C语言程序设计基础

林川

为什么要学C语言？

作为程序员的你，应该熟悉运用三类语言：

- 解释型
  - shell，Python
- 面向对象
  - Java
- 面向过程
  - C

- 培养你的程序思维
- 熟悉程序员工作流程
- 掌握求解问题的思路和方法

如何学习C语言

- 做好预习
- 不懂就问
- 认真完成作业
- LeetCode刷题

学些什么？

数据类型

分支，循环

结构体，指针

函数

结构体

文件

网络

第一章 引 言

- 一个C语言程序
- 程序与程序设计语言
- C语言的发展历史与特点
- 实现问题求解的过程

一个C语言程序

```
#include <stdio.h>                  /* 编译预处理命令 */
int main(void)                      /* 主函数 */
{                                   /* 函数体开始*/
      int n;                        /* 定义变量*/
      int factorial(int n);         /* 声明函数*/
      scanf("%d", &n);              /* 输入一个整数 */
      printf("%d\n", factorial(n));  /* 计算，并输出*/
      return 0;                 /* 返回语句 */
 }                                  /* 函数体结束*/
```

一个C语言程序--续

```
/* 计算 n! 的函数 */
int factorial( int n )                  /* 函数头 */
{                                       /* 函数体开始*/
     int i;                             /* 定义变量 i */
     int fact = 1;                     /* 定义变量 fact */
     for(i = 1; i <= n; i++)            /* 循环 */
           fact = fact * i;             /* 乘法 */
      return fact;                      /* 返回结果 */
 }                                      /* 函数体结束*/
```

程序构成

- 有一些函数
  - main, scanf, printf, factorial
  - 从主函数main开始执行，依次执行
  - 可以输入、输出过程
- 有一些变量
  - i, fact
- 流程控制
  - for

1.2 程序与程序设计语言

- 程序：一系列加工步骤
  - 由计算机可以识别的代码编排而成
  - 指示计算机对数据进行处理
  - 解决实际问题
- 程序设计语言
  - 提供了一种表达数据与处理数据的功能
  - 要求程序员按照语言的规范编程

程序与指令

- 可执行程序
  - 一系列计算机指令的有序组合
- 指令
  - 执行一个最基本的功能
    - 算术运算：加减乘除，比较大小等等
    - 输入输出，控制指令等等
- 指令系统
  - 计算机所能实现的指令集合
  - 不同的计算机有不同指令系统

利用指令编写程序

- 繁琐、效率低下
- 可读性差、不宜维护
- 指令系统相关、难以移植

需要 -- 高级程序设计语言

程序设计语言的功能

- 数据结构：表达所要处理的数据
- 算法：表达数据处理的流程

其他的辅助功能：
优化代码、重用代码，等等

数据结构

- 数据表达：一般将数据抽象为若干类型
- 数据类型：对某些具有共同特点的数据集合的称呼
  - 数据本身的定义
  - 数据可进行的操作和运算

  例如：整数类型
    - 定义：{…，-2，-1，0，1，2，…}
    - 运算：+、-、*、/、%等

数据表达

- C语言提供的数据类型
  - 基本数据类型：整型、浮点型、字符型等。
  - 构造类型：用户定义的，如数组、结构等等。
- 各种数据类型的常量与变量形式
  - 常量（常数）与变量

C语言基本的数据类型

- int（整型）、float(浮点型)、double(双精度)、char（字符型）

变量

```
int k;      定义了一个整数变量k
float x;    定义了一个浮点数变量x
double y;   定义了一个双精度浮点数变量y
char c;     定义了一个字符型变量c
```

常量

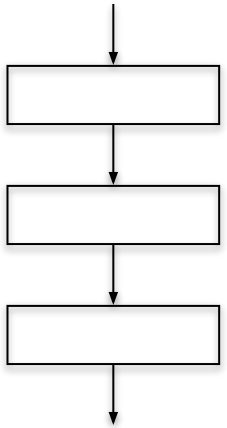
```
'c'         表示一个字符c
100         表示整数100
12.56       表示一个实数12.56
```

