C++语言程序设计
第一章 C++新特征

王焦乐

http://faculty.hitsz.edu.cn/jlwang

哈尔滨工业大学（深圳）

机电工程与自动化学院

本章主要内容

- C语言与面向过程编程
- C++对C的补充
  - 输入输出流（iostream）、命名空间（namespace）、注释（comment）
  - 引用（reference）、字符串变量（String）、函数重载（overload）
  - 默认参数（default argument）、函数模板（template）、变量（variables）
  - 作用域（scope）、声明与定义（declare & define）、创建与释放（new & delete）
- C++程序的编写和实现
- 面向对象编程初步

0、C语言与面向过程编程

□ 面向过程编程

C

- 结构化和模块化
- 面向过程

C++

- 增加了面向对象机制
- 标准模版库

Hello World

```cpp
//main.cpp
#include <iostream>
using namespace std;
int main()
{
    cout << "Hello world!" << endl;
    return 0;
}
```

```c
/*main.c*/
#include <stdio.h>

int main()
{
    printf("Hello world!\n ");
    return 0;
}
```

示例程序1

0、C语言与面向过程编程

C语言的面向过程

- **选择控制结构**
  - **单分支、多分支、多路**

```
if(表达式 1)
     if(表达式 2) 语句 1
     else         语句 2
else
     if(表达式 3) 语句 3
     else         语句 4
```

```
switch(表达式)
   {  case 常量 1: 语句1
      case 常量 2: 语句2
                 ┆
      case 常量 n: 语句n
      default   :  语句n+1
   }
```

- **循环控制结构**
  - **计数控制、条件控制**

```
while (循环控制表达式)
{
    语句序列
}
```

```
do
{
    语句序列
}while(循环控制表达式)
```

```
for (初始化表达式;循环控制表达式;
增值表达式)
{
     语句序列
}
```

- **结构化、模块化程序设计**
  - **采用顺序、选择、循环三种基本结构**
  - **单入口、单出口**
  - **无不可达语句、无死循环**
  - **自顶而下（top-down）**
  - **逐步求精（stepwise refinement）**
  - **高聚合、低耦合，模块相对独立**

```
返回值类型 函数名（类型 形式参数 1,
类型 形式参数 2,  ...）
{
  声明语句序列
  可执行语句序列
}
```

0、C语言与面向过程编程

□ 一个简单的例子

```c
int set_matrix(int n, int m, int (*p)[n][m])
{
  for (int i = 0; i < n; i++)
    for (int j = 0; j < m; j++)
      (*p)[i][j] = (i+1)*10 + j+1;
  return 0;
}

int print_matrix(int n, int m, int (*p)[n][m])
{
  printf("\nElements of matrix A: \n");
  for (int i = 0; i < n; i++)
    for (int j = 0; j < m; j++)
    {
      printf("A%d    ", (*p)[i][j]);
      if (j==m-1) printf("\n");
    }
}
```

```c
/* Print elements of a matrix A*/
int main()
{
  int n=99, m=99;
  do{
    printf("Please input row number (< 10): ");
    scanf("%d", &n);
    getchar();
    printf("\nPlease input col number (< 10): ");
    scanf("%d", &m);
  }while(n>9 || m>9);

  /*Allocate memory*/
  int (*p)[n][m] = malloc(sizeof *p);
  if (!p)  return -1;
  /*Set matrix*/
  set_matrix(n, m, p);
  /*Print matrix*/
  print_matrix(n, m, p);
  /*Release memory*/
  free(p);
  return 0;
  }
```

C++语言的字符集

等同于C语言的字符集，包括：（1）大小写英文字母（2）数字字符（3）其他ASCII码字符

单词及词法规则

单词是构成语句的关键成份之一，通常由若干字符组成，C++有几种单词：1.关键字 2.标识符3.运算符 4.分隔符 5.注释符

关键字

是C++语言中的命令字，它是预定义好的单词，C++编译程序对其有专门的解释

int、float、if、else、while、switch等等

1、C++对C的补充

标识符

程序员用标识符对程序中元素实施命名，包括函数名、类名、对象名、类型名、变量名、常变量名、数组名等。

标识符以字母或下划线开始，后跟字母、数字、下划线，**标识符区分大小写字母**

运算符

运算符代表某种操作的单词，由一个或多个字符组成。

注意运算符的优先级和结合顺序。参考附录B。

分隔符

在语句中关键字和标识符之间、各个语句之间要用分隔符分开。C++常用的分隔符有空格、逗号、分号、冒号、大括号。

注释符

C++提供了两种注释符。 / * 注释* / // 注释

1、C++对C的补充

C

- 结构化和模块化
- 面向过程

C++

- 增加了面向对象机制
- 标准模版库

Hello World

```cpp
//main.cpp
#include <iostream>
using namespace std;
int main()
{
    cout << "Hello world!" << endl;
    return 0;
}
```

```c
/*main.c*/
#include <stdio.h>

int main()
{
    printf("Hello world!\n ");
    return 0;
}
```

