# Многофайловые программы. Библиотеки

### Этапы сборки проекта на языке С++:

1. **Препроцессинг**. Обрабатываются директивы компилятора #include, #define и другие. Удаляются комментарии. Чтобы исполнить только этот шаг, нужно передать компилятору опцию -E:

```
g++ -E main.cpp > preprocessed.cpp
```

2. **Компиляция**: каждый файл исходного кода (файл расширения . cpp) транслируется в код на языке ассемблера. Чтобы исполнить только этапы препроцессинга и компиляции, нужно передать компилятору опцию -S:

```
g++ -S main.cpp
```

3. **Ассемблирование**: каждый файл на языке ассемблера транслируется в машинный код. В результате создаётся объектный файл с расширением .o. Чтобы исполнить процесс до этой стадии включительно нужно передать компилятору опцию -c:

```
g++ -c main.cpp
```

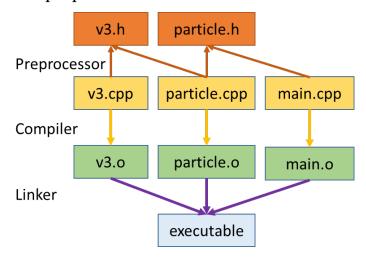
4. **Линковка**: Все объектные файлы сливаются друг с другом, а также с другими библиотеками. Даже если ваш проект состоит из одного файла, вы наверняка используйте как минимум стандартную библиотеку и на этом этапе ваш код соединяется с другими библиотеками.

```
g++ main.cpp или g++ main.o
```

#### Задание:

• В папке Ostages лежит исходный код простой программы. Пройдите поэтапно все стадии сборки с этой программой.

### Сборка многофайловой программы:



Можно собрать всё сразу:

```
g++ main.cpp particle.cpp v3.cpp
```

Либо можно собрать по частям:

```
g++ -c main.cpp
g++ -c particle.cpp
g++ -c v3.cpp
g++ main.o particle.o v3.o
```

#### Виды библиотек:

- 1. **header-only библиотеки:** Весь исходный код хранится в .h файле и подключается с помощью директивы #include (очень просто подключить).
- 2. Исходный код: Библиотека поставляется в виде исходного кода (все .h и .cpp файлы). Для того чтобы использовать эту библиотеку, её нужно сначала скомпилировать, что может быть очень непросто для больших библиотек, так как процесс сборки может сильно отличаться на разных операционных системах и компиляторах.
- 3. Статическая библиотека: Библиотека поставляется в виде header-файлов(.h) и предварительно скомпилированных файлов библиотеки (.a или .lib). Эти библиотеки подключаются на этапе линковки. Такие библиотеки проще подключить к проекту, чем исходный код. Однако, вам обязательно иметь версию библиотеки, скомпилированную на той же ОС и на том же компиляторе, иначе она не подключится. Обратите внимание, что статические библиотеки обязательно должны иметь префикс lib и расширение .a (или .lib). Например, если мы хотим получить библиотеку под названием image, то файл должен называться libimage.a.
- 4. Динамическая библиотека: Библиотека поставляется в виде header-файлов(.h) и предварительно скомпилированных файлов библиотеки (.sp или .dll). Эти библиотеки подключаются на этапе выполнения программы.

#### Задания:

- header: В папке 1image/Oheader-only лежит исходный код программы, которая использует класс Image. Это простой класс для работы с изображениями в формате .ppm. Скомпилируйте и запустите эту программу.
- Случайные отрезки: Используйте этот класс, чтобы создать изображение, состоящее из 100 случайных отрезков случайного цвета. Для случайных чисел используйте функцию rand() из библиотеки <cstdlib>.
- Случайные прямоугольники: Добавьте в этот класс метод void draw\_rectangle(const Vector2i& bottomleft, const Vector2i& torright, const Color& color). Используйте этот метод, чтобы создать изображение, состоящее из 100 случайных прямоуголиников случайного цвета.
- Шум: Добавьте в этот класс метод void add\_noise(float probability), который будет добавлять шум на картинку: каждый пиксель с вероятностью probability должен поменять цвет на случайный. Протестируйте этот метод на картинках.
- Раздельная компиляция: В папке limage/lseparate\_compilation лежит тот же код, но разделённый на 2 файла исходного кода. Скомпилируйте эту программу с помощью g++.Добавьте функции draw\_rectangle и add\_noise из предыдущих заданий в этот проект.
- Статическая библиотека: Чтобы создать свою статическую библиотеку вам нужно:
  - 1. Создать объектный файл необходимого исходного файла.
  - 2. Превратить объектный файл (или файлы) в библиотеку, используя утилиту ат:

```
ar rvs libimage.a image.o
```

3. После этого файл libimage.a можно будет подключить к любому другому проекту примерно так:

```
g++ main.cpp -I<путь до header-файлов> -L<путь до libimage.a> -limage
```

В папке 1image/2static\_library лежит исходный код программы. Вам нужно создать статическую библиотеку из файла image.cpp и поместить полученный файл в папку image/lib, а header-файл поместить в папку image/include. Затем вам нужно удалить файл image.cpp и собрать программу используя только статическую библиотеку (не забывайте про опции -I, -L и -1).

- Статическая библиотека 2: В папке 1image/3static\_test лежит проект с одной очень маленькой статической библиотекой (содержит 1 функцию). Вам нужно собрать этот проект.
- Makefile: make это специальная утилита, предназначенная для упрощения сборки проекта. В limage/4makefile содержится пример проекта с make-файлом. Чтобы скомпилировать его просто:

make <имя цели>
либо просто
make

## Библиотека sfml: