**Список вопросов к коллоквиуму.**

Программирование. Пользовательское программирование. Системное программирование. Технология программирования. Теоретическое программирование. Внешние критерии качества ПО. Внутренние критерии качества ПО. Парадигма программирования. Процедурная декомпозиция. Объектно-ориентированная декомпозиция.

Понятие алгоритма. Общее представление работы программы. Соотношение компонентов программ и компьютерной архитектуры. Что такое поток и процесс.

Этапы разработки программы. Алгоритм. Структура программы. Комментарии. Состав языка. Лексемы. Символы. Выражение. Оператор. Ключевые слова. Константы.

Схема сборки проекта на С++. Препроцессинг. Компиляция. Ассемблерный код. Ассемблирование. Объектный файл. Линковка. Таблица символов.

Динамическая типизация, статическая типизация, интерпретатор, компилятор, семантика, рефакторинг, Debug, псевдокод, транслятор, машинный код, сопроцессор, реверс-инжиниринг, дизассемблирование, ассемблерные вставки, дизассемблирование, трансляция, регистры в asm, эзотерический язык, функциональные языки, логические языки, идентификатор, кодировки юникод, ASCII, utf-8, языки низкого уровня, среднего уровня, языки высокого уровня, компилируемые языки, интерпретируемые языки, нативные языки, аллокация. Императивные и декларативные языки программирования.

Мультипарадигмальный язык, репозиторий, индекс tiobe, высокий уровень безопасности, открытость исходного кода, байт-код, распределенное программирование, автоматическое управление памятью, кроссплатформенность

Апплеты, фреймворк.

Концепция типа данных. Переменная. Тип данных. Диапазон типов по стандарту. Знаковые типы. Беззнаковые типы. Типы фиксированной длины. Тип представления размера. Правила формирования общего типа. Составные типы данных. Структура. Объединение. Перечисление. Битовые поля. Ссылки. Указатели. Понятие, история и характеристика STL. Ассоциативные контейнеры. Последовательные контейнеры.

Стандартные функции ввода вывода. Описание идентификатора. Область видимости. Класс памяти auto. Класс памяти extern. Класс памяти static. Класс памяти register. Пространство имен. Классы идентификаторов. Основные правила использования переменных. Конвенции именования С++. Основные операции С++. Поразрядные операции. Операции инкремента и декремента. Безопасное преобразование типов. Опасное преобразование типов. Форматирующие флаги и методы. Тернарная операция. Условный оператор. Обработка запросов по множеству enum. Оператор Switch.

Оператор цикла: тело цикла, параметр цикла, условие цикла, итерация, счетчик цикла. Условие цикла. Цикл с предусловием. Цикл с постусловием. Цикл с параметром (формат и определение каждого элемента условия). Правила использования циклов. 5 операторов передачи управления в цикле.

Определение функции. Типичный механизм возврата значений. Объявление функций (виды). Возврат нескольких значений из функций. Метод передачи параметров функции по значению: плюсы , минусы, технология. Метод передачи параметров функции по ссылке: плюсы , минусы, технология.Метод передачи параметров функции по константной ссылке: плюсы , минусы, технология.Метод передачи параметров функции по указателю: плюсы , минусы, технология. Стек. Куча. Принцип выбора метода передачи параметров функции. Возможности функций: перегрузка функций. Рекурсия. Правила использования рекурсии. Шаблон функций. Лямбда функция (пример, алгоритмы, итераторы).

Перегрузка операторов. Перегрузка оператора ввода, вывода: особенности, примеры. Перегрузка арифметических операторов: особенности, примеры.

Директивы. Макросы. Препроцессор. Макро-константы. Макрофункции (многострочные). Predefined defines. Минусы использования макросов.

Массив. Статический массив: определение, использование. Виды перебора элементов массива. Обработка элементов массива по два элемента.

Многомерные массивы. Хранение, описание, доступ к элементам. Пример инициализации. Передача параметров одномерного массива (виды). Передача параметров многомерного массива (виды).

Понятие указатели и динамическая память. Схема работы стеккуча при указателях. Оператор new. Инициализация объектов в куче. Контейнеры (динамика). Доступ к элементу в куче. Указатель на функцию. Указатель на объект. Указатель на void. Инициализация указателей (4 вида). Освобождение памяти. Операции с указателями. Арифметические операции с указателями. Ссылки. Взаимосвязь ссылки и указателя. Указатели и массивы. Динамические одномерные массивы. Динамические двумерные массивы: порядок создания и освобождения памяти. Контейнер vector. назначение, инициализация, принцип работы.Понятие указатели и динамическая память. Схема работы стек-куча при указателях. Оператор new. Инициализация объектов в куче.

Контейнеры (динамика). Доступ к элементу в куче. Указатель на функцию. Указатель на объект. Указатель на void. Инициализация указателей (4 вида). Освобождение памяти. Операции с указателями. Арифметические операции с указателями. Ссылки. Взаимосвязь ссылки и указателя. Указатели и массивы. Динамические одномерные массивы. Динамические двумерные массивы: порядок создания и освобождения памяти.

Итераторы. Концепция полуинтервалов. Арифметические операции с итераторами. Шаблонная функция вывода контейнера по двум итераторам. Алгоритмы и итераторы (5 примеров, любых).

