# 面向对象编程基础

本课程入选教育部产学合作协同育人项目

课程主页:http://cpp.njuer.org

课程老师:陈明 http://cv.mchen.org

ppt和代码下载地址

```
git clone https://gitee.com/cpp-njuer-org/book
```

# 第1章

# 开始

- 编写一个简单的C++程序
- 初识输入输出
- 注释简介
- 控制流
- 类介绍
- 书店程序

- *“学习一门新的程序设计语言的最好方法就是练习编写程序.”*

# 编写一个简单的C++程序

- 每个C++程序都必须包含一个或多个函数,其中一个名为main()的函数,它是操作系统执行程序的调用入口.

```
int main() {
    return 0;
}
```

# 一个函数定义包含四部分:

- 返回类型(return type):
  - main()的返回类型必须是int
- 函数名(function name):
  - 用来进行函数调用,这里的函数名是main
- 形参列表(parameter list):
  - 用()包围,指出调用函数时可以使用什么样的实参,允许为空
- 函数体(function body):
  - 用{}包围的语句块,定义了函数所执行的动作.这里语句块里只有一条语句return 0.

*注意不要漏掉return语句后的分号;*
*return 0 ;表示成功*

# 编译、运行程序

- 如何编译程序依赖于使用的操作系统和编译器.
- 集成开发环境（Integrated Development Environment, IDE）将编译器与其它程序创建和分析工具包装在一起. 如VS2022社区版,qt等等.
- 大部分编译器都会提供命令行界面,如g++.

# 编译、运行程序

- 程序源文件（source file）命名约定
  - 不同编译器使用不同的后缀名
  - .h .cpp .cc .cxx .cp .c
- 在linux下命令行运行编译器g++
  - 编译:`g++ test.cpp`
  - 编译:`g++ --std=c++11 test.cpp -o a.out`
  - 用 -Wall选项可对有问题的程序结构发出警告
  - 运行:`./a.out`
  - 查看运行状态:`echo $?`

# 熟悉g++编译器

输入 g++ -v,查看编译器版本:

```
Using built-in specs.
COLLECT_GCC=g++
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-linux-gnu/9/lto-wrapper
OFFLOAD_TARGET_NAMES=nvptx-none:hsa
OFFLOAD_TARGET_DEFAULT=1
Target: x86_64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu 9.3
Thread model: posix
gcc version 9.3.0 (Ubuntu 9.3.0-17ubuntu1~20.04)
```

# 熟悉g++编译器

输入 g++-10 -v,查看编译器版本:

```
Using built-in specs.
COLLECT_GCC=g++-10
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-linux-gnu/10/lto-wrapper
OFFLOAD_TARGET_NAMES=nvptx-none:amdgcn-amdhsa:hsa
OFFLOAD_TARGET_DEFAULT=1
Target: x86_64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu 10.
Thread model: posix
Supported LTO compression algorithms: zlib zstd
gcc version 10.3.0 (Ubuntu 10.3.0-1ubuntu1~20.04)
```

# 熟悉g++编译器

输入 g++ --help,查看编译器选项:

```
Usage: g++ [options] file...
Options:
  -pass-exit-codes         Exit with highest error code from a pha
  --help                   Display this information.
  --target-help            Display target specific command line op
  --help={common|optimizers|params|target|warnings|[^]{joined|sepa
                           Display specific types of command line
  (Use '-v --help' to display command line options of sub-processe
  --version                Display compiler version information.
  -dumpspecs               Display all of the built in spec string
  -dumpversion             Display the version of the compiler.
  -dumpmachine             Display the compiler's target processor
```

输入 g++ -v --help可以看到更完整的指令.

# 初识输入输出

- iostream库包含istream输入流和ostream输出流.一个流就是一个字符序列,随着时间的推移字符是顺序生成或者消耗的.
- 标准输入输出对象
  - 标准输入,名为cin 的istream类型对象
  - 标准输出,名为cout的ostream类型对象
- 另两个ostream对象
  - 标准错误,cerr,输出警告与错误
  - clog,输出程序运行的一般信息

# 一个使用IO库的例子

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

- `#include <iostream>`告诉编译器使用iostream库,一般包含来自标准库的头文件使用<>,不属于标准库的头文件用双引号" ".

