面向对象程序设计

输入/输出流

2020 年 春

耿楠

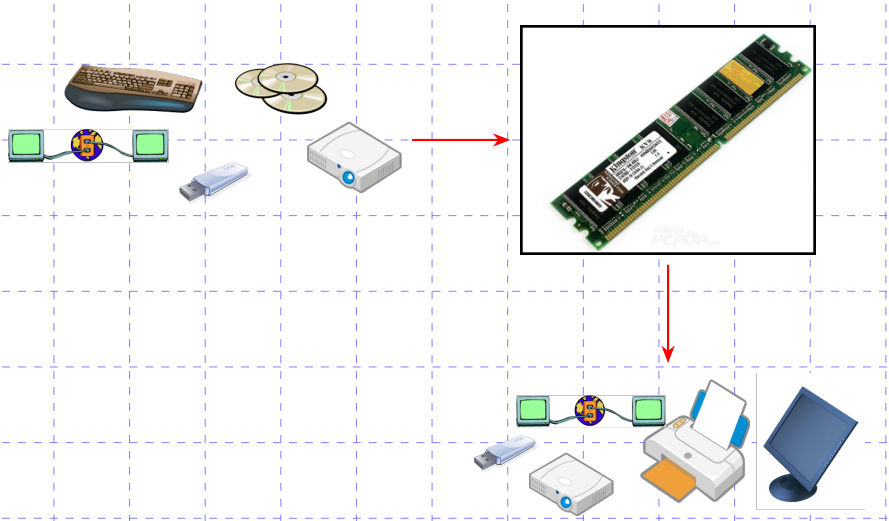
计算机科学系
信息工程学院

西北农林科技大学
NORTHWEST A&F UNIVERSITY
中国 · 杨凌

Object Oriented Programming—OOP

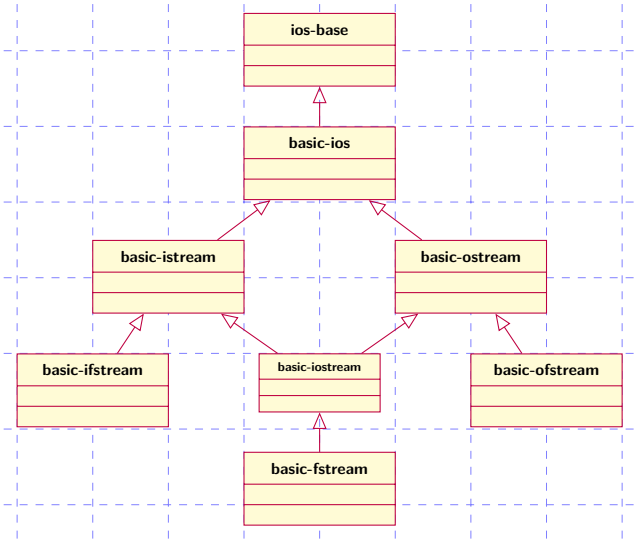
流类 1
输出流
输入流
文本文件
二进制文件
附件下载

- 什么是流 (stream)
  - 与物理输入输出设备无关的数据序列



流类 2
输出流
输入流
文本文件
二进制文件
附件下载

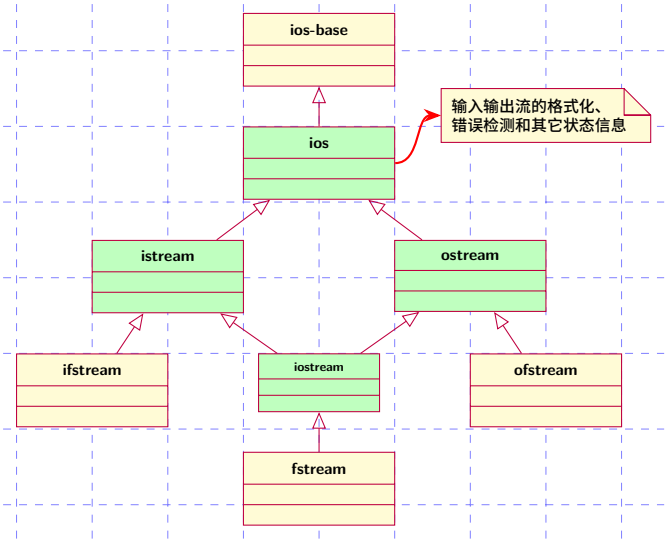
- 输入输出流类模板层次结构图



- 类模板可实例化为窄字符类 (`char`) 和宽字符类 (`wchar_t`)

