University: University Transylvania of Brasov
Faculty: Faculty of Electrical Engineering and Computer Science
Study program: Electrical Engineering and Computer Science in English - IEC

Project
Graph Search

Bănică Ninel-Valentin, Year I, 4LF521 A
01.06.2023

Table of Contents

SPECIFICATION DEFINING THE PROGRAM

DESCRIPTION OF THE THEME

- The project presents a simple Graph Search application with a rather complex menu functionality.

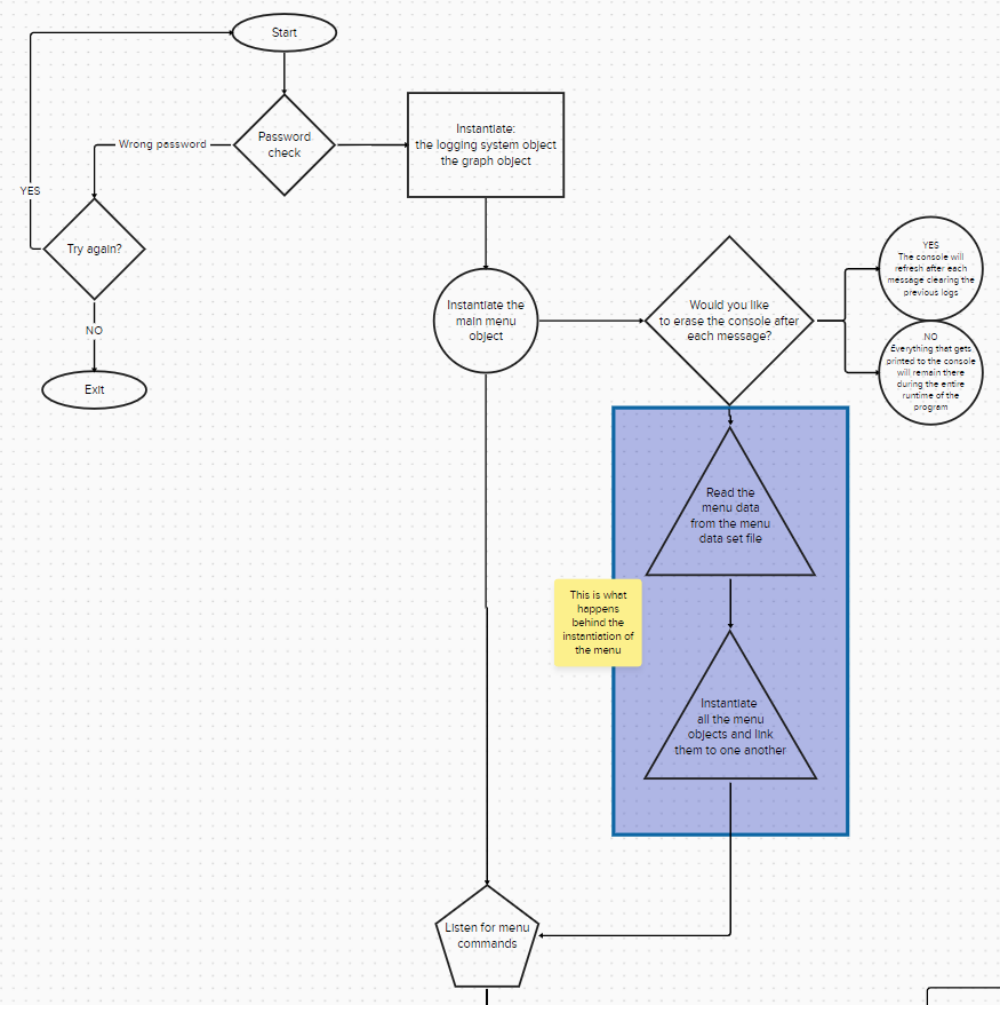
DESCRIPTION OF REQUIREMENTS

- The theme of the project mainly revolves around a very interactive and alive menu application approach, which gives access to an Unoriented Graph application.
- Algorithms used:
    - Bubble sort algorithm
    - Total Breadth First Search algorithm