1、C++对C的补充

iostream

- istream：输入流
- ostream：输出流
- 流：指要从某种IO设备上读入或写出的字符序列
- 标准库中的IO对象
  - cin：标准输入
  - cout：标准输出
  - cerr：标准错误，输出警告和错误信息
  - clog：产生程序执行的一般信息

1、C++对C的补充

iostream

```
#include <iostream>
int main()
{
    std::cout << "Enter two numbers:" <<std::endl;
    int v1,v2;
    std::cin>>v1>>v2;
    std::cout<< "The sum of  " <<v1<< " and  " <<v2
             << " is  " <<v1+v2<<std::endl;
    return 0;
}
```

endl（end of line），将它写入输出流时，具有换行的效果，并刷新与设备相关的缓冲区（buffer）

cout（character output stream），输出流对象，<<是**插入运算符**

cin（character input stream），输入流对象，>>是**提取运算符**

1、C++对C的补充

命名空间

```
#include <iostream>
std::cin
std::cout
```

- 前缀std::表明cin和cout是定义在命名空间std中
- :: 作用域操作符
- 使用命名空间，可以避免使用了与程序中所定义名字相同的名字而引起冲突

```
#include <iostream>
using namespace std;
cout << “Hello world!” << endl;
```

```
#include <iostream>
using std::cout;
cout << “Hello world!” << endl;
```

1、C++对C的补充

注释

两种注释方法：

- **单行注释：　//**
- **成对注释：　/*　*/**
  - **注释对不可嵌套**

- **cout<<“a+b+c”<<endl;**
- **cout<<“/*a+b*/+c”<<endl;**
- **cout<</*　“a+/*b+c*/”　*/<<endl;**

1、C++对C的补充

变量

- 变量提供了程序可以操作的有名字的存储区
- 变量都有一个类型，该类型决定了变量的内存大小和布局、取值范围、操作集合。
- 左值
  - 可以出现在赋值语句的左边或者右边
- 右值
  - 只能出现在赋值的右边
- 定义：用于为变量分配存储空间，变量必须且仅能定义一次
- 声明：用于向程序表明变量的类型和名字。extern。可以多次声明;

1、C++对C的补充

const定义常变量

- 格式：
- const 类型 变量名 = 常数
- 例：const float PI= 3.14159;
- 有数据类型
- 占用储存单元，有地址
- 运行期间变量的值固定，不能变
- 常与指针结合使用，有：
  - 指向常变量的指针
  - 常指针
  - 指向常变量的常指针

□C语言中 `#define PI 3.14159`

```
int a =1; b=2;
#define PI 3.14159
#define R a+b
cout << PI*R*R << endl;
```

预编译时进行置换，容易出错

1、C++对C的补充

声明 与 定义

```
double volumn(double r=1, double h=1);  声明

int main()
{
    cout<< volumn()<<endl;
    cout<< volumn(2)<<endl;
    cout<< volumn(2,2)<<endl;
    return 0;
}

double volumn(double r, double h)  定义
{
    return pi*r*r*h;
}
```

1、C++对C的补充

声明 与 定义

```
//in file a
extern double r;
```

声明

```
//in file b
double r;
```

定义

- 使用前需声明，可多次声明，但只能一次定义
- 变量定义会申请内存空间，也可能为变量赋一个初始值。
- 变量显式初始化的声明也会成为定义
- 函数声明与定义区别在于后者多个函数体
- 声明放在头文件中，定义放在cpp文件中，方便调用。

1、C++对C的补充

函数重载

- **函数重载的规则**
  - **函数名称必须相同。**
  - **参数列表必须不同（个数不同、类型不同、参数排列顺序不同等）。**
  - **函数的返回类型可以相同也可以不相同。**
  - **仅仅返回类型不同不足以成为函数的重载。**

