**План обучения С++**

**Этап 1 – Основы.**

Прочитать книгу по основам С++ (стандарт языка не ниже С++11) – рекомендую «C++11 за 21 день - 2013.pdf» С. Рао (я по ней учился) – книга 2013 года, может есть более поздняя редакция в интернете. Все книги прикрепил. Можно еще книгу Р. Лафоре (Lafore\_4ed.pdf), если что-то непонятно или не хватает из первой книги, или интернет. В любом случае, надо использовать информацию из интернета (вопросы на собеседованиях по С++, Хабр, cppreference.com, видео собеседований - youtube), если что-то непонятно из книг.

1. Основы С++ – применение на практике основ программирования:

* конструкций ветвления if – else if – else, switch – case;
* конструкций повторения (циклов) for, while, do-while;
* статических и динамических массивов, указателей, ссылок, функций, операторов new, delete;
* арифметики указателей.

1. Основы объектно-ориентированного программирования (ООП) в контексте С++ – применение на практике принципов ООП – инкапсуляции, наследования, полиморфизма:

* программирование с использованием классов, объектов;
* использование конструкторов, деструкторов, конструкторов копирования и перемещения, операторов копирования и перемещения;
* использование наследования (создание производных классов);
* реализация и использование полиморфизма на основе виртуальных функций;
* использование перегрузки операторов и операторов приведения типов.

1. Применение на практике стандартной библиотеки С++ и ее самой важной части STL (книга – «Джосаттис Н.М. - Стандартная библиотека C++\_2014.djvu»):

* использование std::string и создание собственного класса строки (разобраться как создается такой класс на примере кода из книг или интернета), чем отличается глубокое копирование (deep copy) от мелкого (shallow copy);
* применение стандартных контейнеров std::vector, std::deque, std::list, std::map, std::unordered\_map и их методов – осмысленный выбор контейнера для решения задач, и важно знание и понимание, чем эти контейнеры отличаются друг от друга, и какой из них надо выбрать (стандартные контейнеры – это структуры данных, в которых заложены базовые алгоритмы сортировки, поиска);
* использование итераторов при обходе контейнеров, какие виды итераторов бывают, арифметика итераторов;
* применение алгоритмов STL: std::sort, std::copy, std::find, std::accumulate, std::transform;
* использование лямбда-выражений;
* применение умных указателей std::unique\_ptr, std::shared\_ptr (их различия);

Разобраться, чем плохо использовать глобальные переменные в программе.

Изучить виды оперативной памяти: статическая, стековая, динамическая. В каких случаях в программе задействуются эти виды памяти? Для чего нужны таблица виртуальных функций и виртуальный деструктор? Разобраться, в чем отличия оператора new (из C++) и функции std::malloc (из языка C).

Учиться устранять ошибки в программе самостоятельно.

Изучить контейнеры-адаптеры std::stack, std::queue, std::priority\_queue и желательно их применять.

**Этап 2 – Фреймворк Qt, правильное использование стандарта С++11, статические и динамические библиотеки, порождающие шаблоны проектирования.**

Изучить основы фреймворка Qt по книге Макса Шлее «Qt\_5\_10\_2018.pdf» (не обязательно читать всю книгу). Изучить по этой книге, как создается программа калькулятор с помощью сигналов и слотов. Как использовать новшества С++11 правильно – по книге Скотта Мейерса «Effective\_Modern\_C++\_2014\_RUS.pdf» (читать параллельно с написанием программы и использовать его рекомендации в своем проекте).

Начать создание собственного проекта с интерфейсом с использованием Qt и с данными, и также с загрузкой данных из файлов форматов JSON или XML. Изучить формат хранения и передачи данных JSON. Научиться структурировать данные в JSON-файлах и создавать классы загрузчиков данных из этих файлов. В Qt есть средства для работы с JSON, использовать их. Реализовать парсинг JSON-файла с данными. Далее эти данные передавать в интерфейс через архитектурный шаблон MVC.

Обязательно использовать в проекте класс QMainWindow, стандартные контейнеры STL, сигналы и слоты, объектные иерархии, архитектурный шаблон MVC, умные указатели (std::unique\_ptr, std::shared\_ptr), обработку событий, лямбда-выражения. Поработать с методами строки std::string (из STL) или QString (из Qt). Создать в проекте модель от класса QAbstractItemModel и использовать ее в классе QTreeView в соответствии с документацией на Qt, которая есть в интернете или в программе Qt Assistant.

По моему опыту на собеседованиях чаще спрашивают про стандартные средства разработки из STL, а не про их аналоги из Qt.

