广州大学

GU CN Computer Networks Institute

面向对象程序设计

Object-Oriented Programming

主讲人：王国军 教授

计算机科学与网络工程学院

csgjwang@gzhu.edu.cn

http://trust.gzhu.edu.cn/

办公室：行政西楼前座532室

根据《C++程序设计基础》（第6版）和《Visual C++面向对象与可视化程序设计》（第4版）制作，仅供广州大学计算机、软件工程、网络工程、网络空间安全、人工智能2023级讲课使用。

第11章 输入流/输出流

11.1 流类和流对象

11.2 标准流和流操作

11.3 格式控制

11.4 串流

11.5 文件处理

第10章 模板

回顾

10.1 什么是模板

10.2 函数模板

10.3 类模板

*10.4 标准模板

单选题 1分

设置

（10.2）假设有函数模板定义如下，则下列选项正确的是（ ）。

```
template <typename T>
void Add( T a, T b, T &c)
{  c = a + b;  }
```

A int x, y; char z; Add(x, y, z);

B double x, y, z; Add(x, y, z);

C int x, y; float z; Add(x, y, z);

D float x; double y, z; Add(x, y, z);

提交

GU CN Computer Networks Institute

单选题 1分

设置

（10.3）建立类模板对象的实例化过程为（ ）。

A 基类→派生类

B 构造函数→对象

C 模板类→对象

D 模板类→模板函数

提交

单选题 1分

设置

（10.3）在一个类层次结构中，（　）。

A 若基类是类模板，派生类必定是类模板。

B 若基类是普通类，派生类也只能是普通类。

C 一个普通类的派生类不能增加类属参数。

D 一个派生类可以对作为基类的类模板提供实例化的类型参数。

提交

单选题 1分

设置

（10.3）若一个类模板定义了静态数据成员，正确的叙述的是（　）。

A 每一个实例化的模板类都有自己的静态数据成员

B 类模板的静态数据成员在声明类模板时定义和初始化

C 一个类模板实例化的不同模板类的全部对象共享一个静态数据成员

D 一个类模板实例化后的每个对象都有自己的静态数据成员

提交

单选题 1分

设置

（10.3）关于类模板与友元叙述错误的是（ ）。

A 一般函数可以声明为类模板的友元

B 函数模板可以声明为类模板的友元

C 一个模板类可以声明为另一个类模板的友元

D 一个模板类的友元只能是模板

提交

MOOC视频与简评

清华郑莉、李超老师第十一章流类库与输入输出

-1-导学（5分钟）：

简评：

1、内存、外存

2、Input、Output

3、I/O Stream：实现文件读写

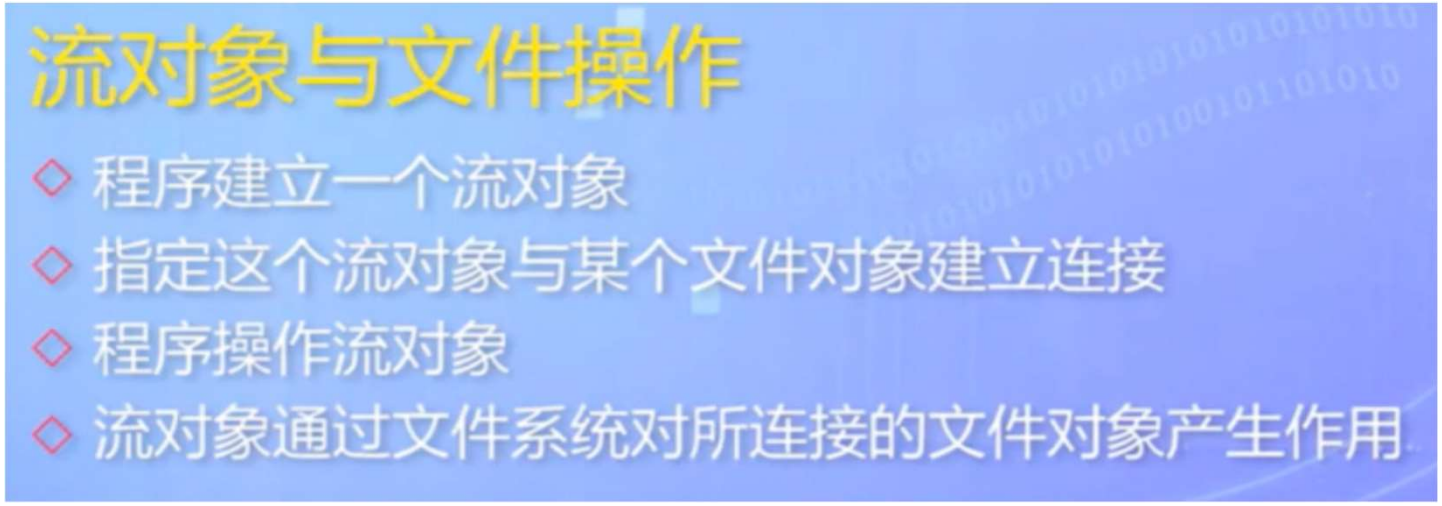
MOOC视频与简评

清华郑莉老师-2-I/O流的概念及流类库结构（4分钟）：

简评：

1、标准输入设备（如键盘）和标准输出设备（如显示器）也视为文件

2、流对象：流是信息流动的一种抽象



类名	说明	包含文件
抽象流基类		
ios	流基类	ios
输入流类		
istream	通用输入流类和其它输入流的基类	istream
ifstream	文件输入流类	fstream
istringstream	字符串输入流类	sstream
输出流类		
ostream	通用输出流类和其它输出流的基类	ostream
ofstream	文件输出流类	fstream
ostringstream	字符串输出流类	sstream
输入/输出流类		
iostream	通用输入/输出流类和其它输入/输出流的基类	istream
fstream	文件输入/输出流类	fstream
stringstream	字符串输入/输出流类	sstream
流缓冲区类		
streambuf	抽象流缓冲区基类	streambuf
filebuf	磁盘文件的流缓冲区类	fstream
stringbuf	字符串的流缓冲区类	sstream

MOOC视频与简评

清华郑莉老师-3-输出流概述（10分钟）：