- `endl`：操纵符，结束当前行，将与设备关联的缓冲区（buffer）中的内容刷到设备中；
- `std::`：cin、cout和endl所在`std`命名空间。作用运算符`::`.
  - 在头文件后加一行 `using namespace std;` 引入std名空间，其后的cin cout endl 就可不必加`std::`
  - 谷歌编程规范建议不要使用 `using namespace std;`,防止命名冲突。

- <<：输出运算符，将右侧的运算对象的值写到左侧运算对象中
  - 比如`cout << v1;`,左侧运算对象cout是一个ostream对象，并返回其左侧运算对象，因此<<可以连写，如 `cout<<v1<<v2;`

```
std::cout << "Enter two numbers:" << std::endl;
等价于
std::cout << "Enter two numbers:";
std::cout << std::endl;
```

- >> 输入运算符，从左侧istream读入数据，存入右侧对象。与输出运算符类似，其返回左侧运算对象，因此>>可以连写，如

```
std::cin >> v1 >> v2;
等价于
std::cin >> v1;
std::cin >> v2;
```

# 编译运行结果

```
$ g++ test.cpp
$ ./a.out
Enter two numbers:
3 7
The sum of 3 and 7 is 10
```

# 练习题

编写程序，在标准输出上打印Hello,world.

# 注释简介

- C++中注释的种类
  - 单行注释以”//"开始，以换行符结束
  - 界定符注释，即多行注释，继承自C语言，以/*开始，以*/结束。此注释方式不可嵌套.

```c
/*
*注释对/* */不能嵌套.
*不能嵌套几个字会被认为是源码，
*像剩余程序一样处理。
*/
int main() {
    // 这是单行注释

    /*
     * 这是多行注释
     * 多行注释不可嵌套
     * /

    // /*
    //  *单行注释中任何内容都会被忽略
    //  * 包括嵌套的注释对也会被忽略
    //  */
    return 0;
}
```

# 练习题

下面输出语句合法吗?

```
std::cout<<"/*";
std::cout<<"*/";
std::cout<</* "*/" */; //(error)
std::cout<</* "*/" /* "/*" */;
```

# 控制流

# WHILE语句

循环执行一段代码,直到给定的条件（condition）为假.

```
//求1到10这10个数的和
#include <iostream>
int main() {
    int sum = 0, val = 1;
    //只要val<=10，while循环就会持续执行
    while(val <= 10) {
        sum += val; // 复合运算符（+=），等价于“sum = sum + val”
        ++val;      // 前缀自增运算符（++），等价于“val = val + 1”
    }
    std::cout << "Sum of 1 to 10 inclusive is "
              << sum << std::endl;
    return 0;
}
```

编译运行结果：

```
$ g++ while.cpp
$ ./a.out
Sum of 1 to 10 inclusive is 55
```

- while 语句：
  - 重复执行语句块statement，直至condition为假

```
while (condition)
 statement;
```

# 练习题

编写程序，使用while循环将50到100整数相加

# FOR语句

for（init-statement; condition; expression)
statement;

- 由三部分组成:
  - 一个初始化语句（init-statement）
  - 一个循环条件（condition）
  - 一个表达式（expression）

```cpp
//用for语句重写从1加到10
#include <iostream>
int main(){
  int sum=0;
  //从1加到10
  for(int val=1;val<=10;++val){
    sum+=val;//等价于sum=sum+val;
  }
  std::cout <<"Sum of 1 to 10 inclusive is"
            <<sum<<std::endl;
  return 0;
}
// Sum of 1 to 10 inclusive is 55
```

# 练习题

编写程序，使用for循环将50到100整数相加

# 读取数量不定的输入数据

```
// 读取数量不定的输入数据
#include <iostream>
int main() {
    int sum = 0, val = 0;
    // 读取数据直到文件尾，计算所有读入值的和
    while(std::cin>>val) {
        sum += val; // 复合运算符（+=），等价于“sum = sum + val”
    }
    std::cout << "Sum is "
              << sum << std::endl;
    return 0;
}
```

- UNIX和Mac下键盘输入文件结束符:`ctrl+d`,
- Windows下:`ctrl+z` ENTER

# 练习题

编写程序，使用cin读一组数，输出其和.

# IF语句:根据CONDITION条件进行执行。如果条件为真，执行IF语句体；否则，执行ELSE语句体（如果存在的话）

```
if(condition)
    statement;
或者
if(condition)
    statement;
else
    statement;
或者
if(condition)
    statement;
else if (condition)
    statement;
...
else
    statement;
```

# 统计输入中每个值连续出现的次数

```
#include <iostream>
int main() {
    // currval: 保存正在统计的数; val: 保存将读入的新值
    int currval = 0, val = 0;
    if (std::cin >> currval) {
        // 保存当前处理值出现的次数
        int cnt = 1;
        while (std::cin >> val) {
            if (val == currval)     // 新读取的值与当前值相同，计数+1
                ++cnt;
            else
            {// 否则打印当前值次数，记住新值并重新开始计数
                std::cout << currval << " occurs "
                    << cnt << " times " << std::endl;
                currval = val;
                cnt = 1;
            }
        }//while
        //最后一个值的个数
        std::cout << currval << " occurs "
```

```
            << cnt << " times " << std::endl;
    }//if
    return 0;
}
```

