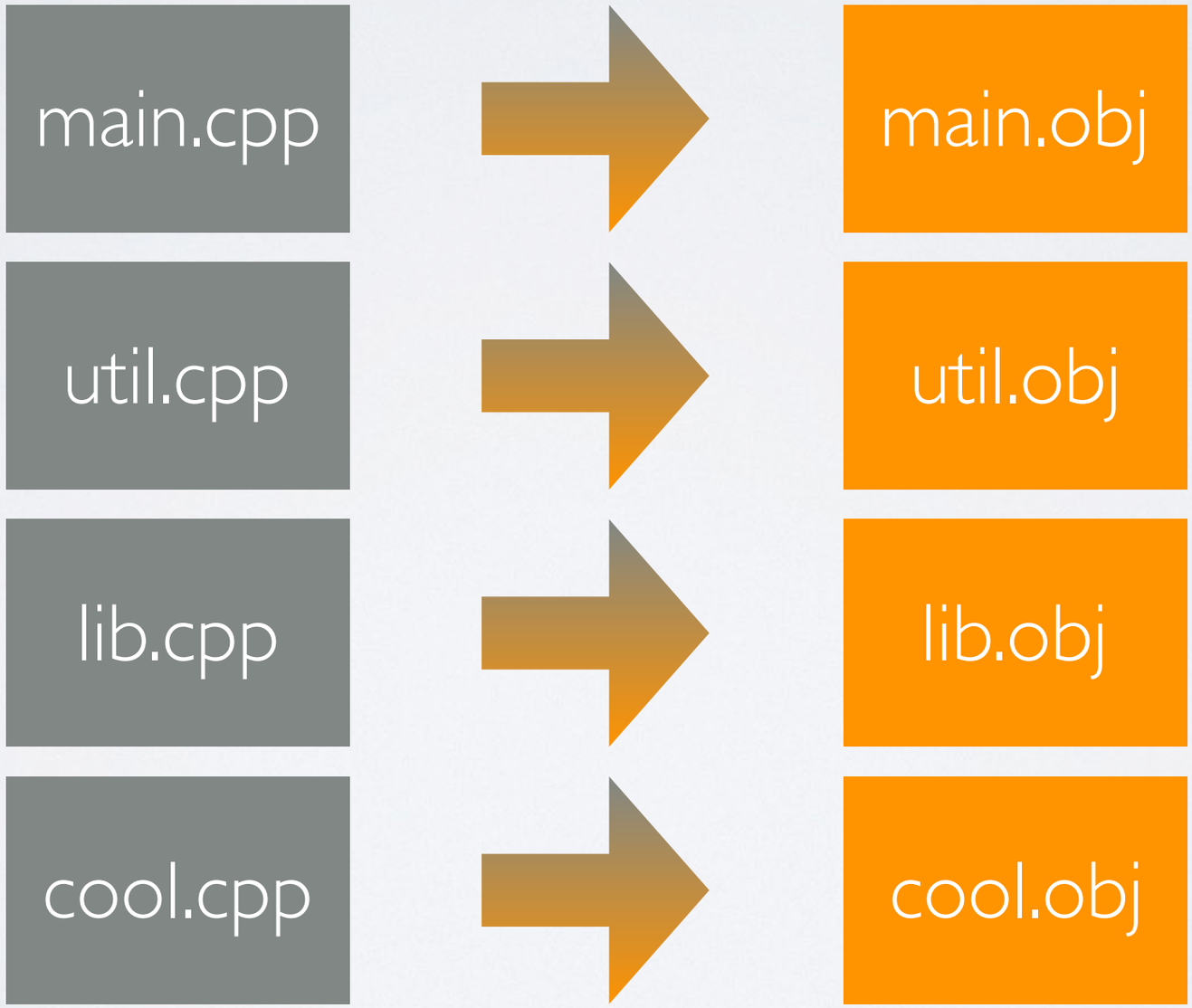
ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

БИБЛИОТЕКИ

КОМПИЛЯЦИЯ

Исходные файлы (модули)