Научиться работать с отладчиком и средой разработки (MS Visual Studio или QtCreator).

Для использования вышеперечисленного изучить следующее.

Основы Qt:

* ограничения, накладываемые на объекты класса QObject;
* механизм сигналов и слотов;
* как создаются объектные иерархии;
* применение на практике архитектурного шаблона MVC (Model-View-Controller), в том числе создание собственной модели от QAbstractItemModel;
* обработка событий в программе (класс QEventLoop).

Умные указатели:

* изучить идиому RAII;
* можно сначала простые указатели пробовать в проекте, а потом заменять их на умные.

Учиться работать с документацией на фреймворк Qt, также изучать основы Qt по документации на Qt, не только по книгам.

Изучить, для чего нужны статические и динамические библиотеки, их различия (интернет и книга Макса Шлее).

Разобраться, что такое ошибки времени компиляции и времени выполнения.

Изучить для начала хотя бы 5 порождающих шаблонов объектно-ориентированного проектирования по классической книге «Приемы объектно-ориентированного проектирования. Паттерны проектирования» Э. Гамма, Р. Хелм [и др.]: абстрактная фабрика, синглтон, строитель, прототип, фабричный метод.

Дополнительно – хотя бы прочитать (в книге Макса Шлее про Qt это есть, и конечно есть в документации Qt):

* про загрузку данных из файлов XML с помощью класса QXmlStreamReader;
* тему исключений – класс std::exception в стандартной библиотеке С++;
* о протоколах TCP, UDP, и понять, чем они отличаются;
* изучить основы языка запросов SQL и баз данных (можно попрактиковаться с СУБД PostgreSQL);
* основы многопоточности, примитивы синхронизации (мьютексы) – для основательного изучения рекомендую книгу «C++ concurrency in action.djvu».

**Этап 3 – Базовые алгоритмы, система контроля версий Git и основы систем сборки qmake, cmake.**

Базовые алгоритмы – это алгоритмы сортировки и поиска, и связанные с ними структуры данных. Книга – «Sedzhvik.pdf».

Изучить алгоритмы сортировки:

* сортировка выбором;
* сортировка вставками;
* пузырьковая сортировка;
* быстрая сортировка;
* сортировка слиянием;
* пирамидальная сортировка.

Изучить структуры данных, используемые в сортировках (да и вообще везде):

* динамический массив;
* односвязный список;
* двусвязный список;

Изучить алгоритмы поиска:

* линейный поиск;
* бинарный поиск.

Изучить структуры данных для поиска:

* что такое бинарные деревья;
* сбалансированное бинарное дерево – например, красно-черное дерево, которое используется, как правило, в контейнерах std::set, std::map.
* хеш-таблица, которая используется в контейнерах std::unordered\_set, std::unordered\_map.

Также изучить структуры данных: стек, очередь.

Что такое асимптотическая сложность алгоритма. Необходимо знать асимптотическую сложность вышеперечисленных алгоритмов, и алгоритмов, заложенных в структуры данных.

Для закрепления алгоритмов можно решать задачи на сайте LeetCode (интернет), начиная с уровня сложности easy (например, в Яндекс без алгоритмических навыков не попасть).

**Git и GitHub.** Сначала прочитать теорию Git (книга «Git.pdf» – главы 1,2,3) – его основы, чем он отличается от других систем контроля версий. Далее использовать на практике в проекте следующие базовые команды (ниже) и разместить свой проект на GitHub.

Применяемые на практике команды (разумеется, разобраться для чего они нужны):

git init

git clone

git status

git commit

git checkout

git reset

git revert

git fetch

git merge

git pull

git push

git rebase

Установить, кроме самого Git (с командной строкой), также TortoiseGit (с интерфейсом). Для лучшего понимания работы с Git пользоваться также параллельно TortoiseGit. Обязательно научиться разрешать конфликты слияния – нагляднее это делать с помощью TortoiseGit.

Изучить, для чего нужны системы сборки qmake, cmake, пользоваться придется ими с самого начала создания проекта. CMake более актуальна, поэтому по возможности лучше с ней работать, однако с qmake не так сложно, поскольку она заточена под Qt. СМаke можно использовать как со средой разработки MS Visual Studio, так и с QtCreator.

**Этап 4 – Шаблоны проектирования и SOLID.**

Прочитать книги:

1. Приемы объектно-ориентированного проектирования. Паттерны проектирования. Э. Гамма, Р. Хелм [и др.]
2. Чистая архитектура. Роберт С. Мартин.

Использовать шаблоны и принципы SOLID на практике.