流程控制

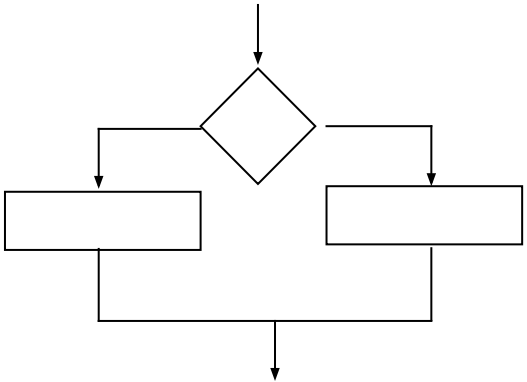
- 结构化程序设计方法
  - 将复杂程序划分为若干个相互独立的模块
  - 模块:若干语句构成的一段程序或一个函数(子程序)等
  - 单入口、单出口

流程控制

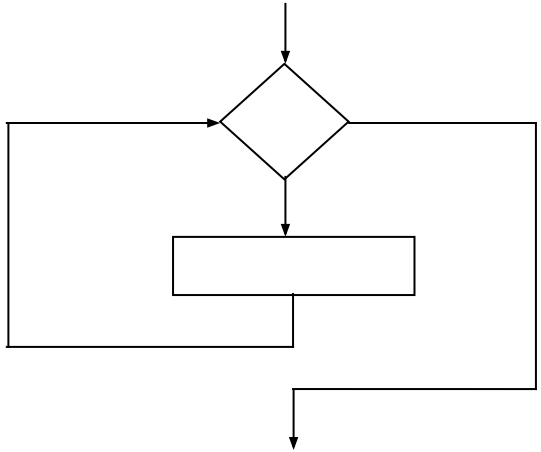
- 3种基本流程控制结构
  - 顺序结构、分支结构、循环结构



顺序结构



分支结构



循环结构

程序设计语言的语法

- 程序必须符合语言的语法规则
- 程序代码由一系列“单词”，按照语法规则组合而成
- 不同的程序设计语言具有不同的语法
- C语言的语法要素
  - 单词：标识符、常量、运算符、分隔符
  - 语法单位：表达式、变量定义、语句、函数定义、函数调用、（输入输出）

C语言的单词

- 标识符：由字母、数字以及下划线组成，且第一个字符必须是字母或下划线
- 保留字(关键字)： C语言规定的、赋予它们以特定含义、有专门用途的标识符
- 自定义标识符：程序中定义的变量名、数据类型名、函数名以及符号常量名
- 常量：常量是有数据类型的，如，整数常量123，浮点数常量12.34
- 运算符：代表对各种数据类型实际数据对象的运算。如，+（加）、-（减）、*（乘）、/（除）、%（求余）、>（大于）、<（小于）等等

C语言的语法单位

- 表达式：运算符与运算对象组合就形成了表达式。如：2 + 3 * 4

- 变量定义：变量也有数据类型，所以在定义变量时要说明相应变量的类型。如：int i;

- 语句：语句是程序最基本的执行单位，程序功能通过执行一系列语句实现。

C语言的语句

- 表达式语句：表达式加分号“;”
- 分支语句：实现分支控制过程

```
if (a > b)  x = a;
else   x = b;
```

- 循环语句：实现循环控制的过程

```
while (i <= 100)
{
    sum = sum + i;
    i = i + 1;
}
```

- 复合语句：用一对“{”和“}”，将若干语句顺序组合在一起就形成了一个复合语句。

C语言的语法单位

- 函数定义: 完成特定任务的独立模块

```
int max( int a, int b )
{
        int x;
        if( a>b) x = a;
        else x = b;
        return x;
}
```

- 函数调用

```
m = max(k, 3);
```

程序的编译与编程环境

- 编译
  - 程序 编译器 可执行代码
  - 可执行代码：计算机能直接理解的指令序列
  - 编译器：对源程序进行词法分析、语法分析、生成可执行的代码、返回编译结果和错误信息

