计算机程序设计

刘龙军 副教授

2025年10月20日

目录

简单文件读写

本质是库函数的使用

上节课回顾

- 程序设计中 数据的封装 技术： 结构体、共用体、枚举 （VS 非复合类型）
- 引用，别名（函数参数传递的 值传递、地址传递、引用传递）

知识点的结合编程：

- 结构体与普通变量
- 结构体与表达式
- 结构体与数组
- 结构体与指针
- 结构体与函数

什么是文件

- 文件有不同的类型，在程序设计中，主要用到两种文件

(1) 程序文件。包括源程序文件(后缀为.c)、目标文件(后缀为.obj)、可执行文件(后缀为.exe)等。这种文件的内容是程序代码

(2) 数据文件。文件的内容不是程序，而是供程序运行时读写的数据，如在程序运行过程中输出到磁盘(或其他外部设备)的数据，或在程序运行过程中供读入的数据。如一批学生的成绩数据，或货物交易的数据等。

例子

- 在以前所处理的数据的输入和输出，从终端的键盘输入数据，运行结果输出到终端显示器上
- 常常需要将一些数据输出到磁盘上保存起来，以后使用
- 这就要用到磁盘文件

定义

- “文件”指存储在外部介质上数据的集合
  - 一批数据是以文件的形式存放在外部介质上的
  - 想找存放在外部介质上的数据，先按文件名找到所指定的文件，然后再从该文件读数据
  - 要向外部介质上存储数据也必须先建立一个文件（以文件名作为标志），才能向它输出数据

例子

- 从程序的观点来看，无论程序一次读写一个字符，或一行文字，或一个指定的数据区，作为输入输出的各种文件或设备都是统一以逻辑数据流的方式出现的。
- 程序把文件看作是一个字符（或字节）的序列。一个输入输出流就是一个字符流或字节(内容为二进制数据)流。

文件名

- 文件要有一个**唯一的文件标识**，以便用户识别和引用。
- 文件标识包括三部分：
  - (1)文件路径
  - (2)文件名主干
  - (3)文件后缀

例子

- 文件路径表示文件在外部存储设备中的位置。如：

  D: \CC\temp\file1.dat

  一般不超过3个字母（doc、txt、dat、c、cpp、obj、exe、ppt、bmp等）

  表示file1.dat文件存放在D盘中的CC目录下的temp子目录下面

文件的分类

- 根据数据的组织形式，数据文件可分为ASCII文件和二进制文件。
  - 数据在内存中是以二进制形式存储的，如果不加转换地输出到外存，就是二进制文件
  - 如果要求在外存上以ASCII代码形式存储，则需要在存储前进行转换
  - ASCII文件又称文本文件，每一个字节放一个字符的ASCII代码

