01 初识 C 程序

从一个具体任务认识代码、运行和结果

本讲学习目标

- 结合完整 C 程序，说明任务、代码、运行和结果的关系
- 解释任务数据的含义，以及程序怎样处理这些数据
- 在指定执行位置，说明相关变量的值怎样变化
- 区分改变输入与修改源代码，判断是否需要重新生成可执行程序
- 根据错误提示或实际结果与预期的差异，修改程序并再次检查

程序：从任务到自动执行

程序 = 数据对象 + 操作

- 程序通过明确的指令描述处理任务的规则，由计算机执行。
- 程序对数据进行操作，并在执行过程中形成可以观察和检查的结果。

1 程序怎样处理一个任务

认识完整 C 程序的代码、运行与结果，再用程序计算矩形面积。

完整 C 程序及其输出

```c
#include <stdio.h>

int main(void)
{
    printf("Hello, world!\n");
    return 0;
}
```

printf 把文字输出到屏幕

已核对的实际输出：Hello, world!

这段输出对应代码中的 printf

源代码、一次运行与输出

对象	当前例子	作用
源代码	文件中的 C 代码	记录程序要执行的指令
一次运行	按程序规定的指令执行一次	程序实际执行的过程
运行结果	屏幕显示 Hello, world!	可观察、可核对的输出

- 程序是一组为完成任务组织的明确指令；用 C 写在文件中的代码称为**源代码**。
- 阅读源代码、观察一次运行和查看运行结果，关注的是三个不同对象。

矩形面积的任务条件与预期结果

- 矩形面积任务
  - 已知条件：长 4、宽 6
  - 目标：求矩形面积
  - 按长乘宽计算，预期面积是 **24**
- 24 是按任务条件算出的预期结果；程序的实际输出要在运行后检查，并与 24 比较。

面积计算的代码与输出

```c
#include <stdio.h>

int main(void)
{
    printf("%d\n", 4 * 6);
    return 0;
}
```

任务与计算

4 表示长，6 表示宽；4 * 6 计算这个矩形的面积

按任务计算，预期面积为 24

结果核对

当前程序的实际输出为 24，与预期一致

本次只检查了长 4、宽 6 这一组数据

面积程序的主要结构

```c
#include <stdio.h>

int main(void)
{
    printf("%d\n", 4 * 6);
    return 0;
}
```

入口与函数体

main 是当前程序的入口

{ ... } 内的代码组成函数体

输出支持与输出操作

#include <stdio.h> 提供输出所需的声明

printf 把结果输出到屏幕

输出格式

%d 标出整数输出的位置

\n 使输出后换行

语句：规定程序执行某个动作的代码结构

printf(...); 执行输出； return 0; 结束当前程序

这两条语句都以分号；结束

{ ... } 是复合语句； } 后不要求加分号

数据及其任务含义

```c
#include <stdio.h>

int main(void)
{
    printf("%d\n", 4 * 6);
    return 0;
}
```

本讲的数据是程序输入、保存、计算或输出的数值及其表示

这里包括代码中的 4、6 和输出的 24

信息是数据在具体任务中表达的含义

在矩形任务中，4 表示长，6 表示宽，24 表示面积

单看 24 不能断定它表示面积，还需结合对应的任务

固定长宽与运行时输入

```c
// 固定版：直接写入本次长宽
printf("%d\n", 4 * 6);

// 输入版：读取本次长宽
scanf("%d%d", &width, &height);
area = width * height;
printf("Area = %d\n", area);
```

两版程序都计算矩形面积，取得长宽的方式不同

固定版：把长宽数值写在计算语句中

输入版：运行时读取本次长宽，再计算和输出

面积程序的输入、计算与输出

```c
#include <stdio.h>

int main(void)
{
    int width = 0;
    int height = 0;
    int area = 0;

    printf("Enter width and height: ");
    scanf("%d%d", &width, &height);
    area = width * height;
    printf("Area = %d\n", area);

    return 0;
}
```

建立变量

定义三个 int 变量，并把初值设为 0

输入

scanf 读取两个整数；&width、&height 指定把数值写入哪个变量

本例以成功读入两个有效整数为前提；输入失败的处理留到后续

计算

计算长乘宽，把乘法结果赋给 area

输出

printf 显示 area 的当前值，再将实际输出与预期面积比较

计算后变量的当前值与任务含义

名称	当前值	任务含义
width	4	本次输入的长
height	6	本次输入的宽
area	24	本次计算出的面积

- 观察位置：输入 4 6 后，面积已算出，输出语句尚未执行

表中列出这个执行位置上各变量的当前值；它们分别表示什么，由矩形面积任务决定。

练习：面积程序的输入、计算与输出

```c
#include <stdio.h>

int main(void)
{
    int width = 0;
    int height = 0;
    int area = 0;

    printf("Enter width and height: ");
    scanf("%d%d", &width, &height);
    area = width * height;
    printf("Area = %d\n", area);

    return 0;
}
```

输入与保存

输入 3 5 后，3 和 5 各表示什么，分别写入哪个变量?