- 编程环境
  - 编辑（Edit）
  - 编译（Compile）
  - 调试（Debug）

C语言的特点

- 一种结构化语言
- 语句简洁、紧凑，使用方便、灵活
- 易于移植：不包含与硬件有关的因素
- 有强大的处理能力
- 目标代码运行效率高
- 数据类型检查不严格
- 区分大小写

1.4 编写程序求解问题示例

求1～100之间所有偶数的和

- 问题分析：
  - 求和
  - 范围
    - 从1到100之间
    - 偶数

1.4 编写程序求解问题示例

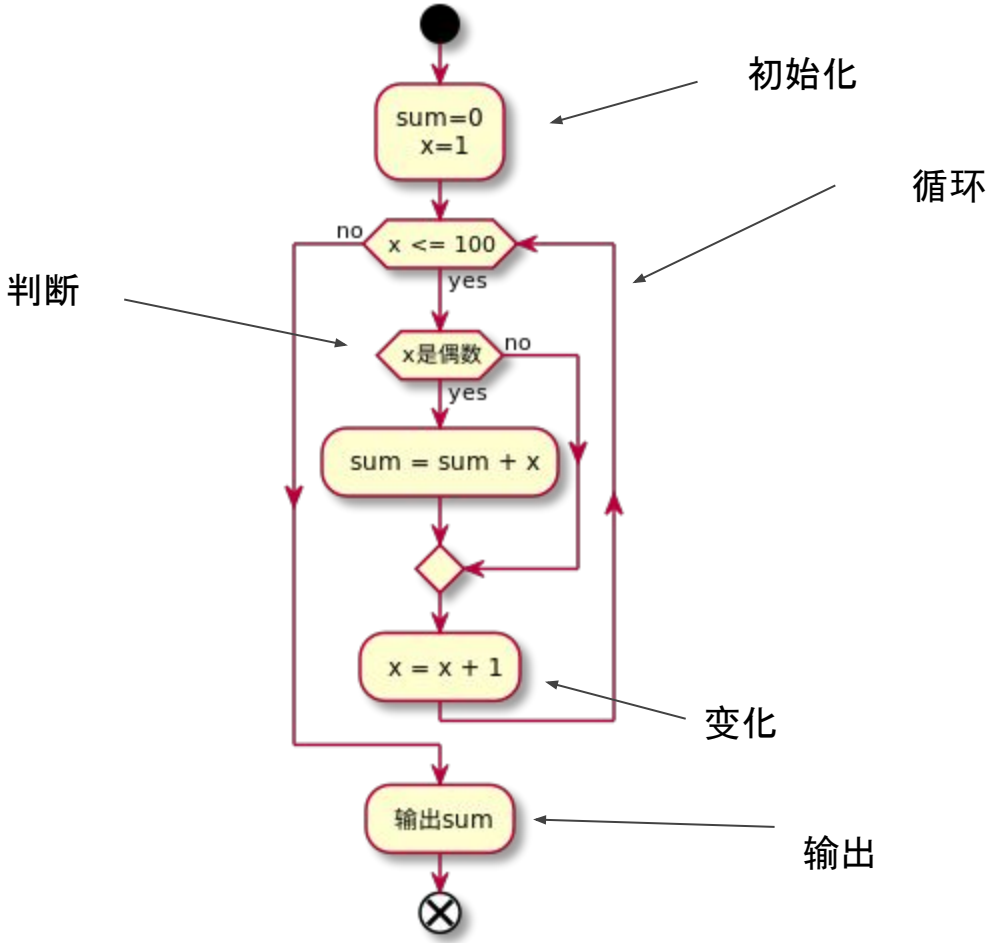
求1～100之间所有偶数的和

- 算法设计
  - 设置一个变量(sum)存储结果
  - 设置一个变量x，x从1开始直到100：如果它是偶数，那么把它累加到sum中
    - 用循环语句实现x从1开始直到100
    - 用if分支语句判断x是否为偶数
      - 被2整除的数为偶数

