Билет будет состоять из трех *вопросов*:

1. вопрос по теории
2. вопрос по языку
3. задача

Далее приведены возможные варианты каждого из трех вопросов. Билеты будут компоноваться из приведенных ниже вариантов вопросов (возможна косметическая правка).

Вопрос №1 (теория)

1. Представление чисел в ЭВМ. Стандартные представления (целое без знака, целое со знаком, Currency, вещественное с плавающей точкой). IEEE стандарт представления вещественных чисел с плавающей точкой без свойств..
2. Машинное эпсилон. Точность представления вещественного числа. Какие числа можно точно представить в виде вещественного числа с плавающей точкой. Нестандартные представления чисел в ЭВМ. BCD. NUMBER из Oracle. Абсолютные и относительные ошибки.
3. Понятие алгоритма. Время работы алгоритма. Сведение задач и алгоритмов. Определения верхних и нижних оценок времени работы алгоритма. Определения верхних и нижних оценок времени решения задачи.
4. Теоремы о верхних и нижних оценках при сведении задач.
5. Теорема о нижней оценке времени решения задачи сортировки на основе дерева решений.
6. Сортировка пузырьком. Случай оптимальности алгоритма слияния пузырьком. Сортировка слиянием с рекурсией и без рекурсии.
7. Сортировка алгоритмом QuickSort. Теорема о среднем времени работы алгоритма QuickSort.
8. Сортировка алгоритмом HeapSort или сортировка с помощью пирамиды.
9. Поиск порядковой статистики в общем случае с верхней оценкой времени работы алгоритма $\Theta(n)$.
10. Поиск порядковой статистики за линейное время в среднем. Поиск порядковой статистики последовательности целых чисел a_i $(i=1,\ldots,n)$ за линейное время для случая $a_i<O(n)$. Задача поиска порядковой статистики в большой окрестности каждой точки серого изображения.
11. Алгоритмы сортировки за время O(N). Сортировка подсчетом. Цифровая сортировка.
12. Структуры данных. Определения. Различные реализации. Вектор. Стек.
13. Структуры данных. Определения. Различные реализации. Очередь. Дек.
14. Структуры данных. Определения. Различные реализации. L1- и L2-списки.
15. Сортировка пузырьком. Случай оптимальности алгоритма слияния пузырьком. Сортировка слиянием с рекурсией и без рекурсии..
16. Бинарные деревья поиска. Поиск, добавление, удаление элемента.
17. Бинарные деревья поиска. Поиск минимального и максимального элемента в дереве. Поиск следующего/предыдущего элемента в дереве.
18. Сбалансированные, идеально сбалансированные и идеально сбалансированные' деревья. Определения. Взаимосвязи данных понятий. Высота сбалансированного и идеально сбалансированного бинарного деревьев в зависимости от количества элементов в дереве.
19. Сбалансированные бинарные деревья поиска. Поиск, добавление, удаление элемента.
20. Сбалансированные бинарные деревья поиска. Поиск минимального и максимального элемента в дереве. Поиск следующего/предыдущего элемента в дереве.

