Сибирский государственный университет телекоммуникаций и информатики
Факультет ИВТ
Программирование на языке высокого уровня
Второй семестр

ЗАДАНИЯ НА КУРСОВОЙ ПРЕКТ
(вариант выбирается студентом и согласовывается у преподавателя)

Требования к курсовым работам

1 Все программы реализуются на языке Си;
2 **Необходима** проверка **всех входных данных** на корректность. В случае ошибки выдается сообщение, поясняющее суть ошибки для пользователя. Далее программу завершаетя с ненулевым кодом;
3 Разработанный программный комплекс должен быть сформирован в дистрибутив, содержащий исходные коды ПО, систему сборки ПО (набор Makefile-файлов) и необходимую документацию;
4 Для хранения **наборов данных заранее неизвестного размера** необходимо использовать **списки**.
5 Для заданий, предусматривающих разработку библиотек,
6 Для сдачи курсового проекта необходимо подготовить отчет, включающий приведенные ниже пункты:
6.1 **Титульный лист**, оформленный как в шаблоне для лабораторных работ. Вместо строки "Лабораторная работа" "Курсовой проект". В поле название пишется название раздела задания к которому относится ваш вариант.
6.2 **Задание —** содержит текст задания.
6.3 **Анализ задачи**, приводится развернутый анализ задачи, описываются методы и алгоритмы решения поставленной задачи (особенно важные фрагменты представляются блок-схемами или псевдокодом). Если используются известные алгоритмы (кодирования, архивации, поиска) для них также необходимо привести описание и привести пример их работы.
6.4 **Тестовые данные:** максимально полный набор тестовых данных демонстрирующий работу разработанного ПО на различных данных (в том числе некорректных). **Данный раздел должен присутствовать обязательно.**
6.5 **Листинг программы**, раздел содержит исходные коды разработанного ПО. При оформлении можно использовать 8 кегль шрифта.

ЗАДАНИЯ

Вариант	Задание
Раздел 1. *Разработка библиотеки сортировок.* ***Примечание***: задания данного раздела имеют сниженную сложность, оценка не выше 3 (удовлетворительно). Реализовать в подпрограммах предусмотренные вариантом алгоритмы сортировки. Подпрограммы выносятся в отдельную библиотеку, которая компилируется как динамическая библиотека. Информация о создании динамических библиотек: http://firststeps.ru/linux/general1.html. Также необходимо разработать демонстрационную программу, использующую созданную библиотеку.	
1	Сортировка методом пузырька, быстрая сортировка
2	Сортировка вставками, пирамидальная сортировка.
3	Сортировка Шелла, блочная сортировка.
Раздел 2. *Разработка утилиты командной строки* Реализовать ПО в соответствии с вариантом. Все входные данные передаются через аргументы командной строки.	

4	Разработать интерпретатор скриптов (ИС) - ***script_interp*** Команда ***script_interp*** принимает в качестве входных данных имя файла, содержащего скрипт. Скрипт состоит из операторов ввода/вывода (read/print) и циклических конструкций (while do). Данные примитивы разрабатываются и реализуются в ИС самостоятельно. ИС последовательно проходит по скрипту и выполняет его программу.
5	Разработать калькулятор для командной строки — ***expr_stud***, выполняющий вычисление арифметического выражения, переданного в качестве входных данных. Арифметическое выражение записывается следующим образом: ***(ApB)***, где ***A*** – левый операнд, ***B***- правый операнд, ***p*** – арифметическая операция. ***A*** и ***B*** – **вещественные числа**, ***p***=+ | - | * | /. Пример: ((((1.1-2) +3) * (2.5*2)) + 10) Если выражение содержит ошибки – необходимо вывести сообщение об ошибке. Иначе – результат вычисления выражения. Пример: expr_stud "((3*2)-(1+3))" должна вывести 2.
6	Разработать ПО для форматирования (расстановки табуляций) программы на языке Си: ***format_stud***. Команда ***format_stud*** принимает в качестве входных данных имя файла, содержащего программу на языке Си. Текст программы форматируется в соответствии с одним из существующих стилей кодирования (например Linux kernel coding style).

Раздел 3. *Полнотекстовый поиск по шаблону*

Полнотекстовый поиск — поиск документа в базе данных текстов на основании содержимого этих документов, а также совокупность методов оптимизации этого процесса. Необходимо разработать ПО выполняющее полнотекстовый поиск строк по шаблону в файлах, находящихся в указанной директории. Если некоторая строка S с номером N в файле FILE подходит по шаблону поиска, то на стандартный вывод (дисплей) выводится сообщение в следующем формате:

FILE (N): S

Имя директории и шаблон передаются в качестве аргументов командной строки. В шаблоне обязательно предусмотреть следующие элементы:

1) точное соответствие (например "abcd" => поиск в файле подстроки "abcd");
2) любой символ (один) - в команде grep это ".";
3) любая последовательность символов - в команде grep ".*";
4) экранирование служебных символов (в выше приведенных примерах – "*",".").

Для поиска точных соответствий необходимо использовать алгоритм соответствующий варианту.

Пример:

Пусть задан следующий шаблон:

"H.llo.*ld"

И дан файл some_text_file.txt (единственный в директории dir) содержащий следующие 4 строки:

Hello world
Hello is a word
I say Hello to this world!
Merry christmas
Hallo world

Шаблону удовлетворяют строки 1,3 и 5.