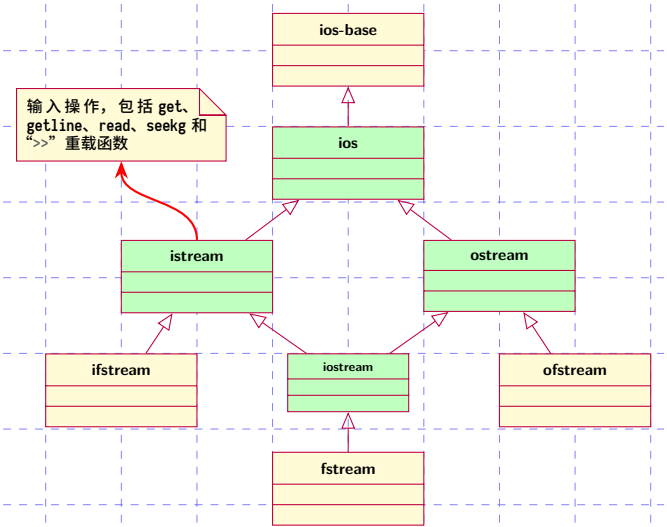
Template Class	Character-based Class	Wide-Character-based Class
basic_ios	ios	wios
basic_istream	istream	wistream
basic_ostream	ostream	wostream
basic_iostream	iostream	wiostream
basic_fstream	fstream	wfstream
basic_ifstream	ifstream	wifstream
basic_ofstream	ofstream	wofstream

