main.cpp

```
/*

    TodX - The cool ToDo app
    Copyright (C) 2016  adiultra

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.

*/


#include <fstream>
#include <iostream>
#include <cstring>

// some local files
#include "fabric.h"
#include "search.cpp"
#include "export.cpp"

using namespace std;


// Variables used throughout the program

List arrayL[20];
List currentL;

int _arrayLindex = 0;
int _currentLindex;
int isOpenL = 0; // For checking wether a list is open or not

char filename[20] = "data.tdx";

// Fxns Used

void    initiate();         // Initiate the data before user inteface
int     parse(char []);     // run the command entered by the users
void    stats();           // display stats of opened list
void    openL(int);         // open the list specified by index
void    displayL();         // display all the lists
int     confirm();          // ask for confirmation (y/n)
void    empty();            // remove data from the program & data-file
void    finish();           // save the data to the data-file
```

```
void initiate() {
    // Fxn to read data from external file int the array of Lists
    fstream file(filename, ios::in|ios::binary);
    List tempOb; // temperary object

    while (file.read((char*)&tempOb, sizeof(tempOb))) {
        arrayL[_arrayLindex++] = tempOb;
    }

    file.close();
}

int parse(char command[80]){
    // Fxn to Parse(execute) the command passed
    /// Here a crazy fun expression is used in if () {} to compare strings
    /// Since we wanted any of them to be zero, hence, they were multiplied
    /// to give zero. Mathematically it is quite accurate and syntactically
    /// beautiful.

    int success = 0; // to mark wether a command completed successfully

    if (!(strcmp(command, "new") * strcmp(command, "n"))) {
        // Create a new List
        arrayL[_arrayLindex++].enter();
        success = 1;
    }

    else if (!(strcmp(command, "stats") * strcmp(command, "stat"))) {
        // Display the stats
        stats();
        success = 1;
    }

    else if (!(strcmp(command, "open") * strcmp(command, "o"))) {
        // Open existing lists
        if (_arrayLindex) {
            if (isOpenL) {
                arrayL[_currentLindex] = currentL;
                finish();
            }

            int choice;
            cout << "Enter The No. of List to be opened" << endl;

            displayL();

            cout << Bblue << " #> " << normal;
            cin >> choice;
            cin.ignore();

            while ( !(0 <= choice && choice < _arrayLindex) ) {
                cout << "Invalid choice choose again" << endl;
                cout << Bblue << " #> " << normal;
                cin >> choice;
                cin.ignore();
            }

            openL(choice);

            isOpenL = 1;
```

```
        }

        else {
            cout << "No List found, create a new one with \'n\' or \'new\'" << endl;
        }

        success = 1;
    }

    else if (!(strcmp(command, "append") * strcmp(command, "a"))) {
        // Append a new Todo to the currently opened list
        if (isOpenL) {
            currentL.append();
        }

        else {
            cout << "No List is opened, Open a list first" << endl;
        }
        success = 1;
    }

    else if (!(strcmp(command, "mark") * strcmp(command, "m"))) {
        // Mark/(change status) a Todo with a given charater
        if (isOpenL) {
            int choice;
            currentL.indexView();

            cout << "Choose the Todo to mark : " << endl << Bblue << " #> " << normal;
            cin >> choice;

            while (choice >= currentL._listIndex || choice < 0) {
                cout << "Invalid Choice Try again" << endl << Bblue << " #> " << normal;
                cin >> choice;
            }

            char status;

            cout << "Enter The new Status" << endl << green << " +> " << normal;
            cin >> status;
            cin.ignore();

            currentL.changeStatus(choice, status);
        }

        else {
            cout << "No List is opened, Open a list first" << endl;
        }

        success = 1;
    }

    else if (!(strcmp(command, "addtag") * strcmp(command, "addt"))) {
        // Append a new Todo to the currently opened list
        if (isOpenL) {
            char newTag[40];