例子

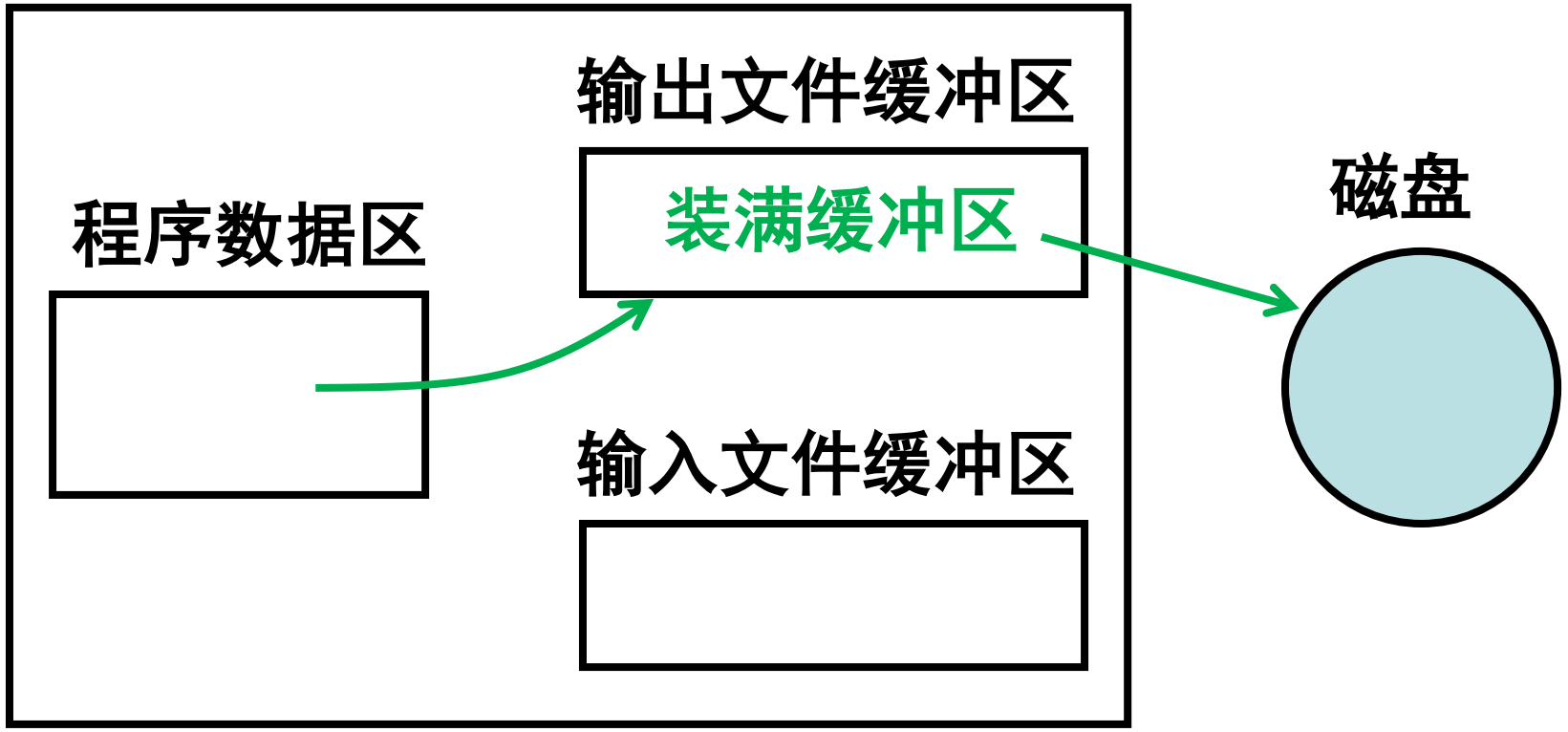
- 字符一律以ASCII形式存储
- 数值型数据既可以用ASCII形式存储，也可以用二进制形式存储
  - 如有整数10000，如果用ASCII码形式输出到磁盘，则在磁盘中占5 个字节(每一个字符占一个字节)，而用二进制形式输出，则在磁盘上只占4个字节

文件缓冲区

- C标准采用“缓冲文件系统”处理数据文件
- 所谓缓冲文件系统是指系统自动地在内存区为程序中每一个正在使用的文件开辟一个文件缓冲区
- 从内存向磁盘输出数据必须先送到内存中的缓冲区，装满缓冲区后才一起送到磁盘去
- 如果从磁盘向计算机读入数据，则一次从磁盘文件将一批数据输入到内存缓冲区（充满缓冲区），然后再从缓冲区逐个地将数据送到程序数据区（给程序变量）

