USTC

Chapters 1 & 2
Welcome to Programming world

王子磊 (Zilei Wang)
Email: zlwang@ustc.edu.cn
http://vim.ustc.edu.cn

Why programming?

- 现代人类的文明离不开软件
  - 软件在几乎所有的工程中发挥着重要作用
  - 事实上，大部分程序并非运行在常见PC平台上
- 编程是一种工具，它是科学和工程中计算机和信息领域的实现技术
  - 不是为了编程而编程
- 编程是思想和理论的实际汇合
  - 不是仅仅学习一种或多种编程语言
  - 不是仅仅编写一些代码

Ships

- 设计 Design
- 建造 Construction
- 管理 Management

- 监控 Monitoring
- 引擎 Engine
- 船身设计 Hull Design
- 水泵 Pumps

Aircraft

- 通信 Communication
- 控制 Control
- 显示 Display

- 信号处理 Signal Processing
- 机械控制 “Gadget” Control
- 监控 Monitoring

Phones

- 声音质量 Voice Quality
- 用户接口 User Interfaces
- 记账 Billing
- 移动处理 Mobility
- 交换 Switching
- 可靠性 Reliability
- 供给 Provisioning
- 图像 Images

Energy

- 控制 Control
- 监控 Monitoring
- 分析 Analysis
- 设计 Design

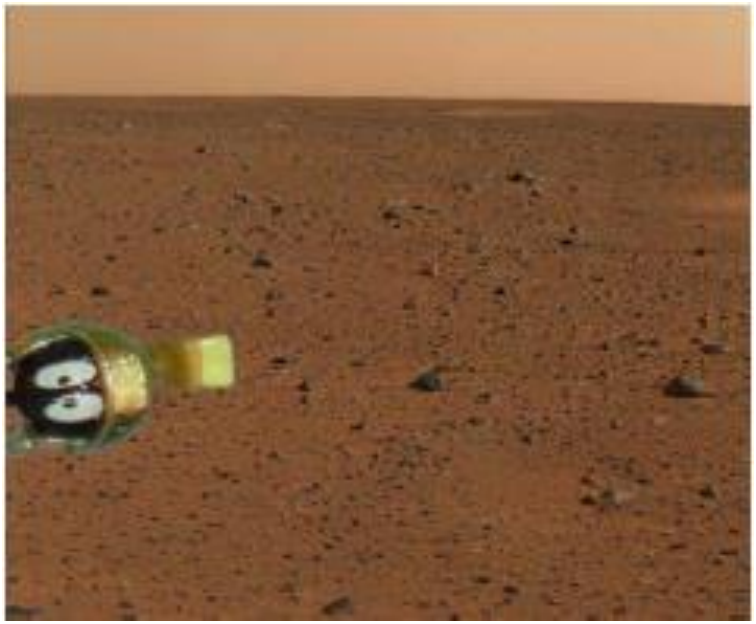
- 通信 Communications
- 可视化 Visualization
- 制造 Manufacturing

PC/workstation

- 除游戏、文档处理、上网等功能外，还有许多的应用
  - 我们最熟悉的场景

Where is C++ used?

- 可能的任何地方



火星探测、动漫、图像、Photoshop、GUI、操作系统、编译器、幻灯片、视频、芯片设计、芯片加工、娱乐终端、半导体工具，等等

参见：http://www.research.att/~bs/applications.html

A first program – just the guts...

```
// ...

int main()                              // main() is where a C++ program starts
                                        //      with no arguments
{
    cout << "Hello, world!\n";          // output the 13 characters  Hello, world!
                                        // followed by a new line
    return 0;                           // return a int value indicating success
}
```

- main具有标准函数结构：返回值、函数名、参数列表、函数体
- 双引号(非单引号)表达字符串常量(注意是""，而不是“”)，编译器不支持中文符号，会报错，但注释除外(因为编译器会忽略它)
- cout (see-out)是标准输出流(输入流为cin)，注意：<<

A first program – complete

```
// a first program:

#include "../../std_lib_facilities.h" // get the library facilities needed for now

int main()                            // main() is where a C++ program starts
{
    cout << "Hello, world!\n";        // output the 13 characters  Hello, world!
                                      // followed by a new line
    return 0;                         // return a value indicating success
}
```

