Chart, funnel chart

Description automatically generated

COURSE: INTRODUCTION TO COMPUTER SCIENCE II (CS162)

FINAL PROJECT: DATA VISUALIZER

Lê Đức Nhuận – 22TT1 – 22125070

HO CHI MINH UNIVERSITY

UNIVERSITY OF SCIENCE

FACULTY OF INFORMATION TECHNOLOGY

1. Summary of Features
   1. Linked List

|  |  |  |
| --- | --- | --- |
| Singly Linked List | Doubly Linked List | Circular Linked List |
| Init from file | Init from file | Init from file |
| Randomized data | Randomized data | Randomized data |
| Insert to the first | Insert to the first | Insert to the first |
| Insert to the last | Insert to the last | Insert to the last |
| Insert to the middle | Insert to the middle | Insert to the middle |
| Delete at the first | Delete at the first | Delete at the first |
| Delete at the last | Delete at the last | Delete at the last |
| Delete at the middle | Delete at the middle | Delete at the middle |
| Update | Update | Update |
| Search | Search | Search |

* 1. Queue and Stack

|  |  |
| --- | --- |
| Stack | Queue |
| Init from file | Init from file |
| Randomized data | Randomized data |
| Push | Add |
| Pop | Remove |
| Clear | Clear |

* 1. Array

|  |  |
| --- | --- |
| Static Array | Dynamic Array |
| Init from file | Init from file |
| Randomized data | Randomized data |
| Insert to the first | Insert to the first |
| Insert to the last | Insert to the last |
| Insert to the middle | Insert to the middle |
| Delete at the first | Delete at the first |
| Delete at the last | Delete at the last |
| Delete at the middle | Delete at the middle |
| Update | Update |
| Search | Search |
| Access | Access |
| Allocate |  |

1. User Manual
   1. A screenshot of a computer

      Description automatically generated with low confidenceA close-up of a data visualizer

      Description automatically generated with medium confidenceMain Menu

*Main Menu (dark theme)*

*Main Menu (light theme)*

* Start by clicking on one of the data structures listed on the menu.
* Clicking on the bulb at top right of the screen toggle between light and dark theme.
  1. A screenshot of a computer

     Description automatically generated with medium confidenceA picture containing text, screenshot, font, diagram

     Description automatically generatedVisualizer’s Main View

*Main View (Light theme)*

Consists of Control Station, Media Control, Settings, Code highlight/description and visualization the data structure

*Main View (dark theme)*

* 1. Control Station
* A screenshot of a computer

  Description automatically generated with low confidenceA picture containing text, screenshot, font, number

  Description automatically generatedStep 1: Click on the button below to select the desired operation.
* A screenshot of a computer program

  Description automatically generated with low confidenceStep 2: Click on the two triangle buttons to switch between different modes of an operation.
* Step 3: Click on typing boxes to enter information of the operation (index, value)

A picture containing text, screenshot, font, number

Description automatically generated

* A picture containing text, screenshot, font, rectangle

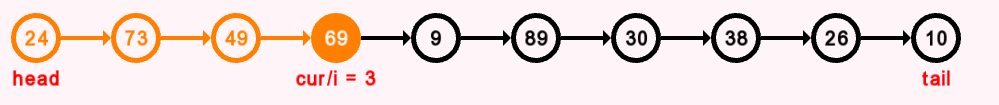
  Description automatically generatedStep 4: Click on “GO” button to start the operation. Users can also press Enter on keyboard for the same effect.
  1. Media Control

A screenshot of a computer program

Description automatically generated with low confidence

* Users can click/hold on the Scrubber to set the visualizer to the desired step.
* Users can also click on 5 buttons below the Scrubber to control the sequence of the visualizer. There are also shortcut keys implemented for each button (From left to right: Home; Left arrow; Space; Right arrow; End).
* Overall, users can control the sequence of the visualizer similarly to popular video players (such as Youtube).
  1. A screenshot of a video player

     Description automatically generated with low confidenceSettings
* Users can toggle the Description Box.
* Users can toggle the Instant Update mode. In this visualizing mode, the visualizer will skip all the animations and only display the result after the operation, which is useful for cases when users want to quickly update the data structure.
  1. A picture containing text, screenshot, font, number

     Description automatically generatedCode highlight/description
* Visualizer will show the code of the current operation and highlight which line of code is running on each step.
* There are also descriptions for each step displayed above the code table. The descriptions display can be toggled in the settings part.
  1. Data Visualization
* The data structures are visualized in the presence of graphs consisting of nodes and edges.
* The visualizer can change color, position, direction of nodes and edges accordingly to the currently running code.

1. Structures of the Project
2. Node struct
3. Summary

* Implemented in Node.h
* Store information of nodes and handle nodes’ animation.

