Jumping & Control Statements in C

Understanding Program Flow, Loops, and Jumps with Code Examples

Part 1: Control Statements

Directing the flow of your program's execution.

Control: `if-else` Statement

How it Works

- `if`: Executes code if a condition is true.
- `else if`: Tests a new condition if the first is false.
- `else`: Executes code if all preceding conditions are false.

Code Example

```c
#include
int main() {
  int num = 10;
  if (num > 10) {
    printf("Number is > 10");
  } else if (num == 10) {
    printf("Number is 10");
  } else {
    printf("Number is < 10");
  }
  return 0;
}
```

Output

```
Number is 10
```

Control: `switch` Statement

How it Works

- Tests a variable for equality against a list of values.
- Each value is a `case`.
- `break` is vital to exit the switch block.
- `default` runs if no other case matches.

Code Example

```c
#include
int main() {
  int day = 3;
  switch (day) {
    case 1: printf("Monday"); break;
    case 2: printf("Tuesday"); break;
    case 3: printf("Wednesday"); break;
    default: printf("Other day");
  }
  return 0;
}
```

Output

```
Wednesday
```

Control: for Loop

How it Works

Repeats a block of code a known number of times. It has three parts:

- ✓ **Init:** `int i = 1`
  Runs once at the start.
- ✓ **Condition:** `i <= 3`
  Checked before each iteration.
- ✓ **Update:** `i++`
  Runs after each iteration.

Code Example

```c
#include
int main() {
  //  (Init;   Condition; Update)
  for (int i = 1; i <= 3; i++) {
    printf("i = %d\n", i);
  }
  return 0;
}
```

Output

```
i = 1
i = 2
i = 3
```

Control: `while` Loop

How it Works

- ✓ Repeats a block of code as long as a condition is true.
- ✓ The condition is checked **before** the loop body executes.
- ✓ If the condition is false at the start, the loop never runs.

Code Example

```c
#include
int main() {
  int i = 1;
  while (i <= 3) {
    printf("i = %d\n", i);
    i++; // Don't forget to update!
  }
  return 0;
}
```

Output

```
i = 1
i = 2
i = 3
```

Part 2: Jumping Statements

Unconditionally transferring program control.

Jump: `break` Statement

How it Works

- Immediately terminates the nearest enclosing loop (`for`, `while`) or `switch`.
- Control passes to the statement immediately following the terminated block.

Code Example

```c
#include
int main() {
  for (int i = 1; i <= 5; i++) {
    if (i == 4) {
      break; // Exit loop when i is 4
    }
    printf("i = %d\n", i);
  }
  printf("Loop finished.");
  return 0;
}
```

Output

```
i = 1
i = 2
i = 3
Loop finished.
```

Jump: `continue` Statement

How it Works

- Skips the remaining code in the **current** loop iteration.
- Control passes to the next iteration of the loop (update in `for`, condition in `while`).

Code Example

```c
#include
int main() {
  for (int i = 1; i <= 5; i++) {
    if (i == 3) {
      continue; // Skip this iteration
    }
    printf("i = %d\n", i);
  }
  return 0;
}
```

Output

```
i = 1
i = 2
i = 4
i = 5
```

Jump: `goto` Statement

How it Works

- Transfers control to a labeled statement within the same function.
- **Caution:** Use of `goto` is highly discouraged. It makes code hard to read and debug ("spaghetti code").

Code Example

```c
#include
int main() {
  int num = 5;
  if (num < 10) {
    goto less; // Jump to 'less' label
  }
  printf("Num is 10 or more");
  goto end;
less:
  printf("Num is less than 10");
end:
  return 0;
}
```

Output

```
Num is less than 10
```

Jump: `return` Statement

How it Works

- ✓ Terminates the execution of the current function.
- ✓ Returns control to the calling function.
- ✓ Can optionally return a value (e.g., `return sum;`).

Code Example

```c
#include
// This function adds two integers
int add(int a, int b) {
  return a + b; // Return the sum
}
int main() {
  int sum = add(5, 3);
  printf("Sum = %d", sum);
  return 0; // Return from main
}
```

Output

```
Sum = 8
```

Questions?

Thank you.

Image Sources

https://as1.ftcdn.net/v2/jpg/03/48/39/74/1000_F_348397404_wXuf22GUPNAh67htBZZnaDSx3Bj92yep.jpg

Source: www.europosters.fr