01 初识C语言

内容提要

- C的历史和特性
- 编写程序所需的步骤
- 关于编译器和链接器的一些知识
- C的标准

1 C语言的起源

- 1972 年，贝尔实验室丹尼斯·里奇（Dennis Ritch）和肯·汤普逊(Ken Thompson）在开发UNIX 操作系统时设计了C语言
  - 在B语言（汤普逊发明）的基础上设计
- C 语言设计的初衷：成为程序员使用的一种编程工具

2 使用C语言的理由

2 使用C语言的理由

- TIOBE开发语言排行榜
  - 每月更新一次，依据的指数是基于世界范围内的资深软件工程师和第三方供应商提供，其结果作为当前业内程序开发语言的流行使用程度的有效指标。
- https://www.tiobe.com/tiobe-index/

使用C语言的理由

- 设计特性
  - 自顶向下的规划，结构化编程，模块化设计高效性：紧凑且运行速度快
- 高效性
  - C 程序相对更紧凑，而且运行速度很快。可以根据具体情况微调程序以获得最大运行速度或最有效地使用内存
- 可移植性
  - 经过很少改动或不经修改就可以其他系统上运行
- 强大而灵活
  - 功能强大，可以解决很多问题，且灵活，使用多种场景

使用C语言的理由

- 面向编程人员
  - 为了满足程序员的需求而设计
  - 允许访问硬件，并可以操纵内存中的特定位。具有丰富的运算符供选择
  - 有丰富的运算符，能让程序员简洁地表达自己的意图
- C语言的缺点
  - 自由的代价是永远的警惕
  - C指针的跟踪
  - 存在极难理解的代码
  - 语言紧凑简洁，结合了大量的运算符，可以编写出让人极其费解的代码

