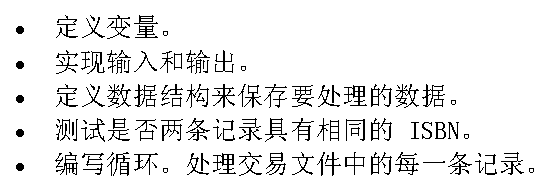
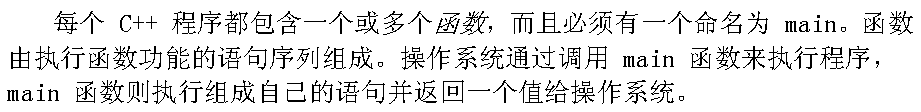
# 快速入门

程序必须



1.1编写简单的C++程序



Int main()

{

Return 0;

}

操作系统通过main函数返回的值来确定程序是否成功执行完成。返回0值表明程序成功执行完毕。

Main函数的返回值必须时int型，该类型表示整数。Int类型是**内置类型**，即该类型是由C++语言定义的。

在大多数系统中，main函数的返回值是一个状态指示器。返回值0往往表示main函数成功执行完毕。任何其他非零的返回值都有操作系统定义的含义。通常非零返回值表明有错误出现。每一种操作系统都有自己的方式告诉用户main函数返回什么内容。

编译与执行程序

程序编写完成后需要进行编译

1.2 初窥输入/输出

Iostream库的基础是两种命名为istream和ostream的类型，分别表示输入流和输出流。流是指要从某种IO设备上读入或写出的字符序列。

1.2.1 标准输入与输出对象

标准库定义了4个IO对象。

处理输入时使用命名为cin的istream类型对象（标准输入）；

处理输出时使用命名为cout的ostream类型对象（标准输出）

[ostream]cerr（标准错误）：用来输出警告和错误信息给程序的使用者；

Clog对象用于产生程序执行对的一般信息

1.2.2 一个使用IO库的程序

#include <iostream>//预处理指示

int main()

{

std::cout << "Enter two numbers:" << std::endl;//<<输出操作符

int v1, v2;

std::cin >> v1 >> v2;

std::cout << "The sum of " << v1 << " and " << v2

<< " is " << v1 + v2 << std::endl;//endl是一个特殊值，称为操纵符，具有输出换//行的效果

return 0;

}

习题1.3

#include <iostream>

int main()

{

std::cout << "Hello,world!!!" << std::endl;

return 0;

}

习题1.5

#include <iostream>

int main()

{

std::cout << "Enter two numbers:" << std::endl;

int v1, v2;

std::cin >> v1 >> v2;

std::cout << "The sum of ";

std::cout << v1;

std::cout << " and ";

std::cout << v2;

std::cout << " is ";

std::cout << v1 + v2;

/\*std::cout << "The sum of" << v1 << " and " << v2

<< " is " << v1 + v2 << std::endl;\*/

return 0;

}

1.4 控制结构

1.1.4 while

#include <iostream>

int main()

{

int sum = 0, val = 1;

while (val <= 10)

{

sum += val;

val++;//++val;

}

std::cout << "1+2+3+4+5+6+7+8+9+10=";

std::cout << sum << std::endl;

return 0;

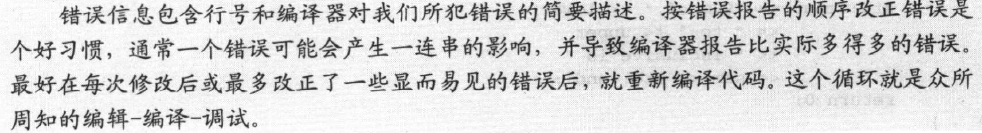
}

条件是一个可求值的表达式，所以可以测试其结果。如果结果值非零，那么条件为真；反之为假

1.4.2 for

(a=0;a<b;++a)

[再谈编译：编译器的部分工作时寻找程序代码中的错误。编译器不能查出程序的意义是否正确，但它可以查出程序形式上的错误。]



1.4.3 if

#include <iostream>

int main()

{

int a, b, up, low;

std::cout << "请输入两个值：>";

std::cin >> a >> b;

if (a > b)

{

up = a;

low = b;

}

else

{

up = b;

low = a;

}

int sum = 0;

for (int i = low; i <= up; ++i)

{

sum += i;

}

std::cout << "这两个值有小到大之后之间的和是" << sum << std::endl;

return 0;

}

1.4.4 读入未知数目的输入

对于不知道有多少个数要进行求和时：

#include <iostream>

int main()

{

int sum = 0, value;

//std::cout << "请输入你要求和的数";

while (std::cin >> value)

{

sum += value;

