Insertion Sort Visualization with SDL

RAMDANI Mouna BOULALOUA Salsabil

NUMIDIA Institute of technology

January 6, 2024

Contents

1 Introduction

The main aim of this code is to provide a **visual** and **interactive** way for users to understand and observe how the **insertion** sort algorithm organizes and sorts data step by step. It serves as an **educational** tool to help individuals comprehend **sorting** algorithms by seeing their actions **visually**.

1.1 Key Components

- **SDL Graphics**: The code utilizes SDL to create a graphical window and render visual elements.
- **Doubly linked List**: It employs a doubly linked list structure to represent the data being sorted.
- **Sorting Visualization**: The sorting process is visually represented through bars, with each bar representing a data element.
- **User Interaction**: Users are given options to generate a random list of numbers, input their list, view sorting information, and exit the program via a console menu.

1.2 How It Works

1. **Initialization**: Initializes SDL and sets up the graphical environment.
2. **Menu System**: Presents a menu for user interaction, allowing users to choose various options for sorting visualization or viewing information.
3. **Sorting**: Implements the insertion sort algorithm on the linked list while visualizing each step of the sorting process using SDL graphics.
4. **User Input**: Allows users to input their data or generate random data for sorting.
5. **Visualization Loop**: Displays the sorting process visually until the user decides to exit.
6. **Timing**: Measures and displays the time taken by the sorting process.

1.3 Purpose

The main aim of this code is to provide a visual and interactive way for users to understand and observe how the insertion sort algorithm organizes and sorts data step by step. It serves as an educational tool to help individuals comprehend sorting algorithms by seeing their actions visually.

2 Libraries and Headers

2.1 Standard C Libraries:

- `#include <stdio.h>`: Standard input/output functions for reading input and printing output.
- `#include <stdlib.h>`: Standard library functions for memory allocation (`malloc`, `free`) and other general utilities.
- `#include <stdbool.h>`: Definition of the boolean type and associated constants `true` and `false`.
- `#include <time.h>`: Functions for manipulating date and time.

2.2 SDL Libraries for Graphics and Text Rendering:

- `#include <SDL2/SDL.h>`: Simple DirectMedia Layer for multimedia applications, handling graphics, audio, and input.
- `#include <SDL2/SDL_ttf.h>`: SDL extension for TrueType font rendering, used for text rendering in SDL applications.
- `#include <SDL2/SDL_render.h>`: SDL extension for 2D rendering operations, used for rendering graphics in SDL applications.

2.3 User-Defined Headers:

- `#include "insertion_info.h"`: A user-defined header file providing additional information or functions related to the insertion sort algorithm.
- `#include "intro.h"`: A user-defined header file containing functions or information for program introduction or initial display.

3 Structure and Functions

3.1 Structure Definition:

Listing 1: Structure Definition

```
typedef struct cel *liste;
typedef struct cel {
    int info;
    liste next;
    liste prev;
} cel;
```

cel: Represents a node in the doubly linked list. It contains an integer info and pointers next and prev pointing to the next and previous nodes, respectively.
liste: An alias for a pointer to a cel structure, used as the type for the linked list.

3.2 Node Insertion:

Listing 2: Node Insertion

```
void ajouter_liste(liste *tete, int e) {
    liste nouv = verifie();

    nouv->info = e;
    nouv->next = *tete;
    nouv->prev = NULL;

    if (*tete != NULL) {
        (*tete)->prev = nouv;
    }

    *tete = nouv;
}
```

ajouter_liste: Adds a new node with the given integer e to the front of the doubly linked list (tete). It creates a new node, sets its data, and updates pointers to ensure proper linking of nodes.

3.3 Insertion Sort Using Doubly Linked List:

Listing 3: Insertion Sort Using Doubly Linked List

```
void Tri_Insertion1(liste tete, SDL_Renderer *renderer, TTF_Font *font) {
    liste i, j, key;
    for (i = tete->next; i != NULL; i = i->next) {
        key = i;
        j = i->prev;
        while (j != NULL && j->info > key->info) {
            j->next = key->next;
            if (key->next != NULL) {
                key->next->prev = j;
            }
```

```
                key->prev = j->prev;
                key->next = j;
                if (j->prev != NULL) {
                    j->prev->next = key;
                } else {
                    tete = key;
                }
                j->prev = key;
                j = key->prev;
                render(renderer, tete, font);
                SDL_Delay(100);
            }
        }
}
```

Tri_Insertion1: Sorts the doubly linked list using the insertion sort algorithm. It traverses the list, compares elements, and reorganizes pointers (prev and next) to rearrange the nodes appropriately.

3.4 Render Function:

This function uses SDL (Simple DirectMedia Layer) to create a graphical representation of a linked list. Here's a breakdown of what each part does:

- **Clear Renderer:** Sets the drawing color to black (0, 0, 0) and clears the renderer, preparing it for the new frame.
- **Iterate Through Linked List:** Iterates through the linked list nodes a total of `list_SIZE` times, representing each node as a graphical bar on the screen.
- **Bar Visualization:** Determines the height and position of each bar based on the node's info field and the screen dimensions. Alternates between two shades of green for the bars based on the index (`i`) in the iteration. Draws a filled rectangle (`barRect`) using SDL, representing the current node as a bar on the screen.
- **Text Rendering:** Converts the node's info field into a text format (`valueText`) to display within the rectangle. Renders the text onto an SDL surface (`textSurface`) and creates an SDL texture (`textTexture`) from the surface.
- **Text Positioning:** Calculates the position to center the text horizontally within the bar and positions it slightly above the bar (`textX`, `textY`).
- **Render Text onto Bars:** Renders the text texture onto the rectangle to display the node's data within the bar.
- **Memory Management:** Frees the allocated surface and texture memory for each node after rendering its representation.
- **Display Update:** Presents the rendered frame on the screen using the renderer and introduces a delay of 100 milliseconds before rendering the next frame.