3 C语言的应用范围

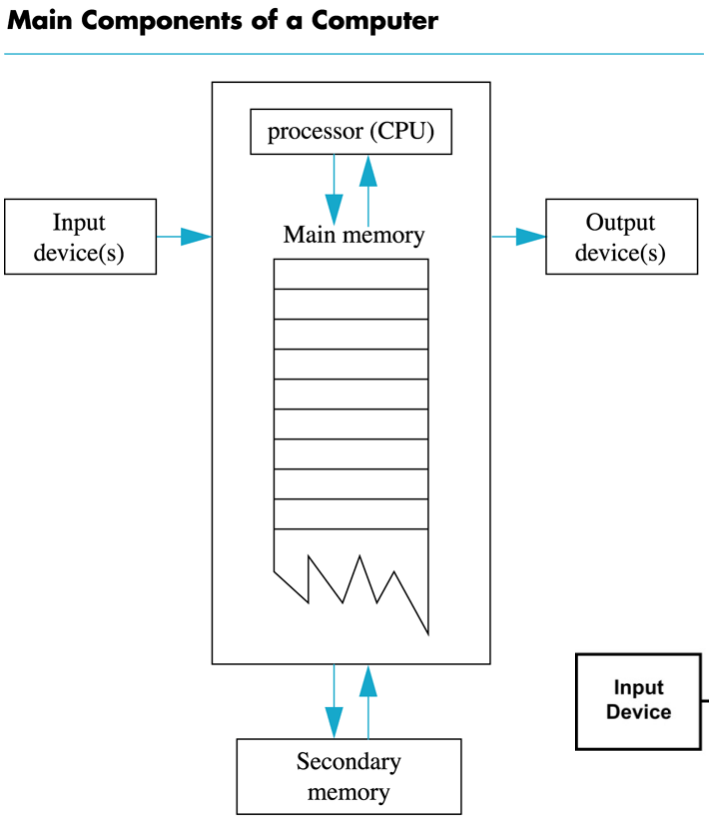
C语言的应用范围

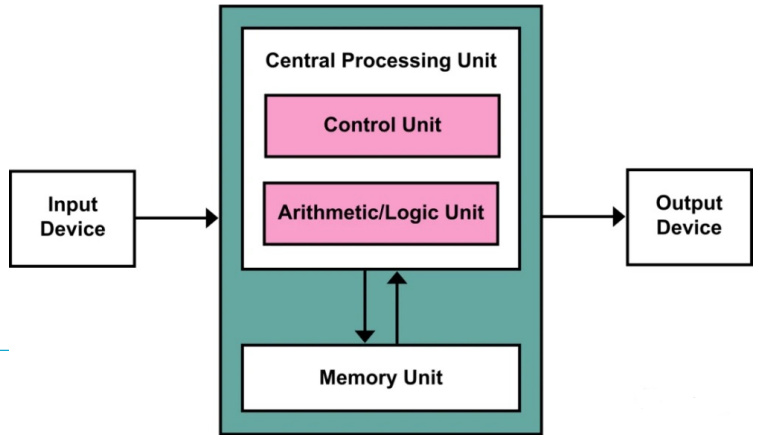
- C 语言编写的程序紧凑而高效
- C 程序很方便修改，而且移植到新型号的计算机中也没什么问题
- 20 世纪90 年代，许多软件公司开始改用C++来开发大型的编程项目。C++在C 语言的基础上嫁接了面向对象编程工具
  - （ 面向对象编程是一门哲学，它通过对语言建模来适应问题，而不是对问题建模以适应语言）。C++几乎是C 的超集，这意味着任何C 程序差不多就是一个C++程序。学习C 语言，也相当于学习了许多C++的知识
- 虽然这些年来C++和JAVA 非常流行，但是C 语言仍是软件业中的核心技能

4 计算机能做什么

4 计算机工作的基本原理

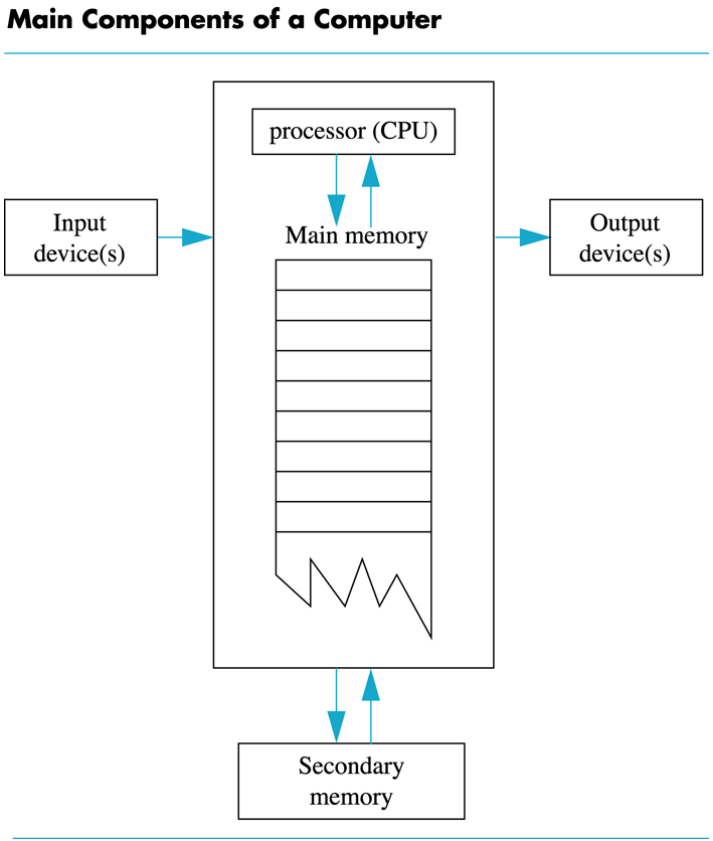
- 5个主要部件
  - 输入设备Input devices
    - 计算机接受信息的设备
  - 输出设备Output devices
    - 计算机输出信息的设备
  - 处理器Processor
    - 中央处理器（CPU）
  - 主存储器Main memory
    - 运行的程序存放于主存储器
  - 辅助存储器Secondary memory
    - 持久性地保存数据





计算机内存Computer Memory

- 主存Main Memory
  - 可以用于存储数据的存储单元（位置）列表带有位置编号
    - 每个存储单元都包含0或1
    - 这些位置的内容可以修改
  - 二进制位Binary Digit，位，比特Bit
    - 每个数位仅仅能存放0或者1
  - 字节Byte（存储一般以字节为单位的）
    - 每8位统称为1字节
  - 地址Address
    - 标识主存位置的数字/编号



内存地址

- 很多数据，一个字节容不下
  - 比如整数和实数
  - 需要多个字节存储
- 地址，指的是内存位置的第一个字节的地址

数据和代码Data or Code?