21. Сбалансированные бинарные деревья поиска. Слияние двух деревьев. Разбиение дерева по разбивающему элементу.
22. Красно-черные деревья. Определение. Красно-черное’ дерево. Оценка высоты красно-черного дерева через количество элементов в дереве. Добавление элемента в красно-черное дерево за два прохода (поиск с проходом сверху вниз + добавление с проходом снизу вверх).
23. Красно-черные деревья. Удаление элемента из красно-черного дерева. Добавление элемента в красно-черное дерево за один проход (с модификацией дерева при поиске элемента, после которого вставляется новый элемент).
24. Графы. Определение. Ориентированные и неориентированные графы. Инцидентность/смежность вершин/ребер. Нахождение пути в графе от вершины к вершине с наименьшим количеством ребер (алгоритм волны). Теорема о времени работы алгоритма волны для случая графа без кратных ребер и петель.
25. Формула Эйлера (с доказательством). Оценка на количество ребер и граней графа через количество вершин для случая планарного графа без кратных ребер и петель.
26. Нахождение кратчайшего пути в графе. Алгоритм Дейкстры. Простой и модифицированный алгоритм Дейкстры. Теорема о времени работы простого и модифицированного алгоритмов Дейкстры.
27. Алгоритма Дейкстры на основе STL (два варианта алгоритма). Потенциальные проблемы при использовании первого алгоритма. Теорема о времени работы алгоритма Дейкстры на основе STL.
28. Хеширование. Метод многих списков. Оценка среднего времени выполнения операций поиска, добавления, удаления элемента.
29. Хеширование. Метод линейных проб. Оценка среднего времени выполнения неудачной операции поиска (=добавления элемента≥удаления элемента).
30. Хеширование. Метод линейных проб. Оценка среднего времени выполнения удачной операции поиска. Алгоритмы поиска свободного места: линейное и квадратичное пробирование (с доказательством корректности), пробирование с помощью двойного хеширования.
31. Хеш-функции на основе деления и на основе умножения. CRC-алгоритмы.
32. Алгоритмы динамического выделения памяти. Использование стека задачи. Списки блоков фиксированного размера. Алгоритм близнецов (для блоков размером 2k). Списки блоков свободной памяти в общем случае. Модифицированные списки блоков свободной памяти в общем случае (алгоритм парных меток).
33. Базовые принципы функционирования вычислительных систем. Общая конфигурация вычислительной системы. Системные шины на IBM PC – совместимых компьютерах. Bus mastering (Управление шиной). AGP. Чем AGP отличается от PCI?. Прерывания.
34. Кэш-память. Организация Кеш-памяти и ассоциативная память в IBM PC-совместимых ЭВМ.
35. Режимы адресации памяти в Intel-процессорах. Реальный режим. Защищенный режим. Страничная организация памяти в защищенном режиме.
36. Операционные системы. Эволюция операционных систем. Последовательная обработка данных. Простые пакетные системы. Многозадачные пакетные системы. Системы, работающие в режиме разделения времени. Системы реального времени. Многопоточность. Потоки и процессы.
37. Основные принципы параллельных вычислений. Конкуренция процессов в борьбе за ресурсы. Сотрудничество с использованием разделения. Сотрудничество с использованием связи. Программная реализация. Алгоритм Деккера. Взаимоисключения. Алгоритм Петерсона. Отключение прерываний. Использование машинных команд.

Семафоры. Задача о производителях и потребителях.

Вопрос №2 (по языку)

1. Общая структура С-программы.
2. Язык С. Типы базовых переменных. Их описание, определение, инициализация.
3. Язык С. Константы (целые, вещественные, строковые, символьные).
4. Язык С. Структуры и объединения. Описания и определения.
5. Язык С. Арифметические операции `+’, `-’, `*’, `/’, `+=’, `-=’, `*=’, `/=’, `++’, `--’.
6. Язык С. Логические операции `&&’, `||’, `!’.
7. Язык С. Указатели. Указатели на функции.
8. Язык С. Область видимости переменных и время жизни переменных.
9. Язык С. Описание и определение функций.
10. Язык С. Операторы цикла.
11. Язык С. Условные операторы. Оператор перехода.
12. Строки в языке С. Функции работы со строками.
13. Язык С. Потоковый ввод/вывод для текстовых файлов. Открытие файла, ввод/вывод с помощью функций fprintf/fscanf/fgets/fputs, закрытие файла.
14. Язык С. Потоковый ввод/вывод для бинарных файлов. Открытие файла, ввод/вывод с помощью функций fread/fwrite, закрытие файла.
15. Язык С. Отведение/очистка памяти с помощью функций malloc/realloc/free.
16. Язык С++. Конструкторы, деструкторы. Отведение памяти. Ввод/вывод с клавиатуры/на экран (cin/cout) и в файл (fstream). Форматированный вывод.
17. Язык С++. Инкапсуляция. Полиморфизм. Создание простых классов. Выделить, какие методы для простых классов можно не создавать. Все необходимое для понимания фразы *a=b+c* для простых классов.
18. Язык С++. Инкапсуляция. Полиморфизм. Создание сложных классов. Идеология *SetZero/Clean/CopyOnly.* Все необходимое для понимания фразы *a=b+c* для сложных классов.
19. Язык С++. Переопределение операторов, в том числе, переопределение операторов преобразования типа.
20. Язык С++. Дружественные функции/классы. С++03 (старый стандарт), С++11 (новый стандарт). Move-constructors.
21. STL. Последовательные контейнеры. Итераторы для них.
22. STL. Ассоциативные контейнеры. Итераторы для них.
23. Python. Арифметические и логические операции.
24. Python. Циклы. Условные операторы.
25. Python. Функции. Модули. Стандартные модули. Создание и использование собственных модулей.
26. Python. Списки. Кортежи. Работа со списками и кортежами. Основные методы списков и кортежей.
27. Python. Словари. Работа со словарями. Основные методы словарей.
28. Python. Строки. Работа со строками. Основные методы строк. Ввод строк (в том числе слов) из файла.
29. Python. Исключения.

