**Исходный C++ файл** — это всего лишь код, но его невозможно запустить как программу или использовать как библиотеку. Поэтому каждый исходный файл требуется скомпилировать в исполняемый файл, динамическую или статическую библиотеки.

**1) Препроцессинг**

**Препроцессор** — это макро процессор, который преобразовывает вашу программу для дальнейшего компилирования. На данной стадии происходит происходит работа с препроцессорными директивами.

Хэдеры, включенные в программу с помощью директивы **#include**, рекурсивно проходят стадию препроцессинга и включаются в выпускаемый файл. Однако, каждый хэдер может быть открыт во время препроцессинга несколько раз, поэтому, обычно, используются специальные препроцессорные директивы, предохраняющие от циклической зависимости.

2) Компиляция

На данном шаге g++ выполняет свою главную задачу — компилирует, то есть преобразует полученный на прошлом шаге код без директив в *ассемблерный код*. Это промежуточный шаг между высокоуровневым языком и машинным (бинарным) кодом.

3) Ассемблирование

Так как x86 процессоры исполняют команды на бинарном коде, необходимо перевести ассемблерный код в машинный с помощью **ассемблера**.

Ассемблер преобразовывает ассемблерный код в машинный код, сохраняя его в *объектном файле*.

**Объектный файл** — это созданный ассемблером промежуточный файл, хранящий кусок машинного кода. Этот кусок машинного кода, который еще не был связан вместе с другими кусками машинного кода в конечную выполняемую программу, называется *объектным кодом*.

Далее возможно сохранение данного объектного кода в *статические библиотеки* для того, чтобы не компилировать данный код снова.

4) Компоновка

**Компоновщик (линкер)** связывает все объектные файлы и статические библиотеки в единый исполняемый файл, который мы и сможем запустить в дальнейшем. Для того, чтобы понять как происходит связка, следует рассказать о *таблице символов*.

**Таблица символов** — это структура данных, создаваемая самим компилятором и хранящаяся в самих объектных файлах. Таблица символов хранит имена переменных, функций, классов, объектов и т.д., где каждому идентификатору (символу) соотносится его тип, область видимости. Также таблица символов хранит адреса ссылок на данные и процедуры в других объектных файлах.

5) Загрузка

Последний этап, который предстоит пройти нашей программе — вызвать загрузчик для загрузки нашей программы в память. На данной стадии также возможна подгрузка *динамических библиотек*.

Используемые источники:

<https://habr.com/ru/post/478124/>