- 输入输出流类模板层次结构图

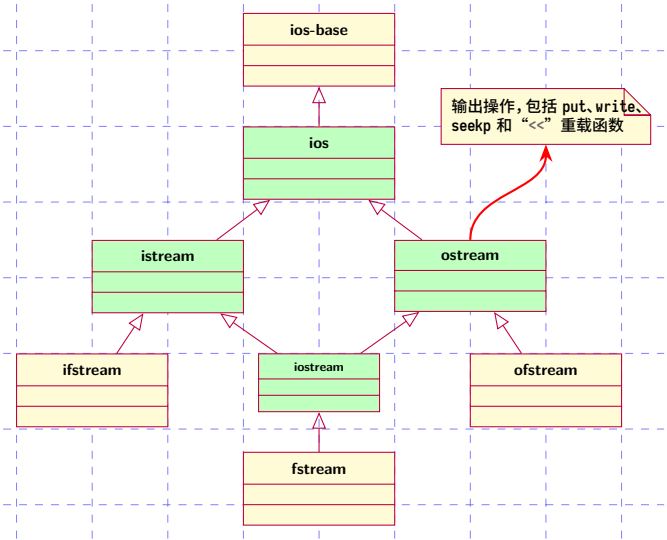


流类 5
输出流
输入流
文本文件
二进制文件
附件下载

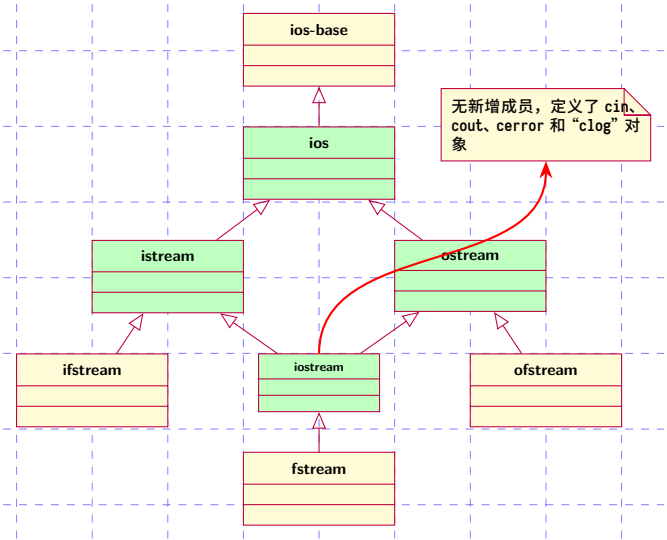
输入输出流类模板层次结构图



输入输出流类模板层次结构图

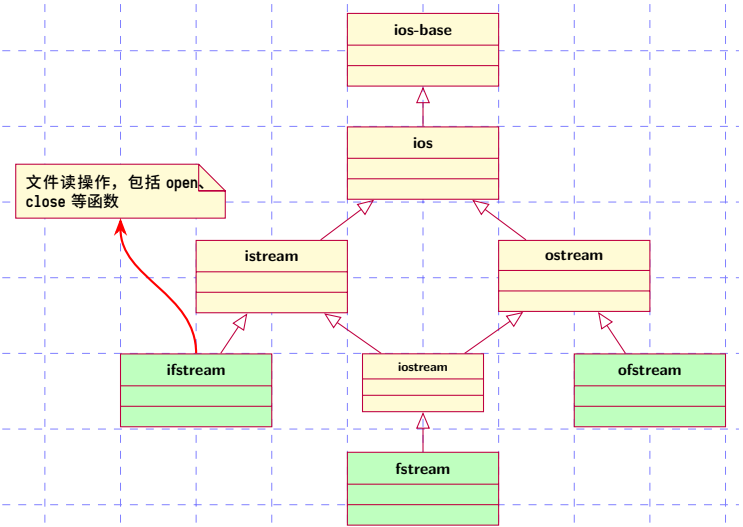


- 输入输出流类模板层次结构图

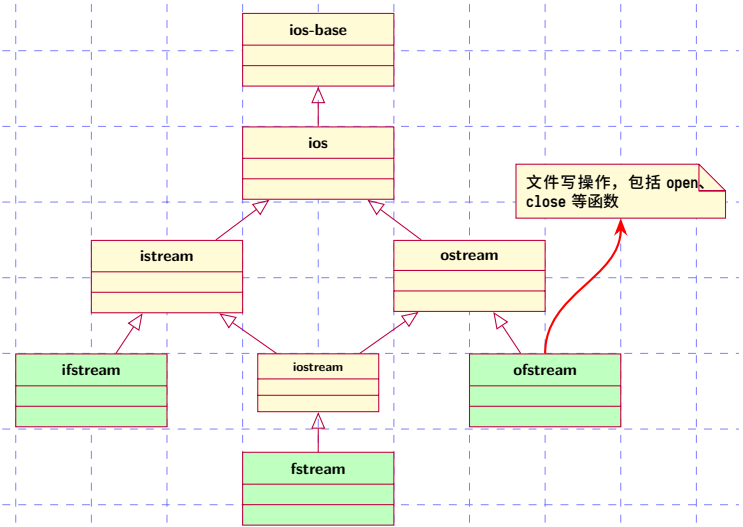


OBJECT ORIENTED PROGRAMMING—OOP
流类 8
输出流
输入流
文本文件
二进制文件
附件下载

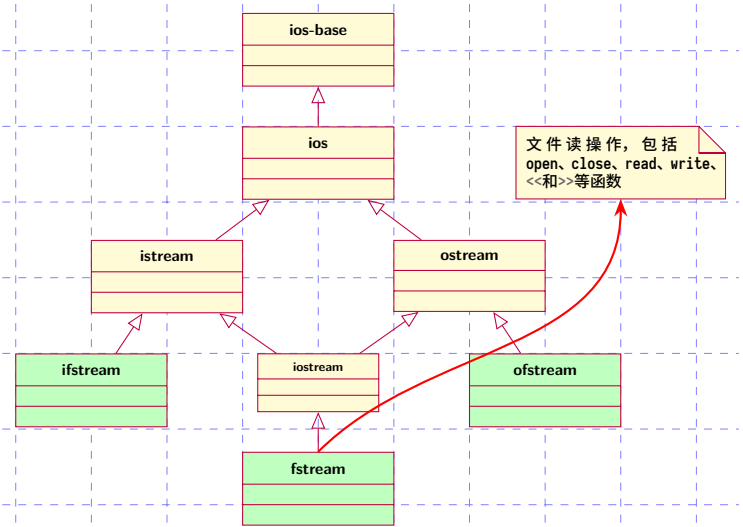
输入输出流类模板层次结构图



输入输出流类模板层次结构图



输入输出流类模板层次结构图



Object Oriented Programming—OOP

流类
输出流 11
输入流
文本文件
二进制文件
附件下载

- 输出流常用函数 (ostream)
  - 运算符重载函数“<<”
  - 输出单个字符到屏幕或文件
    ```
    ostream &put(char ch)
    ```
  - 输出字符块信息到屏幕或文件
    ```
    ostream &write(const char* pch, int nCount)
    ```

- 输出流常用函数 (ostream)

```
#include <iostream>

using namespace std;

int main()
{
   char buff[11] = "0123456789";

   cout << buff << endl;

   for (int i=0; i<10; i++)
      cout.put(buff[i]);
   cout << endl;

   cout.write(buff,10);
   cout << endl;
   return 0;
}
```

- 格式控制
  - ios 类提供直接设置标志字的控制格式函数
  - iostream 和 iomanip 库还提供一批操纵符简化 I/O 格式化操作
- 格式控制标志位 (16 位二进制数)
  - 进制（进制组合 ios::basefield）
  - 对齐（对齐组合 ios::adjustfield）
  - 浮点数（浮点组合 ios::floatfield）
  - 忽略输入流空格（ios::skipws）
  - 显示正号
  - 显示小数点
  - ...

0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0

▶ 格式控制

field	member constant	effect when set
independent flags	boolalpha	read/write bool elements as alphabetic strings (`true` and `false`).
	showbase	write integral values preceded by their corresponding numeric base prefix.
	showpoint	write floating-point values including always the decimal point.