1、C++对C的补充

函数重载

➢ 设计程序计算三个数中的最大数

```
int max(int a, int b, int c) //整数
{ if (b > a)
    a = b;
  if (c > a)
    a = c;
  return a;}
float max(float a, float b, float c) //实数
{ if (b > a)
    a = b;
  if (c > a)
    a = c;
  return a;}
long max(long a, long b, long c) //长整数
{ if (b > a)
    a = b;
  if (c > a)
    a = c;
  return a;}
```

```
int main()
{
  int a, b, c;
  float d, e, f;
  long g, h, i;
  cin >> a >> b >> c;
  cin >> d >> e >> f;
  cin >> g >> h >> i;
  int m;
  m = max(a, b, c); //函数值为整型
  cout << "max_i=" << m << endl;
  float n;
  n = max(d, e, f); //函数值为实型
  cout << "max_f=" << n << endl;
  long int p;
  p = max(g, h, i); //函数值为长整型
  cout << "max_l=" << p << endl;
  return 0;
}
```

1、C++对C的补充

函数重载

➢ 设计程序用一个函数名求两个整数或三个整数中的最大数

```
int max(int a, int b, int c) //求3个整数中的最大者
{
  if (b > a)
    a = b;
  if (c > a)
    a = c;
  return a;
}

int max(int a, int b) //求两个整数中的最大者
{
  if (a > b)
    return a;
  else
    return b;
}
```

```
int main()
{
  int a = 7, b = -4, c = 9;
  //输出3个整数中的最大者
  cout << max(a, b, c) << endl;

  //输出两个整数中的最大者
  cout << max(a, b) << endl;
  return 0;
}
```

1、C++对C的补充

函数模板

- **模板语言实现MAX函数**

```
template<typename T>

T Max( T a, T b)

{ return  a > b ? a : b;}
```

Max(10, 5);

Max(10.5, 5.5);

Max(‘a’ , ‘c’);

顺利运行

```
template <typename T>
T max(T a, T b, T c) //用虚拟类型T表示类型
{
  if (b > a)
    a = b;
  if (c > a)
    a = c;
  return a;
}
```

```
int main()
{
  int i1 = 8, i2 = 5, i3 = 6, i;
  double d1 = 56.9, d2 = 90.765, d3 = 43.1, d;
  long g1 = 67843, g2 = -456, g3 = 78123, g;
  i = max(i1, i2, i3);
  d = max(d1, d2, d3);
  g = max(g1, g2, g3);
  cout << "i_max=" << i << endl;
  cout << "d_max=" << d << endl;
  cout << "g_max=" << g << endl;
  return 0;
}
```

1、C++对C的补充

默认参数

C++允许为函数的参数设置默认值，这时调用函数时，如果**没有实参，就以默认值作为实参值**。

格式：

形参类型 形参变量名 = 常数

功能：调用函数时，如果没有实参，就以常数作为该形参的值; 如果有实参，仍以实参的值作为该形参的值。

注意：有默认值的形参必须放在形参表的右边，**不允许**无默认参数值和有默认参数值的形参**交错排列**。

```
double volumn(double r=1,
              double h=1)
{
    return pi*r*r*h;
}
```

- **必须在函数调用前将默认值的信息通知编译系统**
- **注意二义性**

1、C++对C的补充

默认参数

```
double volumn()
{
    return pi*1*1;
}

double volumn(double r)
{
    return pi*r*r*1;
}

double volumn(double r, double h)
{
    return pi*r*r*h;
}
```

```
double volumn(double r = 1, double h = 1)
{
  return pi * r * r * h;
}

int main()
{
  cout << volumn() << endl;

  cout << volumn(2) << endl;

  cout << volumn(2, 2) << endl;

  return 0;
}
```

1、C++对C的补充

引用

- Reference
- 对象的别名，实际应用中引用主要用作函数的形式参数
  - int a;
  - int &b=a;
  - a=1;
  - cout<<a<<" "<<b<<endl;
  - b=2;
  - cout<<a<<" "<<b<<endl;
- 引用定义必须初始化
- 注意：两个变量不能用同一个别名

例：int a = 3 ,b =4;
int &c = a; // c是a 的别名
int &c = b; // 错误的用法

一个变量可以有多个别名

例：int a = 3;
int & b= a;
int & c= b;
变量a 有两个别名b和c。

```
#include <iostream>
using namespace std;
int main()
{
  int a = 10;
  int &b = a; //声明b是a的引用
  a = a * a;
  cout << a << "  " << b << endl;
  b = b / 5;
  cout << b << "  " << a << endl;
  return 0;
}
```

1、C++对C的补充

引用

- C的例子

```
void swap(int *a, int *b)
{
    int tmp;
    tmp=*a;
    *a=*b;
    *b=tmp;
}
int a=4,b=6;
swap(&a,&b)
cout<<a<< " "<<b<<endl;
```

