Задания к работе №4 по фундаментальным алгоритмам.

1. Разработать программу – интерпретатор операций над целочисленными массивами. Программа оперирует целочисленными массивами произвольной длины с именами A, B, …, Z. Система команд данного интерпретатора (прописные и строчные буквы не различаются):
   1) Load A, in.txt; - загрузить в массив A целые числа из файла in.txt;
   2) Save A, in2.txt; - выгрузить элементы массива A в файл in2.txt;
   3) Rand A, count, lb, rb; - заполнить массив A случайными элементами из отрезка $[lb; lr]$ в количестве count штук;
   4) Concat A, b; - объединить два массива A и B результат сохранить в массив A;
   5) Free(a); - очистить массив A;
   6) Remove a, 2, 7; - удалить из массива a 7 элементов, начиная с элемента с индексом 2;
   7) Copy A, 4, 10, b; - скопировать из массива A элементы с 4 по 10 и сохранить их в b;
   8) Sort A+(-); - сортировать элементы массива A по возрастанию (убыванию);
   9) Permute A; - переставить элементы массива в случайном порядке;
   10) Stat a; - вывести статистическую информацию о массиве a: размер массива, максимальный и минимальный элемент (и их индексы), наиболее часто встречающийся элемент, среднее значение элементов, максимальное отклонение элементов от среднего значения;
   11) Print a, 4, 16; - вывести на экран элементы массива начиная с 4 и по 16;
   12) Print a, all; - вывести на экран все элементы массива.

   Для сортировки массивов используйте стандартную функцию qsort, непосредственно реализовывать алгоритмы сортировки запрещено. Предоставить текстовый файл с инструкциями для данного интерпретатора. Считается, что ошибок в командах нет, однако внимательно обработайте все ошибки исполнения инструкций.
2. Реализовать структуру ЯчейкаПамяти, которая содержит имя переменной и её целочисленное значение. Через аргументы командной строки в программу подается файл с инструкциями вида

   ```
   myvar=15;
   bg=25;
   ccc=bg+myvar;
   print ccc;
   bg=ccc*myvar;
   print;
   ```

   Файл не содержит ошибок и все инструкции корректны. Реализовать чтение данных из файла и выполнение всех простых арифметических операций (+, -, *, /, %) и операции print (вывод на экран либо текущего значения переменной, либо значений всех переменных с указанием имен). В каждом операнде может присутствовать только одна арифметическая операция. При объявлении переменной, необходимо выделить память в динамическом массиве структур. Для поиска переменной в массиве используйте алгоритм дихотомического поиска, для этого поддерживайте всегда ваш массив в отсортированном состоянии. Для сортировки массива используйте стандартную функцию qsort, реализовывать непосредственно какие-либо алгоритмы сортировки запрещается. Имя переменной может иметь произвольную длину и содержать только символы букв. Заглавные и прописные буквы не отождествляются. В конце программы, необходимо очистить всю выделенную память. В случае использования в вычислениях не объявленной переменной вывести сообщение об ошибке.

3. Напишите программу моделирующую приоритетную очередь сообщений. В вашей программе реализуйте абстрактную очередь на основе односвязного списка. Реализуйте операции Enqueue, Dequeue и EnqueuePriority. Данными, которые храняться в очереди, являются структуры с полями текст и приоритет. При вставке нового элемента в очередь вы должны реализовать возможность добавления элемента на основе его приоритета: чем выше приоритет, тем ближе к концу очереди он добавляется, причем элементы с одинаковым приоритетом стоят в очереди в порядке их появления. Входными данными программы являются текстовые файлы, которые имеют следующий формат:

   prior=0 task=’Задание 1’
   prior=0 task=’Задание 3’
   prior=5 task=’Задание 2’
   prior=2 task=’Задание 4’
   prior=3 task=’Задание 3’
   prior=3 task=’Задание 3’
   prior=2 task=’Задание 5’

   Количество входных файлов не ограничено и может быть произвольным. Организуйте считывание имен входных файлов с клавиатуры.
4. В текстовом файле находится информация о жителях некоторого поселения: фамилия, имя, отчество, дата рождения (в формате число, месяц, год), пол, средний доход за месяц. Напишите программу, которая считывает эту информацию из файла в односвязный упорядоченный список (в порядке увеличения возраста). При чем информация о каждом жителе должна храниться в структуре. Реализуйте возможности поиска жителя с заданными параметрами, удаления/добавления информации о жителях и возможность выгрузки данных из списка обратно в файл.
5. A) Написать программу сбора статистических данных по заданному тексту. Результатом работы программы является информация о том сколько раз каждое слово из файла встречается в данном файле. Реализовать дополнительные опции: вывод информации о том сколько раз заданное слово встречалось в файле; вывод первых n наиболее часто встречающихся слов в файле; поиск самого длинного и самого короткого слова. Для решения поставленной задачи необходимо использовать двоичное дерево.

   B) Для построенного дерева в пункте A напишите функцию поиска глубины данного дерева.