	showpos	write non-negative numerical values preceded by a plus sign (+).
	skipws	skip leading whitespaces on certain input operations.
	unitbuf	flush output after each inserting operation.
	uppercase	write uppercase letters replacing lowercase letters in certain insertion operations.
numerical base (basefield)	dec	read/write integral values using decimal base format.
	hex	read/write integral values using hexadecimal base format.
	oct	read/write integral values using octal base format.
float format (floatfield)	fixed	write floating point values in fixed-point notation.
	scientific	write floating-point values in scientific notation.
adjustment (adjustfield)	internal	the output is padded to the *field width* by inserting *fill characters* at a specified internal point.
	left	the output is padded to the *field width* appending *fill characters* at the end.
	right	the output is padded to the *field width* by inserting *fill characters* at the beginning.

Three additional bitmask constants made of the combination of the values of each of the three groups of selective flags can also be used:

flag value	equivalent to
adjustfield	left | right | internal
basefield	dec | oct | hex
floatfield	scientific | fixed

http://www.cplusplus.com/reference/iostream/ios_base/fmtflags/

状态字设置函数

函数	功能
`long flags( long lFlags );` `long flags() const;`	用参数 lFlags 更新标志字 返回标志字
`long setf( long lFlags );` `long setf( long lFlags, long lMask );`	设置 lFlags 指定的标志位 将 lMask 指定的位清 0，然后设置 lFlags 指定位
`long unsetf( long lFlags );`	将 lMask 指定的标志位清 0