Вызов команды поиска должен выглядит следующим образом:

command "H.llo.*ld" dir

Результат работы должен быть следующим:

some_text_file.txt (1): Hello world
some_text_file.txt (3): I say Hello to this world!
some_text_file.txt (5): Hallo world

7	Простейший алгоритм поиска строк в подстроке

8	Алгоритм Рабина-Карпа поиска строк в подстроке
9	Алгоритм Бойера-Мура поиска строк в подстроке

Раздел 4. *Полнотекстовый поиск по шаблону с индексацией*

Полнотекстовый поиск — поиск документа в базе данных текстов на основании содержимого этих документов, а также совокупность методов оптимизации этого процесса.

Необходимо реализовать полнотекстовый поиск по шаблону в файлах определенной директории с предварительной обработкой (индексацией).

Имя директории и шаблон передаются в качестве аргументов командной строки. В шаблоне обязательно предусмотреть следующие элементы:

1) точное соответствие (например "abcd" => поиск в файле подстроки "abcd");
2) любой символ (один) - в команде grep это ".";
3) любая последовательность символов - в команде grep ".*";
4) экранирование служебных символов (в выше приведенных примерах – "*",".").

Предварительная обработка директории осуществляется командой ***index_dir***. В процессе предварительной обработки файлы в директории разбиваются на слова и формируются таблица индексной информации:

Таблица файлов содержит файлы, которые были индексированы и их уникальные (целочисленные) идентификаторы (FID — File ID — File identifier)

Имя файла	FID
filename1	1
filename2	2
.....	
filenameN	N

Таблица слов содержит все слова, найденные в файлах и информацию о файле и номере строки, в котором она найдена. При реализации необходимо использовать 3-х мерную структуру, построенную на базе списков. Первое измерение — элементы слова (wordX), второе — FID-ы файлов, содержащих соответствующее слово wordX, третье — номера строк в файле (FIDY), где найдено слово (wordX).

Пример

Слово	2-мерная структура, содержащая информацию обо всех файлах (по их FID) которые содержат слово из графы "Слово".
Hello	**FID1**(Одномерная структура, содержащая информацию о номерах строк в файле FID1 в которых присутствует слово "Hello"), **FID2**(Одномерная структура, содержащая информацию о номерах строк в файле FID2 в которых присутствует слово "Hello"),.....
.....	
this	**FIDX**(Одномерная структура, содержащая информацию о номерах строк в файле FIDX в которых присутствует слово "this"), **FIDY**(Одномерная структура, содержащая информацию о номерах строк в файле FIDY в которых присутствует слово "this"),.....

Непосредственно поиск выполняется программой ***search_dir***, которая сначала выполняет поиск точных соответствий в индексной информации и на основе данных определяет файлы и строки в них, в которых потенциально может находиться искомая последовательность. Далее в соответствии с алгоритмом поиска подстроки в строке (определяется вариантом) производится окончательный поиск.

10	Простейший алгоритм поиска строк в подстроке

11	Алгоритм Рабина-Карпа поиска строк в подстроке
12	Алгоритм Бойера-Мура поиска строк в подстроке

Раздел 5. *Проверка целостности информации*
Реализовать систему проверки целостности файлов. В данную систему входит 2 команды: ***gen_intergity*** **и** ***check_integrity*****.** Задача данного программного комплекса в создании базы данных целостности (выполняется с помощью ***gen_integrity***) и последующей проверке целостности директории (с помощью ***check_integrity***).
gen_integrity – создает БД целостности. В качестве параметров ей передаются: имя сканируемой директории и имя файла, где сохраняется БД целостности.
check_integrity – проверяет указанную директорию на целостность в соответствии с указанной БД целостности. При обнаружении нарушений целостности – сообщает пользователю.
Для генерации хеш кодов MD5 рекомендуется использовать исходные коды, расположенные на сайте Кафедры ВС (http://csc.sibsutis.ru/files/File/pavu/MD5.tar.bz2) или реализацию в библиотеке OpenSSL (описание ее использования расположено по адресу: http://www.firststeps.ru/linux/r.php?18.

13	При создании базы данных целостности применять вычисление хеш-функции MD5 для каждого из защищаемых файлов. Если при проверке вычисленных хеш-код не совпадает с тем что был сохранен в БД целостности — обнаружен факт модификации файла.

Раздел 6. *Помехоустойчивое кодирование*
Необходимо реализовать систему помехоустойчивой передачи данных через ненадежный канал связи. Канал связи имитируется. Алгоритмы кодирования/декодирования определяется вариантом. Процесс передачи данных выглядит следующим образом:
1) Команда ***code_message*** кодирует сообщение из файла, имя которого передается ей как аргумент командной строки. Закодированное сообщение записывается в выходной файл с именем ***channel***.
2) Команда ***noise_emulation*** вносит помеху в указанный бит сообщения в ***channel*** (также аргументы ком. Строки). Указанный бит инвертируется (заменяется на противоположный). Даная команда может быть вызвана несколько раз для имитации нескольких ошибок.
3) Команда ***decode_message*** считывает сообщение из файла ***channel*** и декодирует его.

14	Код Хэмминга с устранением одного замещения.
15	Код Рида-Соломона

Раздел 7. *Сжатие данных*
Алгоритм сжатия определяется вариантом. Команда ***compress_stud*** принимают в качестве аргументов командной строки:
1) Режим работы: сжатие (c - compress), распаковка (d - decompress)
2) Имя входного файла (в режиме сжатия это исходный файл, в режиме распаковки — сжатый)
3) Имя выходного файла(в режиме сжатия это сжатый файл, в режиме распаковки — исходный)

16	Алгоритм Шенона-Фано (Shannon-Fano)
17	Коды Хаффмана (Huffman)
18	Алгоритм Зива-Лемпела (Ziv-Lempel) LZ77
19	Алгоритм Зива-Лемпела (Ziv-Lempel) LZSS
20	Алгоритм Зива-Лемпела (Ziv-Lempel) LZ78