Цели и задачи практики

Целью практики является знакомство студентов с инструментарием, который используется для создания, отладки и сопровождения программ написанных на языке программирования Си.

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

1. Изучить стадии компиляции программы, научиться компилировать и компоновать программу в командной строке (однофайловый и многофайловый проекты).
2. Получить представление об организации объектных и исполняемых файлов, научиться анализировать информацию, которая в них содержится.
3. Познакомиться с интегрированной средой разработки *Qt Creator*. Научиться
   - создавать проекты в *Qt Creator*;
   - настраивать сборки проектов (release и debug);
   - отлаживать программы в этой среде.
4. Познакомиться с утилитой *make* и научиться ее использовать для автоматизации сборки проектов как в командной строке, так и в среде *Qt Creator*.
5. Познакомиться с утилитой *gcov*. Научиться определять величину покрытия кода тестами.
6. Изучить и закрепить на практике
   - работу с текстовыми файлами;
   - обработку ошибок;
   - работу с аргументами командной строки.

Общее задание

1. Установите дома необходимое программное обеспечение. Кратко опишите ваши основные действия. Если в процессе установки или настройки программного обеспечения вы столкнулись с какими-нибудь проблемами, опишите эти проблемы и способы, с помощью которых вы их преодолели.

2. Познакомьтесь со стадиями компиляции и результатами работы компилятора.
   a. Напишите небольшую программу (не "hello, world!"; обязательно должны присутствовать комментарии, директива define).
   b. Изучите стадии компиляции программы. В отчете приведите команду, которую необходимо написать, чтобы выполнилась очередная стадия компиляция, а так же наиболее существенные результаты работы компилятора.
   c. Какие секции входят в вашу программу, когда вы ее собираете без и с отладочной информацией (воспользуйтесь утилитой *objdump*)? Как сильно отличается размер исполняемого файла?
   d. Определите, в какие секции попадают переменные (глобальные и локальные) и функции.
   e. Добавьте глобальный массив к вашей программе. Не инициализируйте его. В какую секцию он попал? Сравните размер исполняемого файла до и после добавления этого массива.
   f. Проделайте все то же самое, но для проинициализированного массива.
   g. Какие динамические библиотеки использует ваша программа (воспользуйтесь утилитой *objdump* или *dumpbin* из *Visual Studio*)?

3. Познакомьтесь с интегрированной средой разработки *QT Creator*.

4. Выполните задание, которое позволит вам понять насколько вы «освоились» с работой в *QT Creator*.
    a. Скопируйте исходный код программы из Приложения А в отдельный файл.
    b. Создайте в *QT Creator* проект для этой программы. Приведите в отчете команды, с помощью которых вы будете компилировать и собирать программу.
    c. Программа содержит синтаксические (из-за которых она не компилируется) и семантические (из-за которых она работает неправильно) ошибки. Сначала исправьте синтаксические ошибки, затем семантические. При этом ошибки исправляются строго по одной, а в отчет выносится сообщение компилятора об ошибке и соответствующие исправления.
    d. Выполните программу в пошаговом режиме.
    e. Поставьте несколько точек останов, задайте условия для срабатывания этих точек останова. Проверьте правильность срабатывания точек останова.
    f. Выполните программу под отладчиком, контролируя значения различных переменных. Измените с помощью отладчика значение той или иной переменной во время работы программы.

5. Выполните индивидуальное задание согласно варианту.
    a. Исходный код должен соответствовать правилам оформления исходного кода. Выполнение этого пункта контролируется с помощью утилиты *CodeChecker* (эта утилита установлена в 508л).
    b. Для каждой задачи создается отдельный проект в *QT Creator*.
    c. Каждую задачу необходимо уметь компилировать двумя способами: с помощью компилятора *gcc* и с помощью утилиты *make*. Кроме того, необходимо уметь использовать эти способы в настройках проекта в *QT Creator*.

6. Познакомьтесь с утилитой *gcov*.

7. С помощью утилиты *gcov* определите покрытие кода тестами в вашем индивидуальном задании. Добейтесь 100% покрытия, а сам процесс детально опишите в отчете. Если 100% покрытия не удается достигнуть, объясните причину этого.

Индивидуальное задание

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

Задача 1

Пользователь вводит целые числа, по окончании ввода чисел нажимает Ctrl-Z и Enter.

Написать программу, которая

0. находит наибольшее положительное из чисел, которые следуют за отрицательным числом;
1. находит два максимальных элемента последовательности (возможно совпадающих);
2. определяет сколько раз в последовательности чисел меняется знак (нуль считается положительным числом);
3. находит наибольшее число подряд идущих элементов последовательности, которые равны друг другу;

4. находит наибольшую длину монотонного фрагмента последовательности (то есть такого фрагмента, где все элементы либо больше предыдущего, либо меньше);
5. 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.);
6. определяет наименьшее расстояние между двумя локальными максимумами последовательности.

Требования к решению задачи:

1. Программа реализуется как однофайловый проект.
2. Прототип функции, которая реализует решение задачи, должен выглядеть следующим образом:

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

3. Функция process возвращает 0 в случае успешного решения задачи и отрицательный код ошибки в противном случае (например, -1 – входных данных нет и т.д.).
4. При решении задачи два цикла ввода и массивы не использовать.
5. Необходимо подготовить наборы тестовых данных по классам эквивалентности. Каждый набор разместить в текстовом файле.

Задача 2

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

0. найти число, наиболее близкое 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);

   Математическое ожидание – $\bar{x} = \frac{1}{n}\sum_{1}^{n} x_i$, дисперсия – $D = \frac{1}{n}\sum_{1}^{n}\left(x_i - \bar{x}\right)^2$.
3. проверить выполняется ли правило «трех сигм» для чисел (см. статью «Среднеквадратическое отклонение» в wikipedia);
4. найти среднее значение чисел, расположенных между минимальным и максимальным числами («между» - не по значению, а по расположению).

Требования к решению задачи:

1. При решении задачи выделить несколько функций.
2. При решении задачи массивы не использовать.
3. Имя файла берется из аргументов командной строки.
4. Предусмотреть обработку ошибок.
5. Решение любой из этих задач выполняется минимум за два просмотра файла.
6. Подготовить тестовые данные, демонстрирующие правильную работу программы.
7. Подготовить несколько проектов:
   a. многофайловый проект без заголовочных файлов;
   b. многофайловый проект с заголовочными файлами;
   c. многофайловый проект с заголовочными файлами и модульными тестами.

Приложение А

```
#include <stdoi.h>

int main(void)
{
    int max = 0, count = 0;

    scanf("%f", &max);

    while (scanf("%d", num) == 1)
    {
        if (num > max)
            max = num;
        else
            if (num = max)
                count++;
    }

    printf("max %d, count %d\n", &max, &count);

    return 0;
}
```