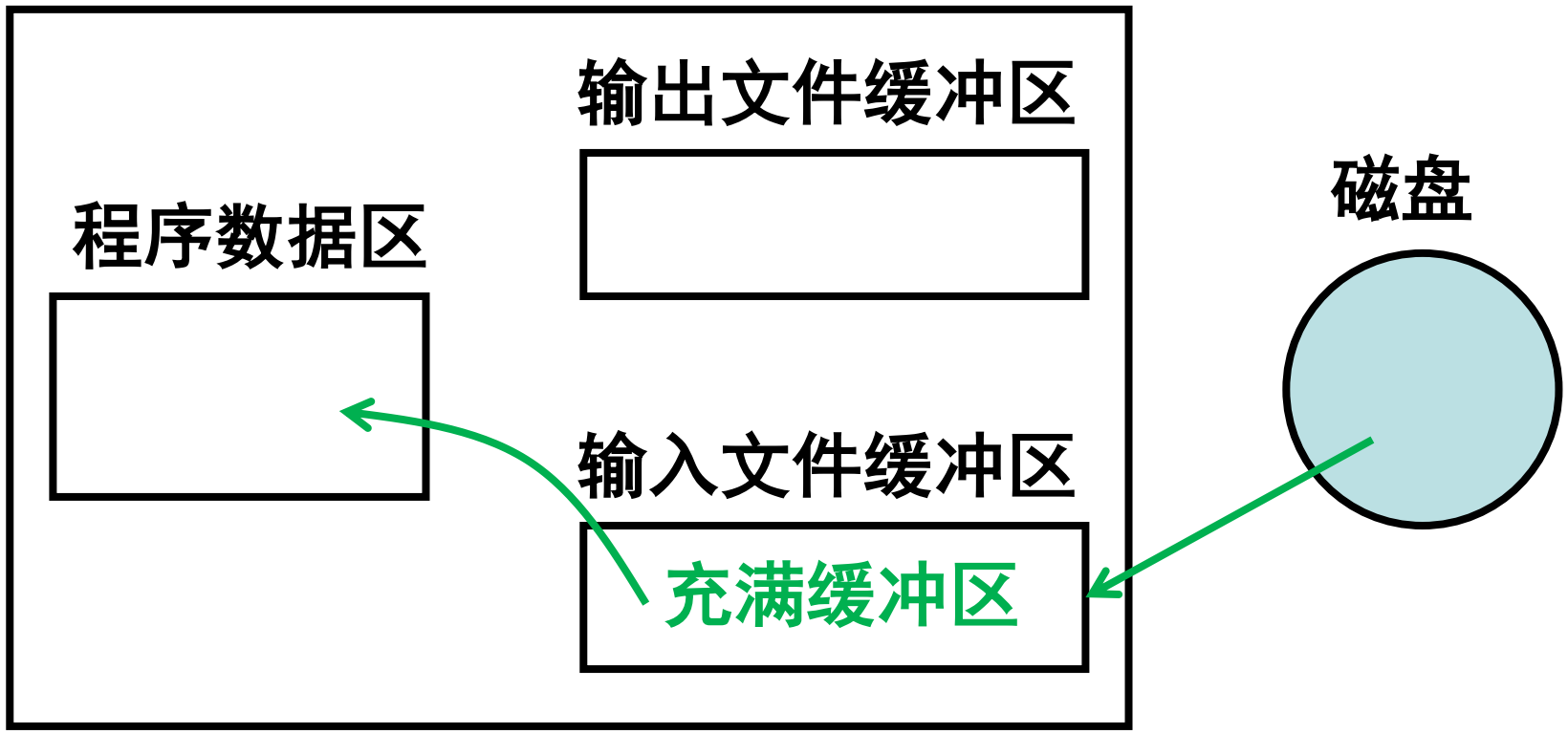
例子

➢从内存向磁盘输出数据



例子

➢从磁盘向计算机读入数据



怎样访问/读写文件?