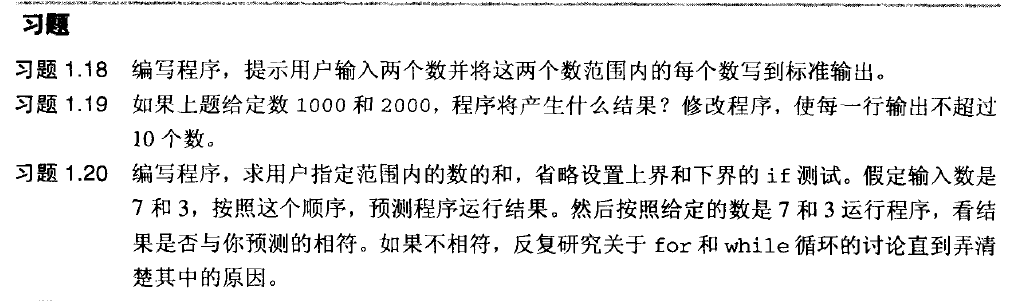
}

std::cout << "这些数的和是 " << sum << std::endl;

return 0;

}

//当我们要结束输入时，输入ctrl+z(文件结束符)



1.5 类的简介

C++中通过定义类来定义数据结构

使用类时我们需要回答三个问题：

1. 类的名字是什么
2. 它在哪里定义
3. 支持什么操作

1.5.1 Sales\_item类

Sales\_item类的目的是存储ISBN并保存该书的销售册数、销售收入和平均售价。

使用类时我们不需要知道这个类是怎样实现的，相反，我们需要知道的是该类提供什么操作。

1. Sales\_item对象上的操作

定义一个sales\_item类型的变量 Sales\_item item;

加法操作 +；

输入操作 >>;

输出操作 <<;

赋值操作 =将一个sales\_item对象赋值给另一个sales\_item对象；

调用same\_isbn函数确定两个sales\_item是否指同一本书

1. 读入与写出sales\_item对象

（<http://www.awprofessional.com/cpp_primer>）

Sales\_item.h

#ifndef SALESITEM\_H

#define SALESITEM\_H

// Definition of Sales\_item class and related functions goes here

#include <iostream>

#include <string>

class Sales\_item {

friend bool operator==(const Sales\_item&, const Sales\_item&);

// other members as before

public:

// added constructors to initialize from a string or an istream

Sales\_item(const std::string& book) :

isbn(book), units\_sold(0), revenue(0.0) { }

Sales\_item(std::istream& is) { is >> \*this; }

friend std::istream& operator>>(std::istream&, Sales\_item&);

friend std::ostream& operator<<(std::ostream&, const Sales\_item&);

public:

// operations on Sales\_item objects

// member binary operator: left-hand operand bound to implicit this pointer

Sales\_item& operator+=(const Sales\_item&);

// other members as before

public:

// operations on Sales\_item objects

double avg\_price() const;

bool same\_isbn(const Sales\_item& rhs) const

{

return isbn == rhs.isbn;

}

// default constructor needed to initialize members of built-in type

Sales\_item() : units\_sold(0), revenue(0.0) { }

// private members as before

private:

std::string isbn;

unsigned units\_sold;

double revenue;

};

// nonmember binary operator: must declare a parameter for each operand

Sales\_item operator+(const Sales\_item&, const Sales\_item&);

inline bool

operator==(const Sales\_item& lhs, const Sales\_item& rhs)

{

// must be made a friend of Sales\_item

return lhs.units\_sold == rhs.units\_sold &&

lhs.revenue == rhs.revenue &&

lhs.same\_isbn(rhs);

}

inline bool

operator!=(const Sales\_item& lhs, const Sales\_item& rhs)

{

return !(lhs == rhs); // != defined in terms of operator==

}

using std::istream; using std::ostream;

// assumes that both objects refer to the same isbn

inline

Sales\_item& Sales\_item::operator+=(const Sales\_item& rhs)

{

units\_sold += rhs.units\_sold;

revenue += rhs.revenue;

return \*this;

}

// assumes that both objects refer to the same isbn

inline

Sales\_item

operator+(const Sales\_item& lhs, const Sales\_item& rhs)

{

Sales\_item ret(lhs); // copy lhs into a local object that we'll return

ret += rhs; // add in the contents of rhs

return ret; // return ret by value

}

inline

istream&

operator>>(istream& in, Sales\_item& s)

{

double price;

in >> s.isbn >> s.units\_sold >> price;

// check that the inputs succeeded

if (in)

s.revenue = s.units\_sold \* price;

else

s = Sales\_item(); // input failed: reset object to default state

return in;

}

inline

ostream&

operator<<(ostream& out, const Sales\_item& s)

{

out << s.isbn << "\t" << s.units\_sold << "\t"

<< s.revenue << "\t" << s.avg\_price();

return out;

}

inline

double Sales\_item::avg\_price() const

{

if (units\_sold)

return revenue / units\_sold;

else

return 0;

}

#endif

1.5.2 初窥成员函数

#include <iostream>

#include "Sales\_item.h"

int main()

{

Sales\_item item1, item2;

std::cin >> item1 >> item2;

// first check that item1 and item2 represent the same book

if (item1.same\_isbn(item2)) {

std::cout << item1 + item2 << std::endl;

return 0; // indicate success

}

else {

std::cerr << "Data must refer to same ISBN"

<< std::endl;

return -1; // indicate failure

}

}

什么是成员函数

if (item1.same\_isbn(item2))

调用命名为item1的Sales\_item对象的成员函数。成员函数是由类定义的函数，有时称为类方法。