- ‘A’ : 01000001
- 65 : 01000001
- 一条指令：01000001

- 计算机怎么知道01000001的意思?
  - 取决于当前的指令

- 程序员很少需要去关心这个问题
  - 因为不是直接去操作这些二进制位

辅助存储器Secondary Memory

- 持久保持数据的设备
- 主存上的数据，掉电就丢失

辅助存储器

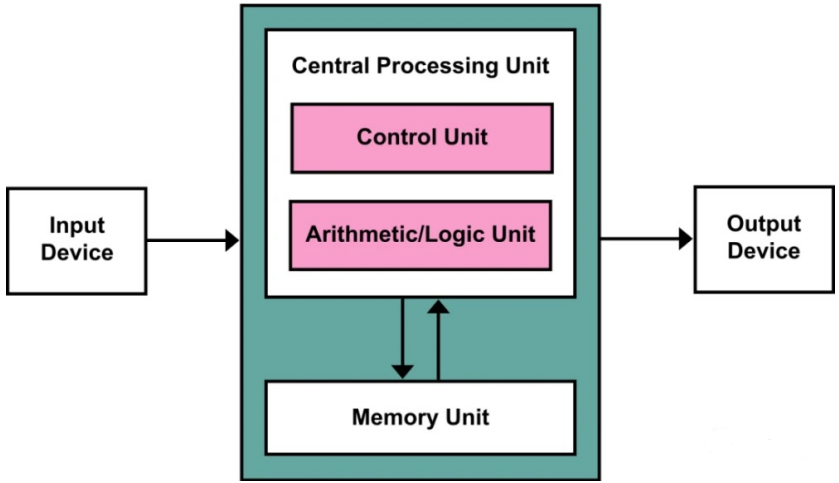
- 常见的辅助存储设备
  - 硬盘Hard disk
    - Fast
    - Fixed in theComputer and not normally removed
  - 软盘Floppy disk（现在比较少）
    - Slow
    - Easily shared with otherComputers
  - 光盘Compact disk
    - Slower than hard disks
    - Easily shared with otherComputers
    - Can be read only or re-writable

内存访问Memory Access

- 随机访问Random Access
  - 又称为 RAM
    - Random Access Memory,随机存取存储器
    - 计算机能直接访问任意位置内存
- 顺序访问Sequential Access
  - 数据存放有顺序，访问必须从头开始
    - 比如，磁带

处理器The Processor

- CPU, Central Processing Unit, 中央处理器
  - 按照程序的指令操作
  - 典型的CPU指令包括
    - add +
    - subtract -
    - multiply ×
    - divide /
    - move 移动数据

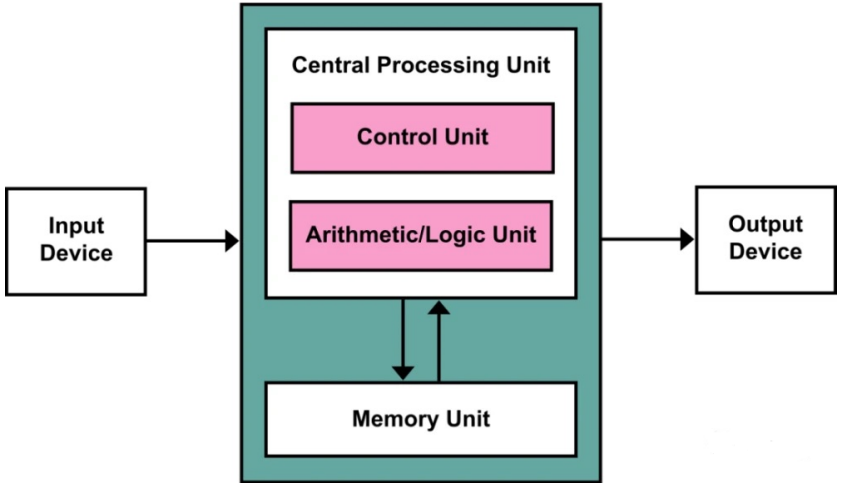


计算机程序（Computer Program）

- 操作系统（OS，Operating System）
  - 用户通过OS和计算机进行交互
  - 是程序
  - 分配和管理计算机资源
    - 内存，CPU
  - 文件
  - …
- 常见的OS…
  - UNIX　　Linux　　DOS
    Windows　　Macintosh　　VMS

