Практические задания к лабораторным работам по дисциплине «Низкоуровневое программирование»

Общие требования к программной реализации работ

Для реализации необходимо использовать язык программирования Си, если в описании к лабораторной работе не сказано иное.

Исходный код лабораторных работ размещать в репозитории Gitlab факультета ПИиКТ.

Также необходимо следовать данным пунктам:

1. **«Нет» статичности.** Все структуры данных должны допускать создание множества их экземпляров.
2. **«Нет» «магическим» константам.** Все значения должны либо вычисляться из обрабатываемых программой данных, либо задаваться с помощью аргументов командной строки или конфигурационных файлов.
3. **«Нет» бесконечным циклам.** Все циклы должны иметь понятные условия выхода: не допускается использовать, например, while (true), for (; ;) и т.д.
4. **«Нет» утечке ресурсов.** Все ресурсы, которые были использованы в программе и требуют освобождения (закрытия), должны корректно освобождаться (закрываться) независимо от возникновения ошибочных ситуаций или исключений. Например, открытый файл должен быть закрыт после того, как он перестал использоваться в программе; аллоцированная вручную память обязательно должна освобождаться.
5. **«Нет» неожиданным завершениям программы.** Все процессы, нити (threads) должны корректно завершаться в результате выполнения работы, а не прерываться функциями вида Abort/Exit.
6. **«Нет» побайтовому вводу-выводу.** Все данные должны обрабатываться частями (блоками) известного размера, с учетом целесообразного размера буфера.

Настоятельно рекомендуется:

- В начале работы подготовить окружение разработчика, включающее отладчик, поддерживающий визуализацию структур данных и возможность отладки программ, выполняющихся под управлением ОС семейств Windows и *NIX.
- Сборку проекта осуществлять с помощью кроссплатформенных средств автоматизации, таких как мэйкфайлы.
- Следовать общим принципам грамотной разработки ПО, таким как SOLID, DRY, и др., грамотно использовать непрозрачные типы данных (opaque data types), разделять публичную и приватную функциональность модулей.

Оформление отчетов

По каждому из заданий должен быть представлен отчет, содержащий следующие части:

1. Цели – описание цели задания (см. текст задания)
2. Задачи – путь достижения цели, что именно нужно было сделать для выполнения задания (план хода вашей работы)
3. Описание работы – внешнее описание созданной программы, состава модулей, способов её использования, примеры входной и выходной информации (модули, интерфейсы, тесты)
4. Аспекты реализации – внутреннее описание созданной программы, особенности алгоритмов, примеры кода
5. Результаты – что было сделано для выполнения задач кратко по пунктам
   (созданные артефакты, результаты тестов, количественные оценки)
6. Выводы – что было достигнуто в отношении цели задания.
   (что показали тесты, почему, как это было достигнуто, чему научились, качественные оценки)

Описание заданий

Задание 1

Создать модуль, реализующий хранение в одном файле данных (выборку, размещение и гранулярное обновление) информации общим объёмом от 10GB соответствующего варианту вида.

Порядок выполнения:

1. Спроектировать структуры данных для представления информации в оперативной памяти
    a. Для порции данных, состоящий из элементов определённого рода (см форму данных), поддержать тривиальные значения по меньшей мере следующих типов: цетырёхбайтовые целые числа и числа с плавающей точкой, текстовые строки произвольной длины, булевские значения
    b. Для информации о запросе
2. Спроектировать представление данных с учетом схемы для файла данных и реализовать базовые операции для работы с ним:
    a. Операции над схемой данных (создание и удаление элементов схемы)
    b. Базовые операции над элементами данными в соответствии с текущим состоянием схемы (над узлами или записями заданного вида)
        i. Вставка элемента данных
        ii. Перечисление элементов данных
        iii. Обновление элемента данных
        iv. Удаление элемента данных
3. Используя в сигнатурах только структуры данных из п.1, реализовать публичный интерфейс со следующими операциями над файлом данных:
    a. Добавление, удаление и получение информации о элементах схемы данных, размещаемых в файле данных, на уровне, соответствующем виду узлов или записей
    b. Добавление нового элемента данных определённого вида
    c. Выборка набора элементов данных с учётом заданных условий и отношений со смежными элементами данных (по свойствам/полями/атрибутам и логическим связям соответственно)
    d. Обновление элементов данных, соответствующих заданным условиям
    e. Удаление элементов данных, соответствующих заданным условиям
4. Реализовать тестовую программу для демонстрации работоспособности решения
    a. Параметры для всех операций задаются посредством формирования соответствующих структур данных
    b. Показать, что при выполнении операций, результат выполнения которых не отражает отношения между элементами данных, потребление оперативной памяти стремится к O(1) независимо от общего объёма фактического затрагиваемых данных
    c. Показать, что операция вставки выполняется за O(1) независимо от размера данных, представленных в файле
    d. Показать, что операция выборки без учёта отношений (но с опциональными условиями) выполняется за O(n), где n – количество представленных элементов данных выбираемого вида
    e. Показать, что операции обновления и удаления элемента данных выполняются не более чем за $O(n*m) > t \rightarrow O(n+m)$, где n – количество представленных элементов данных обрабатываемого вида, m – количество фактически затронутых элементов данных
    f. Показать, что размер файла данных всегда пропорционален количеству фактически размещённых элементов данных
    g. Показать работоспособность решения под управлением ОС семейств Windows и *NIX
5. Результаты тестирования по п.4 представить в составе отчёта, при этом:
    a. В части 3 привести описание структур данных, разработанных в соответствии с п.1
    b. В части 4 описать решение, реализованное в соответствии с пп.2-3
    c. В часть 5 включить графики на основе тестов, демонстрирующие амортизированные показатели ресурсоёмкости по п. 4

