04 循环

循环的执行与终止条件

本单元需要掌握

- 用初始状态、条件、循环体和更新解释循环
- 预测迭代次数、输出和最终状态
- 编写、修正并测试简单循环
- 用状态轨迹和运行结果核查判断

1 从重复语句到循环

逐句复制能够完成小任务，但不能清楚表达规模变化

打印 1 ~ 5：逐句写出能够工作

copied_print.c

```c
printf("1\n");
printf("2\n");
printf("3\n");
printf("4\n");
printf("5\n");
```

五条语句按顺序执行

任务规模写进代码行数

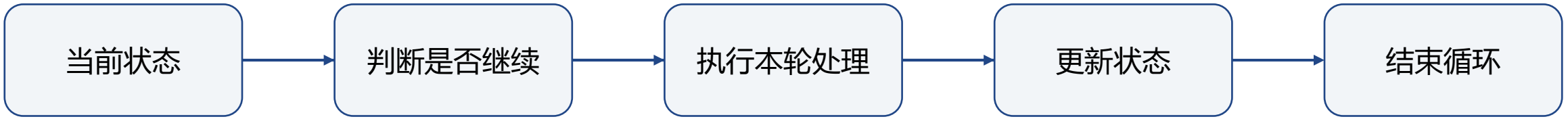
上界变化时需要继续复制

上界改为 100 时，什么发生变化

- 发生变化
  - 输出次数增加
  - 当前数字不断改变
- 保持不变
  - 输出当前数字
  - 输出后前进到下一个数字
- 程序需要表示当前进度，并在每轮决定是否继续。

从固定步骤到可扩展重复

- **能力跃迁** 程序不再需要把每一次重复都写成独立语句，而是用规则让相同处理随状态和任务规模自动重复。
- **实现条件** 这个能力依赖进度、条件和更新的共同作用。
- **本页结论** 循环是程序能力提升的重要机制之一。



第一个 while 循环

while_count.c

```c
int i = 1;

while (i <= 5) {
    printf("%d\n", i);
    i++;
}
```

初始状态 i = 1

继续条件 i <= 5

本轮任务 输出 i

状态更新 i++

2 循环怎样推进并退出

沿执行位置追踪条件、循环体、更新和退出

第一次判断：会进入循环吗

while_count.c

```c
int i = 1;

while (i <= 5) {
    printf("%d\n", i);
    i++;
}
```

当前 i = 1

i <= 5

_____

下一条语句

_____

第一次迭代形成新状态

while_count.c

```c
int i = 1;

while (i <= 5) {
    printf("%d\n", i);
    i++;
}
```

判断

1 <= 5 为真

输出

1

更新

i 由 1 变为 2

返回条件位置

新状态下再次判断

while_count.c

```
1 int i = 1;
2
3 while (i <= 5) {
4     printf("%d\n", i);
5     i++;
6 }
```

当前 i = 2

条件

_____

输出

_____

更新后 i

_____

五次迭代需要六次判断

while_count.c

```c
int i = 1;

while (i <= 5) {
    printf("%d\n", i);
    i++;
}
```

真

i = 1, 2, 3, 4, 5

假

i = 6

5 次循环体

6 次条件判断

循环是条件控制的状态转换

同一个 while，初始状态不同，执行次数不同

初始状态	第一次判断	循环体次数	退出状态
固定设置	条件：i <= 5	更新：i++	只改变初始值
i = 6	6 <= 5 为假	0	i = 6
i = 5	5 <= 5 为真	1	i = 6
i = 1	1 <= 5 为真	5	i = 6
结论	while 先判断	可能执行零次	初始状态决定从哪里开始；更新决定能否退出

3 从计数进度到累加结果

区分控制进度的状态与保存结果的状态

count 与 sum 回答不同问题

- count 表示当前处理到哪个数
  - 初值为 1，每轮增加 1
  - 控制循环何时结束
- sum 表示当前累计结果
  - 初值为 0，每轮加入 count
  - 循环结束后保存结果

预测 1 ~ 5 的累加轨迹

sum_1_to_5.c

```
1 int count = 1;
2 int sum = 0;
3
4 while (count <= 5) {
5     sum += count;
6     count++;
7 }
```

count	sum	条件
1	0	_____
2	__	_____
3	__	_____

累加结果由五轮更新形成

sum_1_to_5.c

```
1 int count = 1;
2 int sum = 0;
3
4 while (count <= 5) {
5     sum += count;
6     count++;
7 }
```

count

1 2 3 4 5 6

sum

0 1 3 6 10 15

退出

count = 6

结果

sum = 15

补全一个固定次数循环

目标：输出 3 ~ 7

```
1 int i = ____;
2
3 while (________) {
4     printf("%d\n", i);
5     ________;
6 }
```

初值决定第一次判断

条件包含 3 ~ 7

更新使循环接近退出

4 三种循环怎样选择

三种结构共享执行模型，但组织控制信息的方式不同

for 集中书写循环控制

while

```c
int i = 1;
while (i <= 5) {
    printf("%d ", i);
    i++;
}
```

for

```c
for (int i = 1; i <= 5; i++) {
    printf("%d ", i);
}
```

for 的书写位置与执行顺序

书写位置

```
1 for (i = 1; i <= 5; i++) {
2     printf("%d ", i);
3 }
```

① i = 1 只执行一次

② 判断 i <= 5

③ 执行循环体

④ 执行 i++

⑤ 回到条件

任务特点帮助选择循环结构

- 固定进度常用 for
  - 范围在开始前清楚
  - 控制信息可以集中阅读
- 条件驱动常用 while
  - 是否继续取决于变化的状态
  - 控制信息不一定集中
- 这些是可读性建议，不是语言限制。

