Understanding the C Language

What is Programming?
Programming is the art of giving instructions to a computer using a specific language. C is a procedural, fast language, offering precise control over hardware.

Why Choose C?
C is incredibly fast and allows direct memory access. It is the backbone of operating systems, device drivers, and compilers, and serves as a foundation for modern languages like C++, Java, and Python.

Basic Structure of a C Program

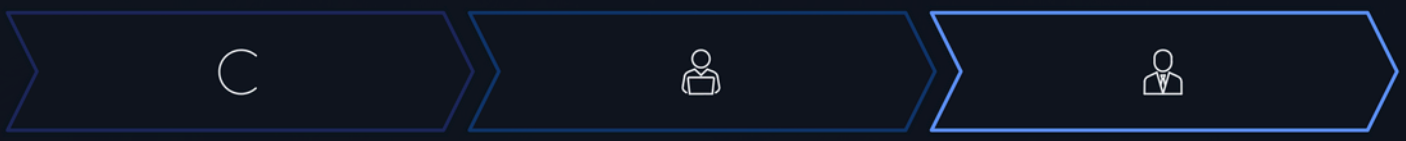
```
#include <stdio.h>

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

- **#include <stdio.h>: Provides access to standard input/output functions (such as printf).**
- **int main(): The entry point where program execution begins.**
- **{ }: Defines a block of code.**
- **printf(): Used to display output on the screen.**
- **return 0;: Indicates that the program executed successfully.**

The Compilation Process and Core E lements

Before your C code can run, it needs to be translated into machine code. This is where the compiler comes in. Understanding this process and the basic building blocks of the language is crucial.



C Code (.c)
Your human-readable source code.

Compiler (gcc)
Translates C code into machine-readable instructions.

Executable Program
The final program ready to run.

Example Compilation

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

Keywords, Identifiers & Comments

Keywords

Reserved words with predefined meanings in C. You cannot use them for variable names.

Examples: int, return, if, else, while

Identifiers

Names you give to variables, functions, and other entities in your program.

- Valid: sum, total_marks, _count
- Invalid: 2num (starts with a number), total (contains a space)

Comments

Notes within your code that are ignored by the compiler. They help explain your code to humans.

- `// Single-line comment`

  `/* Multi-line comment */`

Interacting with Your Program: Input & Output

Programs aren't very useful if they can't communicate! Learn how to display information to the user and receive data from them using printf() and scanf().

Output using printf()

Used to print formatted output to the console. The %d is a format specifier for integers.

```
printf("Welcome to C\n");

int age = 18;

printf("Age is %d\n", age);
```

Input using scanf()

Used to read formatted input from the console. The &symbol is crucial; it provides the memory address of the variable.

```
int marks;
scanf("%d", &marks);
```

Complete Input/Output Example

```
#include <stdio.h>
int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    printf("You entered: %d\n", num);
    return 0;
}
```

Common Beginner Mistakes

Missing Semicolons

Every statement in C must end with a semicolon `;`.

Code outside `main()`

Most executable code belongs within the `main()` function.

Forgetting `#include <stdio.h>`

Essential for input/output functions like `printf` and `scanf`.

Incorrect Input Function

Use `scanf()` for input, not `printf()`.