# 类简介

- C++ 中我们通过定义一个类来定义自己的数据结构。
- 一个类
  - 定义了一个类型
  - 以及与其关联的一组操作。
- 为了使用类
  - 类名是什么
  - 在哪里定义
  - 支持什么操作

# Sales_item类

- 由Sales_item.h定义
- 封装了Sales_item的可用操作
  - isbn函数
  - 重载操作符 >> << 来读写 Sales_item对象
  - 重载操作符 = + +=

```
//Sales_item.h
// 在类和重载操作符的章节会解释，目前不需要理解该文件。
/* This file defines the Sales_item class used in chapter 1.
 * The code used in this file will be explained in
 * Chapter 7 (Classes) and Chapter 14 (Overloaded Operators)
 * Readers shouldn't try to understand the code in this file
 * until they have read those chapters.
*/

#ifndef SALESITEM_H
// we're here only if SALESITEM_H has not yet been defined
```

## 避免多次包含同一头文件:

```
#ifndef SALESITEM_H
#define SALESITEM_H
// Definition of Sales_item class and related functions goes here
#endif
```

定义类Sales_item的对象item

```
Sales_item item;
```

类Sales_item定义了对象item的行为

# 读写Sales_item

```cpp
//item_io.cpp
#include <iostream>
#include "Sales_item.h"
int main()
{
    Sales_item book;
    // read ISBN, number of copies sold, and sales price
    std::cin >> book;
    // write ISBN, number of copies sold, total revenue, and average price
    std::cout << book << std::endl;
    return 0;
}
```

```
$ g++ ./item_io.cpp && ./a.out
asdf 4 24.99
asdf 4 99.96 24.99
```

# Sales_item对象加法

```cpp
//addItems.cpp
#include <iostream>
#include "Sales_item.h"
int main() {
    Sales_item item1, item2;
    std::cin >> item1 >> item2;   //读取一对交易记录
    std::cout << item1 + item2 << std::endl; //打印它们的和
    return 0;
}
```

```
$ g++ ./addItems.cpp && ./a.out
asdf 4 44
asdf 5 50
asdf 9 426 47.3333
//使用文件重定向
$ echo "asdf 4 44 asdf 5 50" >> infile
$ ./a.out <infile >outfile && cat outfile
asdf 9 426 47.3333
```

# 初识成员函数

```cpp
//addItems2.cpp
#include <iostream>
#include "Sales_item.h"
int main()
{
    Sales_item item1, item2;
    std::cin >> item1 >> item2;
    // 检查isbn号是否相同
    if (item1.isbn() == item2.isbn()) {
        std::cout << item1 + item2 << std::endl;
        return 0;   // 表示成功
    } else {
        std::cerr << "Data must refer to same ISBN"
                  << std::endl;
        return -1;  // 表示失败
    }
}
```

```
$ g++ ./addItems2.cpp && ./a.out
asdf 12 22
asdf 6 20
asdf 18 384 21.3333

$ g++ ./addItems2.cpp && ./a.out
asdf 12 22
sdf 6 20
Data must refer to same ISBN
```

## 成员函数（类方法）：

- 定义为类的一部分的函数
- 使用 . 调用.

```
//名为item1的对象的isbn方法
item1.isbn()
```

# 书店程序

读取销售记录，生成销售报告：

- 假设每个isbn号的记录聚在一起保存

```cpp
//avg.cpp
#include <iostream>
#include "Sales_item.h"
int main() {
    Sales_item total; // variable to hold data for the next transaction
    // read the first transaction and ensure that there are data to process
    if (std::cin >> total) {
        Sales_item trans; // variable to hold the running sum
        // read and process the remaining transactions
        while (std::cin >> trans) {// if we're still processing the same bo
            if (total.isbn() == trans.isbn())
                total += trans; // update the running total
            else {// print results for the previous book
                std::cout << total << std::endl;
                total = trans;  // total now refers to the next book
        }}
        std::cout << total << std::endl; // print the last transaction
    } else {// no input! warn the user
        std::cerr << "No data?!" << std::endl;
        return -1;  // indicate failure
    }
```

```
    return 0;}
```

```
$ g++ ./avg.cpp && ./a.out
asdf 12 22
asdf 10 23
sdf  12 11
asdf 22 494 22.4545
sdf 12 132 11


$ echo "asdf 12 22 asdf 10 23 adf 12 11 " > infile &&
 ./a.out <infile >outfile &&cat outfile
asdf 22 494 22.4545
adf 12 132 11
```

# 练习

编译并运行书店程序.

