第八章 输入/输出

8.1 输入/输出概述

- 在C++中，输入/输出是一种基于字节流的操作：
  - 在进行输入操作时，可把输入的数据看成逐个字节地从外设流入到计算机内存。
  - 在进行输出操作时，则把输出的数据看成逐个字节地从内存流出到外设。
- 在C++的标准库中，提供了两大类I/O操作：
  - 基于字节流的I/O操作
  - 基于C++基本数据类型数据的I/O操作（在这些操作的内部实现了基本数据类型与字节流之间的转换）

C++输入输出的实现途径

- 过程式——通过从C语言保留下来的函数库中的输入/输出函数来实现。如：
  printf、scanf、……
- 面向对象——通过C++的I/O类库中的类/对象以及重载的操作符“<<”和“>>”来实现。如：
  ostream、cout，<<
  istream、cin，>>
  ……

I/O的分类

- 面向控制台的I/O：从标准输入设备（如：键盘）获得数据；把程序结果从标准输出设备（如：显示器）输出
- 面向文件的I/O：从外存（文件）获得数据；把程序结果保存到外存（文件）中
- 面向字符串变量的I/O：从程序中的字符串变量中获得数据；把程序结果保存到字符串变量中

8.2 面向控制台的输入/输出

两种实现途径：

- 用C++标准函数库中的输入/输出函数来实现
- 用C++的I/O类库中的I/O类来实现

8.2.1 基于函数库的控制台输入/输出

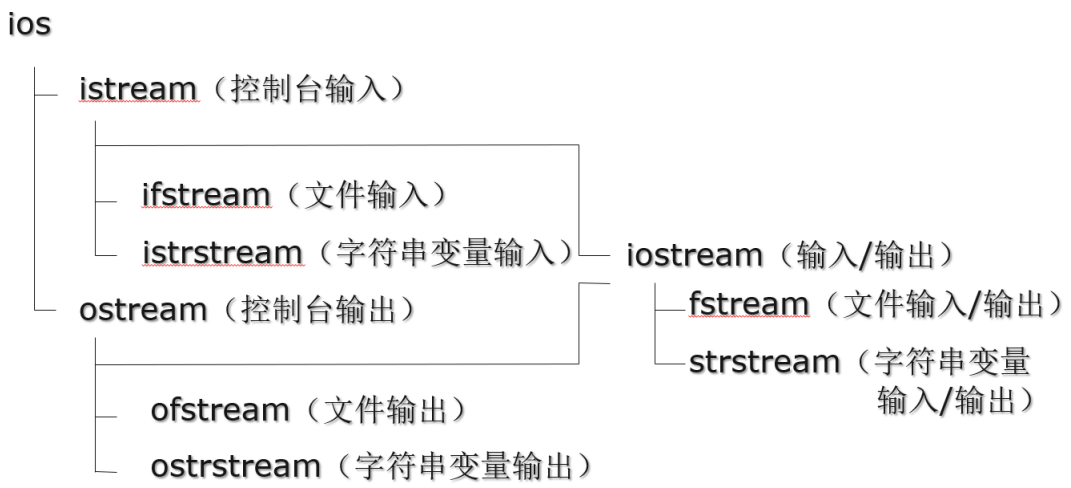
1. 控制台输出：putchar，puts，printf
2. 控制台输入：getchar，gets，scanf

8.2.2 基于I/O类库的控制台输入/输出

printf、scanf的缺陷

- 参数个数和类型不固定，编译时刻不能进行严格的参数类型检查（不是强类型），使用不当会导致运行时刻的错误。
- 需要用一个格式串来指出后面要输出的数据的类型。编译时刻一般不对格式串中指定的数据类型和个数与所提供的数据的实际类型和个数之间是否一致进行的检查（vs2015开始检查了），当格式串描述与实际数据不一致时会导致运行时刻的错误。
- 只能对基本数据类型的数据进行输入/输出，不能对用户定义类型的数据进行输入/输出

C++标准类库提供了以下的类来实现各种输入/输出功能，其中，面向字符串变量的I/O现在已被STL中的string替代。



1. 控制台输出

用插入操作符（<<）进行基本数据类型数据的输出

- 输出指向字符的指针时，并不是输出指针的值，而是输出它指向的字符串。如果要输出字符指针的值，需要把它转换成其它类型的指针
- 为了对输出格式进行控制，可以通过输出一些操纵符（manipulator）来实现

除了用插入操作符（<<）进行基本数据类型数据的输出外，还可以用ostream类的成员函数进行基于字节的输出操作

2. 控制台输入

用抽取操作符（>>）进行基本数据类型数据的输入

- 在输入时，各个数据之间一般要用空白符（空格、\t、\n）分开：
  - 输入一个数据前，先跳过空白符
  - 输入一个数据的过程中，碰到空白符或当前数据类型不允许的字符时结束当前数据的输入
- 可以通过一些操纵符来控制输入的行为

除了用抽取操作符（>>）进行基本数据类型数据的输入外，还可以用istream类的成员函数进行基于字节的输入操作

8.2.3 操作符“>>”和“<<”的重载

8.3 面向文件的输入/输出

用于永久性保存数据的设备称为**外部存储器**（简称：外存），如：磁盘、磁带、光盘等。

在外存（如磁盘）中，每个文件都有一个名字（文件名）（操作系统一般采用树型的目录结构来管理外存中的文件）。

8.3.1 文件概述

- 文本方式（text）
  - 只包含可显示的字符和有限的几个控制字符（如：'\r'、'\n'、'\t'等）的编码。
  - 一般用于存储具有“行”结构的文字数据，数据可用记事本等软件打开察看。
- 二进制方式（binary）

- 包含任意的没有显式含义的纯二进制字节。
- 一般用于存储任意结构的数据，数据由使用它的应用程序来解释。

对文件数据进行读写的过程：

- 打开文件：把程序内部的一个表示文件的变量/对象与外部的一个实际文件关联起来，并创建内存缓冲区。
- 文件读/写：存取文件中的内容。
- 关闭文件：把暂存在内存缓冲区中的内容写入到文件中，并归还打开文件时申请的内存资源（包括内存缓冲区）。

每个打开的文件都有一个内部（隐藏）的位置指针，它指出文件的当前读写位置。进行读/写操作时，每读入/写出一个字节，文件位置指针会自动往后移动一个字节的位置。

8.3.3 基于I/O类库的文件输入/输出

在利用I/O类库中的类进行文件的输入/输出时，程序中需要包含下面的头文件：`#include <iostream>`，`#include <fstream>`

1. 文件的输出操作

（1）打开文件：创建一个ofstream类的对象，并建立与外部某个文件之间的联系。

- 直接方式：`ofstream out_file(<文件名> [,<打开方式>]);`
- 间接方式：

```
ofstream out_file;
out_file.open(<文件名> [,<打开方式>]);
```

打开方式：

- `ios::out`
  - 打开一个外部文件用于写操作。（文件位置指针处于文件的头）
  - 如果外部文件已存在，则首先把它的内容清除；否则，先创建该外部文件。
  - ios::out是默认打开方式。
- `ios::app`
  - 打开一个外部文件用于添加操作。（不清除文件内容，文件位置指针处于文件末尾）
  - 如果外部文件不存在，则先创建该外部文件。
- ios::out | ios::binary 或 ios::app | ios::binary
  - 按二进制方式打开文件。（默认的是文本方式）
  - 用文本方式与二进制方式打开文件的区别：
  - 对以文本方式打开的文件，当输出的字符为'\n'时，在某些平台上（如：DOS和Windows平台）将自动把它转换成'\r'和'\n'两个字符写入外部文件。
  - 对以二进制方式打开的文件，对输出的字节不做任何转换，原样输出。

判断打开操作是否成功

判断文件是否成功打开可以采用以下方式：

```
if (!out_file.is_open())  //或：out_file.fail()
                          //或：!out_file
{ ..... //失败处理
}
```

（2）输出操作

文件成功打开后，可以使用插入操作符“<<”或ofstream类的一些成员函数来进行文件数据的输出操作

（3）关闭文件

文件输出操作结束时，要使用ofstream的成员函数close关闭文件：`out_file.close();`

关闭文件的目的：把文件内存缓冲区的内容写到磁盘文件中

程序正常结束时，系统也会自动关闭打开的文件

2. 文件的输入操作

（1）打开文件：创建一个ifstream类的对象，并与外部文件建立联系。

- 直接方式：`ifstream in_file(<文件名> [,<打开方式>]);`
- 间接方式

  ```
  ifstream in_file;
  in_file.open(<文件名> [,<打开方式>]);
  ```

打开方式

- ios::in
  - 打开一个外部文件用于读操作。（默认）
- ios::in | ios::binary
  - 按二进制方式打开文件。（默认为文本方式）
  - 对以文本方式打开的文件，当读入的字符为连续的'\r'和'\n'两个字符时，在某些平台上（如：DOS和Windows平台）将自动转换成一个字符'\n'。
  - 以二进制方式打开的文件，对输入的字节不做任何转换，原样输入。

打开文件时要判断打开是否成功。判断方式与文件输出打开操作的判断一样。

（2）输入操作

文件成功打开后，可以使用抽取操作符“>>”或ifstream类的一些成员函数来进行文件的输入操作

- 读取数据过程中有时需要判断是否正确读入了数据（尤其是在文件末尾处）。
- 判断是否正确读入了数据，可以调用ios类的成员函数fail来实现：`bool ios::fail() const;`，该函数返回true表示文件操作失败；返回false表示操作成功。

（3）关闭文件

文件输入操作结束后，要使用ifstream的一个成员函数close关闭文件：`in_file.close();`

3. 文件输入/输出与随机存取操作

如果需要打开一个既能读入数据、也能输出数据的文件，则需要创建一个fstream类的对象。

文件内部有两个位置指针，一个用于读，另一个用于写。

在创建fstream类的对象并建立与外部文件的联系时，文件打开方式应为下面之一：

- ios::in|ios::out（可在文件任意位置写）
- ios::in|ios::app（只能在文件末尾写）

文件的随机存取

对于流式文件，要读入/写出第n个字节，一般需要先读入/写出前n-1个字节。

为了能够随机读写文件中的数据，可以显示地指出读写的位置。

下面的操作用来指定文件内部读指针的位置：

- istream& istream::seekg(<位置>)；//指定绝对位置
- istream& istream::seekg(<偏移量>,<参照位置>)；//指定相对位置
- streampos istream::tellg()；//获得读操作的指针位置

下面的操作来指定文件内部写指针的位置：

- ostream& ostream::seekp(<位置>)；//指定绝对位置
- ostream& ostream::seekp(<偏移量>,<参照位置>); //指定相对位置
- streampos ostream::tellp()；//获得写操作的指针位置

<参照位置>可以是：ios::beg（文件头），ios::cur（当前位置）和ios::end（文件尾）。

文件的随机存取一般用于以二进制方式存贮的文件。