   C) Напишите функции сохранения(восстановления) бинарного дерева в(из) файл(а). При этом восстановленное дерево должно иметь точно такую же структуру и вид, как и до сохранения. Иными словами, тривиальный обход дерева как при алгоритме поиска не допустим.
6. Пользователь вводит с клавиатуры различные инструкции, которые являются строками. При этом, каждые N строк, которые ввел пользователь, сохраняются в текстовый файл и у пользователя есть возможность отменить последние $N/2$ введенных строк. Организовать хранение строк в программе в виде односвязного списка, при этом, сохранение строк реализовать посредством очереди, которая содержит введенные команды, а возможность отмены, реализовать с помощью стека, который содержит те же самые элементы.
7. Разработать интерпретатор с настраиваемым синтаксисом. Для настройки интерпретатора через аргументы командной строки подается файл с описанием инструкций и их синтаксиса. Файл настроек содержит сопоставления реальных операций, которые может выполнить интерпретатор и их псевдонимов, которые будут использованы в программах, которые будут поданы на вход. Файл настроек может содержать однострочные комментарии, которые начинаются с символа #. Интерпретатор оперирует 32-х разрядными целочисленными переменными, имена

которых могут содержать более одного символа. Основные команды, которые могут быть выполнены интерпретатором add (сложение), mult (умножение), sub (вычитание), pow (возведение в степень), div (целочисленное деление), rem (остаток от деления), xor (битовое сложение по модулю 2), input (ввод значения с клавиатуры), output (вывод значения переменной на экран), = (присваивание значения выражения переменной или ее инициализация). В файле настроек слева всегда написана оригинальная операция, а справа синоним. Примерный файл настроек выглядит следующим образом:

left= #означает, что результат сохраняется в переменную, которая стоит слева от знака равенства (может быть написано right=, то есть результат сохраняется в переменную, которая стоит справа)

op() #означает, что символ операции предшествует списку аргументов

может быть написано ()op ,то есть операция идет после списка аргументов

add sum

mult prod

sub minus

pow ^

div /

rem %

xor <>

input in

output print

= ->

Синтаксис инструкции имеет вид: A1=<op>(B2, C34); <op> - операция из списка выше;

D12=in(); - ввод значения в переменную D12 в 10-ой системе счисления.

print(D); - вывод значения из переменной D в 10-ой системе счисления.

Разделителем между операторами является символ “;”. Пробелы могут присутствовать произвольно, различий между заглавными и строчными буквами нет. Так же в тексте могут присутствовать многострочные комментарии, которые обрамляются символами [ ]. Написать программу – интерпретатор инструкций в заданном файле, с проверкой корректности многострочного комментария. При завершении работы интерпретатор запоминает последний файл настроек, с которым работал. У каждой команды всегда доступен один идентификатор, который определяется файлом настроек.

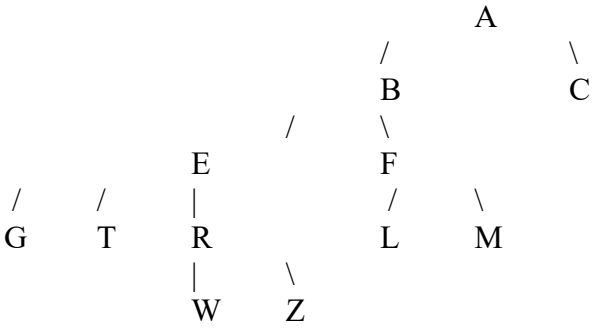
8. Разработайте программу для организации макрозамен в тексте. На вход программы подается текстовый файл, который содержит в начале файла набор директив #define и обычный текст. Синтаксис директивы соответствует стандарту языка C:

   #define <def_name> <value>

   Аргументов у директивы нет. Ваша программа должна обработать текстовый файл, выполнив замены во всем тексте последовательности символов <def_name> на <value>. Количество директив произвольно, некорректных директив нет, размер текста произволен. В имени <def_name> допускается использование символов латинского алфавита (прописные и заглавные буквы не отождествляются) и символов цифр; значение <value> произвольно. Для хранения имен макросов и макроподстановок используйте хеш-таблицу фиксированного размера HASHSIZE (определите величину этой константы равной 128). Для вычисления хеш-функции интерпретируйте <def_name>, как число, записанное в 62-ой системе счисления (алфавит этой системы счисления состоит из символов {0, …,9, A, …, Z, a, …, z}). Переведите это число в 10-ую систему счисления и возьмите остаток от деления на HASHSIZE, полученное значение и будет являться хеш-значением,

соответствующим <def_name>. Для разрешения коллизий используйте метод цепочек.

9. Напишите программу для построения дерева скобочного выражения. На вход программе подается файл, в котором содержится произвольное число строк. Каждая строка является корректным скобочным выражением. Ваша программа должна обработать каждое скобочное выражение из файла и вывести в текстовый файл деревья скобочных записей в наглядной форме (форму вывода определите самостоятельно). Возникающие при работе программы деревья не обязаны быть бинарными. Например, запись A (B (E (G, T, R (W, Z)), F (L, M)), C) соответствует дереву



Формат вывода не фиксирован и определяется удобством восприятия.

10. Напишите приложение, которое по заданной булевой формуле строит ее таблицу истинности. На вход программы подается файл, который содержит одну строку, в которой записана булева формула. В этой формуле могут присутствовать одно символьные имена переменных, константы 0 или 1, булевы операции (&-конъюнкция, | -дизъюнкция, ~ -отрицание, -> - импликация, +> - коимпликация, <> -строгая дизъюнкция (сложение по модулю 2), = -эквиваленция, ! - штрих Шеффера, ? – элемент Вебба и скобки. Вложенность скобок произвольна. Для вычисления булевой формулы постройте бинарное дерево выражения и вычисление значения булевой формулы на конкретном наборе переменных выполняйте с помощью этого дерева. Приоритеты операций: 3(~), 2(?,!,+>,&), 1(|, ->, <>, =).