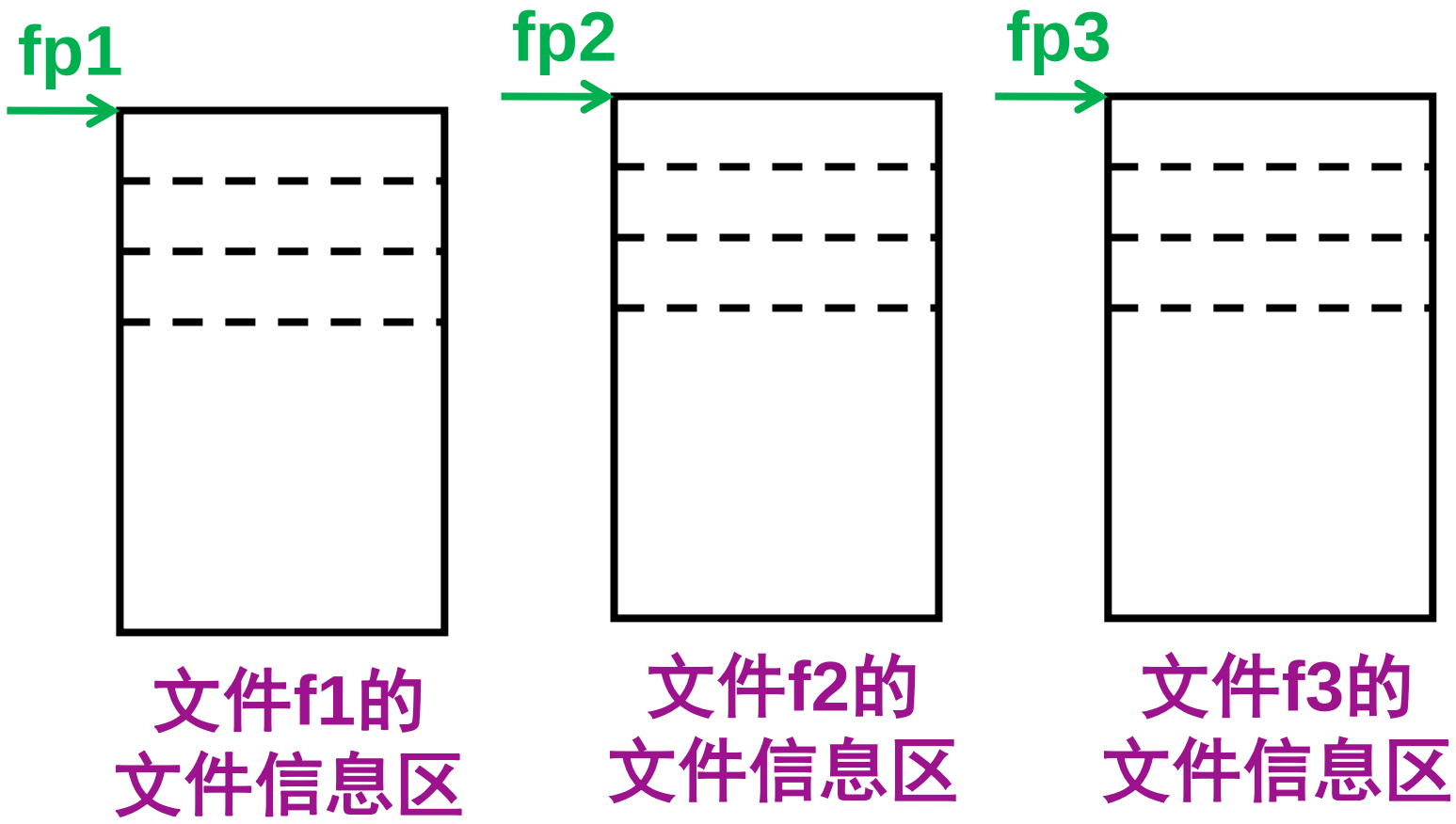
文件类型指针

- 缓冲文件系统中，关键的概念是“文件类型指针”，简称“文件指针”
- 每个被使用的文件都在内存中开辟一个相应的文件信息区，用来存放文件的有关信息（如文件的名字、文件状态及文件当前位置等）
- 这些信息是保存在一个结构体变量中的。该结构体类型是由系统声明的，取名为FILE

- 声明FILE结构体类型的信息包含在头文件“stdio.h”中
- 一般设置一个指向FILE类型变量的指针变量，然后通过它来引用这些FILE类型变量

例子

```
FILE   *fp1,*fp2,*fp3;
```



打开与关闭文件

用fopen函数打开数据文件

- 对文件读写之前应该“打开”该文件，在使用结束之后应“关闭”该文件。
- 所谓“打开”是指为文件建立相应的信息区(用来存放有关文件的信息)和文件缓冲区(用来暂时存放输入输出的数据)。
- 在编写程序时，在打开文件的同时，一般都指定一个指针变量指向该文件，也就是建立起指针变量与文件之间的联系，这样就可以通过该指针变量对文件进行读写
- 所谓“关闭”是指撤销文件信息区和文件缓冲区