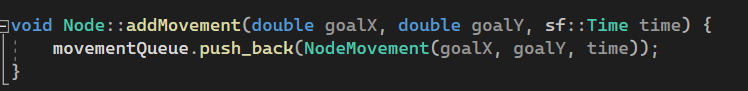
1. A screen shot of a computer program

   Description automatically generated with low confidence Node’s info

* x, y: coordinate of center of node.
* value: the value of the node (visible number on the screen).
* numPointCircle: the number of point used by sf::CircleShape (30 for circle node and 4 for square node in Static Array and Dynamic Array).
* variableList: the list of variable names visualized on the screen of the node.
* fillColor, outlineColor, valueColor, variableColor: color of node’s attributes.
* shape: the sf::CircleShape of the node, used for rendering
* font: a pointer to the font of letter.
* display: a boolean value, if true then the node is visible on the screen, otherwise no.

1. A picture containing text, screenshot

   Description automatically generatedHow nodes’ animation is handled

* Node’s movement:
  + Firstly, we call addMovement function to push to the movementQueue information of the movement.
  + A picture containing text, screenshot, font, software

    Description automatically generatedFor every frame, updateMovement will be called and update the position of the node based on the current information of its position and the desired position in the queue.
  + Node’s zooming and changing color is handled similarly to movement.
* Add / Delete variable:
  + A picture containing text, screenshot, font, line

    Description automatically generatedAdd variable name:
  + A picture containing text, screenshot, font

    Description automatically generatedDelete variable name:
* Stop animation:
  + A picture containing text, screenshot, font

    Description automatically generatedInstantly update the information of the node into its latest version.

1. Edges
2. Edge struct
   * Summary

* Implemented in Edge.h
* Used for rendering edges. An edge is actually a rectangle with fixed height (thickness of the edge).
  + A screen shot of a computer code

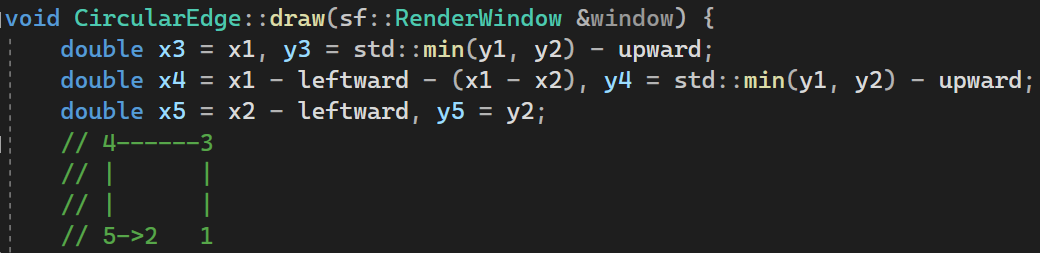
    Description automatically generated with low confidenceEdge’s info
* points[4]: store coordinates of rectangle
* color: color of the edge
* forwardPointer: the small triangle at the end, representing an arrow
* thickness: the thickness of the edge
* type: type of edge (SinglyDirected, DoublyDirected, Undirected)
* Edges don’t have animations implemented in itself, the animation of edges (shrinking, zooming,…) is implemented by using fake nodes (explained in Graph struct).
  + A picture containing text, font, screenshot, number

    Description automatically generatedTrianglePointer struct
* points[3]: store coordinates of three points of the triangle
* color: color of the triangle
* thickness: thickness of the edge

1. CircularEdge struct
   * Summary

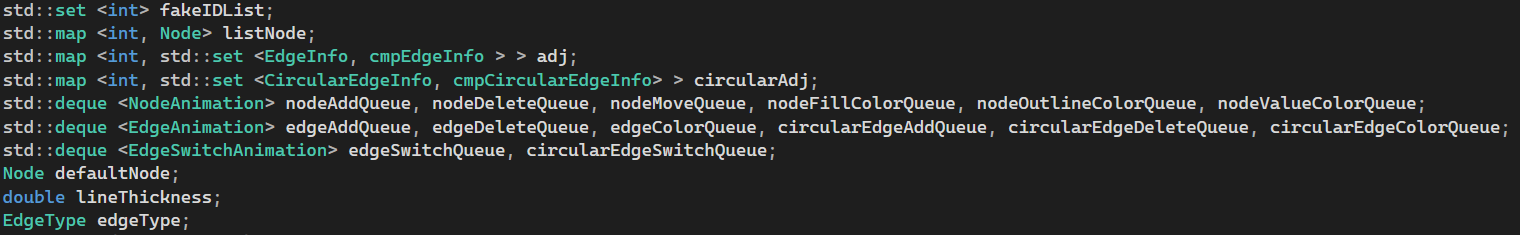
* Implemented in CircularEdge.h
* A picture containing screenshot, circle, design

  Description automatically generatedUsed for the special circular edge in Circular Linked List
  + A black background with white text

    Description automatically generated with low confidenceCircular Edge’s info
* x1, y1: coordinates of the starting point
* x2, y2: coordinates of the ending point
* thickness: the thickness of the edge
* leftward, upward, shorten: constant parameters of the circular edge
* progress: for animating purpose, ranging from 0 to 1, 1 means the edge is fully connected
* color: color of the edge
  + Drawing Circular Edge
* (x1, y1): starting point
* (x2, y2): ending point
* We will draw 4 edges: .

