The Essentials of C Syntax

Shaan Fulton

July 4, 2025

1. Functions: `argc` and `argv`

A C program starts from `main`, which can accept command-line arguments:

```
int main(int argc, char *argv[]) {
    // ...
}
```

- `argc`: # of arguments passed, including program name.
- `argv`: Array of strings (char pointers); each holds one argument.
- **Why:** Enables user input at launch (`./prog foo bar`).

2. Boolean Values: `false` vs `true`

- In C, **0** is considered `false`. Any nonzero value is considered `true`.
- `NULL` is **not** a Boolean value. It is a special macro that represents a null pointer (i.e., a pointer that points to nothing, usually defined as `((void *)0)`).
- Booleans are not built-in in classic C. Instead, you can use `#include <stdbool.h>` to get the `bool` type, and the keywords `true` and `false` (where `false` is 0 and `true` is 1).

```
// All of these are "false":
if (0)   // false
if (NULL) // false, since NULL is 0 as a pointer
```

```
// These are "true":
if (1)   // true
if (42)  // true

#include <stdbool.h>
bool x = false; // x is 0
bool y = true; // y is 1
```

Key Point: In C, `NULL` can only be used in pointer contexts, not as a general Boolean value.

3. Typed Variables and Memory

- Every variable has a type, which defines its memory usage:

```
int age;      // 4 bytes (usually)
char letter;  // 1 byte
float price;  // 4 bytes
```

- Choosing a type = controlling size and range of values.

4. Structs: Custom Types (e.g., `SONG`, `PLAYER`)

```
typedef struct {
    char name[50];
    int duration;
} SONG;
SONG s = {"Hello", 210};
```

- `structs` group variables into one object.
- Use for modeling things: songs, players, points, etc.

5. `if`, `else`, and Brackets

```
if (x > 0) {
    // block runs if x > 0
} else {
    // otherwise, this runs
}
```

- Always use {} for clarity, even for one-line bodies.

6. Loops: `while` vs `do...while`

```
while (condition) {
    // May run zero or more times
}

do {
    // Always runs at least once
} while (condition);
```

- `while`: Checks before running the body.
- `do...while`: Runs body, then checks condition.

7. Switch Statements and `break`

```
switch(choice) {
    case 1:
        // code
        break;
    case 2:
        // code
        break;
    default:
        // code
}
```

- `break` exits the switch. Omitting it leads to **fall-through**, so the other cases just keep running after!

8. Pointers: `&` and `*`

- `*` declares a pointer type or dereferences a pointer (accesses what it points to).
- `&` gives the address of a variable (where it lives in memory).

```
int x = 42;
int *p = &x;  // p stores the address of x
*p = 13;      // changes x through the pointer
```

Key Point: Pointers let you work with memory addresses directly—essential for dynamic memory, arrays, and passing by reference.