Задачи

1. Написать программу на языке С, которая в изображении, заданном в файле */users/staff/staroverov/lectures/images/1.bmp* (изображение с палитрой) заменяет все

пиксели с красным цветом, имеющие синих соседей, на пиксели с зеленым цветом. Предполагается, что указанные цвета в палитре есть. Результат вывести в файл *./1.bmp*. Заголовок BMP-файла можно взять из файла */users/staff/staroverov/lectures/images/header/bmp.h* .

2. Написать программу на языке C, которая транспонирует матрицу изображения, заданного в файле */users/staff/staroverov/lectures/images/1.bmp* (изображение с палитрой). Результат вывести в файл *./1.bmp*. Заголовок BMP-файла можно взять из файла */users/staff/staroverov/lectures/images/header/bmp.h* .
3. Написать программу на языке C, которая транспонирует матрицу изображения, заданного в файле */users/staff/staroverov/lectures/images/32.bmp* (изображение в формате True Color). Результат вывести в файл *./32.bmp*. Заголовок BMP-файла можно взять из файла */users/staff/staroverov/lectures/images/header/bmp.h* .
4. Написать программу на языке C, которая погружает изображения, заданное в файле */users/staff/staroverov/lectures/images/1.bmp* (изображение с палитрой), в черную рамку шириной 2 пикселя. При этом ширина и высота изображения должны увеличиться, соответственно, на 4 пикселя. Результат вывести в файл *./1.bmp*. Заголовок BMP-файла можно взять из файла */users/staff/staroverov/lectures/images/header/bmp.h* .
5. Написать программу на языке C, которая погружает изображения, заданное в файле */users/staff/staroverov/lectures/images/32.bmp* (изображение в формате True Color), в черную рамку шириной 2 пикселя. При этом ширина и высота изображения должны увеличиться, соответственно, на 4 пикселя. Результат вывести в файл *./32.bmp*. Заголовок BMP-файла можно взять из файла */users/staff/staroverov/lectures/images/header/bmp.h* .
6. Написать программу на языке C, которая вводит все слова, разделенные пробелами, табуляциями и переходами на следующую строчку, из файла */users/staff/staroverov/lectures/images/t0.txt* в массив строк (строки длиной не более 256 символов), лексикографически сортирует массив строк методом быстрой сортировки делением пополам с рекурсией и выводит полученный массив в файл *./t0.txt* по одной строке в строку файла.
7. Написать программу на языке C, которая вводит все слова, разделенные пробелами, табуляциями и переходами на следующую строчку, из файла */users/staff/staroverov/lectures/images/t0.txt* в массив строк (строки длиной не более 256 символов), лексикографически сортирует массив строк методом быстрой сортировки делением пополам без рекурсии и выводит полученный массив в файл *./t0.txt* по одной строке в строку файла.
8. Написать программу на языке C, которая вводит все слова, разделенные пробелами, табуляциями и переходами на следующую строчку, из файла */users/staff/staroverov/lectures/images/t0.txt* в массив строк (строки длиной не более 256 символов), лексикографически сортирует массив строк методом быстрой сортировки QSort и выводит полученный массив в файл *./t0.txt* по одной строке в строку файла. Можно использовать любую собственную реализацию алгоритма QSort.
9. Написать программу на языке C, которая вводит все слова, разделенные пробелами, табуляциями и переходами на следующую строчку, из файла */users/staff/staroverov/lectures/images/t0.txt* в массив строк (строки длиной не более 256 символов), лексикографически сортирует массив строк с помощью встроенной функции qsort и выводит полученный массив в файл *./t0.txt* по одной строке в строку файла.
10. Написать программу на языке C, которая вводит все слова, разделенные пробелами, табуляциями и переходами на следующую строчку, из файла */users/staff/staroverov/lectures/images/t1.txt* в массив строк (строки длиной не более 10 символов и состоят только из цифр), лексикографически сортирует массив строк с

помощью цифровой сортировки с использованием сортировки подсчетом и выводит полученный массив в файл *./t1.txt* по одной строке в строку файла.