计算

哪条语句计算面积，预期面积是多少?

输出

哪条语句输出，输出数值表示什么?

解析：面积程序的输入、计算与输出

```c
#include <stdio.h>

int main(void)
{
    int width = 0;
    int height = 0;
    int area = 0;

    printf("Enter width and height: ");
    scanf("%d%d", &width, &height);
    area = width * height;
    printf("Area = %d\n", area);

    return 0;
}
```

输入与保存

同一程序读入 3 5 后，width = 3、height = 5

计算

area = width * height 把计算所得的 15 保存到 area

输出与核对

任务预期 15；已核对的实际输出 Area = 15，两者一致

2 程序怎样运行

观察程序运行时变量值怎样变化，再说明构建与再次运行的区别。

运行中的变量值变化

```
1 int width = 0;
2 int height = 0;
3 int area = 0;
4 scanf("%d%d", &width, &height);
5 area = width * height;
6 printf("Area = %d\n", area);
```

程序状态

当前顺序程序的**程序状态**可用执行位置、相关变量当前值及此时的输出来描述

输入 4 6 的这次运行中，变量值依次变化

括号中的三个数依次是 width、height、area 的当前值

初始化后、输入前：(0,0,0)

输入后、计算前：(4,6,0)

计算后、输出前：(4,6,24)

输出时机

这三个位置都在 printf 执行前；计算后，area 已保存 24，屏幕还没有显示面积

练习：面积程序的变量值变化

```
1 int width = 0;
2 int height = 0;
3 int area = 0;
4 scanf("%d%d", &width, &height);
5 area = width * height;
6 printf("Area = %d\n", area);
```

输入 3 5 后，在以下两个位置，变量的当前值各是多少?

scanf 后、计算前：width、height、area 各是多少?

计算后、输出前：width、height、area 各是多少?

哪条语句改变了哪个变量的值?

解析：面积程序的变量值变化

```
1 int width = 0;
2 int height = 0;
3 int area = 0;
4 scanf("%d%d", &width, &height);
5 area = width * height;
6 printf("Area = %d\n", area);
```

输入 3 5 的这次运行中，输入和赋值改变的变量不同

scanf 后、计算前：(3,5,0)；输入的数值写入 width、height

计算后、输出前：(3,5,15)；赋值语句把乘法结果写入 area

输出时机

此处的 15 是 area 的当前值；printf 尚未执行

程序的信息处理

程序围绕数据执行处理。

先用两个问题观察面积程序：处理什么数据对象？对数据做什么操作？

当前观察视角：数据对象 + 操作。

下面回到同一段代码，指出变量与输入、计算、赋值、输出；再沿执行顺序说明结果。

面积程序中的操作与数据对象

```
1 int width = 0;
2 int height = 0;
3 int area = 0;
4 scanf("%d%d", &width, &height);
5 area = width * height;
6 printf("Area = %d\n", area);
```

