



第九章 输入/输出



本章内容

9.1 概述

9.2 面向控制台的输入/输出

9.3 面向文件的输入/输出

9.4 面向字符串变量的输入/输出



本章内容

9.1 概述

9.2 面向控制台的输入/输出

9.3 面向文件的输入/输出

9.4 面向字符串变量的输入/输出



9.1 输入/输出概述

- **输入/输出**（简称I/O）是程序的重要组成部分
 - 输入：从外设（如：键盘、文件等）得到程序运行所需要的数据
 - 输出：程序的运行结果要输出到外设（如：显示器、打印机、文件等）中
- 输入/输出功能在C++中是作为**标准库**（C++ standard library）的一部分来实现的
 - C++标准库提供了对基本数据类型的输入/输出操作；对于对自定义的类，可以重载这些输入/输出操作。

9.1 输入/输出概述

- 在C++中，输入/输出操作是基于**字节流**的：
 - 把输入的数据看成逐个字节地**从外设流入到内存**
 - 把输出的数据看成逐个字节地**从内存流出到外设**

| C++ standard library | |
|----------------------|-----------------------------|
| 21 | Language support library |
| 22 | Diagnostics library |
| 23 | General utilities library |
| 24 | Strings library |
| 25 | Localization library |
| 26 | Containers library |
| 27 | Iterators library |
| 28 | Algorithms library |
| 29 | Numerics library |
| 30 | Input/output library |
| 31 | Regular expressions library |
| 32 | Atomic operations library |
| 33 | Thread support library |