问题分析与算法设计

- 算法的描述：
  - 自然语言
  - 伪代码
  - 流程图
    - 算法的图形表示法

流程图


初始化
循环
判断
变化
输出
sum=0
x=1
x <= 100
no
yes
x是偶数
no
yes
sum = sum + x
x = x + 1
输出sum

编辑源程序

```
 #include <stdio.h>
int main(void)
{
      int sum = 0, x=1;
      while ( x<=100 )
    {
        if ( x % 2 == 0 )
            sum = sum + x;
        x = x + 1;
    }
    printf("%d", sum);
    return 0;
}
```

编译和调试程序

- 编辑程序后，用相应的编译器对程序进行编译，生成二进制代码表示的目标程序(.o)，与编程环境提供的库函数进行连接形成可执行的程序。

编译程序指出语法错误
调试程序找出逻辑错误

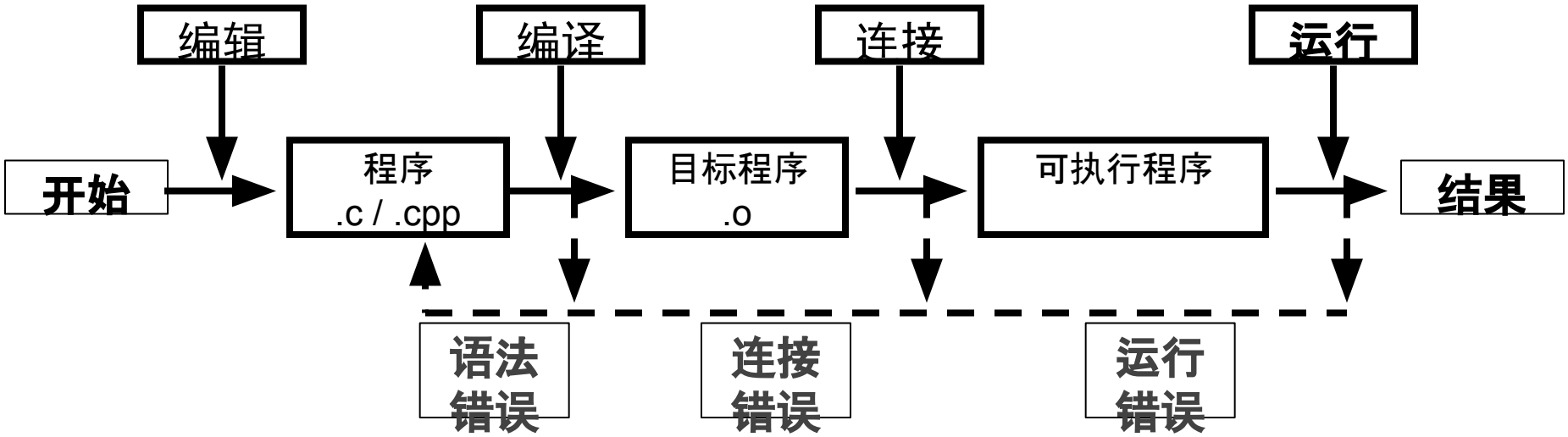
调试程序

如果程序运行所产生的结果不是你想要的结果，这是程序的逻辑错误（语义错误）

- 调试：运行程序，查找并修改错误的过程
- 调试的方法
  - 设置断点
  - 跟踪执行

调试需要耐心和经验，是程序设计中最基本和最重要的技能。

C语言程序的调试、运行步骤



本章要点

- 什么是程序？程序设计语言包含哪些功能？
- 程序设计语言在语法上包含哪些内容？
- 结构化程序设计有哪些基本的控制结构？
- C语言有哪些特点？
- C语言程序的基本框架如何？
- 形成一个可运行的C语言程序主要步骤？
- 如何用流程图描述简单的算法？