            cout << "Enter the New Tag " << endl << green << " +> " << normal;
            cin.getline(newTag, sizeof(newTag));
            currentL.addTag(newTag);
        }
```

```
        else {
            cout << "No List is opened, Open a list first" << endl;
        }

        success = 1;
    }

    else if (!(strcmp(command, "view") * strcmp(command, "v"))) {
        // view todo's of the current list
        if (isOpenL) {
            currentL.view();
        }

        else {
            cout << "No List is opened, Open a list first" << endl;
        }

        success = 1;
    }

    else if (!(strcmp(command, "iview") * strcmp(command, "iv"))) {
        // view todo's of the current list with index
        if (isOpenL) {
            currentL.indexView();
        }

        else {
            cout << "No List is opened, Open a list first" << endl;
        }

        success = 1;
    }

    else if (!(strcmp(command, "search") * strcmp(command, "srch"))) {
        // Search the Database
        char searchTerm[40];

        cout << "Enter the search term " << endl << green << " +> " << normal;
        cin.getline(searchTerm, sizeof(searchTerm));

        if (isOpenL) {
            arrayL[_currentLindex] = currentL;
        }

        search(searchTerm, arrayL);
        success = 1;
    }

    else if (!(strcmp(command, "save") * strcmp(command, "s"))) {
        // Save the data to the file
        if (isOpenL) {
            arrayL[_currentLindex] = currentL;
        }

        finish();
        success = 1;
    }

    else if (!(strcmp(command, "export") * strcmp(command, "exp"))) {
        // Export the data to a text file
        if (isOpenL) {
```

```
            arrayL[_currentLindex] = currentL;
        }

        Export(arrayL, _arrayLindex);

        success = 1;
    }

    else if (!(strcmp(command, "delete") * strcmp(command, "del"))) {
        // Delete Things

        char choice[10];
        cout << "What do you want to delete? (list/todo/tag)" << endl << Bred << " ?> " <<
        cin.getline(choice, sizeof(choice));

        if (!(strcmp(choice, "list") * strcmp(choice, "List"))) {
            int index;

            cout << "Enter the Index of List to Delete " << endl;
            displayL();

            cout << Bblue << " #> " << normal;
            cin >> index;
            cin.ignore();
            cout << "Are you sure, This cannot be undone, This will delete the List ->" <<
            arrayL[index].view();

            if ( confirm() ) {
                for (int i = index; i < _arrayLindex - 1; i++) {
                    arrayL[i] = arrayL[i+1];
                }
                _arrayLindex--;
            }
        }

        else if (!(strcmp(choice, "todo") * strcmp(choice, "Todo") * strcmp(choice, "ToDo")

            if (isOpenL) {
                int index;

                cout << "Enter the Index of Todo to Delete " << endl;
                currentL.indexView();

                cout << Bblue << " #> " << normal;
                cin >> index;
                cin.ignore();
                cout << "Are you sure, This cannot be undone, This will delete the ToDo ->"
                currentL.todoView(index);

                if ( confirm() ) {
                    currentL.removeTodo(index);
                }

                arrayL[_currentLindex] = currentL;  // To Help Improving Finalization
            }

            else {
                cout << "No List is opened, Open a list first" << endl;
            }
        }
```

```
        else if (!(strcmp(choice, "tag") * strcmp(choice, "tags") * strcmp(choice, "Tag")))

            if (isOpenL) {
                int index;

                cout << "Enter the Index of Tag to Delete " << endl;
                currentL.tagIndexView();

                cout << Bblue << " #> " << normal;
                cin >> index;
                cin.ignore();
                cout << "Are you sure, This cannot be undone, This will delete the Tag ->"
                cout << currentL.tags[index] << endl;

                if ( confirm() ) {
                    currentL.removeTag(index);
                }

                arrayL[_currentLindex] = currentL;  // To Help Improving Finalization
            }

            else {
                cout << "No List is opened, Open a list first" << endl;
