Требования к лабораторным работам

по дисциплине «Алгоритмы и структуры данных»

Содержание:

1. Интерфейс программы с внешним окружением

1.1. Ввод данных

Входные данные программы целесообразно разделить на две части:

1) **основной набор данных** (как правило, относительно большого объема, т.е. те данные, которые обременительно вводить при каждом запуске программы) **должен вводиться из файла (это обязательное требование)**;

2) небольшое количество входных параметров, которые предполагается изменять относительно часто при запусках программы, и которые должны вводиться с клавиатуры; эти параметры могут быть либо параметрами алгоритма, либо могут, например, определять текущий размер входных данных, которые будут составлять только часть размещенных в файле.

В некоторых случаях (по рекомендации преподавателя и с учетом режима тестирования) *все* входные данные должны быть прочитаны из входного файла (с оговоренным форматированием).

Не строго обязательные, но **желательные требования к вводу данных:**

1) Наличие ввода из консоли помимо ввода из файла (не требуется в тех случаях, когда данных много и вводить их с консоли было бы неудобно).

2) Ввод данных с консоли должен сопровождаться соответствующим диалогом.

3) При вводе данных из файла они должны быть выведены на консоли.

4) В случае вывода данных в файл перед выходными данными должны сохраняться введенные входные данные.

5) Не слишком плохое поведение программы при некорректных данных на входе: программа не должна зависнуть, не должна потерять способность к нормальному завершению, не должна немедленно завершиться без вывода информативного сообщения об ошибке.

6) Удобство при тестировании. Проверяется корректность входных данных, случайная ошибка пользователя при вводе не должна привести к тому, что всё придётся вводить заново. Если входные данные комплексные, следует предусмотреть возможность выполнять алгоритм повторно (на новых входных данных) без перезапуска программы.

П. 5 и 6 не требуются, если проверка корректности входных данных является самостоятельным заданием (другого варианта).

1.2. Вывод промежуточных (вспомогательных) данных и результатов

Вывод данных должен осуществляться в форме, удобной для анализа и интерпретации.

Кроме собственно результирующих данных в процессе работы программы **должны выводиться промежуточные (вспомогательные) данные, характеризующие вычислительный процесс**. Это требуется, во-первых, для демонстрации способа получения результата, а во-вторых, позволяет преподавателю оценить степень понимания студентом алгоритма и порождаемого алгоритмом вычислительного процесса. Количество и состав выводимых промежуточных данных, а также форма и «динамика» вывода

выбираются студентом, исходя из существа задачи, а также с целью аргументировано убедить преподавателя в правильности и эффективности разработанной программы. Даже если программа очень простая, **какие-то промежуточные данные обязательно должны выводиться**. Промежуточные данные должны быть оформлены в такой степени, чтобы разобраться в них было не слишком сложно.

Выходные данные должны выводиться *на экран* для экспресс-анализа (в режиме *online*) в ходе работы программы и *в файл* для сохранения, документирования, последующего анализа и интерпретации результатов работы программы (в режиме off-line).

В случае использования «автоматического» тестирования все выходные данные должны помещаться в выходной файл. При этом следует заранее определить их состав и формат вывода.

1.3. Организация рабочих и отчетных материалов, предъявляемых студентом

- исходный код;
- запускаемый файл + файлы входных данных;
- отчёт, включающий полный исходный код и тестирование;

2. Внутреннее устройство программы

2.1. Требования к разработке и структуре программы

Общие требования аналогичны использованным в предыдущих семестрах (дисциплина «Программирование»).

При разработке программ следует придерживаться идеологии абстрактных типов данных (АТД) и опираться на триаду «интерфейс-реализация-клиент». В разных лабораторных работах преподавателем могут быть рекомендованы различные парадигмы («процедурно-модульная»или «объектно-ориентированная»), а также использование элементов обобщенного программирования (шаблоны функций и классов).

3. Тестирование и отладка

При защите лабораторной работы студент предъявляет преподавателю проект отчета, набор тестов, краткий протокол испытаний программы (тестирования и отладки). Количество тестов и их наполнение студент определяет самостоятельно с конечной целью убедить преподавателя в корректности разработанной программы, её эффективности и, возможно, надежности. Преподаватель может предложить свои тесты или рекомендовать студенту подготовить дополнительные, а затем провести соответствующие испытания программы. Успешным является тест, с помощью которого были обнаружены дефекты программы.

Требования к тестированию:

1) По крайней мере 1 тест на некорректных данных, который должен быть отделен в отчете от тестов на корректных данных.

2) 1-2 теста на граничных данных.

3) Нумерация тестов.

4) Количество тестов сильно зависит от тестируемого алгоритма. Оно должно быть достаточным, чтобы можно было покрыть различные случаи и убедиться в коррекности написанного алгоритма.

4. Требования к отчету

Отчет представляется в электронной форме в виде текстового файла (например, формата odt), пригодного для оперативного вывода на бумагу.

В разделе «Задание» должен быть указан номер варианта.

Отчет должен быть оформлен по Шаблону оформления отчета по лабораторной работе: https://etu.ru/ru/studentam/dokumenty-dlya-ucheby/

Содержание и форма отчета должны соответствовать требованиям, использованным в предыдущих семестрах (дисциплина «Программирование»). В приложении должны быть представлены тестовые наборы, результаты их обработки и краткий протокол испытаний программы (тестирования и отладки).

Блок-схемы не нужны.

5. Требования к коду.

Язык программирования – C / C++ / Objective-C. **Это обязательное условие.** Кроме того:

• Именование переменных, функций, методов, классов и т.д. таким образом, чтобы это не затрудняло понимание кода.
• Логически правильное и достаточное разбиение исходного кода на функции (методы). Выделение в отдельные файлы тех частей кода, которые имеют самостоятельную значимость.
• В случае использования языка C++ требуется корректное использование и понимание применённых возможностей языка.
• Следование единому стилю форматирования исходного кода на протяжении всего исходного кода программы. Причём:
- имена классов (и других структур данных) с большой буквы (в стиле CamelCase);
- имена переменных и функций с маленькой буквы (lowerCamelCase);
- имена членов перечислений БОЛЬШИМИ БУКВАМИ полностью;
- минимум одна пустая строка между функциями и между структурами данных;

- соблюдениие отступов в коде.

Дополнительно: рекомендация по автоматизации тестирования.

Для программы, требующей относительно большого количества тестов, будет полезна возможность автоматизировать тестирование, включающая:
• Возможность запускать тесты одной командой;
• Хранение тестовых данных в отдельной папке Tests в удобочитаемой форме;
• Наличие поясняющих комментариев при запуске тестов.
Возможные варианты реализации - .bat-файл или другая C/C++ программа.