最重要的三个输出流

- ostream
- ofstream
- ostringstream

预先定义的输出流对象

- cout
- cerr
- clog

MOOC视频与简评

清华郑莉老师-3-输出流概述（10分钟）：

标准输出换向

```
ofstream fout("b.out");
streambuf*  pOld  =cout.rdbuf(fout.rdbuf());
//...
cout.rdbuf(pOld); ——— -
```

MOOC视频与简评

文件输出流成员函数

- open函数
  - 把流与一个特定的磁盘文件关联起来；
  - 需要指定打开模式。
- put函数
  - 把一个字符写到输出流中。
- write函数
  - 将内存中的一块内容写到一个文件输出流中。
- seekp和tellp函数
  - 操作文件流的内部指针。
- close函数
  - 关闭与一个文件输出流关联的磁盘文件。
- 错误处理函数
  - 在写到一个流时进行错误处理。

MOOC视频与简评

清华郑莉老师-4-向文本文件输出（14分钟）：

简评：

1、以标准输出设备（显示器）为例

2、格式控制

MOOC视频与简评

操纵符（manipulator）

- 插入运算符与操纵符一起工作
  - 控制输出格式。
- 很多操纵符都定义在
  - ios_base类中（如hex()）、<iomanip>头文件（如setprecision()）。
- 控制输出宽度
  - 在流中放入setw操纵符或调用width成员函数为每个项指定输出宽度。
- setw和width仅影响紧随其后的输出项，但其它流格式操纵符保持有效直到发生改变。
- dec、oct和hex操纵符设置输入和输出的默认进制。

MOOC视频与简评

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

int main() {
   double values[] = { 1.23, 35.36, 653.7, 4358.24 };
   for(int i = 0; i < 4; i++) {
        cout.width(10);
        cout << values[i] << endl;
   }
   return 0;
}
```

输出结果:

```
      1.23
     35.36
     653.7
   4358.24
```

MOOC视频与简评

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

int main() {
   double values[] = { 1.23, 35.36, 653.7, 4358.24 };
   string names[] = { "Zoot", "Jimmy", "Al", "Stan" };
   for (int i = 0; i < 4; i++)
     cout << setw(6) << names[i]
     << setw(10) << values[i] << endl;
   return 0;
}
```

输出结果:

```
  Zoot      1.23
 Jimmy     35.36
    Al     653.7
  Stan   4358.24
```

MOOC视频与简评

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

int main() {
   double values[] = { 1.23, 35.36, 653.7, 4358.24 };
   string names[] = { "Zoot", "Jimmy", "Al", "Stan" };
   for (int i=0;i<4;i++)
     cout << setiosflags(ios_base::left)//左对齐
          << setw(6) << names[i]
          << resetiosflags(ios_base::left)
          << setw(10) << values[i] << endl;
   return 0;
}
```

输出结果:

```
Zoot        1.23
Jimmy      35.36
Al         653.7
Stan     4358.24
```

MOOC视频与简评

清华郑莉老师-4-向文本文件输出（14分钟）：

MOOC视频与简评

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

int main() {
   double values[] = { 1.23, 35.36, 653.7, 4358.24 };
   string names[] = { "Zoot", "Jimmy", "Al", "Stan" };
   for (int i=0;i<4;i++)
     cout << setiosflags(ios_base::left)
          << setw(6) << names[i]
          << resetiosflags(ios_base::left)//清除左对齐设置
          << setw(10) << setprecision(1) << values[i] << endl;
   return 0;
}
```

输出结果：

```
Zoot          1
Jimmy      4e+001
Al         7e+002
Stan       4e+003
```

MOOC视频与简评

```
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;
int main() {
   double values[] = { 1.23, 35.36, 653.7, 4358.24 };
   string names[] = { "Zoot", "Jimmy", "Al", "Stan" };
   cout << setiosflags(ios_base::fixed);
   for (int i=0;i<4;i++)
     cout << setiosflags(ios_base::left)
          << setw(6) << names[i]
          << resetiosflags(ios_base::left)//清除左对齐设置
          << setw(10) << setprecision(1) << values[i] << endl;
   return 0;
}
```

输出结果：

```
Zoot         1.2
Jimmy       35.4
Al         653.7
Stan      4358.2
```

MOOC视频与简评

```
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;
int main() {
   double values[] = { 1.23, 35.36, 653.7, 4358.24 };
   string names[] = { "Zoot", "Jimmy", "Al", "Stan" };
  cout << setiosflags(ios_base::scientific);
   for (int i=0;i<4;i++)
     cout << setiosflags(ios_base::left)
          << setw(6) << names[i]
          << resetiosflags(ios_base::left)//清除左对齐设置
          << setw(10) << setprecision(1) << values[i] << endl;
   return 0;
}
```

输出结果：

```
Zoot    1.2e+000
Jimmy   3.5e+001
Al      6.5e+002
Stan    4.4e+003
```

MOOC视频与简评

清华郑莉老师-5-向二进制文件输出（5分钟）：

二进制文件流

- 使用ofstream构造函数中的模式参量指定二进制输出模式；
- 以通常方式构造一个流，然后使用setmode成员函数，在文件打开后改变模式；
- 通过二进制文件输出流对象完成输出。

MOOC视频与简评

清华郑莉老师-5-向二进制文件输出（5分钟）：

```
#include <fstream>
using namespace std;
struct Date {
    int mon, day, year;
};
int main() {
    Date dt = { 6, 10, 92 };
    ofstream file("date.dat", ios_base::binary);
    file.write(reinterpret_cast<char *>(&dt),sizeof(dt));
    file.close();
    return 0;
}
```

MOOC视频与简评

清华郑莉老师-6-向字符串输出（7分钟）：

字符串输出流（ ostringstream ）

- 用于构造字符串
- 功能
  - 支持ofstream类的除open、close外的所有操作
  - str函数可以返回当前已构造的字符串
- 典型应用
  - 将数值转换为字符串

MOOC视频与简评

函数模板toString可以将各种支持
“<<“插入符的类型的对象转换为字符串。

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

template <class T>
inline string toString(const T &v) {
    ostringstream os;       //创建字符串输出流
    os << v;                //将变量v的值写入字符串流
    return os.str();        //返回输出流生成的字符串
}

int main() {
    string str1 = toString(5);
    cout << str1 << endl;
    string str2 = toString(1.2);
    cout << str2 << endl;
    return 0;
}
```

输出结果：
5
1.2

MOOC视频与简评

清华郑莉老师-7-输入流概述（8分钟）：

重要的输入流类：

- istream类最适合用于顺序文本模式输入。cin是其实例
- ifstream类支持磁盘文件输入
- istringstream

MOOC视频与简评

清华郑莉老师-7-输入流概述（8分钟）：

构造输入流对象

- 如果在构造函数中指定一个文件名，在构造该对象时该文件便自动打开。

```
ifstream myFile("filename");
```

- 在调用默认构造函数之后使用open函数来打开文件。

```
ifstream myFile;  //建立一个文件流对象
myFile.open("filename");  //打开文件"filename"
```

- 打开文件时可以指定模式

