Лабораторная работа 5, часть 1

Черновик 0.8

Целью лабораторной работы является знакомство студентов с обработкой текстовых и двоичных файлов.

Студенты должны получить и закрепить на практике следующие знания и умения:

1. Обрабатывать текстовые и двоичные файлы (открывать в различных режимах, читать и записывать информацию).
2. Организовывать корректную работу с ресурсами (в данном случае – файловыми описателями).
3. Анализировать информацию об ошибках с помощью функций стандартной библиотеки.
4. Использовать в программе аргументы командной строки.

Общее задание

1. Исходный код лабораторной работы располагается в отдельной ветке lab_51. В ветке lab_51 для каждой задачи создается папка lab_51_X_Y, где X – номер варианта, Y – номер задачи (например, если для первой задачи вы решаете 3 вариант, папка будет называться lab_51_3_1).
   ИСКЛЮЧЕНИЕ
   Для третьей задачи (в ней всего один вариант) папка должна называться lab_51_3.

2. Исходный код должен соответствовать правилам оформления исходного кода.

3. Для каждой задачи создается отдельный проект в *QT Creator*. Для каждого проекта должно быть два варианта сборки: Debug (с отладочной информацией) и Release (без отладочной информации).
   Замечание
   По согласованию с преподавателем, проводящим практические занятия, вы можете использовать другую среду разработки (например, Microsoft Visual Studio Code), но два варианта сборки проекта нужно предусмотреть в любом случае.

4. Созданный проект обязательно должен быть многофайловым.

5. Для каждой задачи студентом подготавливаются тестовые данные, которые демонстрируют правильность ее работы. Входные данные должны располагаться в файлах in_z.txt, выходные out_z.txt, где z – номер тестового случая. Тестовые данные готовятся и помещаются под версионный контроль еще до того, как появится реализация задачи.
   Замечание
   Для третьей задачи расширение файлов с тестовыми данными должно быт bin.

6. Реализовав очередную задачу и проверив правильность ее работы, оцените полноту подготовленных тестовых данных на основе процента покрытия кода этими данными. Добейтесь 100% покрытия кода тестовыми данными. Если это невозможно, необходимо это обосновать.

7. В задачах 1 и 2 ответ выводится на экран. Если он не может быть получен, функция main возвращает код отличный от нуля.

Индивидуальное задание

`Номер задания = Номер в журнале % Количество вариантов.`

Схема распределения вариантов может быть изменена преподавателем, проводящим практические занятия. Прежде чем приступить к работе над вариантом, уточните этот момент у вашего преподавателя.

Задача 1

Пользователь вводит целые числа, по окончании ввода чисел вводит букву (в Windows для ввода в окне cmd признака EOF можно нажать Ctrl-Z и Enter).

Написать программу, которая

0. находит наибольшее положительное из чисел, которые следуют за отрицательным числом (если пользователь вводит “1 2 -3 4 -6 5 -7 -8”, максимум выбирается среди 4 и 5);
1. находит два максимальных элемента последовательности (возможно совпадающих);
2. находит порядковый номер (позиция начинается с 1) максимального из чисел (если чисел с максимальным значением несколько, то должен быть найден номер первого из них);
3. определяет сколько раз в последовательности чисел меняется знак (нуль считается положительным числом);
4. находит количество чисел, которые больше своих «соседей», т.е. предшествующего и последующего;
5. находит наибольшее число подряд идущих элементов последовательности, которые равны друг другу;
6. находит наибольшую длину монотонного фрагмента последовательности (то есть такого фрагмента, где все элементы либо больше предыдущего, либо меньше);
7. определяет количество локальных максимумов в последовательности (Элемент последовательности называется локальным максимумом, если он строго больше предыдущего и последующего элемента последовательности. Первый и последний элемент последовательности не являются локальными максимумами.);
8. определяет наименьшее расстояние между двумя локальными максимумами последовательности (понятие локального максимума описано в пункте 7).

Дополнительные требования к решению задачи 1:

1. Прототип функции, которая реализует решение задачи, должен выглядеть следующим образом:

   ```
   int process(FILE *f [, прочие выходные параметры]);
   ```

2. Функция process возвращает 0 в случае успешного решения задачи и отрицательный код ошибки в противном случае (например, -1 – входных данных нет и т.д.). Для каждого кода ошибки задается мнемоническое имя с помощью директивы define.
3. При решении любого варианта задачи 1 два цикла ввода и массивы не использовать.

Задача 2

Написать программу, которая считывает из текстового файла вещественные числа и выполняет над ними некоторые вычисления:

0. найти число, наиболее близкое по значению к среднему значению всех чисел;

1. найти количество чисел, значение которых больше среднего арифметического минимального и максимального чисел;
2. рассчитать дисперсию чисел (математическое ожидание и дисперсия рассчитываются отдельно);
3. проверить выполняется ли правило «трех сигм» для чисел (если правило «трех сигм» выполняется выводится 1, если нет – 0);
4. найти среднее значение чисел, расположенных между минимальным и максимальным числами («между» - не по значению, а по расположению); предполагается, что минимум и максимум один.

Формулы, используемые в некоторых вариантах

Математическое ожидание	$avg = \frac{1}{n}\sum_{1}^{n} x_i$
Дисперсия	$disp = \frac{1}{n}\sum_{1}^{n} (x_i - avg)^2$
Стандартное отклонение	$\sigma = \sqrt{disp}$
Интервал для правила «трех сигм»	$(avg - 3 * \sigma; avg + 3 * \sigma)$

Дополнительные требования к решению задачи 2:
1. При решении задачи массивы не использовать.
2. Имя файла берется из аргументов командной строки.
3. Предусмотреть обработку ошибок.
4. Решение любой из этих задач выполняется минимум за два просмотра файла.

Задача 3

Написать программу, которая обрабатывает двоичный файл, содержащий целые числа типа *int*. Программа должна уметь
- создавать файл и заполнять его случайными числами (аргументы в произвольной форме, например, *app.exe c number file*; создание не проверяется :));
- выводить числа из файла на экран (*app.exe p file*);
- упорядочивать числа в файле (*app.exe s file*).

Прежде чем реализовывать функцию упорядочивания файла, необходимо реализовать функцию *get_number_by_pos*, которая по заданной позиции, позволяет прочитать число в указанной позиции, и функцию *put_number_by_pos*, которая позволяет записать число в указанную позицию. Функцию упорядочивания необходимо реализовать с помощью этих функций.

В начале файла, содержащего исходный код программы, должен располагаться многострочный комментарий, в котором необходимо указать детали реализации этой задачи: как минимум, выбранные целочисленный тип, алгоритм сортировки, «направление» упорядочивания.