C Programming Training Module

A Comprehensive Guide for Beginners

Prepared by xAI Training Team

May 07, 2025

Contents

1 Introduction to C

C is a powerful, general-purpose programming language known for its efficiency and control over system resources. This training module introduces C fundamentals, with practical examples to build programming skills.

1.1 Why Learn C?

- **Efficiency**: Close-to-hardware performance for system programming.
- **Foundation**: Basis for languages like C++ and operating systems.
- **Versatility**: Used in embedded systems, OS development, and applications.

1.2 Setting Up the Environment

Install a C compiler like GCC (GNU Compiler Collection). On Windows, use MinGW or WSL; on macOS/Linux, GCC is often pre-installed. Verify with:

```
gcc --version
```

Use an IDE like Code::Blocks or a text editor like VS Code.

2 Basic C Syntax

C programs are structured around functions. Below is a simple ”Hello, World!” program.

2.1 First C Program

```c
#include <stdio.h>

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

Explanation:

- `#include <stdio.h>`: Imports input/output functions.
- `int main()`: Program entry point.
- `printf`: Outputs text to the console.
- `return 0`: Indicates successful execution.

2.2 Variables and Data Types

C supports types like `int`, `float`, `char`, and `double`.

```c
#include <stdio.h>

int main() {
    int age = 25;
    float salary = 50000.50;
    char grade = 'A';
    printf("Age: %d, Salary: %.2f, Grade: %c\n", age, salary, grade)
        ;
    return 0;
}
```

3 Control Structures

Control structures manage program flow.

3.1 Conditional Statements

Use `if-else` for decisions.

```c
#include <stdio.h>

int main() {
    int score = 85;
    if (score >= 90) {
        printf("Grade: A\n");
    } else if (score >= 80) {
        printf("Grade: B\n");
    } else {
        printf("Grade: C\n");
    }
    return 0;
}
```

3.2 Loops

Loops repeat code. Below is a `for` loop example.

```c
#include <stdio.h>

int main() {
    for (int i = 1; i <= 5; i++) {
        printf("Number: %d\n", i);
    }
    return 0;
}
```

4 Functions

Functions promote code reuse.

4.1 Defining Functions

Declare functions with return types and parameters.

```c
#include <stdio.h>

int add(int a, int b) {
    return a + b;
}

int main() {
    int sum = add(5, 3);
    printf("Sum: %d\n", sum);
    return 0;
}
```

5 Pointers

Pointers store memory addresses, enabling efficient memory management.

5.1 Pointer Example

```c
#include <stdio.h>

int main() {
    int num = 10;
    int *ptr = &num;
    printf("Value: %d, Address: %p\n", *ptr, ptr);
    *ptr = 20; // Modify value via pointer
    printf("New Value: %d\n", num);
    return 0;
}
```

6 Arrays and Strings

Arrays store multiple values; strings are character arrays.

6.1 Array Example

```c
#include <stdio.h>

int main() {
    int numbers[5] = {1, 2, 3, 4, 5};
    for (int i = 0; i < 5; i++) {
        printf("Number: %d\n", numbers[i]);
    }
    return 0;
}
```

6.2 String Example

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

int main() {
    char name[] = "Alice";
    printf("Name: %s, Length: %lu\n", name, strlen(name));
    return 0;
}
```

7 Structures

Structures group related data.

7.1 Structure Example

```c
#include <stdio.h>

struct Person {
    char name[50];
    int age;
};

int main() {
    struct Person person = {"Bob", 30};
    printf("Name: %s, Age: %d\n", person.name, person.age);
    return 0;
}
```

8 File Handling

Read from and write to files.

8.1 File I/O Example

```c
#include <stdio.h>

int main() {
    FILE *file = fopen("example.txt", "w");
    if (file == NULL) {
        printf("Error opening file!\n");
        return 1;
    }
    fprintf(file, "Hello, C!\n");
    fclose(file);

    file = fopen("example.txt", "r");
    char buffer[100];
    fgets(buffer, 100, file);
    printf("File content: %s", buffer);
    fclose(file);
    return 0;
}
```

9 Conclusion

This module covers C fundamentals, from syntax to pointers, structures, and file handling. Practice these examples and explore advanced topics like memory management and system programming.

10 References

- C Programming Language, Kernighan & Ritchie
- C Reference: `https://en.cppreference.com/w/c`