输出流常用函数 (ostream)

```
#include <iostream>
using namespace std;

struct fmtflags
{
   long flag;
   char flagname[12];
} flags[18] = {{ios::hex, "hex"},
   {ios::dec, "dec"},
   {ios::oct, "oct"},
   {ios::basefield, "basefield"},
   {ios::internal, "internal"},
   {ios::left, "left"},
   {ios::right, "right"},
   {ios::adjustfield, "adjustfield"},
   {ios::fixed, "fixed"},
   {ios::scientific, "scientific"},
   {ios::basefield, "basefield"},
   {ios::showbase, "showbase"},
   {ios::showpoint, "showpoint"},
   {ios::showpos, "showpos"},
   {ios::skipws, "skipws"},
   {ios::uppercase, "uppercase"},
   {ios::boolalpha, "boolalpha"},
   {ios::unitbuf, "unitbuf"}
};

int main()
{
   long IFlags, INewFlags;
   IFlags = cout.setf(ios::hex, ios::basefield);
   INewFlags = cout.flags();
   cout << "Default flag is:" << IFlags << endl;
   cout << "New flag is:" << INewFlags << endl;
   for(int i = 0; i < 18; i++)
      cout << flags[i].flag << '\t' << flags[i].flagname << endl;
   return 0;
}
```

dec	2	10
hec	8	100
oct	40	1000000

0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0		
0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0		

格式控制成员函数

- 设置域宽: `int width(int nw);`
- 填充: `fill(char c);`
- 设置浮点数精度: `int precision(int n);`
- 设置格式: `long flags(long lFlags);`

- 格式操纵符：简化 I/O 格式化操作 (iomanip)
  - 进制 (dec、oct、hex 和 setbase())
  - 对齐
  - 浮点数显示
  - 设置当前域宽 (setw)
  - 填充 (setfill)
  - 设置精度 (setprecision)
  - ...

- 格式操纵符：简化 I/O 格式化操作

```cpp
#include <iostream>
using namespace std;

int main()
{
   long IFlags, INewFlags;
   IFlags = cout.flags();
   cout << hex;
   INewFlags = cout.flags();
   cout << "Default flag is:" << IFlags << endl;
   cout << "New flag is:" << INewFlags << endl;
   for(int i = 0; i < 18; i++)
      cout << flags[i].flag << '\t' << flags[i].flagname << endl;
   return 0;
}
```

格式控制成员函数

```
#include <iostream>
#include <iomanip>

using namespace std;

int main()
{
   double PI = 3.1415926535;
   int precision;
   cout << fixed;
   cout << PI << endl;
   cout.width(8);
   cout.fill('0');
   for(precision = 0; precision <= 9; precision++)
   {
      cout.precision(precision);
      cout << PI << endl;
   }
   cout << PI << endl;
   cout << setw(8) << setfill('0');
   for(precision = 0; precision <= 9; precision++)
   {
      cout << setprecision(precision);
      cout << PI << endl;
   }
   return 0;
}
```

- 输入流常用函数 (istream)

read	无格式输入指定字节数
get	从流中提取字符，包括空格
getline	从流中提取一行字符
seekg	移动输入流指针
operstor>>	输入运算符

输入流常用函数 get()

```
#include<iostream>

using namespace std;
int main()
{
   char c;

   while ( (c = cin.get()) != '\n' )
      cout.put(c);

   cout << endl ;

   char s[ 80 ] ;
   cin.get(s, 10) ;
   cout << s << endl ;
}
```

输入流常用函数 getline

```
#include<iostream>

using namespace std;
int main()
{
   const int max_len = 256;
   char line[max_len];
   int i=0;

   while ( (cin.getline(line, max_len)))
   {
      cout << "[" << i << "]:" << line << endl;;
      i++;
   }
}
```

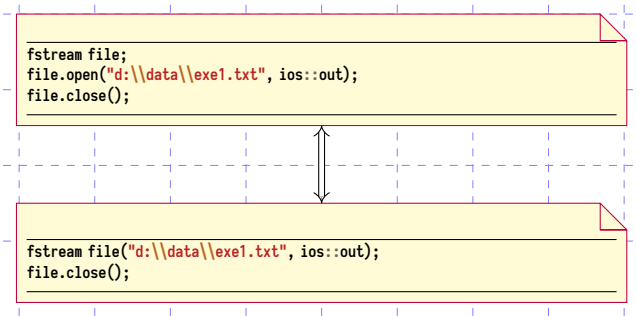
Object Oriented Programming—OOP
流类
输出流
输入流 24
文本文件
二进制文件
附件下载

