Snapch-UTT

Project by Toussaint Baptiste, Martinez Lucas.

1. Introduction

The main objective of this project is to code an application which allows to create and manage student. Each student has personal characteristics: Name, Age, Years of study, Field of studies, City of residence, and Field of interest. The application allows student to follow each other according to their characteristic.

Our social network’s name is “Snapch’UTT”, and it has several features implemented:

* Snapch’UTT has a default implemented list of students.
* Create student and add it to the database.
* Change information about a student.
* Delete student from the database.
* Displays information about a specific student.
* Displays all student according to criteria: (age, field of studies…).
* Allow a specific student to follow others and displays his list of followers.
* Propose a list of followers for a student that matches with his information.

This report is divided in 5 part:

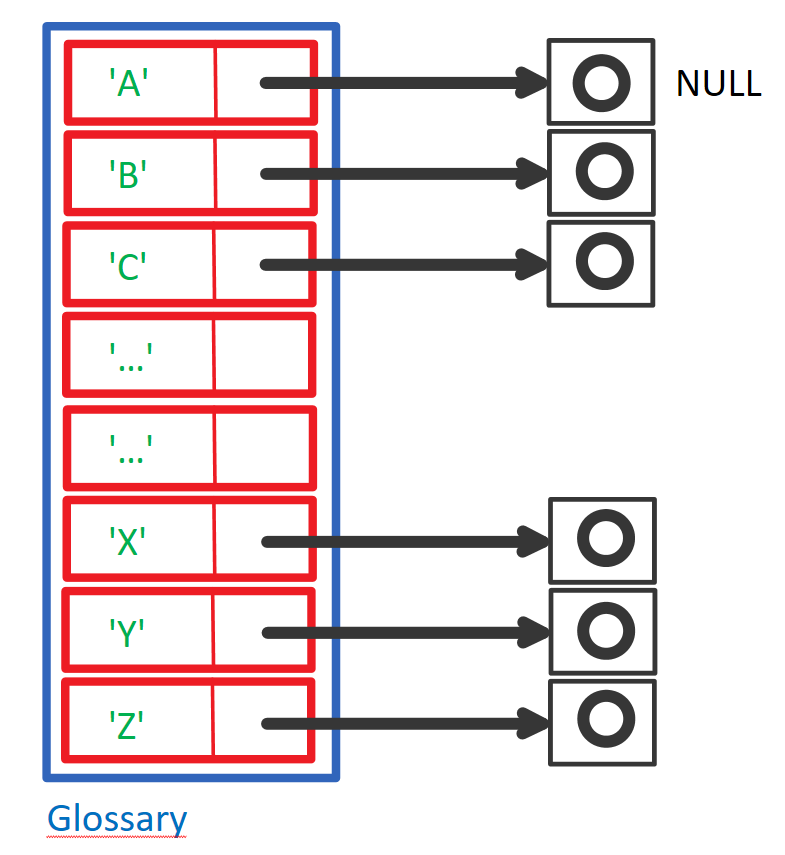
1. Description of the algorithm.
2. Project management: Organization, problem encounters and how resolve it.
3. How use Snapch’UTT?
4. Conclusion and perspective.
5. Appendix.
6. Description of the algorithm
7. Student and Glossary.

The main structure used in this project is the structure: *struct Student.* A Student variable contain several members:

* *name* (Character array)
* *age* (Integer)
* *yearStudy* (Integer)
* *fieldStudy* (Character array)
* *cityResidence* (Character array)
* *interest* (Array of *struct FieldOfInterest*)
* *nextAlphaStudent* (Pointer on *struct Student*)
* *follower* (Pointer on *struct ListOfFollower*)

The 5 first members are easy to understand. Let us first present *nextAlphaStudent* and his utility with the *Glossary.*

To stock and manage all student in the database we choose to create a variable named *Glossary* which is an array of 26 *struct* *LineGlossary*. Each *LineGlossary* is composed by two members: A character (‘A’, ‘B’, ‘C’, …) and a pointer on *struct Student*. At the begin of the execution all these 26 pointers are NULL (a.).



