C Programming: Deck 1

Hello World & Program Structure

Prof. Jyotiprakash Mishra
mail@jyotiprakash.org

Topics Covered

What is a C Program?

- A C program is a collection of functions
- Every C program must have a `main()` function
- Execution begins from `main()`
- Programs are written in plain text files with `.c` extension
- Must be compiled before execution

The Simplest C Program

```c
#include <stdio.h>

int main() {
    printf("Hello, World!");
    return 0;
}
```

Let's break down each component...

Component 1: Preprocessor Directive

```
#include <stdio.h>
```

- `#include` is a preprocessor directive
- Tells the compiler to include the contents of `stdio.h`
- `stdio.h` = **St**an**d**ard **I**nput **O**utput header
- Contains declarations for `printf()`, `scanf()`, etc.
- Preprocessor directives start with #
- No semicolon at the end

Component 2: The main() Function

```c
int main() {
    // function body
    return 0;
}
```

- `int` - return type (returns an integer)
- `main` - function name
- `()` - parameter list (empty here)
- `{...}` - function body enclosed in braces
- `return 0;` - returns 0 to the operating system (success)

Component 3: printf() Function

```c
printf("Hello, World!");
```

- `printf()` = **print f**ormatted
- Used to display output on the screen
- Text to be printed is enclosed in double quotes
- This is a function call statement
- Must end with a semicolon ;
- Defined in `stdio.h`

Component 4: return Statement

```c
return 0;
```

- Returns a value from the function
- `return 0;` indicates successful program termination
- Non-zero values typically indicate errors
- Must match the return type of the function (`int`)
- Must end with a semicolon

Single-Line Comments

```c
// This is a single-line comment

int main() {
    // This prints Hello, World!
    printf("Hello, World!");  // Comment after
        code
    return 0;
}
```

- Start with //
- Everything after // on that line is ignored by compiler
- Can appear on their own line or after code

Multi-Line Comments

```c
/* This is a multi-line comment
   It can span multiple lines
   Useful for detailed explanations */

int main() {
    /*
     * This is also valid
     * Stars are optional but look nice
     */
    printf("Hello, World!");
    return 0;
}
```

- Start with /* and end with */
- Can span multiple lines
- Cannot be nested

Why Use Comments?

- Explain complex logic
- Document your code
- Make code readable for others (and future you!)
- Temporarily disable code during debugging
- Add notes and reminders

Important: Comments are ignored by the compiler and don't affect the program's execution.

Program 1: Basic Hello World

```c
#include <stdio.h>

int main() {
    printf("Hello, World!");
    return 0;
}
```

Program 1: Output

```
Hello, World!
```

Note: The output appears on the screen without a newline at the end.

Program 2: Hello World with Newline

```c
#include <stdio.h>

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

New concept: `\n` is an escape sequence for newline

Program 2: Output

```
Hello, World!
```

Note: The cursor moves to the next line after printing.

Program 3: Multiple printf Statements

```c
#include <stdio.h>

int main() {
    printf("Hello, World!\n");
    printf("Welcome to C Programming.\n");
    printf("Let's learn together!\n");
    return 0;
}
```

Program 3: Output

```
Hello, World!
Welcome to C Programming.
Let's learn together!
```

Observation: Each `printf()` with `\n` prints on a new line.

Program 4: printf Without Newlines

```c
#include <stdio.h>

int main() {
    printf("Hello, ");
    printf("World! ");
    printf("How are you?\n");
    return 0;
}
```

Program 4: Output

```
Hello, World! How are you?
```

Observation: Without `\n`, output continues on the same line.

Program 5: With Comments

```c
#include <stdio.h>  // Include standard I/O

/*
 * Program: Hello World with Comments
 * Purpose: Demonstrate commenting
 */

int main() {
    // Print greeting message
    printf("Hello, World!\n");

    /* Print welcome message */
    printf("Welcome to C!\n");

    return 0;  // Exit successfully
}
```

Program 5: Output

```
Hello, World!
Welcome to C!
```

Observation: Comments don't affect the output at all!

Program 6: Escape Sequences

```c
#include <stdio.h>

int main() {
    printf("Line 1\n");
    printf("Line 2\n");
    printf("Tab\there\n");
    printf("Quote: \"Hello\"\n");
    printf("Backslash: \\\n");
    return 0;
}
```

New concepts: \t (tab), \" (quote), \\ (backslash)

Program 6: Output

```
Line 1
Line 2
Tab     here
Quote: "Hello"
Backslash: \
```

Common Escape Sequences

Escape Sequence	Meaning
`\n`	Newline (line break)
`\t`	Horizontal tab
`\\`	Backslash
`\"`	Double quote
`\'`	Single quote
`\0`	Null character

From Source Code to Executable

1. **Preprocessing**
   - Handles `#include`, `#define` directives
   - Removes comments
   - Produces expanded source code
2. **Compilation**
   - Converts C code to assembly language
   - Checks for syntax errors
3. **Assembly**
   - Converts assembly to machine code (object file)
4. **Linking**
   - Links object files and libraries
   - Produces final executable

Compilation Commands (Reference)

For file `hello.c`:

All steps at once:

```
gcc hello.c -o hello
```

Run the program:

```
./hello
```

Note: You already know this, included for completeness

Program Structure - Quick Reference

1. **Header files** - Include necessary libraries
2. **main() function** - Entry point of program
3. **Statements** - Instructions that end with semicolon
4. **Braces** {} - Group statements together
5. **return 0** - Indicate successful completion

Important Points to Remember

- C is case-sensitive (`main` $\neq$ `Main`)
- Every statement ends with a semicolon ;
- `main()` must be present in every program
- Comments are for humans, ignored by compiler
- `#include` has no semicolon
- Curly braces must be balanced
- Indentation improves readability (not required by compiler)

Common Mistakes

1. Missing semicolon at end of statement
   - `printf("Hello")` [WRONG]
   - `printf("Hello");` [CORRECT]
2. Forgetting to include `stdio.h`
   - Results in "implicit declaration" error
3. Unmatched braces
   - Every { must have a matching }
4. Case sensitivity errors
   - `Printf` instead of `printf`

Try These!

1. Write a program to print your name
2. Write a program to print your name and age on separate lines
3. Write a program to print a simple pattern:

```
*****
*****
*****
```

4. Write a program with at least 5 comments explaining each part
5. Experiment with different escape sequences

Sample Solution: Print Pattern

```c
#include <stdio.h>

int main() {
    printf("*****\n");
    printf("*****\n");
    printf("*****\n");
    return 0;
}
```

Sample Solution: Pattern Output

```
*****
*****
*****
```

End of Deck 1

Questions?

Next: Deck 2 - Data Types & Variables