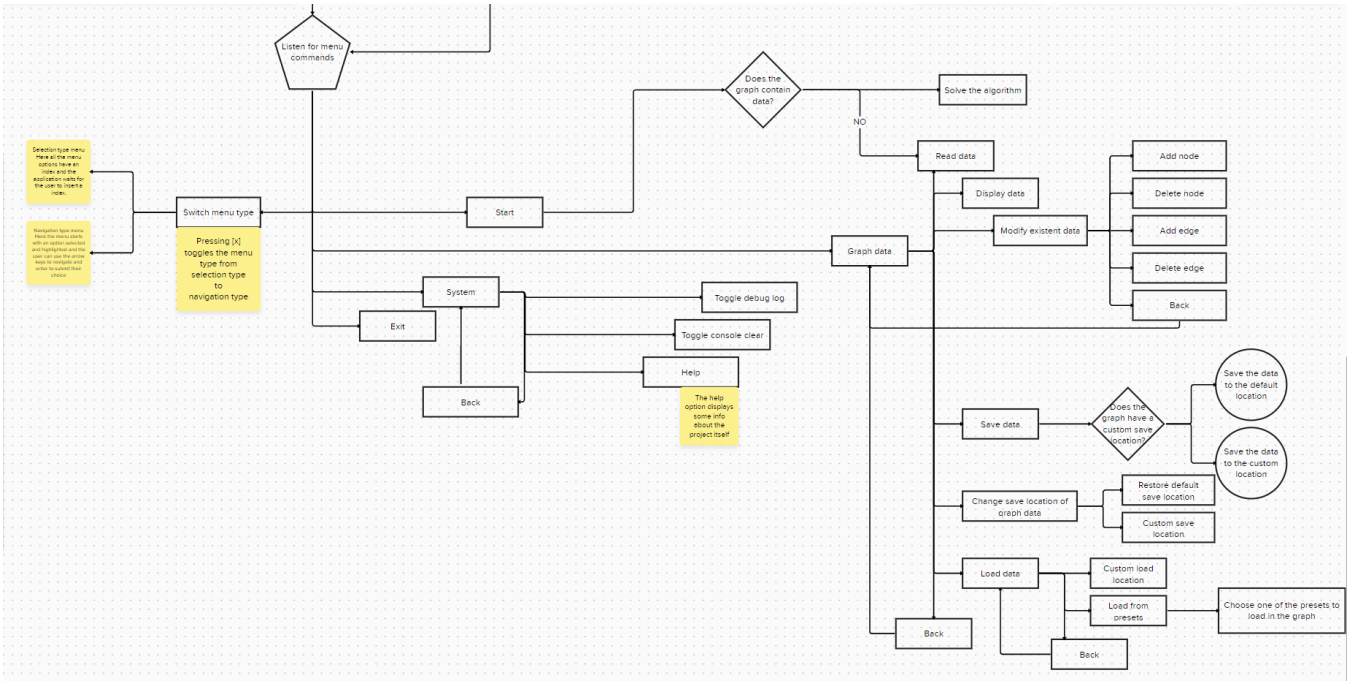
FUNCTIONAL SPECIFICATION

- Below I have created a flow-chart with the logic of the application:

This is available also at:
https://app.mural.co/t/graphsearchdiagrams6069/m/graphsearchdiagrams6069/1685432360195/d01463c6c3bd8cbf60606b2f9b25036415028e2e?sender=u8e371e2eba481a9946f12011



(Fig.1. Flow chart representing the beginning of the application)



(Fig.2. Flow chart representing the main application logic)

User Interface

The User Interface of the application is a rather user-friendly interface:
Firstly, when entering the application, the user is prompted to the following message:

```
Welcome to the Graph Search application!
Insert password:
```

Inserting the password correctly shows the following:

```
Welcome to the Graph Search application!
Insert password: **************
Correct password! Welcome!
Press any key to continue...
```

Inserting the password wrong shows the following:

```
Welcome to the Graph Search application!
Insert password: *
Wrong password provided... Want to try again? (Y/N)
```

Right after the password is correctly inserted, the application will prompt the following message as a question:

```
Would you like to erase the console after each message? (Y/N)
```

This enables or disables the **logging system** console clear flag:

- Everything can be kept in the console.

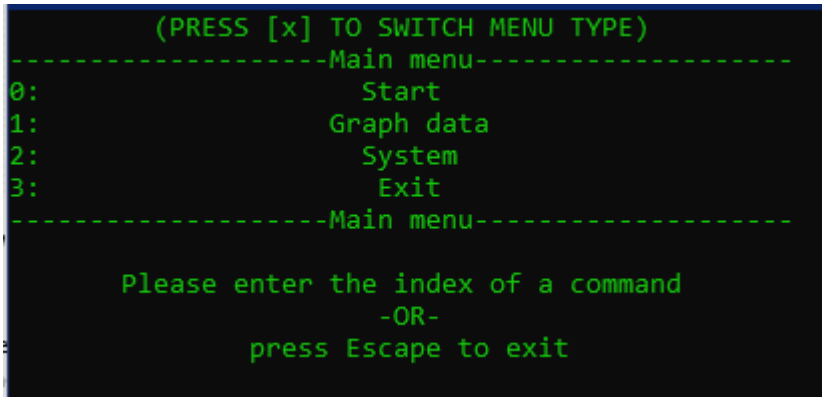
OR

- The console can be erased after each message, this giving the application a more responsive UI rather than a console full of text.

The interface of the application consists of a main menu with options which are either functions or submenus that have functions and submenus as options as well.