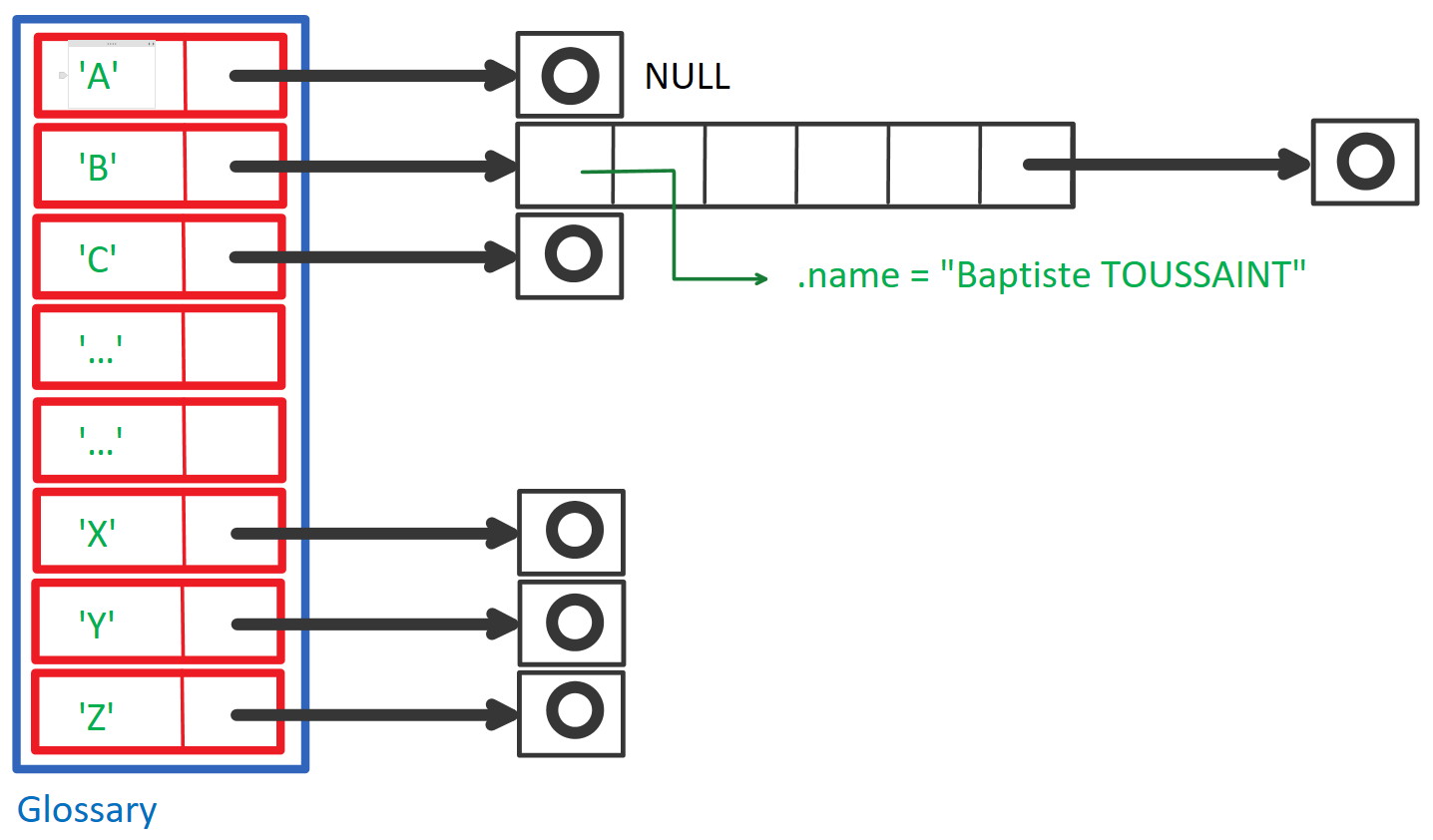
a. Empty Glossary.

When we need to add a *struct Student* to the database, we use a function:

*int add\_student(Student \*stud)*.

We use a chained list to stock every student by alphabetic. And this function finds the position of the new student.