- fopen函数的调用方式为：

  fopen(文件名,使用文件方式);

- 例如：

  fopen(“a1”,”r”);

  表示要打开名为“a1”的文件，使用文件方式为“读入”

  fopen函数的返回值是指向a1文件的指针

- 通常将fopen函数的返回值赋给一个指向文件的指针变量。如：

  FILE *fp;

  fp=fopen(“a1”,”r”);

  fp和文件a1相联系，fp指向了a1文件

- 在打开一个文件时，通知编译系统以下3个信息：

①需要访问的文件的名字

②使用文件的方式（“读”还是“写”等)

③让哪一个指针变量指向被打开的文件

说明

- (1) 用“r”方式打开的文件只能用于向计算机输入而不能用作向该文件输出数据，而且该文件应该已经存在，并存有数据，这样程序才能从文件中读数据。**不能用“r”方式打开一个并不存在的文件**
- (2) 用“w”方式打开的文件只能用于向该文件写数据（即输出文件），而不能用来向计算机输入。
  - 如果原来不存在该文件，**则在打开文件前新建立一个以指定的名字命名的文件。**
  - 如果原来**已存在一个以该文件名命名的文件，则在打开文件前先将该文件删去，然后重新建立一个新文件**

例子

- (3) 如果希望向文件末尾添加新的数据（不希望删除原有数据），则应该用“a”方式打开
  - 但此时应保证该文件已存在；否则将得到出错信息。
  - 打开文件时，文件读写标记移到文件末尾
- (4) 用r+、w+、a+方式打开的文件既可以用来输入数据，也可以用来输出数据。
  - 用r+方式时该文件应该已经存在。
  - 用w+方式则新建立一个文件，先向此文件写数据，然后可以读此文件中的数据。
  - 用a+方式打开的文件，原来的文件不被删去，文件读写位置标记移到文件末尾，可以添加，也可以读。

例子

- (5) 如果打开失败，fopen函数将会带回一个出错信息。fopen函数将带回一个空指针值NULL
- **常用下面的方法打开一个文件：**

```
if ((fp=fopen(“file1”,’r”))==NULL)
  {
    printf(“cannot open this file\n”);
    exit(0);
  }
```

终止正在执行的程序

用fclose函数关闭数据文件

- 关闭文件用fclose函数。fclose函数调用的一般形式为
  - fclose(文件指针);
  - 例如：
  - fclose（fp);
  - 如果不关闭文件将会丢失数据。

顺序读写数据文件

- **在顺序写时，先写入的数据存放在文件中前面，后写入的数据存放在文件中后面**
- **在顺序读时，先读文件中前面的数据，后读文件中后面的数据**
- **对顺序读写来说，对文件读写数据的顺序和数据在文件中的物理顺序是一致的**
- **顺序读写需要用库函数实现**

怎样向文件读写字符

读写一个字符的函数

函数名	调用形式	功能	返回值
fgetc	fgetc(fp)	从fp指向的文件读入一个字符	读成功，带回所读的字符，失败则返回文件结束标志EOF(即-1)
fputc	fputc(ch,fp)	把字符ch写到文件指针变量fp所指向的文件中	写成功，返回值就是输出的字符；输出失败，则返回EOF（即-1）

例子

- 从键盘输入一些字符，逐个把它们送到磁盘上去，直到用户输入一个“#”为止。
- 解题思路：用fgetc函数从键盘逐个输入字符，然后用fputc函数写到磁盘文件即可。

例子（库）

```
#include <stdio.h>
#include <stdlib.h>
int main()
{ FILE *fp;
   char ch,filename[10];
   printf("请输入所用的文件名：");
   scanf("%s",filename);
   if((fp=fopen(filename, "w" ))==NULL)
   {  printf("无法打开此文件\n");
       exit(0);
   }
   ch=getchar( );
```