外设

内存

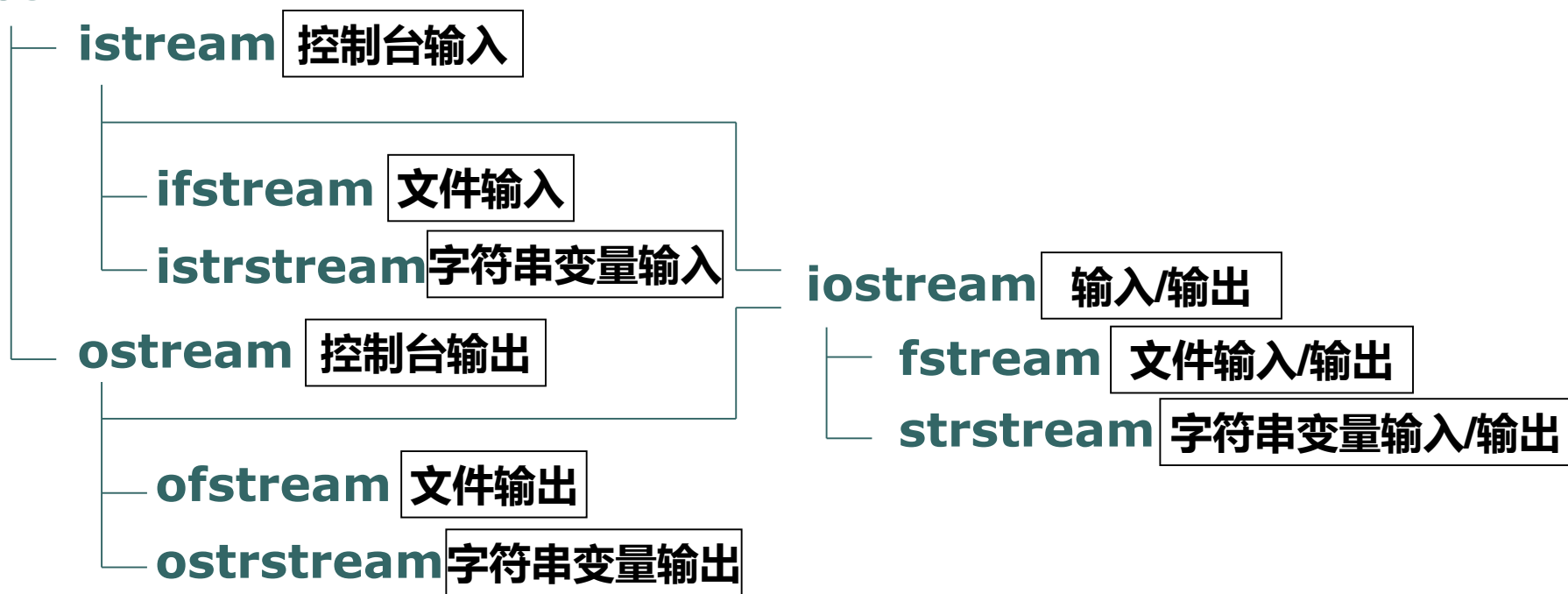


9.1 输入/输出概述

- 根据数据**来源和去向**分类
 - 面向控制台的I/O
 - 面向文件的I/O
 - 面向字符串变量的I/O
- 根据**设计范式**分类
 - 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
 - 面向对象 -- 通过**C++的I/O类库**来实现

9.1 输入/输出概述

ios



□ I/O类库中主要的类以及它们之间的关系

9.1 输入/输出概述

○ istream类的使用举例

1. 通过创建**istream类**（或其派生类）的对象来进行输入操作。
2. **istream类**重载了操作符 “>>”（抽取），用它可进行基本类型数据的输入操作。

➤ 例如：

```
istream in( ... );
```

```
in >> x;  //x是一个变量
```

```
in >> y;  //y是一个变量
```

```
in >> x >> y;
```




9.1 输入/输出概述

o ostream类的使用举例

1. 通过创建**ostream类**（或其派生类）的对象来进行输入操作。
2. ostream类重载了操作符 “<<”（**插入**），用它可进行基本类型数据的输出操作。

➤ 例如：

```
ostream out(...);  
out << e1; //e1是一个表达式  
out << e2; //e2是一个表达式  
out << e1 << e2;
```



本章内容

9.1 概述

9.2 面向控制台的输入/输出

9.3 面向文件的输入/输出

9.4 面向字符串变量的输入/输出

9.2 面向控制台的输入/输出

○ 基于函数库的控制台I/O

- 头文件：include<cstdio> //或<stdio.h>

- 输出：

//输出字符ch，返回输出的字符或EOF

//EOF即end of file，是stdio.h中定义的常量-1

int putchar (int ch);

//输出p指向的字符串，返回非负整数（通常为0）或EOF

int puts(const char *p);

//输出基本类型的数据，返回输出的字符个数或负数

int printf(const char *format, <参数表>)

//常用的格式控制字符串见课本P318

9.2 面向控制台的输入/输出

○ 基于函数库的控制台I/O

- 头文件：include<cstdio> //或<stdio.h>

- 输入：

//从键盘输入一个字符，返回输入的字符或EOF

int getchar ();

//从键盘输入字符串并放入p指向的内存空间，返回p或NULL

char *gets(char *p);

//输入基本类型的数据，返回输入的数据个数或EOF

int scanf(const char *format, <参数表>)

//常用的格式控制字符串见课本**P318**



9.2 面向控制台的输入/输出

- **基于类库的控制台IO**

- 输出
- 输入
- 抽取/插入操作符 >> 和 << 的重载

9.2 面向控制台的输入/输出

- 预定义的静态I/O对象：cin、cout、cerr、clog
 - cin属于istream类的对象，对应计算机系统的标准输入设备；
 - cout属于ostream类的对象，对应着计算机系统的用于输出程序正常运行结果的标准输出设备；
 - cerr和clog属于ostream类的对象，对应计算机系统的用于输出程序错误信息的设备，通常情况下它们都对应着显示器；
- 在进行控制台输入/输出时，程序中需要有下面的包含命令：`#include <iostream>`