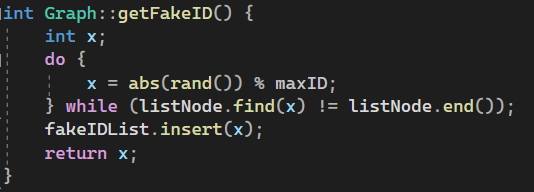
1. Graph struct
2. Summary

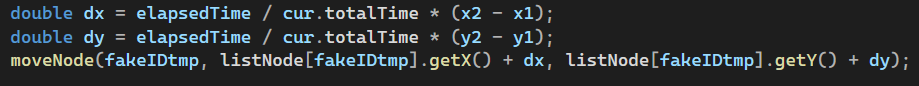
Graph struct is the “data” that is visualized on the screen. It consists of nodes and edges. It also maintains the animation of edges and calls animating function of nodes.

1. Graph’s info

* listNode: list of nodes, each element contains an integer ID of the node and a Node struct for information of that node.
* fakeIDList: list of fake nodes’ id.
* adj, circularAdj: adjacent lists, representing the edges connecting the nodes.
* nodeAddQueue, nodeDeleteQueue, nodeFillColorQueue, nodeOutlineColorQueue, nodeValueColorQueue: queues for nodes’ animation
* edgeAddQueue, edgeDeleteQueue, edgeColorQueue, circularEdgeAddQueue, circularEdgeDeleteQueue, circularEdgeColorQueue: queues for edges’ animation
* edgeSwitchQueue, circularEdgeSwitchQueue: queues for switching edges animation (a special animation that requires info of 3 nodes rather than 2 nodes like above queues)
* defaultNode: a default struct Node to store all default information of a node
* lineThickness: thickness of edge
* edgeType: type of Edge (explained in Edge struct)

1. How fake nodes work
   * + In the Graph struct, there are two types of nodes: real nodes and fake nodes. Real nodes can be displayed on the screen while fakes nodes can not, but the edges between fake nodes or fake nodes and real nodes can still be displayed.
     + The purpose behind using real nodes is to visualize edges’ animation using only nodes’ animation. For example, when an edge is created between 2 real nodes *u* and *v*, a fake node *tmp* is created along with one edge connected between *u* and *tmp*. Then, the visualization of an edge slowly expanding from *u* and reach *v* can be implemented by moving *tmp* slowly from *u’s position* to *v*’s position.
     + Moreover, by implementing all animations using only nodes’ animation, in case of stopping all animation running in the graph, all we need to do is to erase all fake nodes and toggle the display of all real nodes that are currently off.
     + In order to create a new fake node, we first get a random **unused** id by the function getFakeID()
     + A screen shot of a computer code

       Description automatically generated with low confidenceTake a look at the addEdge function work

* Firstly, we add a real edge between *u* and *v*, then we toggle the display of this edge.
* Next, we create 3 fake nodes, one node for *u*, one for *v* and one *tmp* node. All these nodes will be hidden.
* Then we add a new edge between *fakeIDu* and *fakeIDtmp.*
* Later, in the void updateEdgeAdd, we will move *fakeIDtmp* slowly to *fakeIDv.*
* A picture containing text, font, screenshot

  Description automatically generatedFinally, when the animating process is finished, we delete all fake nodes used for this animation and toggle the display of the original edge.

1. Graph’s animation
   * + Each animation has 5 functions associated with it:

* A picture containing text, font, screenshot, line

  Description automatically generatedInstant update function (single object): this function will instantly update the data of the graph (for example, instantly create an edge between *u* and *v*). This kind of function does not directly animate anything, but instead maintain the data in the proper format if the animation is halted and must be updated to its final state .

*addEdge function – Instant update (single object)*

* A computer code on a black background

  Description automatically generated with low confidenceInstant update function (multiple objects): repeat Instant update function (single object) for multiple objects.

*addEdge function – Instant update (multiple objects)*

* A screen shot of a computer program

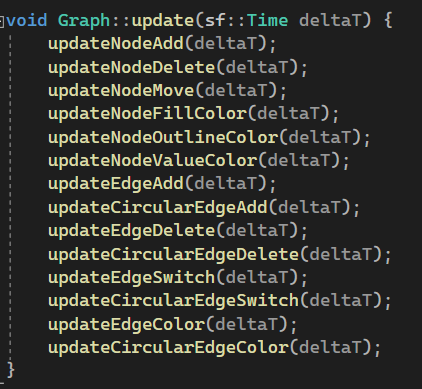
  Description automatically generated with low confidenceAnimating function (single object): creating fake nodes and push the animation into its corresponding animation queue.

*addEdge function – Animating (single object)*

* Animating function (multiple objects): same effect to Animating function (single object).
* A screen shot of a computer code

  Description automatically generated with low confidenceUpdate animation function: process the animation queue given the elapsed time.

*updateEdgeAdd function – Update animation*

* + - For each frame, all the Update animation functions will be called.

1. DataStructure struct
2. Summary
3. How list of frames works
4. Code highlight
5. Set error
6. Stage struct
7. Data Structures’ implementation
8. Fonts and Cursor