用exit函数时加

输入文件名

只写

接收最后输入的回车符

例子

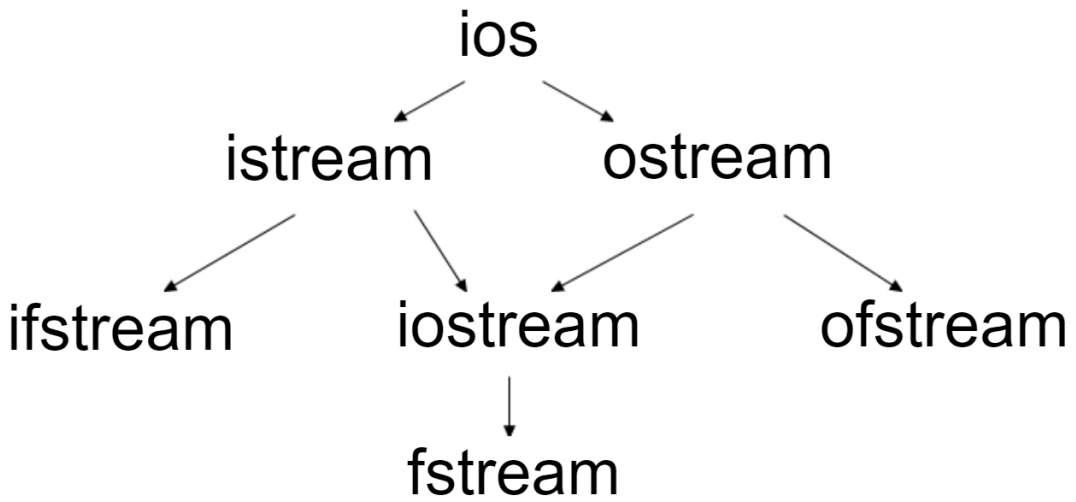
```
    printf(“请输入一个字符串(以#结束)：");
    ch=getchar( );
    while(ch!= ‘#’ )
    {  fputc(ch,fp);
        ch=getchar();
     }
    fclose(fp);
     return 0;
}
```

数据流形式读写文件

- ofstream: 写操作（输出）的文件类 (由ostream引申而来)
- ifstream: 读操作（输入）的文件类(由istream引申而来)
- fstream: 可同时读写操作的文件类 (由iostream引申而来),

- 可以将顺序文件看作一个有限字符构成的顺序字符流，然后象对cin, cout 一样的读写。

回顾一下输入输出流类的结构层次：



顺序文件

```
#include <fstream.h>   // 包含头文件
ofstream outFile( “clients.dat”, ios::out|ios::binary); //
打开文件
```

- **ofstream 是 fstream中定义的类**
- **outFile 是我们定义的ofstream类的对象**
- **“clients.dat”是将要建立的文件的文件名**
- **ios::out 是打开并建立文件的选项**
  - **ios:out** 输出到文件, 删除原有内容
  - **ios::app** 输出到文件, 保留原有内容，总是在尾部添加
  - **ios::ate** 输出到文件, 保留原有内容，可以在文件任意位置添加
- ios::binary 以二进制文件格式打开文件

文件的读写指针

- 对于输入文件,有一个读指针;
- 对于输出文件,有一个写指针;
- 对于输入输出文件,有一个读写指针;
- 标识文件操作的当前位置, 该指针在哪里,读写操作就在哪里进行。

```
ofstream  fout( “a1.out” ,ios::ate);
long location = fout.tellp();
        //取得写指针的位置
location = 10L;
fout.seekp(location);
        // 将写指针移动到第10个字节处
fout.seekp(location,ios::beg); //从头数location
fout.seekp(location,ios::cur); //从当前位置数location
fout.seekp(location,ios::end); //从尾部数location
```