初始条件为假时，还会执行吗

while

```
1 int i = 6;
2 while (i <= 5) {
3     printf("%d", i);
4     i++;
5 }
```

do while

```
1 int i = 6;
2 do {
3     printf("%d", i);
4     i++;
5 } while (i <= 5);
```

do while 先执行再判断

while

```
1 int i = 6;
2 while (i <= 5) {
3     printf("%d", i);
4     i++;
5 }
```

0 次

do while

```
1 int i = 6;
2 do {
3     printf("%d", i);
4     i++;
5 } while (i <= 5);
```

1 次

三种循环的判断时机与常见职责

结构	判断位置	常见任务	能否执行零次
while	循环体之前	条件驱动	能
for	循环体之前	固定进度	能
do while	循环体之后	先做一次再决定	不能
选择边界	先看判断位置	再看任务职责	常见用法不是强制规则

5 循环为什么会出错或不停

从最早偏差检查边界、更新和循环体归属

小于与小于等于规定不同边界

比较条件

```
1 for (i = 1; i < 5; i++)
2     printf("%d ", i);
3
4 for (i = 1; i <= 5; i++)
5     printf("%d ", i);
```

i < 5

1 2 3 4

i <= 5

1 2 3 4 5

最早差异

i = 5

缺少更新时，条件无法接近假

限步追踪，不实际持续运行

```c
int i = 1;
while (i <= 5) {
    printf("%d\n", i);
    /* 没有更新 i */
}
```

第 1 次判断

i = 1，真

循环体后

i 仍为 1

第 2 次判断

仍为真

状态不变，无法接近退出

分号可以构成空循环体

分号就是空循环体

```
1 int i = 1;
2 while (i <= 5);
3 {
4     printf("%d\n", i);
5     i++;
6 }
```

while 的循环体

空语句；

后面的代码块不属于循环

缩进不改变 C 语法

修正循环时检查三个关系

1. 初始状态是否支持第一次正确判断
2. 条件是否包含需要处理的状态
3. 更新是否使状态按正确方向变化并最终退出

- 修正后重新预测，并检查零次、一次和正常次数边界。

6 应用循环并扩展为两层

先把循环模型用于新任务，再用代码和轻度变式扩展到外层行与内层列

外层一次包含内层完整过程

nested_grid.c

```
1 for (row = 1; row <= 3; row++) {
2     for (col = 1; col <= 4; col++)
3         putchar('*');
4     putchar('\n');
5 }
```

外层 3 次
控制行

每行内层 4 次
控制列

内层体共 3 × 4 = 12 次

每轮外层都重新开始内层

两层循环的完整代码

nested_grid.c

```
1 for (row = 1; row <= 3; row++) {
2     for (col = 1; col <= 4; col++) {
3         putchar('*');
4     }
5     putchar('\n');
6 }
```

外层 row
控制行数

内层 col
控制每行列数

每行结束
输出换行

内层完成后，下一轮外层重新开始

应用与迁移：输出递增字符行

目标输出

A
AB
ABC
ABCD

外层控制

_____

内层控制

_____

总字符数

_____

边界检查

第一行与最后一行

独立构造，不给代码骨架

轻度变式：每行长度由外层状态决定

目标输出

1
12
123
1234
12345

外层 row

决定当前行

内层次数

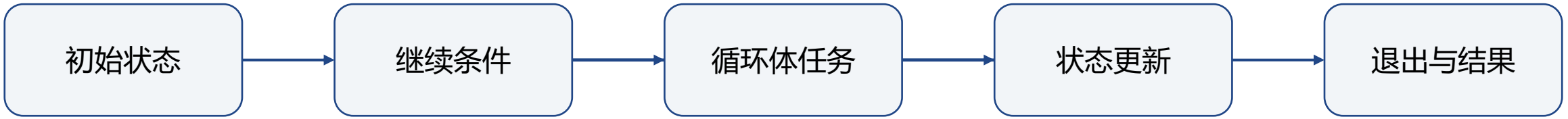
第 1 行 1 次，第 2 行 2 次，依此类推

边界关系

内层条件随 row 改变

静态推演每行长度，不要求实际运行

判断循环是否可靠的五个问题



学习检查与教材对应

学习内容	检查标准	教材范围
状态追踪	次数、输出和最终状态一致	第 5 章循环引入：必读
循环解释	能说明初值、条件、循环体、更新和退出	第 6 章三种循环：必读
应用与迁移	能从任务要求确定循环关系，并用状态轨迹和边界推演说明是否需要修正	循环应用练习：必读
诊断修正	找到最早偏差并核查边界	关系表达式与嵌套循环：选读
独立构造	程序可运行，并能解释结构选择	复杂输入与高级控制：后续
综合证据	预测、轨迹、运行结果和迁移一致	本单元边界内