计算机输入Computer Input

- 包含
  - 程序
  - 一些数据（Data）
    - 向程序提供的输入
    - 计算机可用的任何信息



5 高级计算机语言和编译器

高级语言High-level Languages

- 高级编程语言（如C）以多种方式简化了编程工作
  - 首先，不必用数字码表示指令
  - 其次，使用的指令更贴近你如何想这个问题， 而不是类似计算机那样繁琐的步骤。使用高级编程语言，可以在更抽象的层面表达你的想法，不用考虑CPU 在完成任务时具体需要哪些步骤
- 常见
  - C, C++, Java, Pascal, Visual Basic, FORTRAN, COBOL, Lisp, Scheme, Ada
  - 特点
    - 类似于人类语言
    - 易于读写
    - 指令复杂，远超过CPU直接执行的单条指令
    - 要转化成CPU 能执行的机器指令

低级语言Low-level Languages

- 汇编语言assembly language可能为
  - ADD X Y Z
  - x + y -> Z
- 汇编语言必须转换成机器语言machine language（0,1组合）
  - 0110 1001 1010 1011
- 机器语言可以被CPU直接执行

编译器Compilers

- 编译器：把高级语言程序翻译成计算机能理解的机器语言指令集的程序
  - 程序员进行高级思维活动，编译器则负责处理冗长乏味的细节工作
- 源代码SourceCode
  - 原始的高级语言编写的程序
- 目标代码ObjectCode
  - 转换好的机器语言形式的程序版本

链接器Linkers

- 有些程序已经编译好了
  - 目标码可用
  - 如： 输入输出函数
- 链接器将
  - 用户程序生成的目标码
  - 预先编译好的例程目标码（库函数等，可能别的语言编译过来）
- 合并，生成
  - CPU能运行的机器码

6 语言标准

6 语言标准

- 1972年，贝尔实验室的丹尼斯·里奇(Dennis Ritch)和肯·汤普逊(Ken Thompson)
  - 在开发UNIX 操作系统时设计了C语言
- ANSI C/ ISO C
  - K&R C，1987，C 标准
- C89/C90
  - (ANSI C) 定义了C 语言和C 标准库
- C99
  - 国际化、弥补缺陷和提高计算的实用性
- C11
- C18
  - C语言的现行标准

7 使用C语言的7个步骤

7 使用C语言的7个步骤

- 1．定义程序目标
- 2．设计程序
- 3．编写代码
- 4．编译
- 5．运行程序
- 6．测试和调试程序
- 7．维护和修改程序

第1步： 定义程序目标

- 定义出要解决的问题
  - 想要程序去做什么，首先自己要明确自己想做什么
  - 考虑程序需要的信息，程序需要进行的计算和操作，以及程序应该要报告的信息
    - 用一般概念来描述问题，而不涉及具体的计算机语言

- 对程序应该完成什么任务有概念性的认识

第2步： 设计程序

- 对程序应该完成什么任务有概念性的认识后，考虑如何用程序来完成它
  - 用户界面应该是怎么样的，如何组织程序，目标用户是谁，有多长时间来完成这个程序？
- 在程序中如何表示数据，用什么方法来处理数据
  - 数据+数据处理
- 用一般的概念来考虑问题，二不是具体的代码
  - 某些决策可能要取决于语言的一般特征

第3步： 编写代码

- 把设计的程序翻译成C语言
- 使用文本编辑器创建源代码文件。该文件中内容就是C语言代码
- 程序清单1.1

```
#include <stdio.h>
int main(void)
{
    int dogs;

    printf("How many dogs do you have?\n");
    scanf("%d", &dogs);
    printf("So you have %d dog(s)!\n", dogs);

    return 0;
}
```

第4步：编译

- 编译器是把源代码转换成可执行代码的程序
- 可执行代码是用计算机的机器语言表示的代码。这种语言由数字码表示的指令组成

- C 编译器负责把C 代码翻译成特定的机器语言
- C 编译器还将源代码与C 库（库中包含大量的标准函数供用户使用， 如printf ()和scanf()）的代码合并成最终的程序（更精确地说， 应该是由一个被称为**链接器**的程序来链接库函数）
  - 其结果是，生成一个用户可以运行的可执行文件，其中包含着计算机能理解的代码。