Задание 2

Использовать средство синтаксического анализа по выбору, реализовать модуль для разбора некоторого подмножества языка запросов по выбору в соответствии с вариантом формы данных. Должна быть обеспечена возможность описания команд создания, выборки, модификации и удаления элементов данных.

Порядок выполнения:

1. Изучить выбранное средство синтаксического анализа
    a. Средство должно поддерживать программный интерфейс совместимый с языком C
    b. Средство должно параметризоваться спецификацией, описывающий синтаксическую структуру разбираемого языка
    c. Средство может функционировать посредством кодогенерации и/или подключения необходимых для его работы дополнительных библиотек
    d. Средство может быть реализовано с нуля, в этом случае оно должно быть основано на обобщённом алгоритме, управляемом спецификацией
2. Изучить синтаксис языка запросов и записать спецификацию для средства синтаксического анализа
    a. При необходимости добавления новых конструкций в язык, добавить нужные синтаксические конструкции в спецификацию (например, сравнения в GraphQL)
    b. Язык запросов должен поддерживать следующие возможности:
        - Условия
            - На равенство и неравенство для чисел, строк и булевских значений
            - На строгие и нестрогие сравнения для чисел
            - Существование подстроки
        - Логическую комбинацию произвольного количества условий и булевских значений
        - В качестве любого аргумента условий могут выступать литеральные значения (константы) или ссылки на значения, ассоциированные с элементами данных (поля, атрибуты, свойства)
        - Разрешение отношений между элементами модели данных любых условий над сопрягаемыми элементами данных
        - Поддержка арифметических операций и конкатенации строк не обязательна
    c. Разрешается разработать свой язык запросов с нуля, в этом случае необходимо показать отличие основных конструкций от остальных вариантов (за исключением типичных выражений типа инфиксных операторов сравнения)
3. Реализовать модуль, использующий средство синтаксического анализа для разбора языка запросов
    a. Программный интерфейс модуля должен принимать строку с текстом запроса и возвращать структуру, описывающую дерево разбора запроса или сообщение о синтаксической ошибке
    b. Результат работы модуля должен содержать иерархическое представление условий и других выражений, логически представляющие собой иерархически организованные данные, даже если на уровне средства синтаксического анализа для их разбора было использовано линейное представление
4. Реализовать тестовую программу для демонстрации работоспособности созданного модуля, принимающую на стандартный ввод текст запроса и выводящую на стандартный вывод результирующее дерево разбора или сообщение об ошибке
5. Результаты тестирования представить в виде отчёта, в который включить:
    a. В части 3 привести описание структур данных, представляющих результат разбора запроса
    b. В части 4 описать, какая дополнительная обработка потребовалась для результата разбора, представляемого средством синтаксического анализа, чтобы сформировать результат работы созданного модуля
    c. В части 5 привести примеры запросов для всех возможностей из п.2.b и результирующий вывод тестовой программы, оценить использование разработанным модулем оперативной памяти

Варианты заданий

Вариант	Форма данных	Язык запросов
1	Документное дерево	XPath
2	Реляционные таблицы	SQL
3	Граф узлов с атрибутами	GraphQL
4	Документное дерево	GraphQL
5	Реляционные таблицы	AQL
6	Граф узлов с атрибутами	Gremlin
7	Документное дерево	MongoShell
8	Реляционные таблицы	LINQ
9	Граф узлов с атрибутами	Cypher
10	Документное дерево	XPath
11	Реляционные таблицы	SQL
12	Граф узлов с атрибутами	GraphQL

Расшифровка формы данных

	Документное дерево	Реляционные таблицы	Граф узлов с атрибутами
Организация элементов данных	Дерево узлов, несущих свойства	Таблицы записей, несущих поля	Граф узлов, несущих атрибуты
Способ реализации отношений	Материализованы в представлении родитель-ребенок	Не материализованы, через выражения запросов по значениям	Материализованы в представлении связей между узлами
Примеры	Json, Xml, registry, прикладной уровень файловых систем	РСУБД, метаданные бинарных исполняемых модулей	Семантические сети, сетевые схемы
Состав схемы данных	Виды узлов, виды значений в узлах	Виды записей таблиц, виды значений в полях	Виды узлов, виды связей
Состав модели фильтра данных	Условия по содержимому элементов данных и отношениям между ними		
Примеры языков запросов	XPath, XQuery	SQL, LINQ	GraphQL, Cypher