```
ifstream myFile("filename", ios_base::in | ios_base::binary);
```

MOOC视频与简评

清华郑莉老师-7-输入流概述（8分钟）：

使用提取运算符从文本文件输入

- 提取运算符(>>)对于所有标准C++数据类型都是预先设计好的。
- 是从一个输入流对象获取字节最容易的方法。
- ios类中的很多操纵符都可以应用于输入流。但是只有少数几个对输入流对象具有实际影响，其中最重要的是进制操纵符dec、oct和hex。

MOOC视频与简评

清华郑莉老师-7-输入流概述（8分钟）：

输入流相关函数

- open函数把该流与一个特定磁盘文件相关联。
- get函数的功能与提取运算符（>>）很相像，主要的不同点是get函数在读入数据时包括空白字符。
- getline的功能是从输入流中读取多个字符，并且允许指定输入终止字符，读取完成后，从读取的内容中删除终止字符。
- read成员函数从一个文件读字节到一个指定的内存区域，由长度参数确定要读的字节数。当遇到文件结束或者在文本模式文件中遇到文件结束标记字符时结束读取。

MOOC视频与简评

清华郑莉老师-7-输入流概述（8分钟）：

输入流相关函数（续）

- seekg函数用来设置文件输入流中读取数据位置的指针。
- tellg函数返回当前文件读指针的位置。
- close函数关闭与一个文件输入流关联的磁盘文件。

MOOC视频与简评

清华郑莉老师-8-输入流应用举例（13分钟）：

```
#include <iostream>
using namespace std;
int main() {
    char ch;
    while ((ch = cin.get()) != EOF)
        cout.put(ch);
    return 0;
}
```

MOOC视频与简评

清华郑莉老师-8-输入流应用举例（13分钟）：

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    string line;
    cout << "Type a line terminated by 't' " << endl;
   getline(cin, line, 't');
    cout << line << endl;
   return 0;
}
```

MOOC视频与简评

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

struct SalaryInfo {
    unsigned id;
    double salary;
};
int main() {
    SalaryInfo employee1 = { 600001, 8000 };
    ofstream os("payroll", ios_base::out | ios_base::binary);
    os.write(reinterpret_cast<char *>(&employee1), sizeof(employee1));
    os.close();
```

MOOC视频与简评

```
ifstream is("payroll", ios_base::in | ios_base::binary);
if (is) {
    SalaryInfo employee2;
    is.read(reinterpret_cast<char *>(&employee2), sizeof(employee2));
    cout << employee2.id << " " << employee2.salary << endl;
} else {
    cout << "ERROR: Cannot open file 'payroll'." << endl;
}
is.close();
return 0;
}
```

MOOC视频与简评

```
int main() {
    int values[] = { 3, 7, 0, 5, 4 };
    ofstream os("integers", ios_base::out | ios_base::binary);
    os.write(reinterpret_cast<char *>(values), sizeof(values));
    os.close();

    ifstream is("integers", ios_base::in | ios_base::binary);
    if (is) {
        is.seekg(3 * sizeof(int));
        int v;
        is.read(reinterpret_cast<char *>(&v), sizeof(int));
        cout << "The 4th integer in the file 'integers' is " << v << endl;
    } else {
        cout << "ERROR: Cannot open file 'integers'." << endl;
    }
    return 0;
}
```

MOOC视频与简评

```
int main() {
    ifstream file("integers", ios_base::in | ios_base::binary);
    if (file) {
        while (file) {//读到文件尾file为0
            streampos here = file.tellg();
            int v;
            file.read(reinterpret_cast<char *>(&v), sizeof(int));
            if (file && v == 0)
                cout << "Position " << here << " is 0" << endl;
        }
    } else {
        cout << "ERROR: Cannot open file 'integers'." << endl;
    }
    file.close();
    return 0;
}
```

MOOC视频与简评

清华郑莉老师-9-从字符串输入（6分钟）：

字符串输入流（ istringstream ）

- 用于从字符串读取数据
- 在构造函数中设置要读取的字符串
- 功能
  - 支持ifstream类的除open、close外的所有操作
- 典型应用
  - 将字符串转换为数值

MOOC视频与简评

```
template <class T>
inline T fromString(const string &str) {
    istringstream is(str);          //创建字符串输入流
    T v;
    is >> v;        //从字符串输入流中读取变量v
    return v;       //返回变量v
}