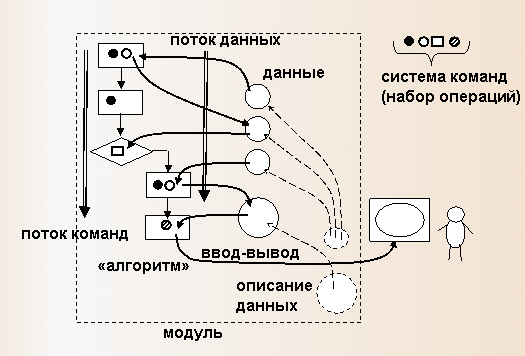
Исключения: механизм, примеры. Типы исключений.

Алгебра (группа, кольцо, ассоциативное кольцо). Отношение сравнения (свойства). Криптография. Стеганография. Классическая задача передачи сообщений от некоторого отправителя А к получателю В. Классическая система секретной связи. Односторонняя функция. Дискретных логарифм. Первая система с открытым ключом Криптопротокол Диффи-Хеллмана. Криптопротокол Шамира. Криптопротокол Эль-Гамаля. Криптопротокол RSA. Простое число. Распределение простых чисел (Эратосфен). Основная теорема арифметики. Функция Эйлера, основные свойства. Каноническое разложение числа. Функция Эйлера. Теорема Эйлера. Теорема Ферма. Наибольший общий делитель – алгоритм Евклида. Расширенный алгоритм Евклида. Обобщенный алгоритм Евклида. Алгоритм возведения в степень по модулю.

**Понятие алгоритма**

* *Алгоритм* — это конечный набор правил, который определяет последовательность операций для решения конкретного множества задач и обладает пятью важными чертами: конечность, определённость, ввод, вывод, эффективность (*Д. Э. Кнут*).
* *Алгоритм* — это всякая система вычислений, выполняемых по строго определённым правилам, которая после какого-либо числа шагов заведомо приводит к решению поставленной задачи (*А. Н. Колмогоров*).
* *Алгоритм* — это последовательность действий, либо приводящая к решению задачи, либо поясняющая, почему это решение получить нельзя.

**Общее представление работы программы**



прежде всего, программа работает не с пользователем, а с данными. Эта первая и основная компонента программы – предметы (объекты), над которыми реализуется алгоритм. Данные состоят из отдельных переменных, связанных как между собой непосредственно (через указатели), так и косвенно (как входные данные – результат);

В языке программирования имеются средства описания данных, которые позволяют программисту конструировать различные формы их представления – типы данных;

программа базируется на наборе операций (системе команд), которые можно выполнять над данными. В этот набор входят арифметические операции, присваивание (сохранение результата в переменной), ввод-вывод, проверка значения переменной и т.п.;

вторая основная компонента программы – описание порядка, последовательности выполняемых действий, также называется алгоритмом «в узком смысле», или алгоритмической компонентой. Она обычно состоит из двух частей. Первая часть – выражения, представляет собой описание линейной последовательности выполнения простейших действий из набора операций (арифметические операции, присваивание, условные выражения). Они включаются во вторую компоненту – операторы, которые задают ту или иную последовательность действий;

**Соотношение компонентов программ и компьютерной архитектуры**

компоненты программы находятся в памяти. В принципе, память является общей для них всех, но логически она разделяется на области, именуемые сегментами. Прежде всего, это сегмент данных, содержащий, естественно, данные программы. Алгоритмическая компонента (выражения, операторы) также находится в памяти в собственном сегменте команд;

одновременное нахождение в памяти «алгоритма» и данных соответствует принципу хранимой программы. Перед загрузкой в память эти же компоненты находятся в программной файле, который представляет собой точную копию  представления программы в памяти – «образ памяти». Это позволяет рассматривать всю программу (в том числе и алгоритм) как данные для работы других программ, например, трансляторов;

набор операций, выполняемый в программе, соответствует системе команд процессора, на котором она выполняется. Сюда же входят команды, которые обеспечивают заданный в программе порядок действий (операторов).

И, наконец, язык программирования также содержит в себе компоненты, предназначенные для описания соответствующих частей программы:

средства описания данных: определение типов данных (форма представления) и переменных;

набор операций над основными типами данных (включая ввод-вывод), а также средства записи выражений;

набор операторов, определяющих различные варианты порядка выполнения выражений в программе (последовательность, условие, повторение, блок);

средства разбиения программы на независимые части – модули (функции, процедуры), взаимодействующие между собой через программные интерфейсы.

***Процесс компиляции программ***

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

**Что такое поток и процесс**

Процессы и потоки связаны друг с другом, но при этом имеют существенные различия.

Процесс — экземпляр программы во время выполнения, независимый объект, которому выделены системные ресурсы (например, процессорное время и память). Каждый процесс выполняется в отдельном адресном пространстве: один процесс не может получить доступ к переменным и структурам данных другого. Если процесс хочет получить доступ к чужим ресурсам, необходимо использовать межпроцессное взаимодействие. Это могут быть конвейеры, файлы, каналы связи между компьютерами и многое другое.

Поток использует то же самое пространства стека, что и процесс, а множество потоков совместно используют данные своих состояний. Как правило, каждый поток может работать (читать и писать) с одной и той же областью памяти, в отличие от процессов, которые не могут просто так получить доступ к памяти другого процесса. У каждого потока есть собственные регистры и собственный стек, но другие потоки могут их использовать.

Поток — определенный способ выполнения процесса. Когда один поток изменяет ресурс процесса, это изменение сразу же становится видно другим потокам этого процесса.