1. Write compilation stages & procedure in c++

**Preprocessing:**

- Preprocessor handles directives, expands macros, and includes headers.

- Output: preprocessed source code (.i)

g++ -E source\_file.cpp -o preprocessed\_file.cpp

**Compilation:**

- Compiler generates assembly code from preprocessed source.

- Output: assembly language file (.s)

g++ -c preprocessed\_file.cpp -o object\_file.o

**Assembly:**

- Assembler converts assembly to object file.

- Output: object file (.o)

g++ -S source\_file.cpp -o assembly\_file.s

**Linking:**

- Linker combines object files and resolves external references.

- Output: executable file

g++ object\_file.o -o executable\_file

**Combine the compilation and linking stages into a single step: g++ source\_file.cpp -o executable\_file**

**Run file: ./excutable\_file**

1. Why we need to compile the c++ code ?
2. **Machine-Readable Code**: C++ is human-readable, but computers need machine code to execute.
3. **Optimization**: Compilers can optimize the code for better performance.
4. **Linking**: Compilers link external libraries to create the final executable.
5. **Error Checking**: Compilation includes checks to identify and report errors in the source code.
6. **Platform-Independence**: Compiled executables can run on different hardware and software platforms.
7. **Memory Management**: Compilers generate code to handle memory allocation and deallocation.
8. Can i use c++ compiler to compile C code?

**Yes**, we can use a C++ compiler to compile C code. This is possible because C++ is a superset of the C programming language, meaning that any valid C code is also valid C++ code.

1. Can i use C compiler to compile C++ code?

**Yes**, It is possible to use a C compiler to compile some C++ code, but this is not recommended because the C compiler cannot handle the advanced features and libraries of C++. Therefore, it is best to use a dedicated C++ compiler, such as g++ or clang++, to ensure that all language-specific parameters are handled correctly and that optimized and correct executables are produced.

5. Is the Python object oriented or non-object oriented programming language?

Python is an object-oriented programming (OOP) language, but it also supports non-object-oriented programming paradigms.