- C++的例子

```
void swap(int &a, int &b)
{
    int tmp;
    tmp=a;
    a=b;
    b=tmp;
}
int a=4,b=6;
swap(a,b)
cout<<a<< " "<<b<<endl;
```

1、C++对C的补充

引用

- **引用与指针**

Reference	Pointer

int a=10;
int &r=a;
r=5;
	int a=10;
	int *r=&a;
	*r=5;

1. **引用必须在创建时初始化，指针可以随时初始化**
2. **引用被初始化到一个变量后，其不能再更改为其他变量的引用，指针可以随时指向其他变量。**
3. **不能有空引用，引用必须关联到一个合法的内存空间。指针可以指向空。**

1、C++对C的补充

引用

(1) 引用变量都具有非void类型
(2) 不能建立引用的数组
(3) 可以建立常引用变量，不允许修改常引用变量的值

例：
```
int i ;
const  int &a = i;
a = 3;   //    错误的用法
i  = 8;   //    i 不是常变量，可以修改
```

(4) 可以建立指针变量的引用变量

例：
```
int i;
int  *p = & i;
int * &pt = p; // pt是p的别名变量，同时
               // 也是指针变量
```

(5) 常量或表达式初始化引用

```
int i=5;
const int &a=i+3;

double d = 3.14;
const int &a = d;
```

临时变量必须用const限定

1、C++对C的补充

内置函数

- 编译时，将函数代码嵌入
- 省去了调用环节，提高了执行速度
- 节省运行时间，但增加了长度

定义格式：

inline 函数类型 函数名(形参表)

{　函数体　}

调用格式：　函数名（实参表）

```
// 计算三个整数中的大数
#include <iostream>
using namespace std;
// 这是一个内置函数，求3个整数中的最大者
inline int max(int a, int b, int c)
{
  if (b>a) a=b;
  if (c>a) a=c;
  return a;
}

int main( )
{
  int i=7,j=10,k=25,m;
  m=max(i,j,k);
  cout<<"max="<<m<<endl;
  return 0;
}
```

1、C++对C的补充

内置函数

```
// 计算三个整数中的大数
#include <iostream>
using namespace std;
// 这是一个内置函数，求3个整数中的最大者
inline int max(int a, int b, int c)
{
  if (b>a) a=b;
  if (c>a) a=c;
  return a;
}

int main( )
{
  int i=7,j=10,k=25,m;
  m=max(i,j,k);
  cout<<"max="<<m<<endl;
  return 0;
}
```

```
m=max(i,j,k);
```

被置换成：

```
{
  a=i ; b = j ; c= k;
  if ( b>a) a=b;
  if ( c>a) a=c;
  m=a;
}
```

1、C++对C的补充

作用域

- 作用域用大括号来界定
- 名字从其声明点开始直到其声明所在的作用域结束处可见
- 全局作用域
  - 定义在所有函数外部的名字具有全局作用域
- 局部作用域
  - 如定义在函数内部
- 语句作用域
  - 如定义在for语句的作用域中
- 通常把一个对象（变量）定义在它首次使用的地方

```
#include <iostream>

using namespace std;

int global_val;

int main()
{
    int local_val;

    for (int i=0;i<10;i++)
    {
        int val=10;

        cout << i*val << endl;
    }

    return 0;
}
```

示例程序3

1、C++对C的补充

作用域

::val表示全局变量val。
注意不能用 :: 访问局部变量

```
#include <iostream>
using namespace std;
int val = 1;

int main()
{
  int val = 2;

  for (int i = 0; i < 3; i++)
  {
    int val = 3;

    cout << "Local: ";
    cout << i * val << endl;
    cout << "Global: ";
    cout << i * ::val << endl;
    cout << "_________" << endl;
  }
}
```

1、C++对C的补充

字符串变量

- **string**
- **字符串赋值操作**
- **字符串内字符操作**
- **cin ； cout**
- **字符串连接操作：+**
- **字符串比较操作：==, >, >=, <, <=, !=**

```
int main()
{
  string a,c;
  string b = "Thee Little Birds";
  a=b;
  cout << a << endl;
  cout << b << endl << endl;

  cout << "Input a word: " << endl;
  cin >> c;
  cout << "The input word is: " << c << endl << endl;

  string word1,word2="C++ ";
  string word3="Language";
  word1=word2+word3;
  cout << "The result is: " << word1 << endl << endl;

  if(word2=="C++")
  {cout<<"This is a cool language!"<< endl << endl;}
  else
  {cout<<"Something bad happened"<< endl << endl;}
}
```

