C PROGRMMING

1. Setting Up Code::Blocks for C Programming 🖥️

- Step 1: Download Code::Blocks with MinGW (GCC Compiler)
- Step 2: Install Code::Blocks
- Step 3: Verify Compiler Installation
- Step 4: Create Your First C Program

2. Setting Up Visual Studio Code (VSCode) for C Programming 🖥️

- Step 1: Install Visual Studio Code
- Step 2: Install the C/C++ Extension
- Step 3: Install GCC Compiler
- Step 4: Configure VSCode to Use GCC
- Step 5: Compile and Run C Code

3. Troubleshooting Common Setup Issues 🚨

- 1. GCC Compiler Not Found
- 2. Build Errors in VSCode
- 3. VSCode Can't Find C Header Files
- 4. Code::Blocks Compiler Not Working

4. Introduction to C Programming 📘

- History of C
  - How C evolved and who created it.
- Features of C
  - Why C is efficient, portable, and simple.
- Basic Structure of C Program
  - How a C program is structured: `main()` function, header files, etc.

5. Identifiers and Modifiers in C 💡

- Identifiers:
  - Names for variables, functions, arrays, etc.
  - Must start with a letter or an underscore, followed by letters, numbers, or underscores.
  - Cannot be a reserved keyword.

- Modifiers:
  - `signed`, `unsigned`: Control the range of integer types.

6. Data Types and Operators 🔢

- Data Types:
  - Basic types: `int`, `float`, `char`, `double`.
  - Other types: arrays, pointers, and structures.
- Operators:
  - Arithmetic: `+`, `-`, ``, `/`, `%`.
  - Relational: `==`, `!=`, `>`, `<`, `>=`, `<=`.
  - Logical: `&&`, `||`, `!`.
  - Assignment: `=`, `+=`, `-=`, `++`, `--`.
  - Bitwise: `&`, `|`, `^`, `~`, `<<`, `>>`.

7. Control Structures ⏯️

- Conditional Statements:
  - `if`, `if-else`, `else-if`, `switch-case`.
- Loops:
  - `for`, `while`, `do-while`.
  - Loop control: `break`, `continue`.

8. Functions 💻

- Function Basics:
  - How to define and use functions in C.
  - Function parameters and return values.
- Types of Functions:
  - Built-in and user-defined functions.
- Passing Data:
  - Passing data by value and by reference (with pointers).

9. Arrays 📊

- Single-Dimensional Arrays:
  - How to declare, initialize, and access arrays.
- Multi-Dimensional Arrays:
  - Arrays with more than one dimension (e.g., 2D arrays).
- String Handling:
  - Working with strings in C using functions like `strcpy()`, `strlen()`, `strcmp()`.

10. Pointers 🔑

- Introduction to Pointers:
  - What pointers are and how to use them.
  - Pointer arithmetic.
- Pointers and Arrays:

  - How pointers can be used to access array elements.
- Dynamic Memory Allocation:
  - Using `malloc()`, `calloc()`, `realloc()`, and `free()` to manage memory.

11. Structures and Unions 🏗️

- Structures:
  - How to define and use structures to store multiple data types.
- Unions:
  - How unions work by sharing memory.
- Enumerations:
  - Defining custom types with `enum`.

12. File Handling 🗂️

- File Operations:
  - How to open, close, and work with files using functions like `fopen()`, `fclose()`, `fread()`, and `fwrite()`.
- File Pointers:
  - How to navigate files using pointers.
- Error Handling:
  - Checking for errors in file operations.

13. Preprocessor Directives ⚙️

- Macros and Constants:
  - Defining simple macros with `define`.
  - Using `const` to define constants.
- Conditional Compilation:
  - Using `ifdef`, `ifndef` to include/exclude code.
- Include Files:
  - Using standard and custom header files.

14. Error Handling in C ⚠️

- Handling Errors:
  - Using functions like `errno`, `perror()`, and `exit()` to handle errors.

15. C Libraries & Tools 🛠️

- Standard Libraries:
  - Using libraries like `stdio.h`, `stdlib.h`, `math.h`, etc.
- IDE and Debugging:
  - Working with IDEs (Code::Blocks, DevC++) and debugging tools like `gdb`.

16. C Programming Practice and Projects 📈

- Hands-On Projects:
  - Create projects like calculators, bank management systems, and library management systems.
- Building Projects:
  - Apply your skills to real-world projects like a student database or library system.

17. Conclusion and Career in C 🏅

- How to Excel:
  - Keep practicing regularly and solve problems on coding platforms.
- Career Opportunities:
  - C programming is useful in embedded systems, system programming, game development, and more.