int main() {
    int v1 = fromString<int>("5");
    cout << v1 << endl;
    double v2 = fromString<double>("1.2");
    cout << v2 << endl;
    return 0;
}
```

输出结果：
5
1.2

MOOC视频与简评

清华郑莉老师-10-输入输出流（2分钟）：

两个重要的输入/输出流

- 一个iostream对象可以是数据的源或目的。
- 两个重要的I/O流类都是从iostream派生的，它们是fstream和stringstream。这些类继承了前面描述的istream和ostream类的功能。

MOOC视频与简评

清华郑莉老师-10-输入输出流（2分钟）：

fstream类

- fstream类支持磁盘文件输入和输出。
- 如果需要在同一个程序中从一个特定磁盘文件读并写到该磁盘文件，可以构造一个fstream对象。
- 一个fstream对象是有两个逻辑子流的单个流，两个子流一个用于输入，另一个用于输出。

MOOC视频与简评

清华郑莉老师-10-输入输出流（2分钟）：

MOOC视频与简评

清华郑莉老师-11-小结（3分钟）：

本章主要内容

- I/O流的概念
- 流类库结构
- 输出流
- 输入流
- 输入/输出流
- 读写文本文件的格式控制。

MOOC视频与简评

清华李超老师-13-实验十一（13分钟）：

预备知识1：

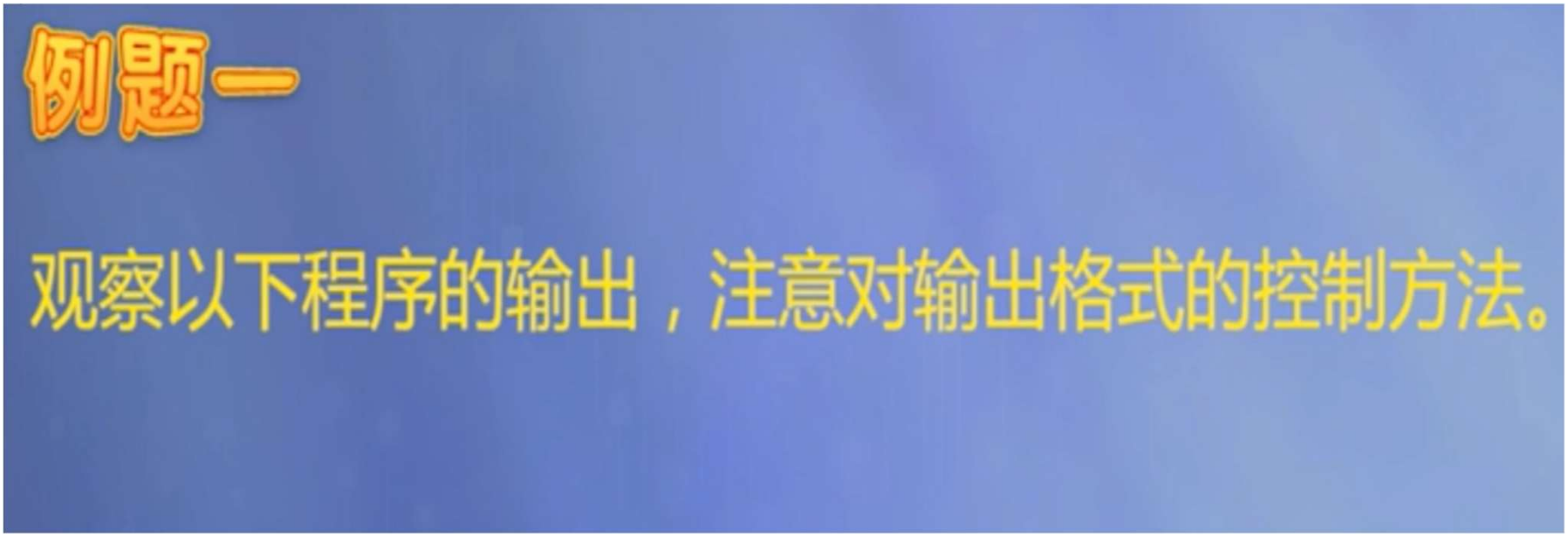
宏定义： #define D(a) T << #a << endl; a
含义：在上述宏定义完成后，在后面出现 D(a)的地方，
用T << #a << endl; a来替代

例如：
D(int i = 53;) //这里的“int i = 53;”就是“a”
所以，实际展开执行的是：
T <<“ int i = 53;” << endl; int i = 53; //这样两条语句

也即：简洁巧妙地实现了将“a”的文字输出到T的同时，执行“a”语句。无需重复书写类似的语句。便于观察和记录语句“a”的执行效果。

MOOC视频与简评

清华李超老师-13-实验十一（13分钟）：



MOOC视频与简评

清华李超老师-13-实验十一（13分钟）：

```
4
5  #include <fstream>
6  using namespace std;
7  #define D(a) T << #a << endl; a
8  ofstream T("output.out");
9
10 int main() {
11     D(int i = 53;)
12     D(float f = 4700113.141593;)
13     char* s = "Is there any more?";
14
```

调用宏D(a)的实际效果即：
把语句“a”的文字输出到文件output.out，并且执行语句“a”。

MOOC视频与简评

清华李超老师-13-实验十一（13分钟）：