location 可以为负值

```
ifstream  fin( “a1.in” ,ios::ate);
long location = fin.tellg();
        //取得读指针的位置
location = 10L;
fin.seekg(location);
        // 将读指针移动到第10个字节处
fin.seekg(location,ios::beg); //从头数location
fin.seekg(location,ios::cur); //从当前位置数location
fin.seekg(location,ios::end); //从尾部数location
```

location 可以为负值

打开文件例子代码

```
fstream fin;
        fin.open("test.txt",ios::in);//打开读
        fstream fin("test.txt",ios::in);
        //派生类ifstream直接有读权限
        ifstream fin;
        fin.open("test.txt");
        ifstream fin("test.txt");
=================================================
        fstream fout;
        fout.open("test.txt",ios::out);//打开写
        fstream fout("test.txt",ios::out);
        //派生类ofstream直接有写权限(清空写)
        ofstream fout;
        fout.open("test.txt");
        ofstream fout("test.txt");
```

检查状态例子代码

1
```
if(fs.is_open())
{
    cout<<"打开成功"<<endl;
}else cout<<"打开失败"<<endl;
```

2
```
if(fs.fail())
{
    cout<<"打开失败"<<endl;
}else cout<<"打开成功"<<endl;
```

3
```
if(fin)
{
    cout<<"打开成功"<<endl;
}else cout<<"打开失败"<<endl;
```

4
```
if(fin.good())
{
    cout<<"打开成功"<<endl;
}else cout<<"打开失败"<<endl;
```

库函数的使用！

- 程序设计中站在巨人肩膀上的思想
- 注意点： 定义 返回值 等

字符数组/数据 处理函数

字符数组处理函数

- 没有提供相应的运算符对字符数组的整体操作(**如复制、比较、连接、计算字符串长度等**)，但以库函数的形式实现了这些操作。
- 在头文件<cctype>中声明了一些测试字符的函数。
- 标准库函数的学习（**自学能力！课外文档**)

头文件<cctype>中声明了一些测试字符的函数。

函数	功能
int isdigit(char c)	判断字符是否是数字(0 – 9)
int isalpha(char c)	判断字符是否是字母(A – Z or a – z)
int isalnum(char c)	判断是否是数字和字母(A – Z, a – z, or 0 – 9)
int islower(char c)	判断字符是否是小写字母
int isuper(char c)	判断字符是否是大写字母
int isspace(char c)	判断字符是否是空格、换行符、制表符等
int ispunct (char c)	判断字符是否是标点符号(字母、数字、空格以外的字符)
int tolower(char c)	将字符转化为小写字符
int toupper(char c)	将字符转化为大写字符

字符统计的例子

- 编程统计一段文字中字母、数字字符、空格的个数,同时统计大写字母、小写字母的个数

```
//char_stat.cpp
//character statistics
#include <iostream>
#include <cstring>
#include <cctype>

using namespace std;
int main()
{
   char str[255]="She said: 1 car comes, 1 car goes, 2 car peng peng, 1
man dies!";
   int alpha = 0, upper = 0, lower=0, digit=0;
   int i=0;
   char c;

   for (i = 0; str[i]; i++)
   {
      c = str[i];
      if(isalpha(c))
      {
         alpha++;
         if(isupper(c))
            upper++;
         else
            lower++;
      }
      if(isdigit(c))
         digit++;
   }
   cout<<str<<endl;
   cout<< "Letters: "<< alpha<<endl;
   cout<< "Upper case: "<<upper<<endl;
   cout<< "Lower case: "<<lower<<endl;
   cout<< "Number: "<<digit<<endl;

   return 0;
}
```

字符串应用

- 编程统计一段文字中有多少个单词
  - 解题关键：单词的数目可以由空格的数目得到
  - 思路：
  - 从头到尾扫描字符串。当发现当前字符为非空格,且当前字符以前的字符是空格,则表示找到了一个新的单词。

```
//word_count.cpp
#include <iostream>
#include <cstring>
using namespace std;

int main()
{
   char str[255]= "He said: 1 car came, 1 car went, 2 car peng peng, 1 man died!";
   char prev = ' ';
   int i, num = 0;
   int word_flag=0;

   for (i = 0; str[i] != '\0'; i++)
   {
      word_flag = (prev == ' ') && (str[i] != ' ');//is it enough?
      if (word_flag){
         num++;
      }
      prev = str[i];
   }
   cout << str << endl;
   cout << "# of words: " <<num<<endl;

   return 0;
}
```

