Экзамен по предмету "Информатика" 2 семестр 2016-2017 уч.г.

1 Общие соображения для получения хорошей оценки.

Экзамен проводится в письменной форме. Все решения обязательно оформлять как пользовательские функции, с явным указанием формальных параметров (входа/выхода). Выделение подзадач с использованием вспомогательных функций обязательно.

Ниже приведены:

1. Программа курса.
2. Список тем для практических задач.
3. Примерный вариант экзаменационного билета. Для каждого вопроса указан список тем, знание которых необходимо для решения соответствующей задачи.

2 Программа курса

1. Текстовые файлы. Форматированный и неформатированный ввод-вывод при обработке текстовых файлов.
2. Структуры. Работа со структурами. Вложенные структуры.
3. Бинарные файлы. Неформатированный ввод-вывод. Обработка бинарных файлов, содержащих разнотиповую информацию.
4. Обработка бинарных файлов из структур.
5. Обработка упорядоченных и неупорядоченных файлов.
6. Метод состояний для задач обработки упорядоченных файлов из структур.
7. Метод диаграмм для задач обработки упорядоченных файлов из структур.
8. Динамические и статические переменные, их разница.
9. Тип указатель. Операции над указателями. Выделение и освобождение памяти, операция разадресации. Пустой указатель.

10. Динамические и статические массивы. Их сравнение. Работа с массивами в терминах индексов и в терминах указателей.

11. Выделение и освобождение памяти под динамический массив. Работа с динамическими массивами.

12. Двумерные динамические массивы.

13. Указатели на функции. Передача функций в качестве параметров в функцию.

14. Тип ссылка.

15. Рекурсивные функции. Оформление и исполнение рекурсивных функций. Нисходящая и восходящая ветви рекурсии. Задача "Ханойские башни" и рекурсивный алгоритм ее решения. Пример необоснованного выбора реккурсии как способа реализации вычисления последовательности Фибоначи в соответствии с ее определением. Возможные ошибки при использовании рекурсивных функций.

16. Итерация и рекурсия, их сравнение. Элиминация рекурсии.

17. Алгоритм Merge_Sort.

18. Односвязные линейные списки. Основные операции работы со списками: вставка, удаление элемента, переход к следующему элементу, доступ к элементу, проверка на конец списка.

19. Реализация односвязных линейных списков при помощи массивов.

20. Рекурсивное определение линейного списка. Реализация рекурсивных функций обработки линейного списка, основанных на рекурсивном определении списка.

21. Разновидности линейных списков: кольцевой список, двусвязный список.

22. Стек. Основные операции работы со стеком: создание стека, добавление, удаление элемента, проверка на пустоту. Две реализации стека: на основе массива и на основе линейного односвязного списка.

23. Очередь. Основные операции работы с очередью: создание очереди, добавление, удаление элемента, проверка на пустоту. Две реализации очереди: на основе массива и на основе линейного односвязного списка.

24. Стековые алгоритмы.

25. Линейные списки как структуры данных. Представление полинома в виде списка. Основные операции, их реализация и сложность.

26. Структуры данных поиска. Хэш-таблица. Различные варианты Хэш-таблиц: с прямой адресацией и с использованием хэш-функции. Проблема коллизии и способы ее разрешения. Реализация и сложность основных операций: найти, добавить, удалить.

27. Бинарные деревья. Основные понятия: лист, корень, сын, отец, поддерево, путь, уровень, высота, ширина дерева. Программная реализация двоичного дерева.

28. Операции над двоичными деревьями: добавить лист, удалить лист, переход по дереву.

29. Реализация двоичного дерева при помощи массивов.

30. Обход дерева в глубину. Различные виды обхода в глубину. Реализация обхода в глубину при помощи стека.

31. Обход дерева в ширину. Реализация обхода в ширину при помощи очереди.

32. Двоичное дерево поиска. Основные операции: поиск элемента, вставка элемента, удаление элемента, поиск элемента.

33. Рекурсивное определение двоичного дерева. Реализация рекурсивных функций обработки двоичного дерева, основанных на его рекурсивном определении.

