Loops

C Reference

while & for loops

A loop 循环 repeats a block. A `while` loop runs as long as a condition is true. A `for` loop packs the start, the test, and the increment 自增 (`i++`) into one line — best when you know how many times to repeat.

```c
#include <stdio.h>

int main(void) {
    int i = 1;
    while (i <= 3) {
        printf("%d ", i);
        i++;
    }
    printf("\n");                    // 1 2 3
    for (int j = 0; j < 5; j++) {
        printf("%d ", j);
    }
    printf("\n");                    // 0 1 2 3 4
    return 0;
}
```

while — check first

```
while (cond) {
    body;
}
```

for — count & step

```
for (i=0; i<n; i++) {
    body;
}
```

Exam tip: both need a way to end; for packs init, test, step; while is open-ended

while checks a condition; for counts with a step

Accumulation

A common pattern: start an accumulator 累加器 at `0`, then add to it inside a loop. The same idea counts items or finds a running total.

```c
#include <stdio.h>

int main(void) {
    int total = 0;
    for (int i = 1; i <= 5; i++) {
        total += i;              // 1 + 2 + 3 + 4 + 5
    }
    printf("%d\n", total);       // 15
    return 0;
}
```

Nested loops & patterns

A loop inside another loop is a nested 嵌套 loop. The inner loop finishes fully for each step of the outer loop — perfect for grids and patterns.

```c
#include <stdio.h>

int main(void) {
    for (int row = 0; row < 3; row++) {
        for (int col = 0; col < 3; col++) {
            printf("*");
        }
        printf("\n");
    }
    return 0;
}
```

Common mistakes

- `for (i = 0; i < n; i++)` runs `n` times; a semicolon right after `for (...)` makes an empty loop.
- A `while` whose condition never becomes false loops forever.
- Declare the counter (`int i`) before or inside the `for` header.