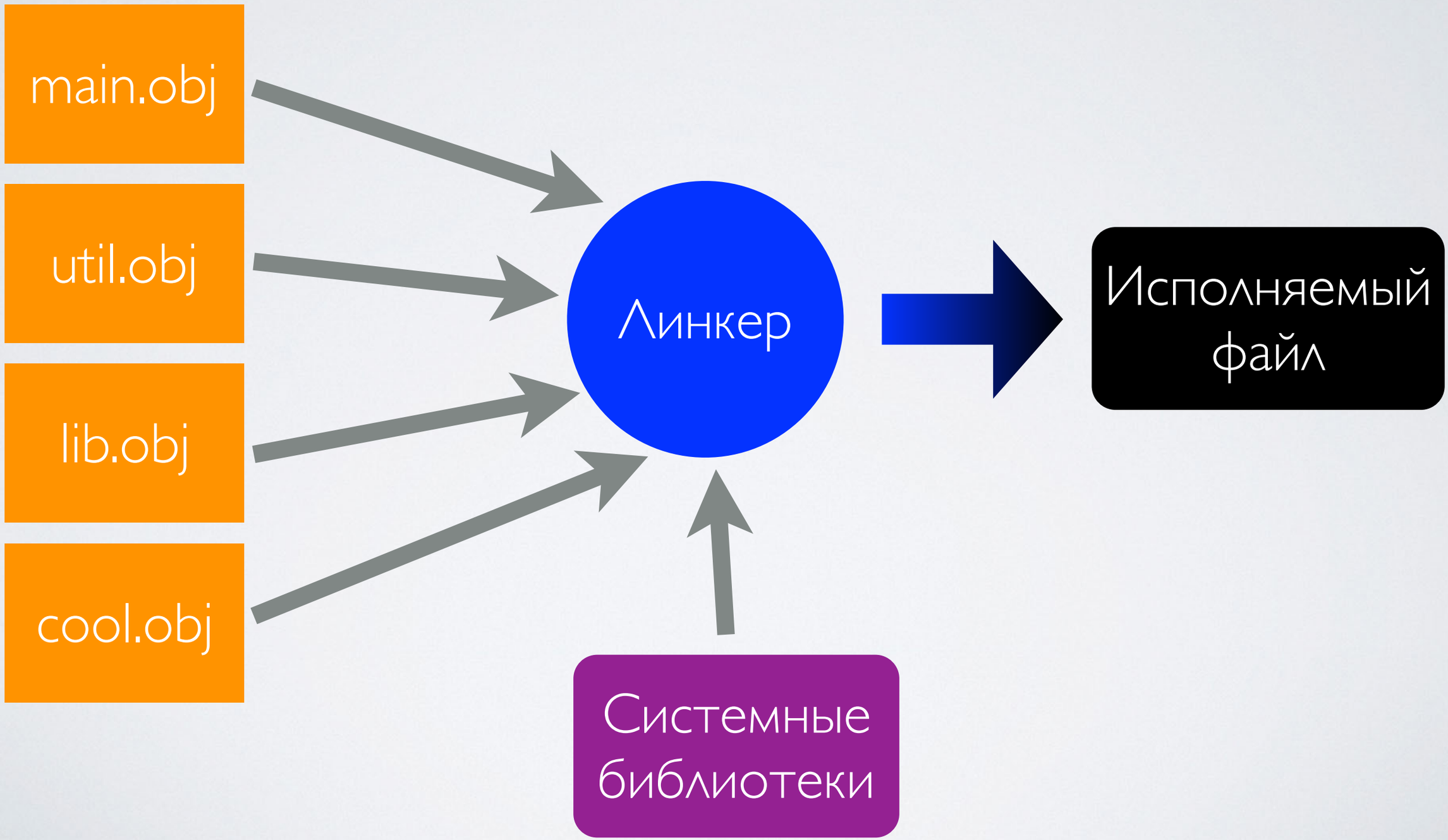
Объектные файлы



ОБЪЕКТНЫЙ ФАЙЛ FILE.OBJ

- Машинный код функций и методов, объявленных в **file.cpp**.
- Память под объявленные глобальные и статические переменные.
- Ссылки на внешние функции и методы.
- Ссылки на глобальные переменные.

ЭТАП 2. СВЯЗЫВАНИЕ



РАБОТА ЛИНКЕРА

- Операционная единица: **имя**. Каждый объектный модуль (в т.ч. библиотечный):
  - Предоставляет какие-то имена (функции, переменные...)
  - Требует какие-то имена.
- Линкер удовлетворяет зависимости (все начинается с имени *main*).
- Ошибки:
  - Имя требуется одним из модулей, но не предоставляется ни одним из них.
  - Одно и то же имя предоставляется более, чем одним модулем.

ИСПОЛНЯЕМЫЙ ФАЙЛ

- Содержит все нужные имена (и ничего лишнего). Все ссылки на имена в объектных файлах были разрешены.
- По построению зависит от объектных файлов и библиотек (те зависят от исходных файлов).
- Для выполнения не нужно больше ничего (за исключением динамических библиотек).

КАК ПОДКЛЮЧИТЬ БИБЛИОТЕКУ?

- Первый вариант — просто скопировать её исходные ***.cpp** и ***.h** файлы в свой проект.

- Второй вариант — воспользоваться готовой сборкой.

Подключаем готовую сборку библиотеки

LIB-ФАЙЛ

Архив obj-файлов

ДИНАМИЧЕСКИЕ БИБЛИОТЕКИ

- Код и данные, соответствующие именам, находятся в отдельном файле (*.dll).

- Статическая линковка DLL — в исполняемый файл включаются ссылки на DLL-файлы. Без них он просто не запустится.

- Динамическая подгрузка DLL — файл открывается прямо в процессе исполнения программы. Используется для плагинов.

КАК ВЫБРАТЬ БИБЛИОТЕКУ?

ЧЕМ ОТЛИЧАЕТСЯ ХОРОШАЯ БИБЛИОТЕКА ОТ ПЛОХОЙ?