9.2 面向控制台的输入/输出

➤ 插入操作符 << 的控制台输出示例：

```
#include <iostream>
```

```
using namespace std;
```

```
.....
```

```
int x;
```

```
char ch;
```

```
int *p = &x;
```

```
.....
```

```
cout << x ;           //输出x的值
```

```
cout << ch;           //输出ch的值
```

```
cout << "hello"       //输出字符串"hello"
```

```
cout << p;             //输出变量p的值，即变量x的地址
```

```
//以连续方式输出，并使用操纵符以十六进制输出x的值，然后换行
```

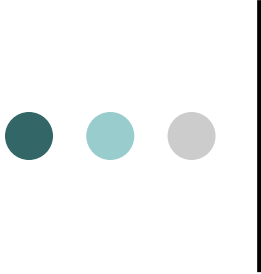
```
cout << hex << x << endl << ch << "hello" << p;
```

特例！

```
const char *q = "abcd";
```

```
cout << q;           //输出字符串
```

```
cout << (void *)q;    //输出q
```



常用输出操纵符

| 操纵符 | 含义 |
|---|---|
| <code>endl</code> | 输出换行符，并执行flush操作 |
| <code>flush</code> | 使输出缓存中的内容立即输出 |
| <code>dec</code> | 十进制输出 |
| <code>oct</code> | 八进制输出 |
| <code>hex</code> | 十六进制输出 |
| <code>setprecision(int n)</code> | 设置浮点数有效数字的个数或小数点后数字的位数 |
| <code>setiosflags(long flags)/ resetiosflags(long flags)</code> | 设置/取消输出格式，flags的取值可以是： <code>ios::scientific</code> （以指数形式显示浮点数）， <code>ios::fixed</code> （以小数形式显示浮点数），等等 |



常用输出操纵符

- 对于浮点数 (float、double和long double)
 - 当输出格式为ios::scientific或ios::fixed时，操纵符setprecision(int n)用于设置浮点数的小数点后面的位数（分别采用科学计数法和固定小数点的格式）
 - 当输出格式既不为ios::scientific也不为ios::fixed时（称为自动方式），操纵符setprecision(int n)用于设置浮点数的有效数字个数（采用科学计数法）
 - 需要首先导入头文件<iomanip>



9.2 面向控制台的输入/输出

- 除了通过插入操作符 << 进行输出外，也可以用 ostream 类提供的一些基于字节流的操作来进行输出，例如：

//输出一个字节

```
ostream& ostream::put(char ch);
```

//输出p所指向的内存空间中count个字节

```
ostream& ostream::write(const char *p, int count);
```



9.2 面向控制台的输入/输出

- **基于类库的控制台IO**

- 输出
- 输入
- 抽取/插入操作符 >> 和 << 的重载

9.2 面向控制台的输入/输出

- 任何基本类型的数据都可以通过**抽取操作符 >>**输入。在输入时，各个数据之间用空白符分开，最后输入一个回车符。例如：

```
#include <iostream>
```

```
using namespace std;
```

```
.....
```

```
int x;
```

```
double y;
```

```
char str[10];
```

```
cin >> x; cin >> y; cin >> str;
```

```
//使用操纵符控制输入格式，把输入的前9个字符和一个'\0'放入str中
```

```
cin >> setw(10) >> str;
```

9.2 面向控制台的输入/输出

- 可以使用**istream类的基于字节流的成员函数**来进行输入，例如：

//输入一个字节

istream::get(char &ch);

//输入一个字符串直到输入了count-1个字符或遇到

//delim指定的字符为止，并自动加上一个'\0'字符

istream::getline(char *p, int count, char delim='\n');

//读入count个字符至p所指向的内存空间中

istream::read(char *p, int count);



9.2 面向控制台的输入/输出

- **基于类库的控制台IO**

- 输出
- 输入
- 抽取/插入操作符 `>>` 和 `<<` 的重载

9.2 面向控制台的输入/输出

- 为了能用抽取操作符 >> 和插入操作符 << 对自定义类的对象进行输入/输出操作，就需要为自定义的类重载插入操作符 << 和抽取操作符 >>
- 两者分别是对象cin和cout的成员函数，但可以作为全局函数重载。
- 见下页例

cin.op|

```
operator void *  
operator =  
operator >>  
operator!  
out  
peek  
precision
```