Описание примеров реализации

Организация страниц и данных в реестре Windows: https://github.com/msuhanov/regf/blob/master/Windows%20registry%20file%20format%20specification.md

Организация данных в графовом хранилище Neo4j: https://neo4j.com/developer/kb/understanding-data-on-disk/

Организация страниц в MSSQL: https://docs.microsoft.com/en-us/sql/relational-databases/pages-and-extents-architecture-guide?view=sql-server-2017

Организация страниц в Postgre: https://www.postgresql.org/docs/9.4/storage-page-layout.html

Кучи данных в MSSQL: https://docs.microsoft.com/en-us/sql/relational-databases/indexes/heaps-tables-without-clustered-indexes?view=sql-server-2017#heap-structures

Организация данных в Postgre: http://rachbelaid.com/introduction-to-postgres-physical-storage/, http://www.interdb.jp/pg/pgsql01.html

Организация данных в MongoDb: https://www.quora.com/What-is-the-internal-file-structure-for-the-collection-in-MongoDB, https://docs.mongodb.com/manual/core/storage-engines/, http://bsonspec.org/spec.html

Графы в MSSQL: https://docs.microsoft.com/en-us/sql/relational-databases/graphs/sql-graph-architecture?view=sql-server-2017

Примеры исходного кода к этапам

Задание 1, п.2, пример тестовой программы для базовых операций над файлом данных реляционной модели

```
void checkCreateTables() {
	char *filename = "/home/ged/projects/TableStoreNix/test.tabs.data";

	tableStoreTableSchemaInfo *tabSchema1 = tableStoreInitTableSchema();
	tableStoreExpandTableSchema(tabSchema1, "col1", TABLE_STORE_TYPE_INT32, 1);
	tableStoreExpandTableSchema(tabSchema1, "col2", TABLE_STORE_TYPE_BOOL, 1);
	tableStoreExpandTableSchema(tabSchema1, "col3", TABLE_STORE_TYPE_STRING, -1);
	tableStoreExpandTableSchema(tabSchema1, "col4", TABLE_STORE_TYPE_SINGLE, 1);

	tableStoreTableSchemaInfo *tabSchema2 = tableStoreInitTableSchema();
	tableStoreExpandTableSchema(tabSchema2, "id", TABLE_STORE_TYPE_INT32, 1);
	tableStoreExpandTableSchema(tabSchema2, "str", TABLE_STORE_TYPE_STRING, -1);

	tableStoreFileHandle *f = tableStoreFileOpenOrCreate(filename);

	tableStoreTableHandle *t1 = tableStoreFileCreateTable(f, "tab1", tabSchema1);
	tableStoreFileSeekTable(t1, 0, SEEK_SET);
	for (int i = 0; i < 100; i++) {
		int b = i % 2 == 0;
		char s[100];
		float f = 1.0f / i;
		sprintf(s, "some data [#%d, %s, %f]", i, (b?"T":"F"), f);
		void *p[] = { &i,  &b,  s, &f };
		tableStoreRecordData rec = { 0, p, 0 };
		tableStoreFileAddRecord(t1, &rec);

		printf("[%d] %d, %s, %s, %f\n",
			i,
			*(int*)rec.data[0],
			(*(int*)rec.data[1] ? "T" : "F"),
			(char*)rec.data[2],
			*(float*)rec.data[3]
		);
	}
	tableStoreFileCloseTable(t1);

	// TODO test tab2

	tableStoreFileClose(f);
}

void checkReadTables() {
	char *filename = "/home/ged/projects/TableStoreNix/test.tabs.data";

	tableStoreFileHandle *f = tableStoreFileOpenOrCreate(filename);

	tableStoreTableHandle *t1 = tableStoreFileOpenTable(f, "tab1");
	tableStoreRecordData *rec = tableStoreFilePrepareRecordDataStructure(t1);
	tableStoreFileSeekTable(t1, 0, SEEK_SET);
	for (int i = 0; i < t1->schema->recordsCount; i++) {
		tableStoreFileReadRecord(t1, rec);
		printf("[%d] %d, %s, %s, %f\n",
			i,
			*(int*)rec->data[0],
```

```
                        (*(int*)rec->data[1] ? "T" : "F"),
                        (char*)rec->data[2],
                        *(float*)rec->data[3]
                );
                tableStoreFileSeekTable(t1, 1, SEEK_CUR);
        }
        tableStoreFileCleanupRecordDataStructure(rec);
        tableStoreFileCloseTable(t1);

        // TODO test tab2

        tableStoreFileClose(f);
}

int main(int argc, char** argv)
{
        (void)&argc;
        (void)&argv;

        if (argc > 1) {
                if (strcmp(argv[1], "-w") == 0) {
                        // checkWrite();
                        checkCreateTables();
                }
                if (strcmp(argv[1], "-r") == 0) {
                        // checkRead();
                        checkReadTables();
                }
        }
}
```