Compilation process in C++

What is a compilation?
Compilation is the transformation of source code into object code. It is accomplished with the assistance of the compiler. The compiler checks the source code for syntactical and structural faults, and then creates the object code if the source code is error-free.
The c++ compilation process translates source code into object code, often known as machine code. Pre-processing, Compiling, Assembling, and Linking are the four steps in the compilation process.

The process of converting a source file into an executable file is divided into four stages.

1) Preprocessor

The compilation phase is carried out on each of the preprocessor's outputs. The compiler transforms the pure C++ source code into assembly code (without any preprocessor directives). The underlying back end (toolchain's assembler) then assembles the code into machine code, resulting in a binary file in some format (ELF, COFF, a.out, ...). The compiled code (in binary form) of the symbols defined in the input is stored in this object file. Object files use names to refer to symbols.

2) Compiler

The compilation phase is carried out on each of the preprocessor's outputs. The compiler transforms the pure C++ source code into assembly code (without any preprocessor directives). The underlying back end (toolchain's assembler) then assembles the code into machine code, resulting in a binary file in some format (ELF, COFF, a.out, ...). The compiled code (in binary form) of the symbols defined in the input is stored in this object file. Object files use names to refer to symbols.

3) Assembler

Object code is created by the assembler. On a UNIX system, files with the suffix.o (.OBJ on MSDOS) signify object code files. The assembler translates those object files from assembly code to machine level instructions in this phase, and the resulting file is a relocatable object code file. As a result, the compilation process produces a relocatable object program, which can be utilized in several locations without requiring recompilation.

4) Linking

The linker is responsible for generating the final compilation output from the assembler's object files. This result can be a shared (or dynamic) library (which, despite the name, has little in common with the static libraries discussed above) or an executable.
It connects all the object files by replacing undefined symbol references with the right addresses. Other object files or libraries can define each of these symbols. You must inform the linker if they are specified in libraries other than the standard library.