# Program Description

My program has been created for the storing of hydrological information from boreholes, for the purpose of monitoring the salinity of groundwater in an area. Each entry in the database stores information for a particular sample taken from a borehole.

Header file:CBorehole.h

The header file stores the class which contains the relevant data for a particular borehole and is used to create the linked list in which each entry is for a water sample. The class stores a sample ID (which is unique), a borehole ID (i.e. which borehole the sample came from), latitude and longitude of the borehole, the date that the sample was taken and the salinity of the sample. The class CLBorehole stores all this but also has the variable next, which is a pointer of type CLBorehole, which is used to point to the next item in the list.

Beginning of the program

When the program is first run it gives the user two options: either to create a new data file, or to load one that currently exists by asking for an input of “1”(create a new data file) or “2”(load a file that currently exists) from the user.

Creating a new file

If the user wishes to create a new file the function then calls the add function with arguments start and current. The add function has two arguments of type CLBorehole, “lstart” which is for the start of the list and “lcurrent” which is to store the current item in the list. The values are passed by reference so that the global variables “current” and “start” are changed. The add function allocates a chunk of memory of type CLBorehole to “lcurrent”, and asks the user to input data for each of the variables located in CLBorehole, and assigns these values to the corresponding variable in “lcurrent”. “lcurrent” is added to the start of the list and “lstart” is set to point at it, and the user is asked if they wish to enter another entry. If “y” is entered then the process repeats. Else the file is saved(See below under subtitle “saving a file”), and the new file data is displayed using the print\_all function (see below under subtitle print\_all).

Saving a file

When the function save\_file is called then the list that is indicated by the pointer “start” is saved. The function asks for a file name for the user, and then opens this file for writing to in binary. If the file is unable to be opened then the user is given the option to retry saving, and save\_file is called again. When the file has been opened successfully the list is run through once and the number of items in the list is counted, and this is stored in the variable num, and is the first thing written to file. This is used by load file so that it knows how many items there are in the list before it is loaded. Then the list is run through again, this time writing the data to file for each of the entries in the list.

Loading a file

If the user wishes to load a currently existing file, then the name of the file is requested. This is then input by the user and load\_list is called with the file name as the argument. The file is then opened for reading in binary, and the first thing that is read is the number of entries in the list stored in that file (variable “num”). Then each of the variables in current is set to the corresponding value in the file. This is done “num” times, thereby loading all the entries in the list. The data is then printed to screen using the print\_all function.

print\_all function

This firstly prints out a header, and then loops through the list pointed to by “start”, and prints out each entry. It then prints out a menu containing options for the user. These options are to search the data, delete the data, add new data or quit. If searching the data is selected then the user has the option of searching by a particular borehole or by year. This calls the function search\_borehole which searches through the entire list, and when it finds the borehole the user was searching for, prints it to the screen. search\_year does exactly the same, but instead searches for the year that the user wanted to find. Once these have been printed the user then must press enter, and the print\_all function is called again and the user is returned to the whole database.hen

If the user wishes to delete the data then they must enter the unique sample ID that they wish to delete. The list is then searched through until this value is found, and the function delete\_data() is called on it. This takes in “start” and “current” by reference and removes the item that the user wishes to delete. Then the option to save the data is given to the user, and then the altered database is printed to the screen using print\_all.

If the user wishes to add data to the list the function add\_data is called on “start” and “current”, which, after the data has been altered, asks the user if they wish to save the data. Then print\_all is called and the altered database is displayed on the screen.

The final option that is given to the user is to exit the program.