```
13    char* s = "Is there any more?";
14
15    D(T.setf(ios::unitbuf);)
16
17    D(T.setf(ios::showbase);)
18    D(T.setf(ios::uppercase);)
19    D(T.setf(ios::showpos);)
20    D(T << i << endl;)
21    D(T.setf(ios::hex, ios::basefield);)
22    D(T << i << endl;)
23    D(T.unsetf(ios::uppercase);)
```

MOOC视频与简评

output.out - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
int i = 53;
float f = 47011113.141593;
T.setf(ios::unitbuf);
T.setf(ios::showbase);
T.setf(ios::uppercase);
T.setf(ios::showpos);
T << i << endl;
+53
T.setf(ios::hex, ios::basefield);
T << i << endl;
0X35
T.unsetf(ios::uppercase);
T.setf(ios::oct, ios::basefield);
T << i << endl;
065
T.unsetf(ios::showbase);
T.setf(ios::dec, ios::basefield);
T.setf(ios::left, ios::adjustfield);
T.fill('0');
T << "fill char: " << T.fill() << endl;
fill char: 0
```

MOOC视频与简评

清华李超老师-13-实验十一（13分钟）：

预备知识2：

在调用一个可执行程序时，某些情况下需要向程序传递参数。

例如：
我们可以在控制台中键入notepad.exe，回车后将执行记事本程序。
如果我们希望在打开notepad时同时打开一个文本文件，
可以在notepad.exe 后面跟上文件的路径和名字，
如notepad.exe example.txt（文件在当前路径）。

MOOC视频与简评

清华李超老师-13-实验十一（13分钟）：

预备知识2：

那么程序中如何能得到这些输入参数呢？

这个工作是编译器帮我们完成的，编译器将输入参数的信息
放入main函数的参数列表中，即：int main(int argc,char*argv[])。
里面的两个型参是有什么作用？