第5步：运行程序

- 可执行文件是可运行的程序
- 在常见环境中运行程序要输入可执行文件的文件名
  - 包括Windows 命令提示符模式、UNIX 终端模式和Linux 终端模式

第6步： 测试和调试程序

- 检查程序是否按照所设计的思路运行
- 调试（Debugging）：发现并修正程序错误
- 现在的编译器会捕获许多错误，而且自己也可以找到编译器未发现的错误

第7步： 维护和修改程序

- 发现程序有错，或者想扩展程序的用途，需要修改程序
  - 如果在编写程序时清楚地做了注释并采用了合理的设计方案，这些事情都很简单

总结

- 编程并非是一个线性的过程，要在不同的步骤之间往复
- 对程序做文字注释为今后的修改提供了方便

- 养成在编写代码前先进行规划的习惯！
  - 1，2步很重要

- 磨刀不误砍柴工，应该养成先规划再动手编写代码的好习惯用纸和笔记录下程序的目标和设计框架

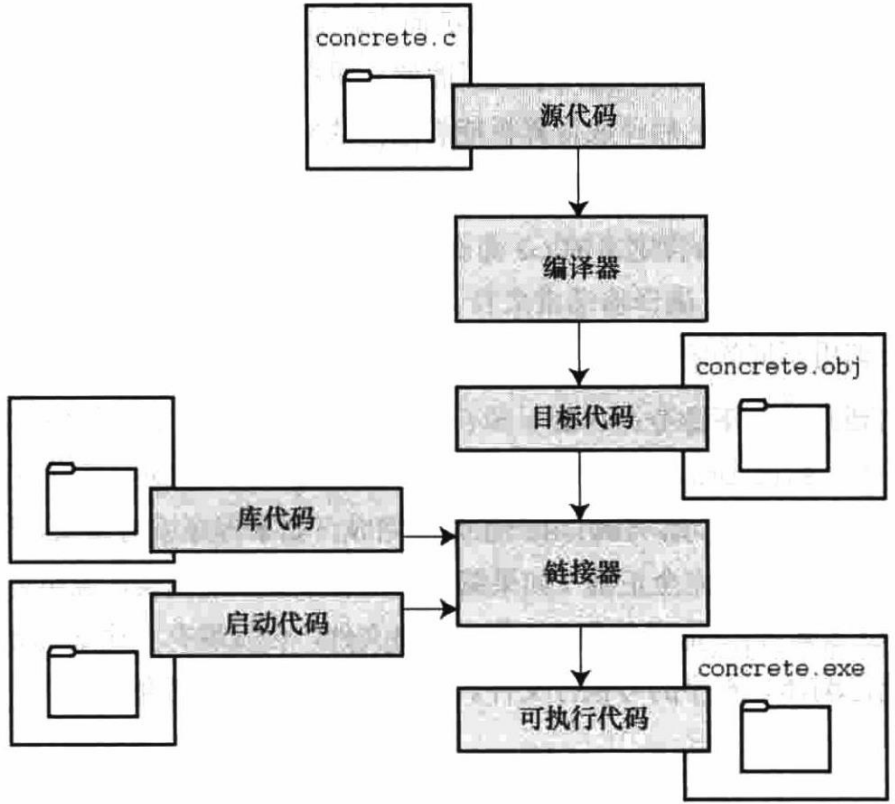
8 编程机制

编程机制

- 源代码文件(source code file)：程序的内容
  - 文件名以.c 结尾（如，budget.c）
  - 在文件名中，点号（ . ）前面的部分称为基本名(basename)，点号后面的部分称为扩展名(extension)
- 程序清单1.2

1 目标代码文件、可执行文件和库

- 实现通过编译和链接两个步骤来完成源代码文件转换为可执行文件过程
  - 编译器：把源代码转换成中间代码
  - 链接器：把中间代码和其他代码合并，生成可执行文件
- C使用这种分而治之的方法
  - 对程序进行模块化，可以独立编译单独的模块，稍后再用链接器合并已编译的模块
- 中间文件
  - 普遍的一种为目标代码文件
- 目标文件包含机器语言代码
  - 缺失启动代码(startup code)，不能运行
  - 启动代码充当着程序和操作系统之间的接口



Linux 系统 & Window

- gcc inform.c
- gcc -std=c11 inform.c
- http://www.gnu.org/software/gcc/index.html