输入流常用函数 getline

```
#include<iostream>

using namespace std;
int main()
{
   const int max_len = 256;
   char line[max_len];
   int i=0;

   while ( (cin.getline(line, max_len, ',')))
   {
      cout << "[" << i << "]:" << line << endl;;
      i++;
   }
}
```

- 文件的创建
  - 方法 1
    - 使用 ifstream、ofstream 或 fstream 类建立文件流对象（无参构造函数）
    - 调用成员函数 open 建立文件
  - 方法 2
    - 使用 ifstream、ofstream 或 fstream 类建立文件流对象（调用带参数构造函数）

文件的创建


fstream file;
file.open("d:\\data\\exe1.txt", ios::out);
file.close();
fstream file("d:\\data\\exe1.txt", ios::out);
file.close();

文件的创建

```
fstream file;
file.open("d:\\data\\exe1.txt", ios::out);
file.close();
```

```
ofstream file;
file.open("d:\\data\\exe1.txt");
file.close();
```

Object Oriented Programming—OOP

流类
输出流
输入流
文本文件 28
二进制文件
附件下载

文件的创建

```
fstream file;
file.open("d:\\data\\exe1.txt", ios::in);
file.close();
```

```
ifstream file;
file.open("d:\\data\\exe1.txt");
file.close();
```

OBJECT ORIENTED PROGRAMMING—OOP

流类
输出流
输入流
文本文件 29
二进制文件
附件下载

文本文件的写操作

```cpp
#include <fstream>
#include <iostream>
#include <string>

using namespace std;

int main()
{
   string name;
   float score;
   fstream myfile;
   myfile.open("record.txt", ios::out);
   if(!myfile)
   {
      cerr << "File open or create error!" << endl;
      return 0;
   }

   while( cin >> name && name != "exit")
   {
      cin >> score;
      myfile << name << ' ' << score << endl;
   }

   myfile.close();
   return 0;
}
```

OBJECT ORIENTED PROGRAMMING—OOP

流类
输出流
输入流
文本文件 30
二进制文件
附件下载

文本文件的读操作

```
#include <fstream>
#include <iostream>
using namespace std;

int main()
{
   string name;
   float score;
   ifstream myFile;

   myFile.open("record.txt");
   if(!myFile)
   {
      cerr << " record.txt open error!" << endl;
      return 0;
   }

   while(!myFile.eof())
   {
      myFile >> name >> score;
      if(myFile) cout << name << '\t' << score << endl;
   }

   myFile.close();
   return 0;
}
```

Object Oriented Programming—OOP

流类
输出流
输入流
文本文件 31
二进制文件
附件下载

- 文件合并排序操作
  - 高考成绩
  - 姓名、准考证号、所考大学、总成绩
  - 准考证号是关键字
  - 有重复记录
  - 合并，去掉重复记录，并按准考证号升序排列

```
struct StuNode
{
   string name;
   int ID;
   string univ;
   int score;
   StuNode(string _name = "", int _ID = 0, string _univ = "", int _score = 0)
      : name(_name), ID(_ID), univ(_univ), score(_score) {}
   friend bool operator == (const StuNode &l, const StuNode &r)
   {
      return (l.ID == r.ID);
   }
   friend bool operator <(const StuNode &l, const StuNode &r)
   {
      return (l.ID < r.ID);
   }
};
```

文件合并排序操作

```
void ReadFile(char *fileName, list <StuNode> &buff)
{
   ifstream inFile(fileName);
   StuNode stu;

   if(!inFile)
   {
      cout << fileName << " read error!" << endl;
      exit(0);
   };

   while(!inFile.eof())
   {
      inFile >> stu.name;
      inFile >> stu.ID;
      inFile >> stu.univ;
      inFile >> stu.score;
      buff.push_back(stu);
   }
   inFile.close();
}
```

