User Manual Document

By:

Jacob Hopkins

Misky Abshir

Tyler Willard

Anuska Shrestha

Sequence Set Group Project

Objective: Generate a sequence set as a file composed of (doubly) linked blocks of fixed-sized records comprising fixed-sized fields.

**Class Name**

SequenceSet

**Public Methods**

SequenceSet(); //! default constructor.

SequenceSet(int b\_size, int r\_size, float d\_cap, std::string i\_filename, std::string o\_filename); //! copy constructor

~SequenceSet(); //! destructor

void create(); /\*! function prototype for create() that creates empty file for the header any it contains \*/

void load(); /\*! function prototype for load() that load block of sequence set file into ram \*/

void close(); /\*! function prototype for close() that is called when file needs to be closed \*/

bool is\_empty(int flag, int block, int record, int field); /\*! prototype for is\_empty() to know the state of the structure \*/

std::vector<int> search(std::string search\_term); /\*! function prototype for search(string) to search for specific record in the file from user input \*/

std::string get\_field\_from\_record(int field, int record, int block);

void populate(); /\*! function prototype for populate() that creates an empty node for a btree \*/

void insert(std::string new\_record); /\*! function prototype for insert(strint) that inserts a new record into the file from user input \*/

void delete\_record(int block, int record); /\*! function prototype for delete\_record(int, int) that deletes specific record from user input \*/

void update(int block, int record, int field, std::string new\_field); /\*! function prototype for update(int, int, string) that updates a record, field or adds new field \*/

void display\_record(int record, int block); /\*! function prototype display\_record(int, int) displays specific record request by user input \*/

void display\_field(int field, int record, int block); /\*! function prototype display\_field(int, int, int) displays specific field request by user input \*/

void display\_file(int limit); /\*! function prototype display\_file(int) displays file request by user input \*/

void display\_SS(); /\*! function prototype display\_SS() to display the sequence set \*/

void validate(); /\*! function prototype validate() to validate a record in the file \*/

//void addIndex(int primKey, Block \*b); /\*! function prototype addIndex(int, Block) that adds an index in a record \*/

//void delIndex(int primKey); /\*! function prototype delIndex(int) that removes an index in a record \*/

void developer\_show(); /\*! function prototype developer\_show() that creates the columns the record will be diplayed into\*/

int search\_file(int primKey); /\*! function prototype search\_file(int) searches for a file \*/

std::vector<int> get\_field\_range\_tuple(int field\_index); /\*! function prototype get\_field\_range\_tuple(int) for extracting the range of character index in a record\*/

void nsew\_most(std::string state);

void state\_and\_place\_from\_zip(std::string zip);

**Compile Class**

Simple Compiler:

G++ main.cpp [Give it a name].exe

[Whatever name you gave].exe

Our Case

**Examples of Public Methods**

int main(int arg\_count, char\*\* arg\_values){

/\*

Here we declare and initialize the sequence set data. This will call load().

\*/

SequenceSet data;

data.create(); //!

data.populate(); //!

data.display\_field();

data.display\_record(); //works most of the time, with the occasional exit

data.display\_file(); //this works great

data.display\_SS(); //this works great

data.developer\_show(); //this works great

std::vector<int> loc = data.search(data.get\_field\_from\_record(0,0,0));

std::cout << "\n" << data.get\_field\_from\_record(0,0,0) << "\nBlock:\t" << std::to\_string(loc[0]) << "\nRecord:\t" << std::to\_string(loc[1]) << "\n";

loc = data.search(data.get\_field\_from\_record(4,1,0));

std::cout << "\n" << data.get\_field\_from\_record(4,1,0) << "\nBlock:\t" << std::to\_string(loc[0]) << "\nRecord:\t" << std::to\_string(loc[1]) << "\n";

loc = data.search(data.get\_field\_from\_record(4,1,1));

std::cout << "\n" << data.get\_field\_from\_record(4,1,1) << "\nBlock:\t" << std::to\_string(loc[0]) << "\nRecord:\t" << std::to\_string(loc[1]) << "\n";

loc = data.search("42.1934"); // from line 28

std::cout << "\n42.1934" << "\nBlock:\t" << std::to\_string(loc[0]) << "\nRecord:\t" << std::to\_string(loc[1]) << "\n";

loc = data.search("yeeeet"); // from line 28

std::cout << "\nyeeeet" << "\nBlock:\t" << std::to\_string(loc[0]) << "\nRecord:\t" << std::to\_string(loc[1]) << "\n";

data.insert("");

data.update(0,0,0,"12345");

data.update(0,0,1,"12345");

data.update(0,0,2,"12345");

data.update(0,0,3,"12345");

data.update(0,0,4," 12.345");

data.update(0,0,5,"-12.345");

data.display\_SS();

data.validate();

**Test Driver Program**

Upon running the program:

1. You will be prompted with a menu. Begin by choosing which option you would like to test first
   1. The test will run, and if it works, you will be brought back to the menu. If it doesn’t work, the tester will break. (We wanted to add messages to display what happened, but we ran out of time. We would have had it display what was tested, if it worked, and other information related to the test)
2. After the test runs, you will be prompted to choose another test. Repeat the steps to run all tests.
3. Once complete, enter “x” to terminate the program.

**Known Errors and Deficiencies**

SequenceSet:

Tester:

The tester doesn’t have user friendly descriptions regarding the test results. It doesn’t give feedback on why a test failed, which we would have loved to add.

Main – a program to use and demonstrate the class SequenceSet

1. Takes in arguments C:\GroupProject>./main.exe [a/b] [input:{NY/30005}]

First argument

a or b

A: find the furthest zipcode by state

Second arg is the state “NY”

B: show state name and place name by zip code

Second arg is zip code “30005

2. We initialize the default constructor to a Sequence set called data

Constructors call load()

3. We call create which makes the output file and writes the header to the file.

4. We call load to

4. We call populate

5. Based on the arguments we call the function to return the Northernmost, Southernmost, Easternmost, and Westernmost zip code for a specified state

Or using a different set of command line flags, will display the State and Place Name for a specified Zip Code or set of zip codes.

Wait for the users enter to pause the output

Return