9.2 面向控制台的输入/输出

- 抽取/插入操作符按**友元全局函数**的方式重载：

```
class A
{
    int x, y;
public:
    .....
    friend ostream& operator << (ostream& out, const A &a);
};
ostream& operator << (ostream& out, const A &a)
{
    out << a.x << ',' << a.y;
    return out;
}
.....
A a;
cout << a << endl;
```


9.2 面向控制台的输入/输出

- 抽取/插入操作符在派生类中的再次重载:

```
class B: public A
{
    int z;
public:
    .....
    friend ostream& operator << (ostream& out, const B &b);
};
ostream& operator << (ostream& out, const B &b)
{
    out << (A&) b << ',' << b.z;
    return out;
}
.....
B b;
cout << b << endl;
```

9.2 面向控制台的输入/输出

- **复习**：如何实现动态绑定？
- 因此，再次重载子类的抽取/插入操作符，
也不能实现以下的动态绑定：

```
A *p;  
p = new B; // p = new A;  
//无论p指向父类或子类，都将调用A的重载插入操作符 <<  
cout << *p;
```

➤ **解决办法见下页**

```
class A
{
    int x,y;
    public:
        .....
        //在类A中定义虚函数display来完成打印功能，并由operator <<调用
        virtual void display(ostream& out) const
        { out << x << ',' << y ; }
};

ostream& operator << (ostream& out, const A& a)
{
    a.display(out); //动态绑定到A类或B类对象的display
    return out;
}
```

```
class B: public A
{
    double z;
    public:
        .....
        //在类B中覆盖虚函数display
        void display(ostream& out) const
        { A::display(out); out << ',' << z; }
};
```



本章内容

9.1 概述

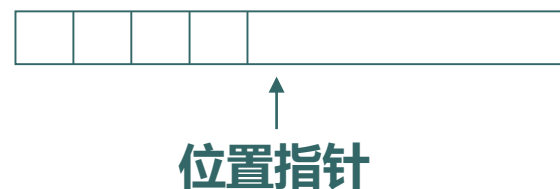
9.2 面向控制台的输入/输出

9.3 面向文件的输入/输出

9.4 面向字符串变量的输入/输出

9.3 面向文件的输入/输出

- 在C++中，把文件看成是由**一系列字节**所构成的字节串，称为**流式文件**。对文件中数据的操作通常是逐个字节地进行：
 - 打开文件**：目的是把程序中表示文件的变量/对象与外部的具体文件联系起来，并建立内存缓冲区。
 - 文件读写**：每个打开的文件都有一个**隐式的读写位置指针**，它指出文件的当前读写位置；进行读写操作时，每读入或写出一个字节，该指针会自动往后移动一个字节。
 - 关闭文件**：目的是把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。





9.3 面向文件的输入/输出

○ 文件中数据的存储方式

- **文本方式** (text)：用于存储具有“行”结构的文本数据，只包含可显示字符和有限的几个控制字符（如：\n、\t）。
 - 例：文本方式存储整数1234567：把1、2、3、4、5、6、7的ASCII码（共7个字节）写入文件。
- **二进制方式** (binary)：用于存储无显式结构的数据，可以包含任意的二进制字节。
 - 二进制方式存储整数1234567：把整数1234567的计算机内部表示（比如每个整数的补码占4个字节）分解成字节写入文件。

9.3 面向文件的输入/输出

基于函数库的文件I/O

- 头文件: `include<cstdio>` //或`<stdio.h>`

- 打开文件、关闭文件、位置指针:

//打开文件, 若打开成功则返回FILE*类型的指针, 否则NULL

File *fopen(const char *filename, const char *mode);

//关闭文件, 若关闭成功则返回0

int fclose(FILE *stream);