34. Структуры данных поиска. Приоритетные очереди и их реализация на основе массива. Реализация и сложность основных операций: Найти минимальный элемент, добавить, удалить.

35. Двоичная куча как реализация приоритетной очереди. Основные операции, их реализация и сложность. Построение двоичной кучи за линейное время.

36. Алгоритм Heap_Sort.

37. Двоичные деревья как структуры данных. Дерево арифметического выражения. Основные операции, их реализация и сложность.

3 Список тем для экзаменационных практических задач

1. Обработка текстовых файлов с использование как форматированного, так и неформатированного ввода. Правильный выбор наиболее подходящей команды ввода для каждого конкретного случая. Функции >> (форматированный ввод), get (для ввода символа), >> (для ввода слова), getline (для ввода строки).

2. Работа со структурами. Структуры с полями – массивами. Массивы из структур. Передача структур в качестве параметров в функции. Описание вложенных структур, доступ к полям.

3. Бинарные (двоичные) файлы. Обработка бинарных файлов, содержащих как однотипную, так разнотипную информацию, с использованием неформатированного ввода.

4. Обработка бинарных файлов из структур.

5. Метод состояний в задачах обработки бинарных упорядоченных файлов из структур.

6. Метод диаграмм в задачах обработки бинарных упорядоченных файлов из структур.

7. Динамические одномерные и двумерные массивы: создание, обработка в терминах указателей, удаление.

8. Обработка динамических массивов строк.

9. Указатели на функции. Передача функций в качестве параметров в функцию.

10. Обработка односвязных линейных списков. Задачи, которые раньше решались для типов массив или файл, теперь - для списков. Например, поиск элемента, проверка упорядоченности, удаление вхождения одного списка в другой, сортировка списка методом вставки, вычисление предикатов $\forall$ и $\exists$ (простых и вложенных), удаление повторных вхождений элементов,копирование списка, проверка на равенство двух списков,....

11. Обработка упорядоченных последовательностей, заданных в виде списков (объединение, пересечение, разность, поиск, вставка).

12. Представление полинома в виде списка. Операции над полиномами, заданными в виде списка (сложение, умножение, дифференцирование, взятие первообразной, приведение подобных, вычисление в точке).

13. Стек, очередь. Основные операции.

14. Бинарное (двоичное) дерево. Обход в глубину с использованием стека. Задачи на обработку дерева с использованием обхода в глубину. Например: подсчет глубины дерева, поиск элемента, распечатать листья дерева, переставить минимум и максимум, вычисление $\forall$ и $\exists$-свойств (простых и вложенных), копирование дерева, проверка на равенство двух деревьев, ...

15. Бинарное (двоичное) дерево. Обход в ширину с использованием очереди. Задачи на обработку дерева с использованием обхода в ширину. Например: найти ширину дерева, поиск элемента, распечатать листья дерева, переставить минимум и максимум, вычисление $\forall$ и $\exists$-свойств (простых и вложенных).

16. Рекурсивные функции. Например: вычисление факториала, вычисление степени, проверка упорядоченности массива, сортировка слиянием, алгоритмы, основанные на дихотомии (поиск элемента в упорядоченном массиве, нахождение решения уравнения $f(x) = 0$ на итервале $[a, b]$), вычисление значения формулы, проверка правильности формулы, ...

17. Рекурсивные функции обработки односвязного линейного списка.

18. Рекурсивные функции обработки двоичного дерева.

19. Массивы списков, списки массивов, списки списков. Операции над ними.

20. Дерево двоичного поиска. Основные операции: создать дерево поиска, вставка элемента, поиск элемента, удаление элемента. Использование дерева двоичного поиска в задачах для организации эффективного поиска.

21. Двоичная куча. Основные операции: извлечь минимум, добавить элемент, удалить элемент, создать кучу из массива, проверить, является ли заданное дерево двоичной кучей. Операции Heapify_up и Heapify_Down как основные операции, поддерживающие свойство кучи.

22. Алгоритм сортировки кучей.

23. Дерево арифметического выражения, основные операции, сложность.

4 Образец экзаменационного билета 2 семестр 2016-2017 учебный год.