11. Написать функцию
    float Deg2N(float x,int n);
    умножающую число *x* типа *float* на множитель 2^n , использующую только сложение степени в представлении числа с числом *n*.
12. Написать функцию
    double Deg2N(double x,int n);
    умножающую число *x* типа *double* на множитель 2^n , использующую только сложение степени в представлении числа с числом *n*.
13. Написать программу на языке C, которая вводит все слова, разделенные пробелами, табуляциями и переходами на следующую строчку, из файла */users/staff/staroverov/lectures/images/t0.txt* в массив строк (строки длиной не более 256 символов), и находит k-ую порядковую статистику для данного массива (в смысле лексикографического сравнения) алгоритмом, обеспечивающим поиск порядковой статистики за линейное время в среднем. Полученное слово вывести на экран.
14. На языке C++ написать собственную реализацию стека строк (строки длиной не более 256 символов) неограниченной длины на базе вектора (с реаллокацией памяти в случае заполнения стека). Заполнить стек словами из файла */users/staff/staroverov/lectures/images/t0.txt* (слова разделяются пробелами, табуляциями и переходами на следующую строчку). Вывести все элементы стека в файл *./t0.txt* по одному слову в строке.
15. На языке C++ написать собственную реализацию очереди строк (строки длиной не более 256 символов, количество элементов в очереди не более 20). Последовательно заполнять очередь словами из файла */users/staff/staroverov/lectures/images/t0.txt* (слова разделяются пробелами, табуляциями и переходами на следующую строчку) и при заполнении очереди выводить все элементы очереди в файл *./t0.txt* по одному слову в строке (таким образом, все слова из исходного файла должны быть выведены в выходной файл).
16. На языке C++ написать собственную реализацию однонаправленного списка строк с фиктивным элементом (строки длиной не более 256 символов). Последовательно заполнить список словами из файла */users/staff/staroverov/lectures/images/t0.txt* (слова разделяются пробелами, табуляциями и переходами на следующую строчку). После считывания всех строк вывести все элементы списка в файл *./t0.txt* по одному слову в строке (таким образом, все слова из исходного файла должны быть выведены в выходной файл).
17. На языке C++ написать собственную ссылочную реализацию двунаправленного списка строк (строки длиной не более 256 символов). Последовательно заполнить список словами из файла */users/staff/staroverov/lectures/images/t0.txt* (слова разделяются пробелами, табуляциями и переходами на следующую строчку). После считывания всех строк вывести в обратном порядке все элементы списка в файл *./t0.txt* по одному слову в строке (таким образом, все слова из исходного файла должны быть выведены в выходной файл, но в обратном порядке).
18. На основе STL создать вектор из *K* списков целых чисел. Заполнить списки числами, записанными в файле */users/staff/staroverov/lectures/images/t2.txt* таким образом, чтобы число, остаток от деления которого на *K* был бы равен *i*, заносился бы в список с индексом *i*. Число K должно вводиться с клавиатуры. Содержимое каждого списка должно быть выведено в отдельную строку файла *./t2.txt* (через пробел). При выводе перебирать элементы вектора и списков надо через соответствующие итераторы.
19. Написать собственную реализацию двунаправленного списка строк (строки длиной не более 256 символов) на основе двух стеков, реализованных с помощью STL.

Последовательно заполнить список словами из файла */users/staff/staroverov/lectures/images/t0.txt* (слова разделяются пробелами, табуляциями и переходами на следующую строчку). После считывания всех строк вывести в обратном порядке все элементы списка в файл *./t0.txt* по одному слову в строке (таким образом, все слова из исходного файла должны быть выведены в выходной файл, но в обратном порядке).