字符串复制

- strcpy函数用于既想修改又需保留原值的情况,通常是将字符串的值复制到一个临时字符串的值复制到一个临时变量中,然后对临时变量进行操作

```
//strcpy.cpp

#include <iostream>
#include <cstring>

using namespace std;

int main()
{
   char str1[10]="Green";
   char str2[10]="Red";

   cout<<"str1: "<<str1<<endl;
   cout<<"str2: "<<str2<<endl;

   strcpy(str2,str1);

   cout<<"str1: "<<str1<<endl;
   cout<<"str2: "<<str2<<endl;

   return 0;
}
```

请点评

- 使用strcpy时一定要保证目标字符串的空间足够大

```
……
int main()
{
   char str1[6]="China";
   char str2[6];

   strcpy(str2,"A_long_string");

……
   return 0;
}
```

字符串连接

- strcat函数将str2连接到str1的尾部，一般用来将较小的字符串合成一个大字符串

```
//strcat.cpp

#include <iostream>
#include <cstring>

using namespace std;

int main()
{
   char str1[150]="People's republic of ";
   char str2[10]="China";

   cout<<"str1: "<<str1<<endl;
   cout<<"str2: "<<str2<<endl;

   strcat(str1,str2);

   cout<<"str1: "<<str1<<endl;
   cout<<"str2: "<<str2<<endl;

   return 0;
}
```

字符串比较

- strcmp函数比较两个字符串同一位置的一对字符，若它们的ASCII码相同，则继续比较下一对字符。常用于判断两字符串是否相同。
- 譬如if(strcmp(s1,s2))
- 不能用以下形式：
- if(s1==s2)

```
//strcmp.cpp
#include <iostream>
#include <cstring>

using namespace std;

int main()
{
   char str1[150]="United Kindom";
   char str2[150]="United States";
   int diff;

   cout<<"str1: "<<str1<<endl;
   cout<<"str2: "<<str2<<endl;

   diff = strcmp(str1,str2);

   if(!diff)
      cout<<"The two char arrays are the same.\n";
   else
      cout<<"The distance is " << diff << endl;

   return 0;
}
```

不使用strcmp函数

```
//strcat2.cpp
#include <iostream>//
using namespace std;

int main()
{
    char str1[150]={};
    char str2[150]={};
    int diff=0;
    int flag=1;
    char *p1=str1, *p2=str2;

    cin.getline(str1, 150);
    cin.getline(str2, 150);
    cout<<"str1: "<<str1<<endl;
    cout<<"str2: "<<str2<<endl;

    flag = *p1&&*p2;
    while(flag){
        diff = *p1-*p2;
        if(diff)
            break;
        p1++;
        p2++;
        flag = *p1&&*p2;
    }
                        diff=*p1-*p2;

    if(!diff)
        cout<<"The two char arrays are the same.\n";
    else
        cout<<"The distance is " << diff << endl;

    return 0;
}
```

练习 字符数组长度

- strlen函数返回字符串的长度,且长度不包含结束符’\0’。
- 要求不调用strlen函数，实现strlen的功能。

练习

- 输入一个字符串str，再输入一个字符ch
- 返回str中第一次出现字符ch的位置,没有出现则返回0。
- 输入：
  - we are XJTUer
  - J
- 输出：
  - 9

练习

- 比较字符串dst和src,遇到第一个不一样的字符，返回第一次出现不一样字符的位置；如果dst和src完全相同，则返回0。
- 第一个输入的为dst，第二个为src
- 例如输入：
  - We are XJTUer
  - We r XJTUer
- 输出：
  - 4

数据处理函数

- math.h
- 参考课外资料

本章结束！