**Задания к контрольной работе**

В рамках данной контрольной работы предлагается осуществить формирование требований, проектирование, реализацию и тестирование программы. Типовые темы заданий приводятся ниже для десяти вариантов. Возможен выбор темы, отличающейся от типовой. Выбор темы из приводимого ниже списка альтернативных тем производится без согласования с преподавателем. При согласовании с преподавателем допускается выбор собственной темы, отличающейся от типовой или альтернативной из списка.

**Общий для всех вариантов и тем план выполнения работ разбит на следующие этапы:**

1. Формирование требований. Представьте себя заказчиком и потенциальным пользователем программы, которая решала бы задачу Вашего варианта. Сформируйте требования пользователя к ней. Можно использовать как неформальное описание на естественном языке (русском, английском), так и какой-либо формальный язык для спецификаций. В любом случае важен не объем текста (это может быть, например, всего 0.5-2 страницы), а содержательное описание функций с обоснованием их выбора.

2. Проектирование

2.1. Выберите и обоснуйте аппаратную платформу, ОС, язык программирования и компилятор для него (или некоторую интегрированную среду разработчика). При необходимости, также можно выбрать какую-либо заемную библиотеку функций. Выбор производится, исходя из сформированных в задании 1 требований, имеющихся у Вас аппаратных ресурсов, знания языков программирования и имеющегося набора программного обеспечения. В данном разделе желательно избегать подробное изложение достоинств выбираемых сред, которые не относятся непосредственно к обоснованию их выбора (достаточно указания ссылок на печатные или электронные материалы).

2.2. Нарисуйте в виде блок-схемы алгоритм работы программы.

2.3. Если использован объектно-ориентированный подход, перечислите все классы, их атрибуты и методы, отношение наследования (в виде текста на естественном языке или текста на выбранном языке программирования, поддерживающем объектно-ориентированное программирование, или в виде диаграммы классов).

2.4. Нарисуйте диаграмму с набором модулей, на которые будет разбита программа. Стрелками для каждого модуля укажите, какие модули используются данным модулем.

3. Реализация

3.1. Программирование

По результатам проектирования реализуйте программу на выбранной платформе и на выбранном языке.

В процессе реализации для устранения ошибок пользуйтесь отладчиком. Для этого соберите программу в отладочном варианте. Если Вы выбрали GNU C/C++, то командная строка для сборки может быть такой: *gcc –g –o program.bin program.c.* Запустите собранный бинарный файл в отладчике. Пример для GCC и GDB приведен в гл. 6.

Результат выполнения задания 3 – все тексты программы и использовавшаяся для ее компиляции команда. Тексты программы оформляются в соответствии с рекомендациями по стилю, которые приведены в гл. 4 конспекта лекций данного курса.

**3.2. Документирование – разработка модели.** Возьмите за основу модель документации из приложения 2. Предложите измененную версию модели, которая на Ваш взгляд была бы удобна для описания программ, подобных построенной в задании 3. Для каждого добавленного, измененного и исключенного пункта объясните причину, почему это было необходимо сделать.

**3.3. Документирование.** На основе полученной в предыдущем пункте модели документации, составьте документацию для программы, реализованной в этом задании (п. 3.1).

4. Оценка производительности. Проведите измерение времени выполнения получившейся в задании 4 программы зависимости от объема данных. Если Вы выбрали в качестве языка программирования C/C++, то для замеров можно воспользоваться функциями из файла *benchmark.1.c* (или *benchmark.2.c*). При этом функция main будет иметь такой вид:

*main(){*

*benchmark\_start();*

*…*

*рrintf(“duration = %d\n”, benchmark\_stop());*

*}*

Сделайте это для нескольких вариантов, изменив флаги для компилятора, которые отвечают за уровни оптимизации. Можно попробовать и флаги, включающие и выключающие отдельные виды оптимизации. Если Вы выбрали GNU C/C++, то минимальный рекомендуемый набор вариантов для тестирования таков:

*gcc –o program.bin program.c*

*gcc –O2 –o program.bin program.c*

*gcc –O3 –o program.bin program.c*

*gcc –O4 –o program.bin program.c*

Постройте графики, где по ось X задана в соответствии с Вашим вариантом, а ось Y – время выполнения программы в миллисекундах.

**Выполнение этапов 1, 2.1, 2.2., 3.1. и 3.3. - обязательно.** Выполнение остальных этапов желательно. Важность их выполнения варьируется в зависимости от выбранной темы.

**Типовые темы заданий для десяти вариантов.**

**Вариант 2.** Калькулятор для выражений с инфиксной формой записи. В минимальном виде калькулятор – это утилита, которой в качестве параметров передается имя входного файла, содержащего выражение. Утилита 1) считывает входные параметры и проверяет их корректность, 2) открывает исходный, обрабатывая возможные ошибочные ситуации, 3) производит вычисление результата выражения, 4) печатает и завершает работу.

Выражение для калькулятора– это:

 десятичная константа;

 (выражение)

 -выражение

 выражение + выражение

 выражение - выражение

 выражение \* выражение

 выражение / выражение

 sin(выражение)

 cos(выражение)

Для задания 4 ось X – размер входного файла со строкой выражения.