20. В файле */users/staff/staroverov/lectures/images/v.txt* заданы координаты вершин графа по паре координат в каждой строке. В каждой строке файла */users/staff/staroverov/lectures/images/e.txt* заданы номера вершин очередного ребра графа и длина данного ребра. С помощью STL реализовать алгоритм Дейкстры (модификацию алгоритма Дейкстры для STL), находящий кратчайший путь в графе (вершины и ребра графа можно загружать из указанных файлов) от первой до последней вершины из списка. Список вершин найденного пути и длину пути вывести в файл *./rez.txt* .
21. Написать с использованием STL реализацию хеш-таблицы на основе метода многих списков множества строк длины не более 256 (при этом придется написать свою реализацию простого (!) класса строк длины не более 256). Хеш-функцию реализовать произвольным образом. Заполнить таблицу словами из файла */users/staff/staroverov/lectures/images/t0.txt* (слова разделяются пробелами, табуляциями и переходами на следующую строчку). Для каждого значения *i* хеш-функции вывести все слова из соответствующего списка в файл *i.txt* по одному слову в строке (в имени файла *i* – текстовое представление соответствующего числа).
22. Написать с использованием STL реализацию хеш-таблицы на основе метода линейных проб множества строк длины не более 256 (от STL здесь используется реализация вектора и итераторов; при этом придется написать свою реализацию простого (!) класса строк длины не более 256). Хеш-функцию реализовать произвольным образом. Заполнить таблицу словами из файла */users/staff/staroverov/lectures/images/t0.txt* (слова разделяются пробелами, табуляциями и переходами на следующую строчку; длину таблицы задать как двойное количество слов во входном файле). Для каждого значения *i* хеш-функции вывести все слова из соответствующего списка в файл *i.txt* по одному слову в строке (в имени файла *i* – текстовое представление соответствующего числа).
23. Написать на языке Python реализацию класса, задающего дробное число. С помощью элементов данного типа написать функцию, считающую в лоб сумму *1/(1*2)+1/(2*3)+1/(3*4)+…+1/((n-1)*(n))*, где n является параметром функции.
24. Написать программу на языке Python, которая вводит все слова, разделенные пробелами, табуляциями и переходами на следующую строчку, из файла */users/staff/staroverov/lectures/images/t0.txt* в список строк, лексикографически сортирует список стандартной функцией и выводит полученный массив в файл *./t0.txt* по одной строке в строку файла.
25. Написать на языке Python функцию, реализующую метод Гаусса решения системы линейных уравнений. Матрицу задавать как словарь, индексированный кортежами. Протестировать написанную функцию.
26. Написать на языке C++ функцию, реализующую метод Гаусса решения системы линейных уравнений. Матрицу задавать как указатель на указатель. Размерность матрицы передавать через параметры вместе с самой матрицей и правой частью уравнения. Ввести матрицу в указанном виде из файла, где она записана естественным образом (по одной строке матрицы в строке файла без отдельного указания размера матрицы). Пусть правая часть СЛУ задается в том же файле дополнительным столбцом данной матрицы. Решить СЛУ созданной функцией. Результат вывести на экран.
27. Реализовать на языке C хеш-функцию на основе деления. Создать массив целых чисел, в котором в элементе с индексом *i* будет храниться количество слов из файла

/users/staff/staroverov/lectures/images/t0.txt (слова длиной не более 256 символов, разделяющиеся пробелами, табуляциями и переходами на следующую строчку), которым соответствует значение хеш-функции = *i*. Вывести на экран значения данного массива, среднеквадратичное отклонение значений данного массива и нормированное среднеквадратичное отклонение значений данного массива (D/M).

28. Реализовать на языке С хеш-функцию на основе умножения. Создать массив целых чисел, в котором в элементе с индексом *i* будет храниться количество слов из файла */users/staff/staroverov/lectures/images/t0.txt* (слова длиной не более 256 символов, разделяющиеся пробелами, табуляциями и переходами на следующую строчку), которым соответствует значение хеш-функции = *i*. Вывести на экран значения данного массива, среднеквадратичное отклонение значений данного массива и нормированное среднеквадратичное отклонение значений данного массива (D/M).
29. Реализовать функцию расчета CRC-значения для строки, дополненной двумя нулевыми байтами. Занести в два последних нулевых байта найденное CRC-значение и проверить, что у полученной строки CRC-значение =0 .