文件合并排序操作

```
void PrintFile(char *fileName, const list <StuNode> &buff)
{
   list <StuNode>::const_iterator it;
   ofstream outFile(fileName);

   if(!outFile) return;
   for (it = buff.begin(); it != buff.end(); it++)
   {
      outFile << it->ID << "\t" << it->name << "\t"
            << it->univ << "\t" << it->score << endl;
   }
   outFile.close();
}
```

Object Oriented Programming—OOP
流类
输出流
输入流
文本文件 34
二进制文件
附件下载

文件合并排序操作

```
#include <iostream>
#include <string>
#include <list>
#include <fstream>
#include <cstdlib>

using namespace std;

int main()
{
   list <StuNode> first, second;

   ReadFile("first.txt", first);
   ReadFile("second.txt", second);

   first.sort();
   second.sort();

   first.merge(second);
   first.unique();

   PrintFile("result.txt", first);

   return 0;
}
```

二进制文件的输出

```cpp
#include <fstream>
#include <iostream>
#include <cstring>

using namespace std;

int main()
{
   char name[8];
   float score;
   fstream myfile;
   myfile.open("record.bin", ios::out | ios::binary);
   if(!myfile)
   {
      cerr << "File open or create error!" << endl;
      return 0;
   }

   while( cin >> name && strcmp(name, "exit"))
   {
      cin >> score;
      myfile.write(name, 8 * sizeof(char));
      myfile.write((char *)&score, sizeof(float));
   }

   myfile.close();
   return 0;
}
```

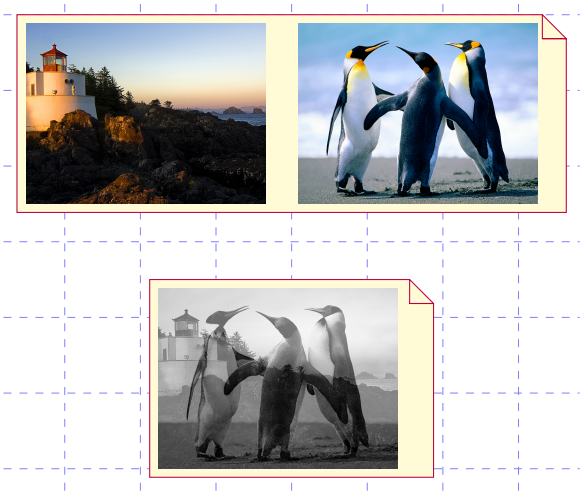
```cpp
ostream& write(const char* s , streamsize n);
```

二进制文件的输入

```
istream& read(char* s, streamsize n);
```

```
#include <fstream>
#include <iostream>
#include <string>

using namespace std;

int main()
{
   char name[8];
   float score;
   fstream myfile;
   myfile.open("record.bin", ios::in | ios::binary);
   if(!myfile)
   {
      cerr << "File open or create error!" << endl;
      return 0;
   }

   while(!myfile.eof())
   {
      myfile.read((char *)name, 8 * sizeof(char));
      myfile.read((char *)(&score), sizeof(float));
      if(myfile) cout << name << '\t' << score << endl;
   }

   myfile.close();
   return 0;
}
```

- 二进制文件的输入



Object Oriented Programming—OOP

流类
输出流
输入流
文本文件
二进制文件 38
附件下载

二进制文件的输入

```
#include <iostream>
#include <fstream>

using namespace std;

//下列代码处理图像宽度必须为 4 的倍数!
#define MIN3(x,y,z)  ((y) <= (z) ? ((x) <= (y) ? (x) : (y)) : ((x) <= (z) ? (x) : (z)))
#define MAX3(x,y,z)  ((y) >= (z) ? ((x) >= (y) ? (x) : (y)) : ((x) >= (z) ? (x) : (z)))

#pragma pack(1)
struct FILEHEADER
{
    char type[18];     // 标识"BM"
    int width;         // 图像宽
    int height;        // 图像高
    char dummy[28];    //无用信息
};

struct RGB
{
    unsigned char b; // 蓝
    unsigned char g; // 绿
    unsigned char r; // 红
};
```