            }
        }

        finish();
        success = 1;
    }

    else if (!(strcmp(command, "qdelete") * strcmp(command, "qdel"))) {
        // Delete Todo of current list using index
        /// qdelete -> Quick Delete

        if (isOpenL) {
            int index;

            cout << "Enter the Index of List to Delete " << endl;
            currentL.indexView();

            cout << Bblue << " #> " << normal;
            cin >> index;
            cin.ignore();
            cout << "Are you sure, This cannot be undone, This will delete ->" << endl;
            currentL.todoView(index);

            if ( confirm() ) {
                currentL.removeTodo(index);
            }

            arrayL[_currentLindex] = currentL;
        }

        else {
            cout << "No List is opened, Open a list first" << endl;
        }

        finish();
        success = 1;
    }
```

```
    else if (!(strcmp(command, "clear") * strcmp(command, "clr"))) {
        // Refresh the data file
        // XXX WARN XXX -> Strictly, Not to be used By users, Deletes all Data

        if ( confirm() ) {
            ofstream file(filename, ios::trunc|ios::binary|ios::out);

            file.write("", sizeof(arrayL));
            file.close();

            empty();
            initiate();
        }

        success = 1;
    }

    else if (!(strcmp(command, "quit") * strcmp(command, "q"))) {
        // exit the program after saving it to the file

        if ( confirm() ) {
            if (isOpenL) {
                arrayL[_currentLindex] = currentL;
            }

            finish();
            success = -1;
        }

        else {
            success = 1;
        }
    }

    else if (!(strcmp(command, "help") * strcmp(command, "-h"))) {
        // View help
        char line[80];
        ifstream helpfile("help.txt");

        while (!helpfile.eof()) {
            helpfile.getline(line, sizeof(line));
            cout << line << endl;
        }

        helpfile.close();

        success = 1;
    }

    return success;
}

void stats(){
    // Display wether a list is open or not
    if (isOpenL) {
        cout << "This List is open : " << currentL.title << endl;
        cout << "Total No. of todos : " << currentL._listIndex << endl;
    }

    else {
        cout << "No list is open" << endl;
```

```
    }

    cout << "Total lists = " << _arrayLindex << endl;
}

void openL(int index) {
    currentL = arrayL[index];
    _currentLindex = index;

    cout << "Opened : " << currentL.title << endl;
    currentL.view();
}

void displayL() {
    for (int i = 0; i < _arrayLindex; i++) {
        cout << i << ". " << arrayL[i].title << endl;
    }
}

int confirm() {
    // can be used as if( confirm() ) /// Improves yes/no prompts
    char confm[10];
    cout << "Enter \'yes\' to continue" << endl << Bred << " ?> " << normal;
    cin.getline(confm, sizeof(confm));

    if (!(strcmp(confm, "yes") * strcmp(confm, "y"))) {
        return 1;
    }

    else{
        return 0;
    }
}

void empty() {
    List tarrayL[20];
    memcpy(arrayL, tarrayL, sizeof(arrayL));
    _arrayLindex = 0;
    isOpenL = 0;
}

void finish() {
    // Write the changes back to file
    ofstream file (filename, ios::out);
    for (int i = 0; i < _arrayLindex; i++) {
        file.write((char*)&arrayL[i], sizeof(arrayL[i]));
    }

    file.close();
}

int main() {
    initiate();

    cout << "=== --- --> \033[5;33;1m TodX \033[0;m <-- --- ===" << endl;
    cout << "Welcome to TodX the ultimate Todo list" << endl;
    cout << "v1.00a = Linux Build, docs at -> http://todx.rtfd.io" << endl;

    char command[80];

    while (1) {
```

```
        cout << yellow << "\n *> " << normal;
        cin.getline(command, sizeof(command));

        if (!cin) {
            cout << "Have a Great Day :)" << endl;
            break;
        }

        int result = parse(command);

        if(result > 0){
            continue;
        }

        else if (result == 0) {
            cout << "Command not found, try `help` for help" << endl;
        }

        else if (result == -1) {
            cout << "Data Saved, Have a Great Day :)" << endl;
            break;
        }

        else {
            cout << "Something went terribly wrong, we apologize :(" << endl;
        }
    }

    finish();

    return 0;
}
```