任务中的信息

输入的 4 6 表示长和宽，计算所得的 24 表示面积

数据对象：程序中具有类型、保存数据并可由代码访问的对象

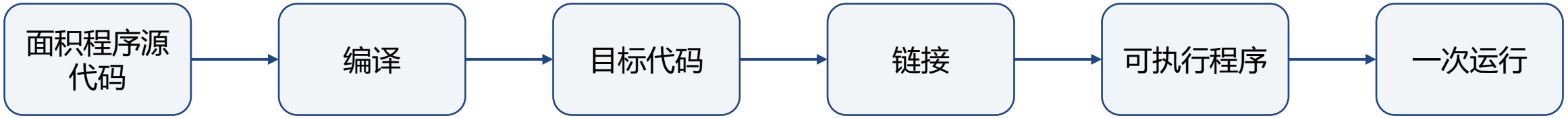
当前程序中的三个 int 变量是 width、height、area

操作：程序对数据执行的处理

本例包括输入、计算与赋值、输出

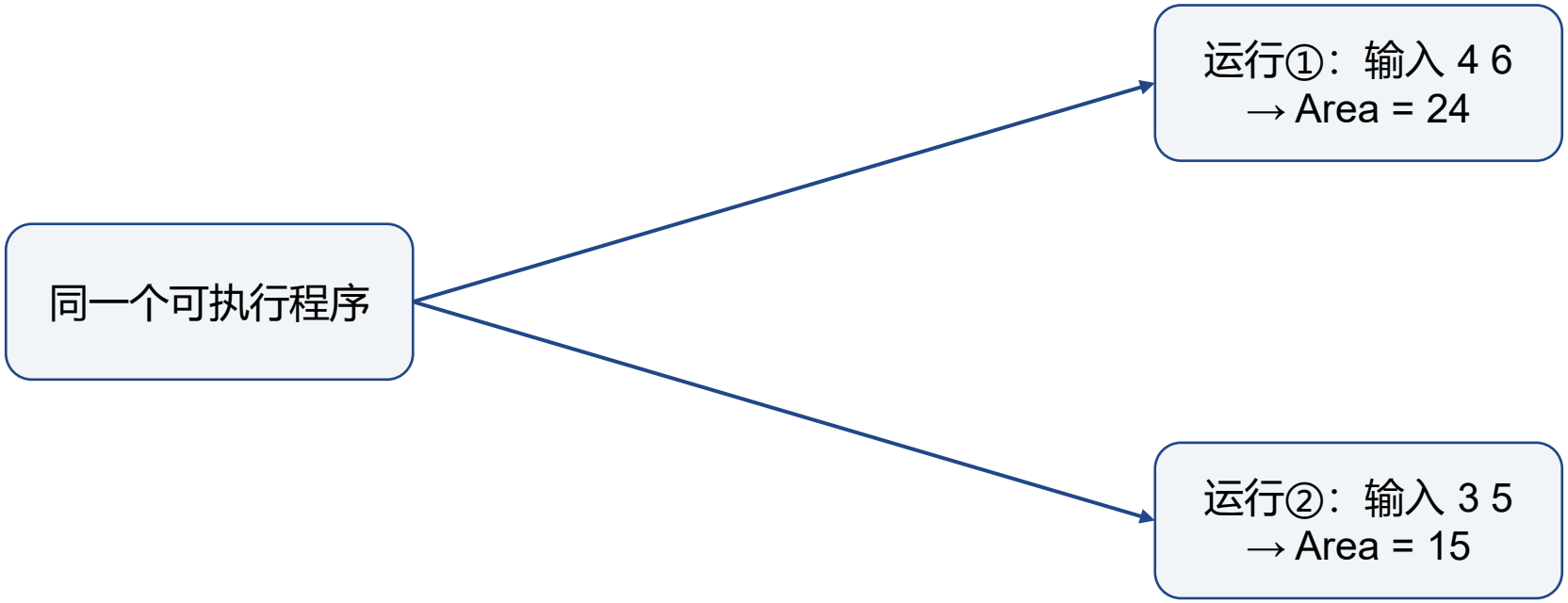
从源代码到可执行程序

- 矩形面积程序保存在 C 源代码文件中。运行前，要把源代码转换为能在当前环境启动的**可执行程序**。
- 从当前源代码生成可执行程序的过程叫**构建**。其中，编译和链接的作用分别是：
  - **编译**：把 C 源代码转换为中间产物，称为**目标代码**，供链接使用
  - **链接**：把目标代码与程序所需的其他代码组合起来，生成可执行程序
- 启动可执行程序后，开始一次运行。
- 日常可把源代码和可执行程序都简称“程序”；需要区分时，分别使用完整名称。



同一可执行程序的两次运行

- 输入版面积程序生成的同一个可执行程序，可以接收不同输入，运行多次
- 两次运行使用相同的源代码和可执行程序，但输入数据不同。



练习：是否需要重新构建

```
1 // 当前程序的提示语句
2 printf("Enter width and height: ");
3
4 // 若把提示语句改为
5 printf("Enter sides: ");
```

比较下面两种改动对原有可执行程序的影响

条件 A：只把运行输入换成 3 5，源代码不变

条件 B：输入仍为 4 6，但修改提示语句

分别判断原有可执行程序能否直接使用、是否需要重新构建；条件 B 下，原有可执行程序会显示哪句提示？

解析：是否需要重新构建

```
1 // 原提示语句
2 printf("Enter width and height: ");
3
4 // 修改后的提示语句
5 printf("Enter sides: ");
```

条件 A：只换输入，可直接使用原有可执行程序

原有可执行程序接收输入 3 5，显示 Area = 15

换输入产生一次新运行

条件 B：修改源代码，须重新构建

原有可执行程序仍显示 Enter width and height:

重新构建生成的新可执行程序显示 Enter sides:，包含修改后的提示语句

3 怎样判断和修正

结合错误提示和运行结果，检查并修改程序。

判断 → 检查 → 修改 → 复查

练习：判断面积程序能否构建

```c
#include <stdio.h>

int main(void)
{
    int width = 0;
    int height = 0;
    int area = 0;

    printf("Enter width and height: ");
    scanf("%d%d", &width, &height);
    area = width * height
    printf("Area = %d\n", area);

    return 0;
}
```

这份面积程序能否成功构建，生成新的可执行程序?

