Team-H

produced by Jeongmin Hwang’s Notion

Problem solving steps 1-4

Step 1: Understand the Problem

The aim of this project is to create a C program that reads data from a text file, stores it in an array of structures and a linked list, and processes this data according to specific criteria. The main functionalities include sorting, searching, and updating the data. The data contains information like registration number, date, fee payment status, name, age, organization, and occupation.

The core objective of this project is to develop a data management program using C language. This program is tasked with reading structured data from a text file, which involves parsing and extracting key information from each line of the file. The data includes several fields: registration number, date of registration, fee payment status, name, age, organizational affiliation (company or university), and occupation.

This project is not merely about reading and storing data; it involves implementing complex functionalities such as sorting, searching, and updating the data based on specific criteria. For instance, the program needs to be able to sort records by age in ascending order, search for individuals based on their fee payment status or organizational affiliation, and update the records by adding new entries or removing existing ones based on certain conditions (e.g., last name).

Furthermore, this program must efficiently handle the data by utilizing appropriate data structures, such as arrays and linked lists. The choice of data structures and algorithms will significantly impact the program's performance, especially considering operations like sorting and searching which are central to the project's requirements.

Step 2: Outline a Solution (Including Logic Sketch)

1. Data Reading and Initial Processing:

- **Objective**: Efficiently read and parse structured data from a text file.
- **Detailed Steps**:
    - Open the text file for reading.

    - For each line in the file:
        - Parse the line into fields: registration number, registration date, fee status, name, age, organization, occupation.
        - Store these fields in a struct.
    - Close the file after reading all lines.

2. Storing Data in Structures and Linked Lists:

- **Objective**: Organize the parsed data into an array of structures and a linked list for efficient access and manipulation.
- **Detailed Steps**:
    - As each line is parsed, simultaneously:
        - Add the struct to an array.
        - Create a linked list node with the struct data and append it to the linked list.

3. Sorting the Data:

- **Objective**: Sort the data based on age in ascending order.
- **Detailed Steps**:
    - Apply a sorting algorithm (e.g., merge sort) on the array based on the age field.
    - Reconstruct the linked list according to the sorted order.

4. Writing Sorted Data to a File:

- **Objective**: Persist the sorted data for future reference or use.
- **Detailed Steps**:
    - Open a new text file for writing.
    - Write each record from the sorted array to the file.
    - Close the file.

5. Searching for Specific Records:

- **Objective**: Implement search functionalities based on criteria like fee payment status and university affiliation.
- **Detailed Steps**:
    - Create functions to search through the array or linked list.
    - For each record, check if it matches the criteria (e.g., paid fees, belongs to Gachon University).
    - Store and return the search results.

6. Updating the Data:

- **Objective**: Modify the dataset based on specific conditions such as adding or removing records.

- **Detailed Steps**:
  - For removal, iterate through the data structures, find and remove records with specific attributes (e.g., last name “Choi”).
  - For adding, insert new records into both the array and linked list.

7. Final Output and Integration:

- **Objective**: Ensure that all functionalities work together seamlessly and produce accurate results.
- **Detailed Steps**:
  - Integrate all modules (reading, sorting, searching, updating).
  - Perform end-to-end testing to validate the workflow and output.

Logic Sketch

```
[Start]
    |
    |--> [Read Data from File]
    |       |
    |       |--> [Parse Data into Structs]
    |               |
    |               |--> [Store in Array & Linked List]
    |
    |--> [Sort Data in Array]
    |       |
    |       |--> [Reconstruct Linked List in Sorted Order]
    |
    |--> [Write Sorted Data to New File]
    |
    |--> [Search Functionality]
    |       |
    |       |--> [Criteria-Based Filtering]
    |
    |--> [Update Data]
    |       |
    |       |--> [Add/Remove Records as Needed]
    |
    |--> [Integrate & Test All Functionalities]
    |
[End]
```

Step 3: Form a Program Structure

1. main() Function:

- The `main()` function serves as the entry point of the program.

- It creates and initializes an array of `struct regi` data.
- It creates and initializes a linked list of type `struct NODE*`.
- It displays a menu and prompts the user to choose an operation, then calls the corresponding function accordingly.

2. Insertdata() Function:

- The `Insertdata()` function adds a new data entry (`struct regi data`) to the given linked list (`struct NODE* head`).

3. printlist() Function:

- The `printlist()` function prints the contents of the given linked list (`struct NODE* head`).

4. sortData() Function:

- The `sortData()` function sorts the data array (`struct regi data[]`). It sorts the given data array and updates it with the sorted data.

5. writeDataToFile() Function:

- The `writeDataToFile()` function writes the sorted data array to a file. It takes the sorted data array and a filename as input and writes the data to the specified file.

6. createLinkedList() Function:

- The `createLinkedList()` function creates and initializes a linked list (`struct NODE*`) based on the data array (`struct regi data[]`).

7. printLinkedListToFile() Function:

- The `printLinkedListToFile()` function prints the contents of the linked list (`struct NODE* head`) to a file.