- 分号表示一个语句的终止，注意宏指令不是语句
- 此处的大括号表示一个程序块，约定一个范围(range)

A second program

```
// modified for Windows console mode:

#include "../../std_lib_facilities.h" // get the facilities for this course

int main()                          // main() is where a C++ program starts
{
    cout << "Hello, world\n";       // output the 13 characters  hello, world!
                                    // followed by a new line
    keep_window_open();             // wait for a keystroke
    return 0;                       // return a value indicating success
}

// without keep_window_open() the output window will be closed immediately
// before you have a chance to read the output (on Visual C++ 2003)
```

- keep_window_open()在std_lib_facilities.h中定义

为什么需要注释？

- 程序有两个读者
  - 计算机(编译器)——翻译成机器语言
  - 程序员是主要读者——可读性
    - 代码是人与人之间交流的一种有效方式！
- 注释是给程序员看的，被编译器忽略，其形式有
  - 单行注释(推荐) //.....
  - 块注释(可以单行和多行) /*.....*/
- 注释是多级的
  - 文件注释、类注释、函数注释、语句注释、参数注释等

Why “Hello, world!”？

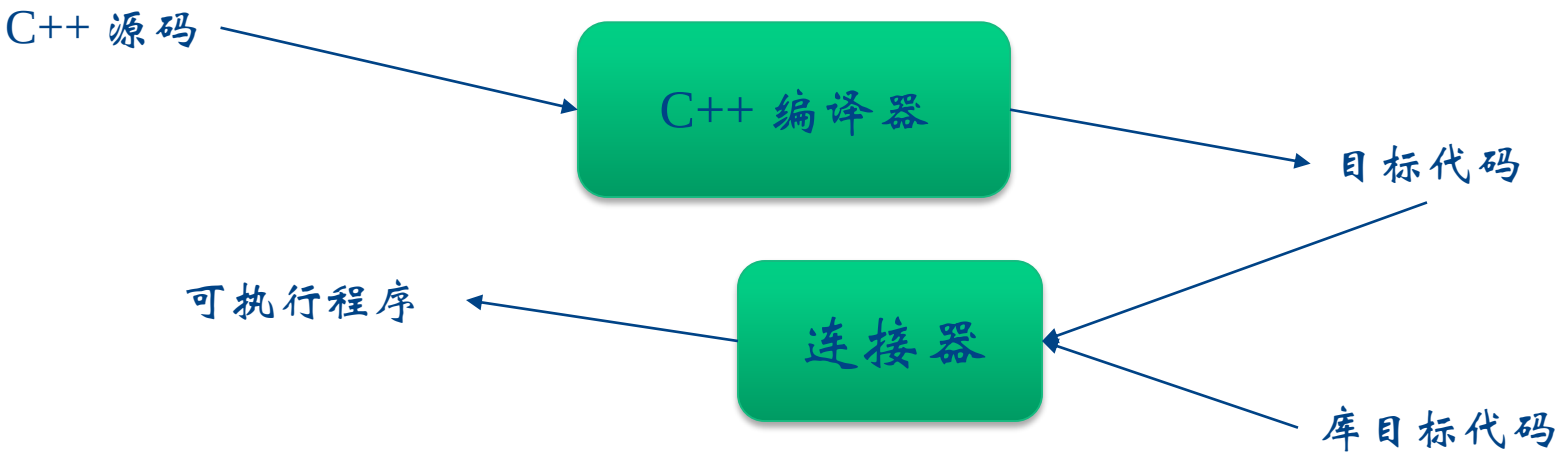
- “Hello, world!” 永远是最重要的程序
  - 它能够帮助熟悉相关编程工具
    - 编译器 Compiler
    - 程序开发环境 Program development environment
    - 程序执行环境 Program execution environment
  - 小心输入程序代码
    - 编译器是严格不留情面的，容不得错误语法和失误
    - 当”Hello world!”正常工作后，尝试做些修改，看看编译器的反应，如：
      - 忘记包含头文件 (// #include …)
      - 字符串忘记终止 (“Hello, would!)
      - 拼写错误 **return** (e.g. **retrun**)
      - 忘记分号终止符
      - 忘记大括号{ or }

练习：参考教材第28页，并在自己搭建的环境中进行测试!