4 Main Function

4.1 Initialization:

Initializes SDL for graphical rendering and font loading. Loads the required font for text rendering.

4.2 Menu System:

Enters a continuous loop to provide a menu for user interaction and choice selection. Displays an introductory message or options for the user. Takes user input for their choice:

- Generates a random array and visualizes the sorting process based on the user's choice.
- Allows the user to input their own elements and visualizes the sorting process.
- Displays sorting information.
- Exits the program.

Executes the corresponding functionalities based on the user's choice.

4.3 Visualization Loop:

Executes a loop that allows the program to continuously visualize sorting or perform other actions based on user input. Contains conditional blocks based on user choices to generate random data, sort, and visualize it, or display information.

4.4 SDL Event Loop:

Manages SDL events using an event loop to handle window events. Checks for specific events like quitting the program (`SDL_QUIT`) triggered by the user closing the window.

4.5 Looping Structure:

Contains a while loop that continuously prompts the user for choices and handles them until the user decides to exit the program. Provides an interactive environment for the user to make choices, visualize sorting, view information, or exit.

5 Strategies and Challenges

5.1 Strategies:

- **Interactive Visualization:** Create a dynamic and engaging sorting visualization tool using SDL graphics to enhance user understanding of the insertion sort algorithm.
- **Flexible User Input:** Enable users to generate random arrays or input their own data for sorting, fostering exploration and experimentation.
- **GitHub Version Control:** Facilitate collaboration and track changes effectively through GitHub, ensuring code integrity and managing updates efficiently.

5.2 Challenges and Solutions:

5.2.1 Graphical Representation:

- **Improve Element Mapping:** Refine the mapping of array elements to graphical bars to ensure a precise and visually accurate representation of the sorting process.
- **Optimize Rendering Efficiency:** Explore techniques to enhance rendering performance for smoother animations and a seamless user experience.

5.2.2 Code Structuring:

- **Function Modularity:** Break down large functions into well-defined, reusable components to improve code readability, maintainability, and testability.
- **Adhere to Coding Conventions:** Follow consistent naming conventions and indentation practices for enhanced code clarity and collaboration.

6 summary

The provided code is a C program that demonstrates the Insertion Sort algorithm visually using the SDL library for graphics and text rendering. It defines structures for a doubly linked list to represent data being sorted. The main functionalities include:

- **Rendering Function:** The `render` function visualizes the sorting process by drawing bars representing the elements.
- **Insertion Sort Implementation:** The `Tri_Insertion1` function sorts the linked list and calls the rendering function for visualization.
- **Additional Functions:**
  - Functions for adding elements to the list.
  - Display function to show the elements in the list.
  - Initialization functions for SDL.
  - Handling user inputs through a menu system.
  - Cleanup functions for SDL resources.

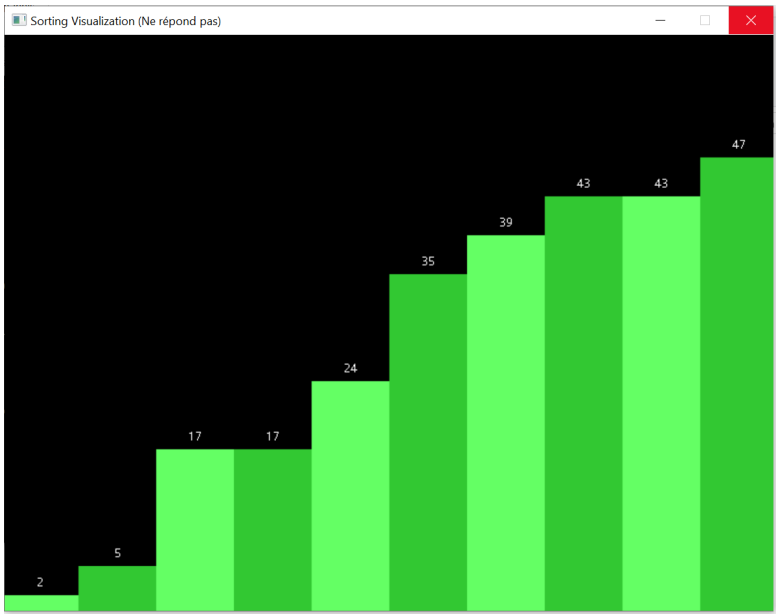

Figure 1: Insertion sort visualization

7 Site References

Here are some references to websites:

1. *Visualgo.net*. Available at: https://visualgo.net/en.
2. *Csvistool.com*. Available at: https://csvistool.com/InsertionSort.
3. *Cs.usfca.edu*. Available at: https://www.cs.usfca.edu/~galles/visualization/Algorithms.html.
4. *ChatGPT*. Available at: https://chat.openai.com/.
5. *SDL (Simple DirectMedia Layer)*. Available at: https://www.libsdl.org/.
6. *Lazyfoo.net*. Available at: https://lazyfoo.net/tutorials/SDL/index.php.