8. searchFeePaid() Function:

- The `searchFeePaid()` function searches for students who have paid their fees in the data array (`struct regi data[]`).

9. searchFeePaidLinkedList() Function:

- The `searchFeePaidLinkedList()` function searches for students who have paid their fees in the linked list (`struct NODE* head`).

10. searchGachonUniversityInArray() Function:

- The `searchGachonUniversityInArray()` function searches for Gachon University students in the data array (`struct regi data[]`).

11. searchGachonUniversityInLinkedList() Function:

- The `searchGachonUniversityInLinkedList()` function searches for Gachon University students in the linked list (`struct NODE* head`).

12. cancelChoi() Function:

- The `cancelChoi()` function cancels the registration of students with the last name "Choi" in the data array (`struct regi data[]`).

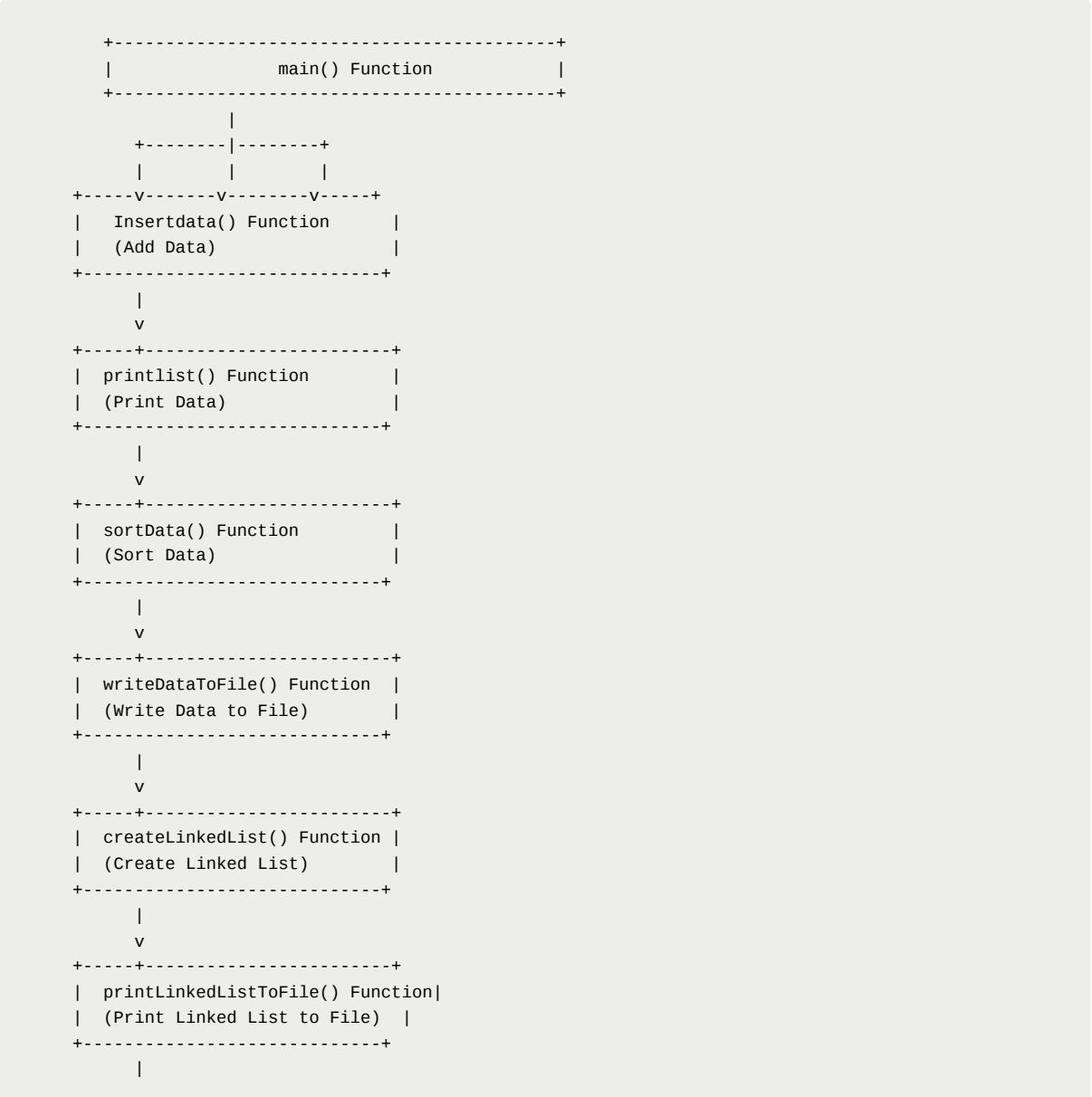
13. cancelChoiLinkedList() Function:

- The `cancelChoiLinkedList()` function cancels the registration of students with the last name "Choi" in the linked list (`struct NODE** head`).