1、C++对C的补充

new与delete

- 动态的创建和释放空间函数

```
int *p=new int;
delete p;
int *p=new int(10);
delete p;
int *p=new int[10];
delete [] p;
```

- 执行delete操作后，要

```
p=nullptr;
```

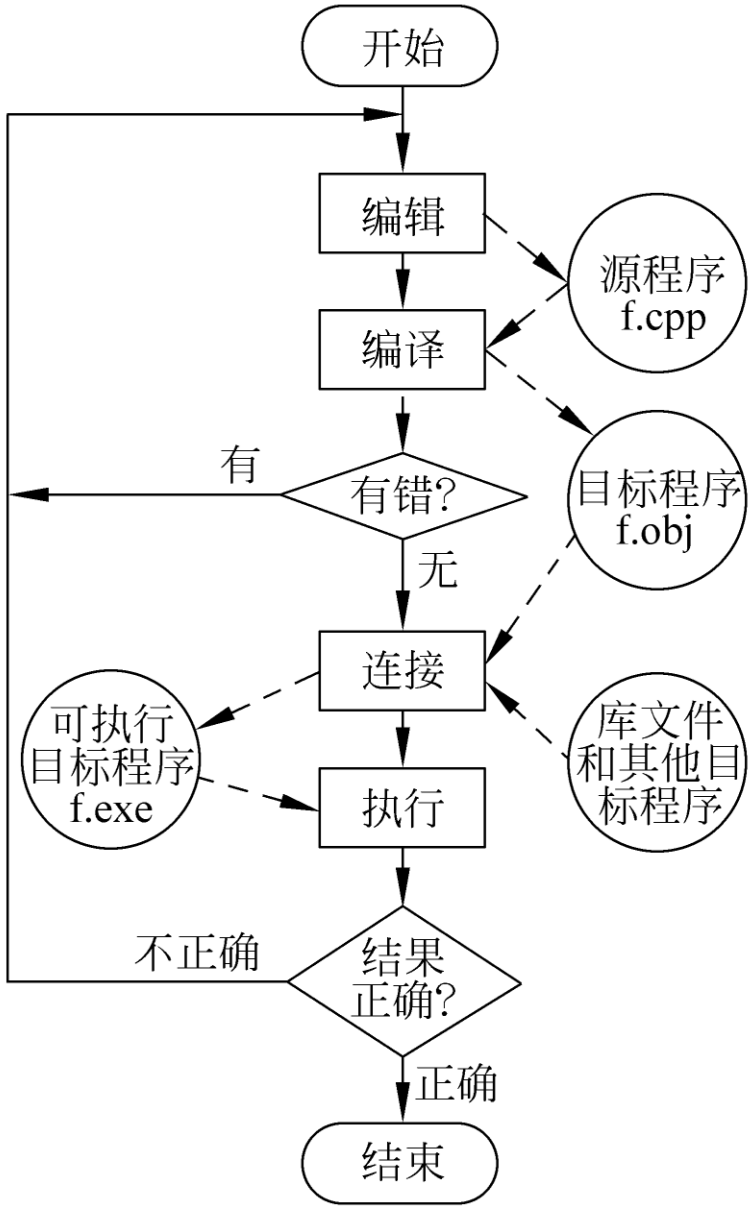
- 注意不要混合使用new、delete、malloc、free。要正确搭配，不要用new分配内存后，又用free释放内存。

```
int *p1=new int;
int *p2=new int(10);
int *p3=new int[10];
p3[0]=1;
p3[1]=2;
p3[2]=3;
cout<<*p1<<endl<<*p2<<endl<<*p3<<endl;

delete p1;
delete p2;
delete [] p3;
p1= nullptr;
p2= nullptr;
p3= nullptr;
```

2、C++程序的编写和实现

1. 用C++语言编写程序
2. 对源程序进行编译
3. 将目标文件连接
4. 运行程序
5. 分析运行结果



2、C++面向对象编程初步

- 面向对象编程（Object-oriented programming，OOP）
  - 出发点：更直接地描述客观世界中存在的事物(对象)以及它们之间的关系。
  - 特点：
    - 高级语言
    - 将客观事物看作具有属性和行为的对象
    - 通过抽象找出同一类对象的共同属性和行为，形成类
    - 通过类的继承与多态实现代码重用
  - 面向对象与面向过程不矛盾
    - 函数、结构化