1. *(Обработка текстовых и бинарных файлов из структур, обработка упорядоченных файлов, форматированный и неформатированный ввод-вывод).*

   - Двоичный файл из структур "Температура" хранит следующие данные по городам: Название города (менее чем 50 символов), среднесуточную температуру в данном городе по каждому месяцу, по каждому году (показания за 20 лет). Файл упорядочен по названию города.
   - Имеется текстовый файл "Показания" содержащий информацию о замерах температуры в городах за последний 2014 год. Каждая строка файла содержит следующую информацию (через пробелы): название города, в котором производился замер среднесуточной температуры, дата, когда производился замер среднесуточной температуры (три целых числа через пробел, обозначающих год, месяц, день). Предполагается, что в одном и том же городе замер среднесуточной температуры мог производиться в течение нескольких дней месяца, количество дней замера достаточно для установления среднесуточного значения температуры в месяце. Файл упорядочен по названию города, дате.
   - Вывести в текстовый файл "Превышение" название городов и название месяцев, в которых показания среднесуточной температуры в 2014 году превысили показания среднесуточной температуры за последние 20 лет. Также вывести число, на сколько градусов превышено показание среднесуточной температуры в данном городе в данном месяце. Предполагается, что файл "Показания" может содержать показания по городам, отсутствующим в файле "Температура".

Один проход по файлам (использовать упорядоченность исходных файлов)!

2. *(Односвязные линейные списки и их разновидности (стеки, очереди, кольцевые, двухсвязные списки, списки списков, массивы списков, хеш-таблица), вставка, удаление элементов списка, построение списка, обращение к элементам списка, передвижение по списку).*

   Множество из элементов некоторого типа задается списком. Каждый элемент списка содержит элемент, принадлежащий множеству. Множество не может содержать повторных вхождений элементов. Над множеством определены операции:

   (a) *create* – создание пустого множества;

   (b) *is_empty* – проверка множества на пустоту;

   (c) *is_element* – проверка на принадлежность заданного элемента множеству;

   (d) *add* – добавление элемента в множество;

   (e) *del* – удаление элемента из множества;

   (f) *count* – количество элементов в множестве.

   "Множество из целых" – это множество из элементов – целых чисел. "Супермножество" – это множество, каждый элемент которого является "Множеством из целых".

   Создать "Супермножество" $\mathcal{S} = \{S_1, \ldots, S_n\}$ из элементов, заданных в тектовом файле. Формат файла: в первой строке задано число n – количество элементов в супермножестве. Далее идут $2n$ строк. Первая строка из каждых двух строк содержит целое число k – количество элементов в "Множестве из целых" S, вторая содержит k целых чисел – элементы S. Удалить из каждого множества элементы, которые есть в каких либо последующих множествах. Вывести в выходной файл количество множеств, которые стали пустыми после преобразования. Вывести на экран номера и элементы множеств, имеющих наибольшую мощность. Удалить все множества и супермножество.

   Пример.
   4
   4
   1 2 9 7
   3
   2 8 3
   5
   1 3 2 1 2
   2
   2 8
   Результат:
   kol of empty sets=1
   number=1, elements=$\{9, 7\}$
   number=3, elements=$\{1, 3\}$

number=4, elements=$\{2, 8\}$

3. *(двоичные деревья и их разновидности (дерево поиска, двоичная куча, дерево выражения), обходы в глубину и ширину с помощью стека, очереди или рекурсивно).*

   Дан текстовый файл из целых чисел. Преобразовать его дерево двоичного поиска. Вывести элементы построенного дерева в порядке обхода в ширину с использование очереди. Операции работы с очередью (Create, Empty, EnQueue, DeQueue) не описывать, тип очереди описать необходимо. Рекурсию не использовать. Описать и использовать функции

   (a) Добавить элемент в дерево поиска;

   (b) Обход дерева в ширину с выводом его значений в текстовый файл. Каждый новый уровень печатать с новой строки.

4. *(Рекурсия, сортировка, стековый алгоритм).*

   Написать функцию Merge_Sort – рекурсивный алгоритм сортировки массива слиянием.