第一个参数argc记录了输入参数的个数，
第二个参数argv[]记录了输入的字符串数组。

MOOC视频与简评

清华李超老师-13-实验十一（13分钟）：

预备知识2：

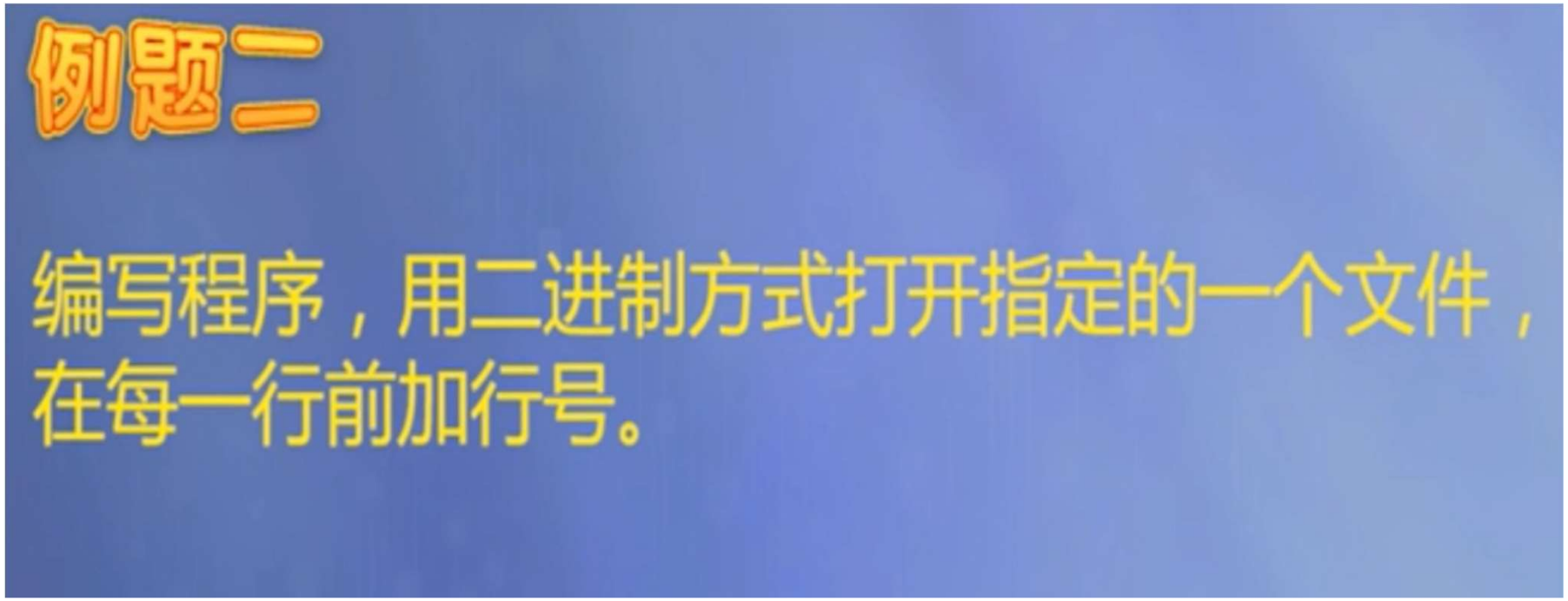
以notepad.exe　example.txt为例

argc是2，就是说argv数组中有两个有效单元
第一单元指向的字符串是"notepad.exe"
第二单元指向的字符串是"example.txt"

** argv数组中的第一个单元指向的字符串总是可执行程序的名字，
以后的单元指向的字符串依次是程序调用时的参数。
** 这个赋值过程是编译器完成的，我们只需要读出数据就可以了。

MOOC视频与简评

清华李超老师-13-实验十一（13分钟）：



MOOC视频与简评

清华李超老师-13-实验十一（13分钟）：

```
3  #include <iostream>
4  #include <fstream>
5  using namespace std;
6
7  int main(int argc, char* argv[])
8  {
9      ifstream in;
10     in.open(argv[1], ios::binary);
11     if (!in) {
12         cout << "Cannot open file.";
13         return 1;
14     }
```

MOOC视频与简评

清华李超老师-13-实验十一（13分钟）：

```
12          cout << "Cannot open file.";
13          return 1;
14      }
15      const int bsz = 1024;
16      char buf[bsz];
17      int line = 0;
18      while(in.getline(buf, bsz)) {
19          cout << ++line << ": " << buf << endl;
20      }
21      return 0;
22  }
```

MOOC视频与简评

清华李超老师-13-实验十一（13分钟）：

MOOC视频与简评

管理员: C:\Windows\system32\cmd.exe

```
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Users\Administrator>IOtest.exe output.out
1: int i = 53;
2: float f = 4700113.141593;
3: T.setf(ios::unitbuf);
4: T.setf(ios::showbase);
5: T.setf(ios::uppercase);
6: T.setf(ios::showpos);
7: T << i << endl;
8: +53
9: T.setf(ios::hex, ios::basefield);
10: T << i << endl;
11: 0X35
12: T.unsetf(ios::uppercase);
13: T.setf(ios::oct, ios::basefield);
```

单选题 1分

设置

（11.2）在下列流类中，可以用于处理输入/输出的是（　）。

A ios

B iostream

C strstream

D fstream

提交

GU CN Computer Networks Institute

单选题 1分

设置

（11.2）在下列选项中，（　）是istream类的对象。

A cerr

B cin

C clog

D cout

提交

单选题 1分

设置

（11.2）在下列选项中，不可以作为输出流对象的是（　）。

A 文件

B 内存

C 键盘

D 显示器

提交

单选题 1分

设置

（11.2）在下列选项中，用于处理字符串流的是（　）。

A strstream

B ios

C fstream

D iostream

提交

单选题 1分

设置

（11.2）能够从输入流中提取指定长度的字节序列的函数是（　）。

A get

B getline

C read

D cin

提交

单选题 1分

设置

（11.2）能够把指定长度的字节序列插入到输出流中的函数是（　）。

A put

B write

C cout

D print

提交

单选题 1分

设置

（11.2）getline函数的功能是从输入流中读取（　）。

A 一个字符

B 当前字符

C 一行字符

D 指定若干个字节

提交

单选题 1分

设置

（11.2）要进行文件的输出，除了包含头文件iostream外，还要包含头文件（　）。

A ifstream

B fstream

C ostream

D cstdio

提交

单选题 1分

设置

（11.2）用标准输入流对象cin与提取操作符>>连用进行输入时，将空格与回车当作分隔符，使用（　）成员函数进行输入时可以指定输入分隔符。

A get()

B put()

C read()

D gcount()

提交

单选题 1分

设置

（11.2）在ios类中，状态字用于记录流错误状态，其每位对应一种流的错误状态，其中（　）表示流数据已遭到损坏。

A goodbit

B eofbit

C failbit

D badbit

提交

单选题 1分

设置

（11.3）在下列选项中，用于清除基数格式位设置以十六进制数输出的语句是（　）。

A　cout<<setf(ios::dec, ios::basefield);

B　cout<<setf(ios::hex, ios::basefield);

C　cout<<setf(ios::oct, ios::basefield);