“Hello world”的启示

- 它是最简单和普通的
  - 只有 cout << “Hello, world!\n” 是真正工作的
  - 程序主要采用的是已有系统代码，使其更简洁高效
- Why so simple, comprehensible, and efficient?
  - 采用标准的代码形式、注释、程序库
  - 无需在底层撰写复杂的代码 (machine code)
- 启示：编程不仅仅是使其工作结果正确，还要强调优美
  - 正确性——计算机
  - 高效性——计算机
  - 可维护性——程序员
- 将上述要求作用于编写代码 → Style Matters!

编译和链接



- C++ 源码由程序员编写
  - 原则上要求具有较高的可读性
- 编译器将源代码翻译成目标代码(也称作机器代码)
  - 机器代码是计算机可理解的
- 链接器将编写目标代码和系统目标代码链接为可执行程序
  - 系统代码包括输入输出库、操作系统代码、窗口代码等，也可以是其它中间件和编程工具
  - 可执行程序是windows上的 **.exe** 文件或Unix上的**.out** 文件

Notes

- 什么是库？
  - 形式上：代码集合，通过#include 声明来引用
  - 设计上：基础设施，提供工具
- 移植性
  - 标准C++源码可以运行在不同的平台上
  - 目标代码和可执行程序不能在系统之间移植
    - 为什么 .dll 库和 .so 库不能互用?
- 程序员教条
  - 编译时错误比链接时错误更容易理解和修正！
  - 链接时错误比运行时错误更容易发现和修正！

关于编程环境

- 集成开发环境 IDE
  - 如Microsoft的Visual C++ (Appendix C)
  - 集成编辑器、编译器、链接器、调试器等
- 替代的
  - 命令行形式
  - 采用每个独立工具进行操作，如linux下
    - vim ——编辑
    - g++ (makefile)——编译、链接
    - gdb——调试

What is programming?

- 编程的传统定义
  - 告诉傻瓜机器如何做一件事
  - 计算机上解决问题的方法
  - 指定程序的执行顺序
    - 以数据操作为中心
- 一种严密的学术定义
  - A … program is an organized and directed accumulation of resources to accomplish specific … objectives …
  - 规范，但没有指明具体的特点(太笼统)

Our definition

- 定义
  - Specifying the structure and behavior of a program, and testing that the program performs its task correctly and with acceptable performance
    - structure需要设计——优美
    - behavior代表功能——正确
    - Performance代表性能——可接受(注意：不是越高越好)
  - 再次强调：三者有机结合，缺一不可！

- 什么是软件
  - 一个或多个程序构成的集合

编程的简单与复杂

- 定义上看，编程是简单的
  - 只要告诉机器你打算“做什么”
- 为什么编程又那么难呢？
  - 我们希望“机器”做那些难以完成的事
    - 但计算机是挑剔的、无情的和不会说话的
  - 世界比我们想象的复杂
    - 我们并不理解自己需求的实际含义
  - 编程就是理解
    - 当你需要编写一个任务时，你需要理解它
    - 编程中，你会花费很多时间尝试理解你试图自动化的任务
  - 编程是半理论半实践的
    - 只注重实践，会编写出不可扩展的糟糕代码
    - 只重视理论，只能编写不实用的toys

程序开发的四个阶段

- 软件工程过程
  - 分析
    - 问题是什么？用户想要什么？用户可以负担什么？我们需要哪种可靠性？
  - 设计
    - 如何解决问题？系统的整体结构将是什么？系统包括哪些部分？这些部分之间如何通信？用户接口？
  - 编程
    - 用代码表达问题求解的方法，以满足所有约束(时间、成本、可靠性等)的方式编码，并保证代码的正确性和可维护性
  - 测试(单元、集成、系统)
    - 系统化地进行各种尝试，保证这些代码的正确和可维护性

Next …

- 进入编程的细节
  - 类型 types
  - 值 values
  - 变量 variables
  - 声明 declarations
  - 简单的输入输出 simple input and output
  - 类型安全 type safety
- 强调概念的理解