- Прикладная установка: как можно лучше решить задачу, но быстро и ничего лишнего.
- Библиотечная установка: реализовать инструменты для решения прикладных задач с рассмотрением ВСЕХ частных случаев.

- Пример: создать процедуру для решения квадратного уравнения

$$Ax^2 + Bx + C = 0.$$

```cpp
#include <cmath>
#include <utility>

std::pair<double> quadraticEquation(double A, double B, double C)
{
    double D = B*B - 4*A*C;
    double SD = std::sqrt(D);

    return std::make_pair((-B-SD)/(2*A),
                          (-B+SD)/(2*A));
}
```

Проблемы ? Проблемы !!!

```
class QuadraticEquation
public:
    QuadraticEquation(double A, double B, double C);

    enum Type {
        NO_ROOTS,
        INFINITE_ROOTS,
        SINGLE_ROOT,
        DOUBLE_ROOT,
        TWO_ROOTS,
        COMPLEX_ROOTS
    };

    Type getType() const;
    double root1() const;  // SINGLE_ROOT, DOUBLE_ROOT, TWO_ROOTS
    double root2() const;  // DOUBLE_ROOT, TWO_ROOTS

    std::pair<std::complex> complexRoots() const; // COMPLEX_ROOTS
};
```

Уже лучше

- Но если вы настоящий «библиотекарь», то…
  - Google: *«quadratic equation numeric stability»*.
- Один из корней склонен к *underflow*, если $b^2 \gg 4ac$ (например, корень с «плюсом», если $b > 0$).

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

СТАБИЛЬНОЕ РЕШЕНИЕ

$$x_1 = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \quad \& \quad x_2 = \frac{2c}{-b - \sqrt{b^2 - 4ac}}$$

если b ≥ 0

$$x_1 = \frac{2c}{-b + \sqrt{b^2 - 4ac}} \quad \& \quad x_2 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

если b < 0

CHECK-LIST ПО ВЫБОРУ БИБЛИОТЕКИ ПОД ЗАДАЧУ

- История («откуда ноги растут?»).
- Поддержка стандартов C++.
- Поддерживается или заброшена? Когда была выпущена последняя версия?
- Наличие открытых исходных кодов (желательно на GitHub).
- Поддержка различных компиляторов и платформ (Windows, Linux, OS X, Android, iOS, …).
- Отзывы и рекомендации в Интернетах (stackexchange.com и проч.).
- http://fffaraz.github.io/awesome-cpp/

QT

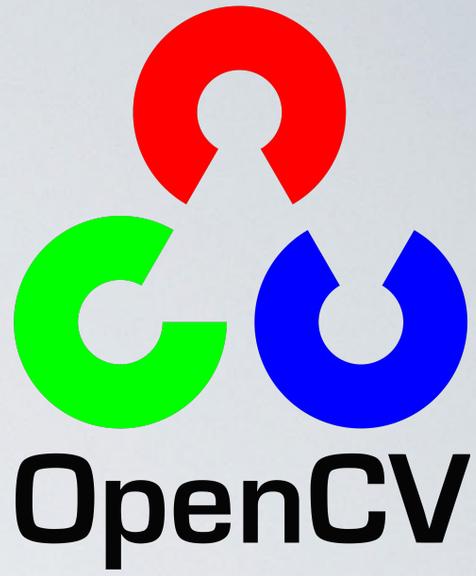
- http://www.qt.io/
- Кроссплатформенная библиотека для GUI, но не только:
  - Работа с изображениями, аудио, видео.
  - Работа с базами данных.
  - Создание сетевых приложений.
  - Удобные инструменты для создания многопоточных приложений.
  - Встроенные скрипты на JavaScript.
  - …

BOOST

- http://www.boost.org.

- Совокупность библиотек на различные темы (работа с текстом, парсинг, контейнеры и АТД, алгоритмы, метапрограммирование и т.д. и т.п.)

OPENCV



- http://opencv.org/
- Основная тема: Computer Vision и Machine Learning.
- Помимо всего прочего:
  - Работа с графическими файлами.
  - Всевозможные алгоритмы обработки изображений.
  - Работа с видео.

СТРУКТУРИРОВАНИЕ КОДА

СТРУКТУРИРОВАНИЕ ИСХОДНЫХ ФАЙЛОВ

- Проще всего: один класс (или **namespace**) — 1 **.h**-файл и 1 **.cpp**-файл.
- Объявления пространств имён, типов, классов, констант и т.п. — в **.h** файл.
- Определения **inline**-функций — туда же.
- Шаблоны классов, специализации, шаблонные методы обычных классов — туда же.
- Определения обычных функций и методов — в **.cpp**-файл! Рядом можно положить объявления и определения вспомогательных классов (внутри **namespace { }**).

КРУПНЫЕ БЛОКИ ПРОЕКТА

- **Главный результат:** набор исполняемых файлов.
- **Вспомогательные результаты:** DLL и LIB.
- Всё общее выносится в библиотеки.