OBJECT ORIENTED PROGRAMMING—OOP

流类
输出流
输入流
文本文件
二进制文件 39
附件下载

二进制文件的输入

```
int main()
{
   FILEHEADER imgHead1, imgHead2;
   RGB pixel1, pixel2, pixel;
   float blendRatio = 0.5;
   fstream imgFile1, imgFile2, imgFileResult;

   imgFile1.open("Lighthouse.bmp", ios::in | ios::binary);
   imgFile2.open("Penguins.bmp", ios::in | ios::binary);
   imgFileResult.open("Result.bmp", ios::out | ios::binary);

   if(!imgFile1 || !imgFile2 || !imgFileResult)
   {
      cerr << "File open or create error!" << endl;
      return 0;
   }

   imgFile1.read((char *)(&imgHead1), sizeof(FILEHEADER));
   if(!imgFile1)
      return 0;

   cout << imgHead1.type[0] << imgHead1.type[1] << endl;
   cout << imgHead1.width << "," << imgHead1.height << endl;

   imgFile2.read((char *)(&imgHead2), sizeof(FILEHEADER));
   if(!imgFile2)
      return 0;

   cout << imgHead2.type[0] << imgHead2.type[1] << endl;
   cout << imgHead2.width << "," << imgHead2.height << endl;
```

Object Oriented Programming—OOP
流类
输出流
输入流
文本文件
二进制文件
附件下载

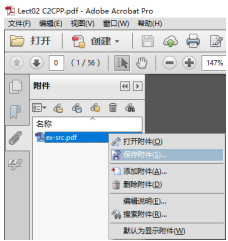
二进制文件的输入

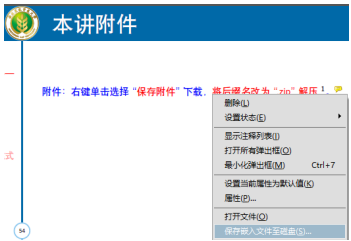
```
    if(imgHead1.width != imgHead2.width || imgHead1.height != imgHead2.height)
        return 0;

    imgFileResult.write((char *)(&imgHead1), sizeof(FILEHEADER));

    for(int i = 0; i < (imgHead1.width * imgHead1.height); i++)
    {
        imgFile1.read((char *)(&pixel1), sizeof(RGB));
        if(!imgFile1) return 0;
        imgFile2.read((char *)(&pixel2), sizeof(RGB));
        if(!imgFile2) return 0;
        pixel = pixel1;

        pixel.b = pixel1.b * blendRatio + pixel2.b * (1 - blendRatio);
        pixel.g = pixel1.g * blendRatio + pixel2.g * (1 - blendRatio);
        pixel.r = pixel1.r * blendRatio + pixel2.r * (1 - blendRatio);

        pixel.r = pixel.g = pixel.b;
        imgFileResult.write((char *)(&pixel), sizeof(RGB));
    }

    imgFile1.close();
    imgFile2.close();
    imgFileResult.close();
    return 0;
}
```

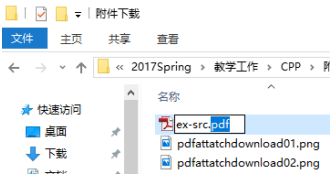
Object Oriented Programming—OOP

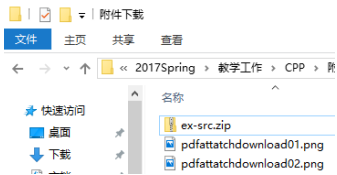
流类
输出流
输入流
文本文件
二进制文件
附件下载 41

附件：右键单击该链接，选择“保存附件”下载，将后缀名改为“.zip”解压[1] [2]。









[1] 请退出全屏模式后点击该链接。

[2] 以 Adobe Acrobat Reader 为例。

本讲结束，谢谢!
欢迎多提宝贵意见和建议

西北农林科技大学
NORTHWEST A&F UNIVERSITY
中国 · 杨凌