D　cin>>setf(ios::hex, ios::basefield);

提交

单选题 1分

设置

（11.3）下列格式控制符，既可以用于输入，又可以用于输出的是（　）。

A setbase

B setfill

C setprecision

D setw

提交

单选题 1分

设置

（11.3）若在I/O流的输出中使用控制符setfill()设置填充字符，应包括的头文件是（　）。

A cstdlib

B iostream

C fstream

D iomanip

提交

单选题 1分

设置

(11.3) 以下语句的输出结果是（ ）。

```
cout<<setw(3)<<25<<oct<<25<<hex<<endl;
```

A 25 25

B 2531

C 31 19

D 25 31

提交

单选题 1分

设置

（11.3）以下语句的输出结果是（ ）。

```
cout<<setfill('*')<<setw(10)<< "Hello!"<<endl;
```

A ****Hello!

B Hello!****

C Hello!

D Hello!

提交

单选题 1分

设置

(11.3) 以下语句的输出结果是（ ）。

```
cout.fill('*');  cout.width(10);
cout<<setiosflags(ios::left)<<123.45<<endl;
```

A ****123.45

B **123.45**

C 123.45****

D ***123.45*

提交

单选题 1分

设置

（11.4）使用串流类需要包含（　）头文件。

A iostream

B iomanip

C fstream

D strstream

提交

单选题 1分

设置

（11.4）串流在提取数据时，对字符串按（　）解释。

A 整型数据

B 浮点型数据

C 变量类型

D ASCII码

提交

单选题 1分

设置

（11.4）串流在插入数据时，把各种类型数据转换成（　）。

A 二进制码

B 十进制码

C 格式化ASCII码

D 计算结果

提交

单选题 1分

设置

（11.5）在下列流类中，可以用于处理文件的是（　）。

A ios

B iostream

C strstream

D fstream

提交

单选题 1分

设置

（11.5）在文件操作中，表示以追加方式打开文件的模式是（　）。

A iso::ate

B iso::app

C iso::out

D iso::trunc

提交

单选题 1分

设置

（11.5）下列打开文件的语句中，（ ）是错误的。

A ofstream ofile; ofile.open("abc.txt", ios::binary);

B fstream iofile; iofile.open("abc.txt", ios::ate);

C ifstream ifile("abc.txt");

D cout.open("abc.txt", ios::binary);

提交

单选题 1分

设置

（11.5）以下关于文件操作的叙述中，不正确的是（　）。

A 打开文件的目的是使文件对象与磁盘文件建立联系

B 文件的读写过程中，程序将直接与磁盘文件进行数据交换

C 关闭文件的目的之一是保证输出的数据写入磁盘文件中

D 关闭文件的目的之一是释放内存中的文件对象

提交

单选题 1分

设置

（11.5）以下不能正确创建输出文件对象并使其与磁盘文件相关联的语句是（　）。

A
```
ofstream myfile;
myfile.open("d:\\ofile.txt");
```

B
```
ofstream *myfile=new ofstream;
myfile->open("d:\\ofile.txt");
```

C
```
ofstream myfile("d:\\ofile.txt");
```

D
```
ofstream *myfile=new ("d:\\ofile.txt");
```

提交

单选题 1分

设置

（11.5）要打开文件 D:\file.dat，并能够写入数据，正确的语句是（　）。

A　ifstream infile("D:\\file.dat", ios::in);

B　ifstream infile("D:\\file.dat", ios::out);

C　ofstream outfile("D:\\file.dat", ios::in);

D　fstream infile("D:\\file.dat", ios::in|ios::out);

提交

GU CN Computer Networks Institute

单选题 1分

设置

（11.5）能实现删除文件功能的语句是（ ）。

A ofstream fs("date.dat", ios::trunc);

B ifstream fs("date.dat", ios::trunc);

C ofstream fs("date.dat", ios::out);

D ifstream fs("date.dat", ios::in);

提交

单选题 1分

设置

（11.5）设已定义浮点型变量data，以二进制代码方式把data的值写入输出文件流对象outfile中，正确的语句是（　）。

A　outfile.write((double *) &data, sizeof(double));

B　outfile.write((double *) &data, data);

C　outfile.write((char *) &data, sizeof(double));

D　outfile.write((char *) &data, data);

提交

单选题 1分

设置

（11.5）把二进制数据文件流fdat的读指针移到文件头的语句是（　）。

A fdat.seekg(0, ios::beg);

B fdat.tellg(0, ios::beg);

C fdat.seekp(0, ios::beg);

D fdat.tellp(0, ios::beg);

提交

作　　业

习题