# 实践课

- 本场景将使用一台配置了Aliyun Linux 2的ECS实例（云服务器）
  - 使用Vim编辑C++代码
  - 使用g++编译运行这段代码
  - 编辑一个 README.md 文档，键入本次实验心得。

```
- 云服务器（Elastic Compute Service，简称ECS）
- Aliyun Linux 2是阿里云推出的 Linux 发行版
- Vim是从vi发展出来的一个文本编辑器。
- g++ 是c++编译器
```

# C++代码

```cpp
#include <iostream>

int main() {
  std::cout << "!!!Hello World!!!" << endl; // prints !!!Hello World!!!
  return 0;
}
```

或

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

int main() {
  cout << "!!!Hello World!!!" << endl; // prints !!!Hello World!!!
  return 0;
}
```

# 附加题：

- 借助Sales_item.h头文件(slide内有提供），使用本机环境或阿里云提供的编程环境(IDE界面,图形用户界面，shell界面均可)，编译并运行最后一个书店程序avg.cpp

# 实践课

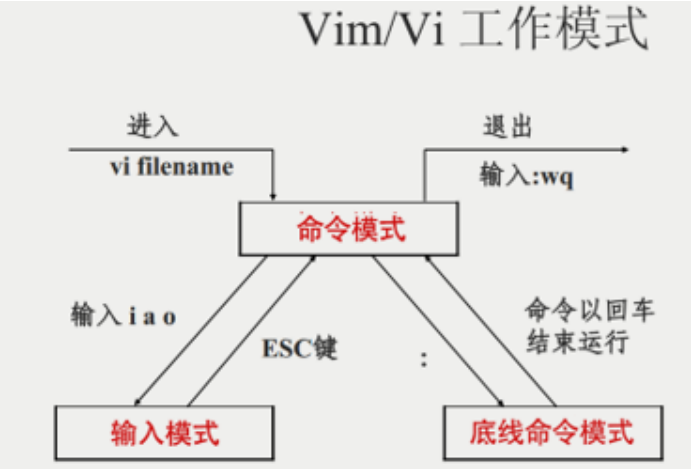
- 打开课程主页 cpp.njuer.org
- 点击本课程阿里云平台实验网址
- 选择实验—开始体验 （提示注册阿里云账号并登录）
    - 单击屏幕右侧创建资源（本次实验2小时）
    - 资源创建完毕后， 使用命令安装git g++ vim工具`yum install -y git gcc-c++ vim`
    - 在终端内使用VIM编辑一个C++程序，可参照vim说明帮助

```
使用g++编译，并执行，
编译命令 g++ 文件名
执行 ./a.out
```

# 提交

- 截图或复制文字，提交到群作业。
- 填写网页实验报告栏。并将报告链接填入 https://www.aliwork.com/o/cpphomework
- 本学期选做：填写问卷调查 https://rnk6jc.aliwork.com/o/cppinfo

# VIM 共分为三种模式




- 命令模式
  - 刚启动 vim，便进入了命令模式。其它模式下按ESC，可切换回命令模式
    - i 切换到输入模式，以输入字符。
    - x 删除当前光标所在处的字符。
    - : 切换到底线命令模式，可输入命令。
- 输入模式
  - 命令模式下按下i就进入了输入模式。
    - ESC，退出输入模式，切换到命令模式
- 底线命令模式

- 命令模式下按下`:`（英文冒号）就进入了底线命令模式。
  - wq 保存退出

# VIM 常用按键说明

除了 i, Esc, :wq 之外，其实 vim 还有非常多的按键可以使用。命令模式下：

- 光标移动
  - j下 k上 h左 l右
  - w前进一个词 b后退一个词
  - Ctrl+d 向下半屏  ctrl+u 向上半屏
  - G 移动到最后一行 gg 第一行 ngg 第n行
- 复制粘贴
  - dd 删一行 ndd 删n行
  - yy 复制一行 nyy复制n行
  - p将复制的数据粘贴在下一行 P粘贴到上一行
  - u恢复到前一个动作 ctrl+r重做上一个动作
- 搜索替换
  - /word 向下找word  ? word 向上找
  - n重复搜索 N反向搜索
  - :1,$s/word1/word2/g从第一行到最后一行寻找 word1 字符串，并将该字符串取代为 word2

# VIM 常用按键说明

底线命令模式下：

- `:set nu`　显示行号
- `:set nonu` 取消行号
- `:set paste`　粘贴代码不乱序

【注：把caps lock按键映射为ctrl，能提高编辑效率。】

# MARKDOWN 文档语法

```
# 一级标题
## 二级标题
*斜体* **粗体**
- 列表项
  - 子列表项
> 引用
[超链接](http://asdf.com)
![图片名](http://asdf.com/a.jpg)

|表格标题1|表格标题2|
|-|-|
|内容1|内容2|
```

# 谢谢