**Object-Oriented Programming (OOP) Lab 1**:

**1. History of Programming Languages**

Programming languages have evolved through multiple generations, each bringing significant advancements:

1. **First Generation (1950s-1960s)**
   * Early machine languages (binary) and Assembly language.
   * Used for direct hardware control and system programming.
2. **Second Generation (1950s-1970s)**
   * Introduction of procedural languages like **Fortran (1957)** and **COBOL (1959)**.
   * Focused on numerical computing and business applications.
3. **Third Generation (1970s-1980s)**
   * Emergence of high-level languages like **C (1972)** and **Pascal (1970)**.
   * Increased portability and better abstraction from hardware.
4. **Fourth Generation (1980s-1990s)**
   * Object-Oriented Programming (OOP) languages such as **C++ (1983)** and **Objective-C (1984)**.
   * Allowed modular programming with encapsulation, inheritance, and polymorphism.
5. **Fifth Generation (1990s-Present)**
   * Web development and scripting languages like **Java (1995)**, **Python (1991)**, and **JavaScript (1995)**.
   * Modern programming focuses on security, networking, and artificial intelligence ([The Complete History of Computer Programming Languages - DivNotes](https://divnotes.com/blog/history-of-computer-programming-languages)).

**2. Features of C++**

C++ is an extension of C with additional features for OOP. Some key features include:

* **Object-Oriented Programming:** Supports encapsulation, inheritance, and polymorphism.
* **Strongly Typed:** Requires explicit variable declarations to ensure type safety.
* **Standard Template Library (STL):** Provides reusable components for data structures and algorithms.
* **Memory Management:** Includes manual memory control via new and delete operators.
* **Multi-Paradigm Language:** Supports procedural, object-oriented, and generic programming.
* **Portability & Efficiency:** Runs on various hardware and operating systems with high performance.
* **Operator Overloading:** Allows defining custom behavior for operators like + and -.
* **Exception Handling:** Provides error management through try, catch, and throw blocks.

**3. Overview of Basic C++ Concepts**

* **Variables & Data Types:** Integer (int), floating-point (float), character (char), and Boolean (bool).
* **Operators:** Arithmetic (+, -, \*, /), relational (==, !=, <, >), and logical (&&, ||, !).
* **Control Structures:** if-else, switch, for, while, and do-while loops.
* **Functions:** Modular code organization, return values, and function overloading.
* **Pointers & References:** Direct memory access using \* and & operators.
* **Classes & Objects:** Core OOP features for structuring code efficiently.
* **File Handling:** Read/write operations using file streams (ifstream, ofstream).
* **Templates:** Enable generic programming for reusable code.

**4. Required Tools and Their Configurations**

To set up a **C++ development environment**, you need the following tools:

1. **Integrated Development Environment (IDE):**
   * **Code::Blocks**, **Dev-C++**, **Eclipse**, or **Visual Studio**.
   * Recommended for debugging, syntax highlighting, and project management.
2. **Compilers:**
   * **GCC (GNU Compiler Collection):** For Linux and Windows (via MinGW).
   * **Clang:** Known for performance optimization.
   * **Microsoft Visual C++ (MSVC):** Part of Visual Studio.
3. **Debugging Tools:**
   * **GDB (GNU Debugger):** Helps in debugging C++ programs.
   * **Valgrind:** Memory leak detection tool.
4. **Configuration Steps:**
   * Install a compiler (e.g., MinGW for Windows, GCC for Linux).
   * Set up environment variables (PATH for compiler access).
   * Use a text editor (Notepad++, VS Code) or an IDE for development.

**5. Edit, Compile, and Run C++ Applications**

The process involves the following steps:

1. **Editing:**

**Write a simple C++ program using a text editor or an IDE.**

#include <iostream>

using namespace std;

int main() {

cout << "Hello, World!" << endl;

return 0;

}

1. **Compiling:**
   * Use a compiler like **Dev c++** to compile the code.
   * Command: C++ program.cpp output.exe (Windows)
2. **Running:**
   * Execute the compiled file:
   * Windows: output.exe
   * Linux: ./output.out
3. **Debugging & Fixing Errors:**
   * If errors occur, check for syntax issues and use debugging tools.
   * Enable debugging mode in an IDE.