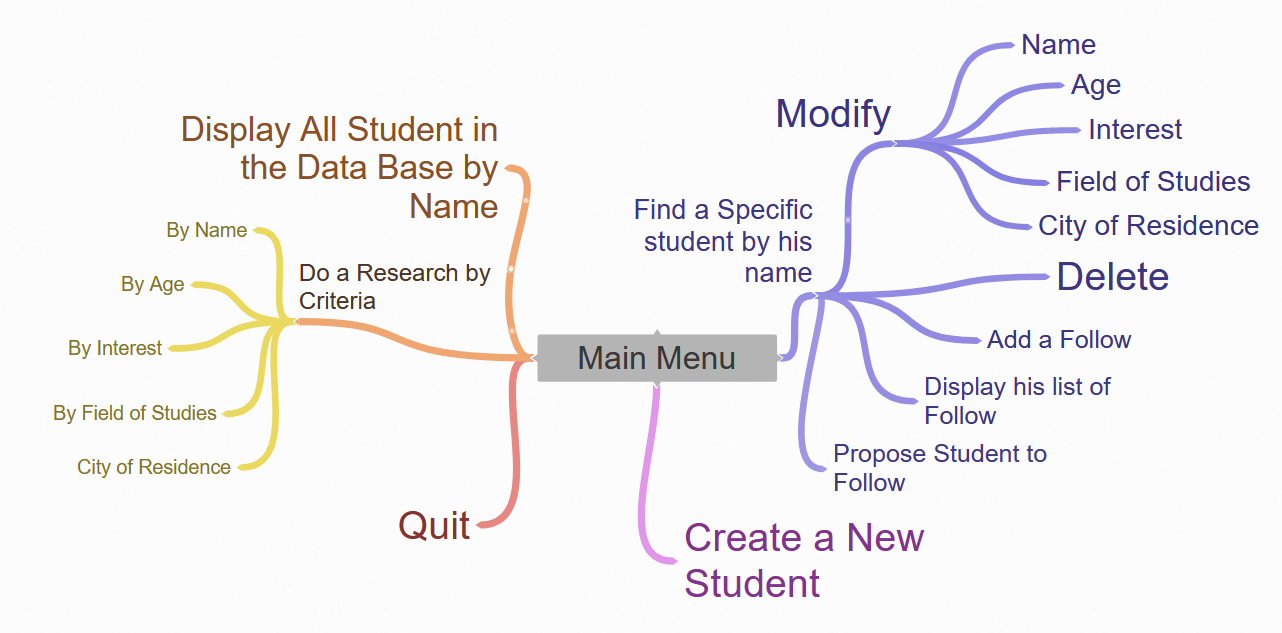
For example, if I want to add a student named Baptiste TOUSSAINT, the pointer of the second element of the *Glossary* will point on this new student (b.)



b. Initialized Glossary.

1. Architecture.

We choose to create Snapch’UTT according to a specific architecture (a).

According to this architecture (a) we choose to create a Main Menu and several sub-menus. Each menu is following the same principle : A *do()* loop in which a *switch()* condition (see more in appendix).

c. Features Tree.

1. Follow system.

Each student has a member named : *follower*. We use a structure names *struct ListOfFollower* with the following’s members :

* *maxElement* (integer)
* *nbrFollower* (integer)
* *listFollower* (Pointer on array of Student)

When a Student A follow a Student B, we add the address of B in the array. If the array is full, we use the *realloc()* function to add space in the array.

1. Project management: Organization, problem encounters and how resolve it.
2. Organization.

To easily manage the project, we choose to use GitHub. Our priority was to create all functions relative to the management of the student.

1. Problems encounters.

During this project there were 2 major problem encounters:

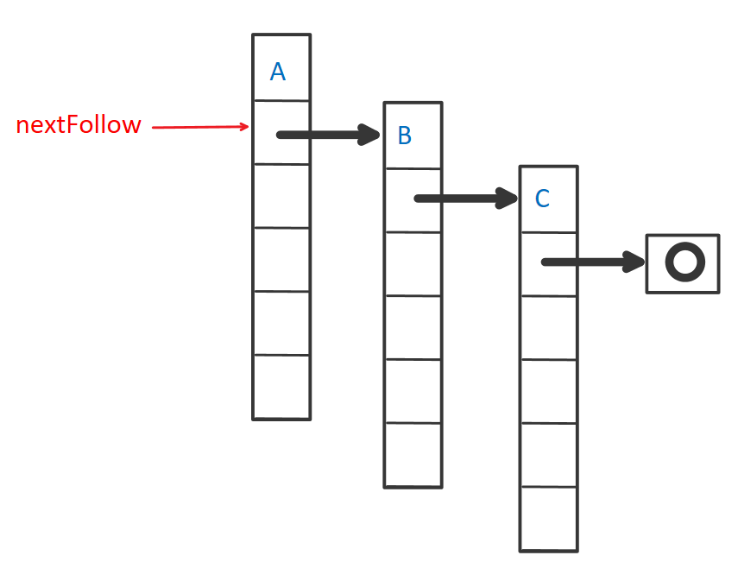
* Create and implement the follower system.
* Manage the dynamic allocation of the *Student* variable.

The difficulty of the follower system is that one person can follow many other and there are many different situations.

For example, student A can follow B, but B is not following A.

Our first idea was to use a linked chain: Each student had a member : *nextFollow* a pointer on *Student*.

For example, student A want to follow B and C, then *A.nextFollow* point on B and *A.nextFollow* point on C (d.)



d. First follow system.

Here A has a correct list of follows. The important mistake we made with this system is that B cannot have a personal list of follows.

1. How to use Snapch’UTT ?
2. Conclusion and perspectives.
3. Appendix.
4. Menu function algorithm.
5. Find one function algorithm.
6. Find per criteria algorithm.
7. Modify algorithm.