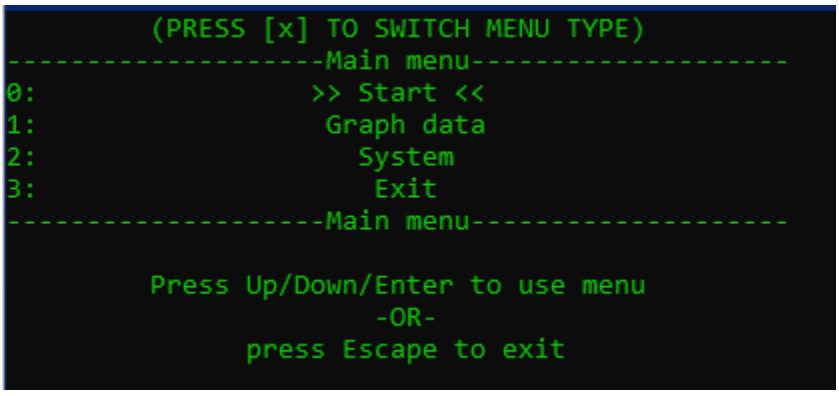
The menu has 2 types of use:

- **Selection Mode**: The menu waits for one of the **commands' indexes** to be pressed to select the option.



*Application user interface for **selection type menu***

- **Navigation Mode**: The menu shows one option as highlighted/selected and we can navigate through it using the **up/down arrows** and using the **Enter** key to select the option.



*Application user interface for **navigation type menu***

PROJECT DEVELOPMENT

GENERAL OBJECTIVES

The objective of the project was to create an application with a friendly user interface that provides the functionality of working with undirected graphs.

The main objective was to make the application look more like an application rather than a information prompter that just displays information based on fed information.

Another not planned yet achieved goal was gathering additional knowledge about C++.

DESCRIPTION OF DATA PROCESSED BY THE PROGRAM

The program has been created to be compact and all the data it processes is safely stored in initial objects that are used along the application and are linked together until the very end of the program:

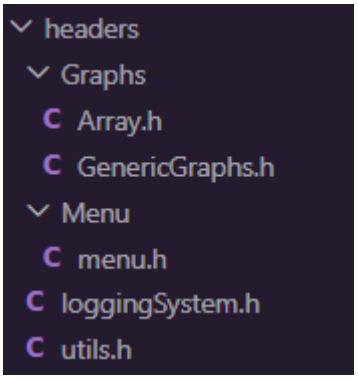
```
// logging system initialization
loggingSystem *log = new loggingSystem();
// log->setDebug(true);

// Instantiate the graph of the application
GenericGraph *mainGraph = new GenericGraph();

// menu tree initialization
menu *mainMenu = menu::InstantiateMenu(log, mainGraph);
```

Main objects used for the application.

There are 3 main objects linked that “work together” to make the application run:

- **loggingSystem** is the system that keeps the data regarding the information logging into the console.
- **GenericGraph** is the main graph object that stores the graph data during the application runtime.
- **mainMenu** array which is the array of menu options that are linked to one another as pairs of parent-children menus.

The program’s data structures are the following:



- **Array.h**:
  - A custom Array module defined to store an array of **int** values.
  - The array class has custom functions.
- **GenericGraph.h**:
  - The main graph class for working with graph data.
- **menu.h**:
  - The menu option object that links all the project together.
- **loggingSystem.h**:
  - The logging system object manages the console logging options.
- **utils.h**:
  - A header with custom functions and constants used across multiple files.

The Input/Output of files compatible with the program is the following:

entries_count nodes_count source_node edges_count edge_start_1 edge_end 1 edge_start_2 edge_end 2 …

Where:

- entries_count = is the number of entries in the file (except the entries_count itself)
- nodes_count = is the number of nodes in the graph
- source_node = is the node from which all the paths will start
- edges_count = is the number of edges in the graph
- edge_start_n & edge_start_n = represent pairs of edges' ends

Examples:

- 17 7 0 7 0 1 1 2 2 3 3 0 2 4 4 5 4 6
- 15 7 0 6 0 1 0 2 3 4 3 5 4 6 5 6

The program's menu objects and links are read and initialized from a file. That way, a menu can easily be implemented into the program:

```
22
-1 4 -1 9 Main menu
0 0 0 5 Start
-1 6 0 10 Graph data
1 0 2 9 Read data
2 0 2 12 Display data
-1 4 2 20 Modify existent data
3 0 5 8 Add node
4 0 5 11 Delete node
5 0 5 8 Add edge
6 0 5 11 Delete edge
-1 3 0 6 System
7 0 2 9 Save data
-1 2 2 34 Change save location of graph data
10 0 12 29 Restore default save location
11 0 12 20 Custom save location
-1 2 2 9 Load data
8 0 15 20 Custom load location
9 0 15 17 Load from presets
12 0 10 16 Toggle Debug Log
13 0 10 20 Toggle console clear
14 0 10 4 Help
15 0 0 4 Exit
```

The menu initializer data

The format is the following:

- The first row represents the number of the menu rows in the file.
- Each of the other rows is structured as follows:

function_index children_count parent_index name_length name_array

Where:

- function_index = the id of the function it executes in a switch logic in the code
- children_count = this menu is a submenu that opens this amount of submenus/function options
- parent_index = the index of the parent submenu
- name_length = this is the number of characters the name_array has
- name_array = the name of the menu

Description Of program modules

All main and more complex functions have descriptive comments and descriptions in the code.

Conclusions

- Could be improved by adding a **Change source node** option.
- Main conclusion is that having an interactive UI is an important note when working on a project.

References

- https://cplusplus.com/
- https://stackoverflow.com/
- Applied Informatics courses for the Graph Algorithm