//将位置指针从origin偏移offset, 若成功则返回0, 否则失败

int *fseek(FILE *stream, long offset, int origin);

//获取位置指针的当前位置

long *ftell(FILE *stream);

//判断位置指针是否在文件末尾, 若是则返回0

int *feof(FILE *stream);

9.3 面向文件的输入/输出

○ 基于函数库的文件I/O

● 输出:

//输出字符，若成功则返回输出的字符

int fputc(int c, FILE *stream);

//输出字符串，若成功则返回一个非负整数

int fputs(const char *string, FILE *stream);

//输出基本类型的数据，返回输出的字符数

int fprintf(FILE *stream, const char *format, <参数表>)

//按字节块输出：size为字节块的尺寸，count为字节块的个数

//返回输出的字符个数或负数

**size_t fwrite(const void *buffer, size_t size, size_t count,
FILE *stream)**

9.3 面向文件的输入/输出

基于函数库的文件I/O

输入:

//输入一个字符，返回字符的编码

int fgetc(int c, FILE *stream);

//输入字符串，若成功则返回该字符串，否则NULL

char* fgets(const char *string, int n, FILE *stream);

//输入基本类型的数据，返回值为读入并存储的数据个数

int fscanf(FILE *stream, cost char *format, <参数表>)

//按字节块输入。size为字节块的尺寸，count为字节块的个数

//返回值为读入的字节块的个数

**size_t fread(const void *buffer, size_t size, size_t count,
FILE *stream)**



9.3 面向文件的输入/输出

基于类库的文件IO

- 输出
- 输入
- 输出/输入

9.3 面向文件的输入/输出

○ 打开/关闭文件

- 在利用I/O类库中的类进行外部文件的输入/输出时，程序中需要下面的包含命令：**#include <fstream>**
- 创建ofstream类的对象，建立与外部文件之间的联系：
ofstream out_file(<文件名>, <打开方式>); //或下面的方式
ofstream out_file; out_file.open(<文件名>, <打开方式>);
- 判断文件是否成功打开可以采用以下方式
if (!out_file) //或 out_file.fail() 或 !out_file.is_open()
{ } //失败处理
- 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();**

9.3 面向文件的输入/输出

○ 打开方式

- **ios::out**: 若外部文件已存在, 则先把它的内容清除; 否则, 先创建该外部文件。文件位置指针指向文件中的第一个字节。与fopen的打开方式w相同。
- **ios::app**: 若外部文件不存在, 则先创建该外部文件。文件位置指针指向文件末尾。与fopen的打开方式a相同。
- **ios::out | ios::binary**或者**ios::app | ios::binary** 表示按二进制方式打开文件写或添加。与fopen的打开方式wb, ab相同。
- 默认的打开方式是**文本方式**

9.3 面向文件的输入/输出

- 操作文件指针的位置使用下面的函数：

//指定绝对位置

ostream& ostream::seekp(<位置>);

//指定相对位置，<参照位置>可以取以下值：

//ios::beg(文件头)、ios::cur(当前位置)、ios::end(文件尾)

ostream& ostream::seekp(<偏移量>, <参照位置>);

//获得指针位置

streampos ostream::tellp();

9.3 面向文件的输入/输出

- 使用插入操作符 << 或 ofstream 的成员函数进行输出。例如：

```
int x; double y; .....
```

```
//以文本方式输出数据到文件
```

```
ofstream out_file("d:\\myfile.txt", ios::out);
```

```
if (!out_file) exit(-1);
```

```
out_file << x << ' ' << y << endl;
```

```
//以二进制方式输出数据
```

```
ofstream out_file("d:\\myfile.dat", ios::out | ios::binary);
```

```
if (!out_file) exit(-1);
```

```
out_file.write((char *)&x, sizeof(x));
```

```
out_file.write((char *)&y, sizeof(y));
```



