分内容*/

```
typedef signed char           int8_t;
typedef unsigned char         uint8_t;
typedef short                 int16_t;
typedef unsigned short        uint16_t;
typedef int                   int32_t;
typedef unsigned              uint32_t;
typedef long long             int64_t;
typedef unsigned long long    uint64_t;
```

标准语言补充

/*头文件：某实现 stdint.h 中关于准确长度类型数值范围定义的部分内容*/

#define INT8_MIN	(-128)	/*int8_t 类型数值的最小值*/
#define INT16_MIN	(-32768)	/*int16_t 类型数值的最小值*/
#define INT32_MIN	(-2147483647 - 1)	/*int32_t 类型数值的最小值*/
#define INT64_MIN	(-9223372036854775807LL - 1)	/*int64_t 类型数值的最小值*/
#define INT8_MAX	127	/*int8_t 类型数值的最大值*/
#define INT16_MAX	32767	/*int16_t 类型数值的最大值*/
#define INT32_MAX	2147483647	/*int32_t 类型数值的最大值*/
#define INT64_MAX	9223372036854775807LL	/*int64_t 类型数值的最大值*/
#define UINT8_MAX	0xff	/* 255U */
#define UINT16_MAX	0xffff	/* 65535U */
#define UINT32_MAX	0xffffffff	/* 4294967295U */
#define UINT64_MAX	0xffffffffffffffffULL	/*18446744073709551615ULL */

标准语言补充

❖ 最小长度类型：

- ◆ `int_least8_t`、`int_least16_t`、`int_least32_t`、`int_least64_t`
- ◆ `uint_least8_t`、`uint_least16_t`、`uint_least32_t`、`uint_least64_t`

/*头文件：某实现 stdint.h 中关于最小长度类型定义的部分内容*/

<code>typedef signed char</code>	<code>int_least8_t;</code>
<code>typedef unsigned char</code>	<code>uint_least8_t;</code>
<code>typedef short</code>	<code>int_least16_t;</code>
<code>typedef unsigned short</code>	<code>uint_least16_t;</code>
<code>typedef int</code>	<code>int_least32_t;</code>
<code>typedef unsigned</code>	<code>uint_least32_t;</code>
<code>typedef long long</code>	<code>int_least64_t;</code>
<code>typedef unsigned long long</code>	<code>uint_least64_t;</code>

标准语言补充

/*头文件：某实现 stdint.h 中关于最小长度类型定义数值范围的部分内容*/

#define INT_LEAST8_MIN	INT8_MIN /*与 INT_8_MIN 相同*/
#define INT_LEAST16_MIN	INT16_MIN
#define INT_LEAST32_MIN	INT32_MIN
#define INT_LEAST64_MIN	INT64_MIN
#define INT_LEAST8_MAX	INT8_MAX
#define INT_LEAST16_MAX	INT16_MAX
#define INT_LEAST32_MAX	INT32_MAX
#define INT_LEAST64_MAX	INT64_MAX
#define UINT_LEAST8_MAX	UINT8_MAX
#define UINT_LEAST16_MAX	UINT16_MAX
#define UINT_LEAST32_MAX	UINT32_MAX
#define UINT_LEAST64_MAX	UINT64_MAX

标准语言补充

❖ 快速长度类型

- ◆ `int_fast8_t`、`int_fast16_t`、`int_fast32_t`、`int_fast64_t`、
- ◆ `uint_fast8_t`、`uint_fast16_t`、`uint_fast32_t`、`uint_fast64_t`

/*头文件：某实现 stdint.h 中关于快速长度类型定义的部分内容*/

<code>typedef char</code>	<code>int_fast8_t;</code>
<code>typedef unsigned char</code>	<code>uint_fast8_t;</code>
<code>typedef short</code>	<code>int_fast16_t;</code>
<code>typedef unsigned short</code>	<code>uint_fast16_t;</code>
<code>typedef int</code>	<code>int_fast32_t;</code>
<code>typedef unsigned int</code>	<code>uint_fast32_t;</code>
<code>typedef long long</code>	<code>int_fast64_t;</code>
<code>typedef unsigned long long</code>	<code>uint_fast64_t;</code>

标准语言补充

❖ 指针长度类型

/*头文件：某实现 stdint.h 中关于指针长度类型定义的部分内容*/

```
typedef int intptr_t;
typedef unsigned uintptr_t;

#define INTPTR_MIN INT32_MIN
#define INTPTR_MAX INT32_MAX
#define UINTPTR_MAX U INT32_MAX
```

标准语言补充

❖最大长度类型

/*头文件：某实现 stdint.h 中关于最大长度类型定义的部分内容*/

```
typedef long long intmax_t;
typedef unsigned long long uintmax_t;
#define INTMAX_MIN INT64_MIN
#define INTMAX_MAX INT64_MAX
#define UINTMAX_MAX UINT64_MAX
```

标准语言补充

❖例 阅读下面的程序，体会stdint.h 中定义的类型及宏的使用

```
#include <stdio.h>
#include <stdint.h>
int main(void)
{
    int8_t x = 4;
    intmax_t mv = 5;
    intptr_t ptr = (intptr_t)&x;
    printf("x 的宽度为: %d\n", sizeof(int8_t));
    printf("x 的值为: %d\n", *(int8_t *)ptr);
    printf("int8_t 类型的最小值为: %d\n", INT8_MIN);
    printf("int8_t 类型的最大值为: %d\n", INT8_MAX);
    printf("mv 的宽度为: %d\n", sizeof(intmax_t));
    printf("mv 的值为: %d\n", mv);
    printf("intmax_t 类型的最小值为: %llu\n", INTMAX_MIN);
    printf("intmax_t 类型的最大值为: %llu\n", INTMAX_MAX);
    return 0;
}
```

标准语言补充

❖注意：

1. C语言可以干很多事儿，需要相应的库
2. C语言中常用的字符处理函数
3. C语言中常用的字符串处理函数
4. C语言中常用的文件操作函数

本章小结

- ◆ 讲述了标准库的作用。
- ◆ 简单阐述了有哪些标准头文件，各头文件中主要声明什么样的函数。
- ◆ 讲述了字符及字符串处理的标准库函数的用法。
- ◆ 讲述了内存管理方面的标准库函数的用法。
- ◆ 讲述了文件操作的标准库函数，掌握文件与内存间数据“交换”的方法。
- ◆ 讲述了标准输入输出函数的使用方法及区别。
- ◆ 简单阐述了标准语言补充方面的相关知识。

Thank You !