14. registerLatePaik() Function:

- The `registerLatePaik()` function registers students with the last name "Paik" in the data array (`struct regi data[]`). It also updates the array size (`int* size`) and adds them to the linked list (`struct NODE** head`).

Program Structure Diagram



```
      v
 +-----+------------------------+
 |  searchFeePaid() Function    |
 |  (Search Fee-Paid Students - Array) |
 +-----------------------------+
      |
      v
 +-----+------------------------+
 |  searchFeePaidLinkedList() Function|
 |  (Search Fee-Paid Students - Linked List)|
 +-----------------------------+
      |
      v
 +-----+------------------------+
 |  searchGachonUniversityInArray() Function|
 |  (Search Gachon University Students - Array) |
 +-----------------------------+
      |
      v
 +-----+------------------------+
 |  searchGachonUniversityInLinkedList() Function|
 |  (Search Gachon University Students - Linked List)|
 +-----------------------------+
      |
      v
 +-----+------------------------+
 |  cancelChoi() Function       |
 |  (Cancel Choi - Array)       |
 +-----------------------------+
      |
      v
 +-----+------------------------+
 |  cancelChoiLinkedList() Function|
 |  (Cancel Choi - Linked List) |
 +-----------------------------+
      |
      v
 +-----+------------------------+
 |  registerLatePaik() Function |
 |  (Register Late - Array and Linked List) |
 +-----------------------------+
```

Step 4: Write a Pseudo Code (Including Logic Sketch)

```
# Main Function

function main()
    # Read data from the file
    data = readDataFromFile("registration_data.txt")

    # Build a linked list from the data
    linkedList = buildLinkedList(data)

    # Sort the data
    sortedData = sortData(data)
```

```
    # Write sorted data to a file
    writeSortedDataToFile(sortedData, "sorted_data.txt")

    # Update the linked list with sorted data
    updateLinkedList(linkedList, sortedData)

    # Search for paid members
    paidMembers = searchData(sortedData, feePaid="yes")

    # Search for members from Gachon University
    gachonMembers = searchData(sortedData, organization="Gachon University")

    # Cancel registrations for members with the last name "Choi"
    cancelRegistrations(data, linkedList, lastName="Choi")

    # Add a late registration
    addLateRegistration(data, linkedList, newMemberInfo)
end function

# Read Data from File

function readDataFromFile(filename)
    # Open the file
    Open filename

    # Initialize an empty array for data
    Initialize an empty array for data

    # Read lines from the file
    While not end of file
        Read line
        Parse line into a structure (registrationNo, date, feePaid, name, age, organization, occupation)
        Append the structure to the data array
    end while

    # Close the file
    Close file

    # Return the data
    return data
end function

# Build Linked List

function buildLinkedList(data)
    # Create an empty linked list
    Create an empty linked list

    # Iterate over each element in data
    For each element in data
        Create a new node with the element
        Append the node to the linked list
    end for

    # Return the linked list
    return linkedList
end function

# Sort Data

function sortData(data)
    # Use a sorting algorithm (e.g., quicksort) to sort data by age
    Use a sorting algorithm (e.g., quicksort) to sort data by age
```

```
    # Return the sorted data
    return the sorted data
end function

# Write Sorted Data to File

function writeSortedDataToFile(sortedData, filename)
    # Open the file for writing
    Open filename for writing

    # Write each element in sortedData to the file in a formatted way
    For each element in sortedData
        Write element to file in a formatted way
    end for

    # Close the file
    Close file
end function

# Update Linked List

function updateLinkedList(linkedList, sortedData)
    # Clear the existing linkedList
    Clear the existing linkedList

    # Rebuild the linkedList with sortedData
    Rebuild the linkedList with sortedData
end function

# Search Data

function searchData(data, criteria)
    # Initialize an empty list for results
    Initialize an empty list for results

    # Iterate over each element in data
    For each element in data
        # Check if the element meets the criteria
        Check if element meets the criteria

        # If criteria match, add element to results
        If criteria match, add element to results
    end for

    # Return the results
    Return the results
end function

# Cancel Registrations

function cancelRegistrations(data, linkedList, lastName)
    # Iterate over each element in data
    For each element in data
        # If element's lastName matches the given lastName
        If element's lastName matches the given lastName
            # Remove element from data
            Remove element from data

            # Remove the corresponding node from linkedList
            Remove the corresponding node from linkedList
        end if
    end for

    # Update the data array and linkedList to reflect the removals
    Update the data array and linkedList to reflect the removals
```

```
end function

# Add Late Registration

function addLateRegistration(data, linkedList, newMemberInfo)
    # Append newMemberInfo to the data array
    Append newMemberInfo to the data array

    # Create a new node with newMemberInfo
    Create a new node with newMemberInfo

    # Append the node to the end of the linkedList
    Append the node to the end of the linkedList

    # Optionally, sort data and linkedList if sorting is required after addition
    Optionally, sort data and linkedList if sorting is required after addition
end function
```

Contribution percentage

김현태 - 20 %

배혜림 - 18 %

이우정 - 18 %

최경민 - 18 %

황정민 - 26 %