9.3 面向文件的输入/输出

基于类库的文件IO

- 输出
- 输入
- 输出/输出

9.3 面向文件的输入/输出

○ 打开/关闭文件

- 在利用I/O类库中的类进行外部文件的输入/输出时，程序中需要下面的包含命令：**#include <fstream>**
- 创建一个ifstream类的对象，建立与外部文件之间的联系
ifstream in_file(<文件名>, <打开方式>);
ifstream in_file; in_file.open(<文件名>, <打开方式>);
- 判断文件是否成功打开可以采用以下方式：
if (!in_file) //或 in_file.fail() 或 !in_file.is_open()
{ } //失败处理
- 文件输入操作结束时，要使用ifstream的成员函数close关闭文件：**in_file.close();**



9.3 面向文件的输入/输出

○ 打开方式

- **ios::in**：打开文件进行读操作。如果外部文件不存在则打开文件失败。位置指针指向文件的第一个字节。与fopen的打开方式r相同。
- **ios::in | ios::binary**：表示按二进制方式打开文件。与fopen的打开方式rb相同。
- 默认的打开方式是**文本方式**。

9.3 面向文件的输入/输出

- 操作文件指针的位置使用下面的函数：

//指定绝对位置

istream& istream::seekg(<位置>);

//指定相对位置，<参照位置>可以取以下值：

//ios::beg(文件头)、ios::cur(当前位置)、ios::end(文件尾)

istream& istream::seekg(<偏移量>, <参照位置>);

//获得指针位置

streampos istream::tellg();

//ios类的成员函数eof用于判断文件是否结束，

//该函数返回非0表示上一次读操作遇到了文件结尾

int ios::eof();

9.3 面向文件的输入/输出

- 使用抽取操作符 `>>` 或 `ifstream` 的成员函数进行输入。例如：

```
int x; double y; .....
```

```
//以文本方式输入数据
```

```
ifstream in_file("d:\\myfile.txt", ios::in);
```

```
if (!in_file) exit(-1);
```

```
in_file >> x >> y;
```

```
//以二进制方式输入数据
```

```
ifstream in_file("d:\\myfile.dat", ios::in | ios::binary);
```

```
if (!in_file) exit(-1);
```

```
in_file.read((char *)&x, sizeof(x));
```

```
in_file.read((char *)&y, sizeof(y));
```



9.3 面向文件的输入/输出

基于类库的文件IO

- 输出
- 输入
- 输出/输出



9.3 面向文件的输入/输出

- 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**。
- 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：
 - **ios::in | ios::out**：可在文件任意位置读写
 - **ios::in | ios::app**：可在文件任意位置读，但只能在文件末尾写



本章内容

9.1 概述

9.2 面向控制台的输入/输出

9.3 面向文件的输入/输出

9.4 面向字符串变量的输入/输出



9.4 面向字符串变量的输入/输出

- 将**字符串变量**作为输入源/输出目的地：
 - 程序中的有些数据并不直接输出到标准输出设备或文件，而是需要保存在**某个字符串变量**中。
 - 程序中的有些数据并不直接从标准输入设备或文件输入，而是需要从**某个字符串变量**中读入。

9.4 面向字符串变量的输入/输出

- 首先, **#include <stringstream>**
- 再创建类**istream**、**ostream**或**stringstream**的对象。
- 然后, 用与**基于I/O类库的文件输入/输出**类似的操作进行输入/输出即可, 例如:

- 对于ostream类:

```
char buf[100];
```

```
.....
```

```
ostream str_buf(buf); //或ostream str_buf(buf,100);
```

```
str_buf << x << y << endl;
```

- 对于istream类

```
char buf[100];
```

```
istream str_buf(buf); //或istream str_buf(buf,100);
```

```
.....
```

```
str_buf >> x >> y;
```


9.4 面向字符串变量的输入/输出

- 在C++新标准中，类`istream`、`ostream`和`stringstream`分别被`istringstream`、`ostringstream`和`stringstream`所替代（在头文件`<sstream>`中声明）