哪里可能有问题？理由是什么？

判断 → 检查 → 修改 → 复查

解析：判断面积程序能否构建

```
1 area = width * height
2 printf("Area = %d\n", area);
```

诊断与定位

本机 MSVC x64 给出编译诊断 C2146，提示 printf 前缺少分号；提示位置是查找错误的线索

缺少分号的是上一行计算语句的末尾

修正

在计算语句末尾补上分号：area = width * height;

补回分号后再次检查

重新构建成功

输入 4 6 后显示 Area = 24

判断 → **检查** → 修改 → 复查

练习：构建结果与计算结果

```c
#include <stdio.h>

int main(void)
{
    int width = 0;
    int height = 0;
    int area = 0;

    printf("Enter width and height: ");
    scanf("%d%d", &width, &height);
    area = width + height;
    printf("Area = %d\n", area);

    return 0;
}
```

任务条件

任务是计算矩形面积；输入 4 6，预期面积为 24

判断与依据

这份程序能否成功构建？运行结果是否符合面积任务？分别说明判断依据

判断 → 检查 → 修改 → 复查

解析：构建结果与计算结果

```
1 // 缺分号变式
2 area = width * height
3
4 // 加法变式
5 area = width + height;
```

能否生成新的可执行程序：检查构建诊断

缺分号的程序出现编译诊断，不能生成新的可执行程序

是否完成面积任务：比较实际结果与任务预期

使用加法的程序能成功构建；输入 4 6 后，实际显示 Area = 10

预期面积为 24，实际结果与预期不同

判断 → **检查** → 修改 → 复查

练习：修正面积计算

```c
#include <stdio.h>

int main(void)
{
    int width = 0;
    int height = 0;
    int area = 0;

    printf("Enter width and height: ");
    scanf("%d%d", &width, &height);
    area = width + height;
    printf("Area = %d\n", area);

    return 0;
}
```

结果差异

输入 4 6，实际输出 10，预期面积 24

修改与理由

找出计算面积的语句，写出修改后的语句，并说明为什么这样修改能符合面积任务

判断 → 检查 → **修改** → 复查

解析：修正面积计算

```
1 // 修改前：计算关系不符合矩形面积任务
2 area = width + height;
3
4 // 修改后：恢复长乘宽
5 area = width * height;
```

重新构建

将加法改为乘法后，重新构建，得到新的可执行程序

运行与核对

输入 4 6，显示 Area = 24，与任务预期一致

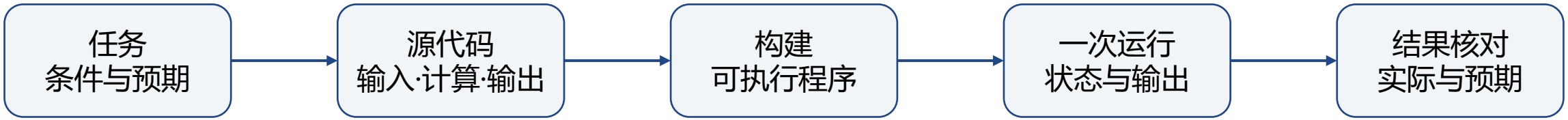
验证范围

本次只检查了这组输入和这处修改

判断 → 检查 → 修改 → **复查**

本讲总结：任务、程序与结果

- 任务给出已知条件、目标和预期结果。
- 源代码规定怎样输入、保存、计算和输出；构建后得到可执行程序。
- 在一次运行中，变量值随执行位置变化，输出语句产生实际输出；实际结果还须与任务预期比较。



本讲总结：关键认识

- 程序处理任务中的信息
  - width、height、area 保存本次数据；输入、计算、赋值与输出作用于这些对象
  - 操作按程序规定的顺序执行，使输入的长宽得到对应的面积结果
- 一次运行是状态变化的过程
  - 结合执行位置和变量的当前值，可以说明程序运行到了哪一步、保存了什么值
- 输入变化与源代码变化不同
  - 源代码经构建得到可执行程序；换输入可再次运行，修改源代码须重新构建
- 检查结果时，要与任务预期比较
  - 编译诊断帮助查找构建问题；比较实际结果与任务预期，判断是否完成任务
  - 修改后重新构建、运行并检查结果；本次验证只针对当前输入和修改

教材阅读指引

- 重点阅读《C Primer Plus》第六版与本讲直接对应的内容
  - 第 1 章“准备工作”：使用 C 的步骤、程序形成与运行、测试和调试的基本关系
  - 第 2 章“C 语言概述”：简单程序、变量与基本输入输出
- 按需查阅本讲已经用到的写法
  - 第 3 章整数类型、第 4 章基本 printf / scanf
  - 第 5 章赋值、简单算术运算和顺序语句
- 阅读重点是本讲的简单顺序程序和已讨论的问题，不要求掌握上述章节的全部内容。