Data Structure Project #1

2023년 09월 15일

Due date: 2023년 10월 12일 목요일 23:59:59 까지

본 프로젝트에서는 이진 탐색 트리(Binary Search Tree), 연결 리스트(Linked List), 큐(Queue) 자료구조를 이용하여 간단한 개인정보 관리 프로그램을 구현한다. 이 프로그램은 데이터 파일로부터 회원이름, 나이, 개인정보수집일자, 가입약관종류 정보를 읽어 Queue를 구축하며, 해당 Queue를 Member_Queue라 부른다. 가입약관은 A, B, C, D 총 4종류가 존재하며, 각 6개월(A), 12개월(B), 24개월(C), 36개월(D)의 개인정보 보관 유효기간을 갖는다. 각 Queue에서 pop 명령어를 실행하면 데이터를 방출하여 가입약관종류를 기준으로 정렬된 자료구조(Terms_List, Terms_BST)와 회원이름을 기준으로 정렬된 자료구조(Name_BST)에 저장한다.

- 가입약관종류를 기준으로 정렬된 자료구조

Terms_List는 가입약관의 종류 별로 노드가 구성되며, 각 노드는 가입약관종류, 해당 가입약관의 회원 수, 해당 가입약관의 BST 포인터 정보를 갖는다. 입력된 가입약관의 순서대로 노드가 생성되어 연결되며, 이미 존재하는 가입약관 정보가 입력될 경우 새로운 노드를 생성하지 않고 해당하는 기존 노드의 회원수를 증가시킨다. List의 정보가 입력된 후 해당 가입약관의 BST에서 노드 구성을 수행한다.

Terms_BST는 가입약관의 종류 별로 구성되며, 각 BST의 노드는 회원이름, 나이, 개인정보수집일자, 개인정보만료일자 정보를 갖는다. 개인정보만료일자는 개인정보수집일자에 가입약관 별 개인정보 보관 유효기간이 더해져 계산되며, 각 BST는 개인정보만료일자 정보를 기준으로 정렬된다.

- 회원이름을 기준으로 정렬된 자료구조

Name_BST는 회원이름, 나이, 개인정보수집일자, 개인정보만료일자, 가입약관종류 정보를 갖는 노드로 구성되며, 회원이름 정보를 기준으로 정렬된다.

각 자료구조의 구축 방법과 조건에 대한 자세한 설명은 **Program Implementation**에서 설명한다.

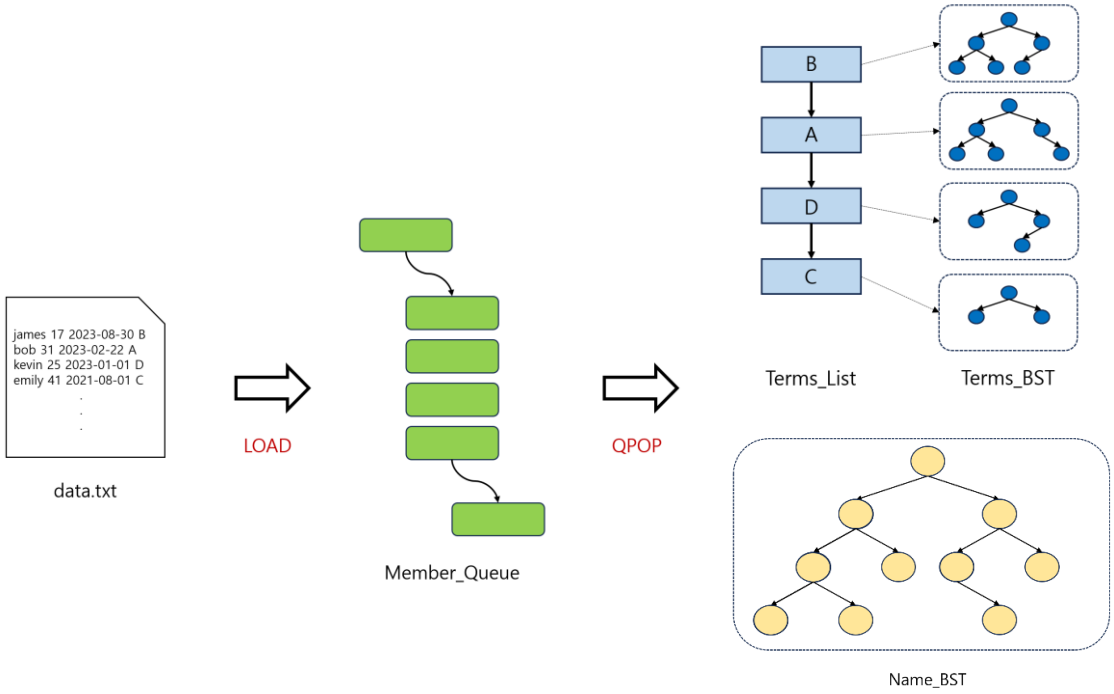

그림 1. 프로그램 구조

☐ Program Implementation

1) Member_Queue

- 주어진 data.txt에 저장된 데이터를 이용하여 구축한다. Queue의 자료구조에 따라 저장되는 순서대로 방출된다. Queue에는 data.txt에 첫번째 줄부터 마지막 줄까지 순서대로 저장된다.
- 입력 데이터는 "회원이름 나이 개인정보수집일자 가입약관종류" (ex. "james 17 2023-08-30 B") 의 형식을 갖는다. 이때 회원이름은 공백문자 없이 소문자로 구성되며, 중복되는 회원이름은 없다고 가정한다.
- Queue에 저장되는 데이터는 MemberQueueNode class로 구현되며, 회원이름, 나이, 개인정보수집일자, 가입약관종류 정보를 갖는다.
- Queue의 크기는 100으로 초기화되고 이후 크기가 변하지 않으며, Queue가 비어 있을 때 POP 또는 전부 차 있을 때 PUSH가 수행되는 경우 프로그램이 종료된다.
- Queue에서 QPOP 명령어를 통해 방출되는 데이터는 List와 BST의 입력으로 사용된다.

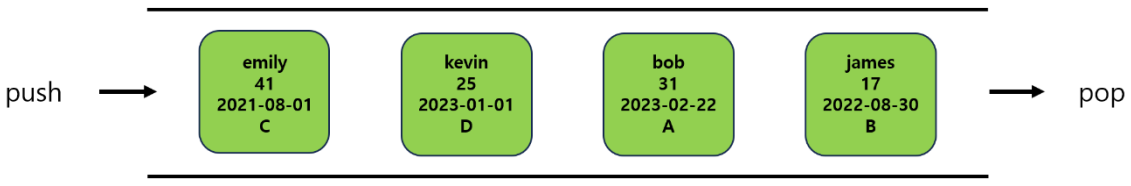

그림 2. Member_Queue 예시

2) Terms_List

- Queue에서 방출된 데이터를 이용하여 구축한다.
- List의 노드는 TermsListNode class로 구현되며, 가입약관종류, 해당 가입약관의 회원 수, 해당 가입약관의 BST 포인터 정보를 갖는다.
- 입력된 데이터의 가입약관을 확인하여 List에 존재하지 않는 경우 List에 노드를 추가하고, 존재하는 경우 해당 가입약관 노드의 회원수를 증가시킨다. List는 입력된 가입약관 순서대로 정렬된다.
- DELETE 명령어를 통해 BST 노드의 삭제 연산이 수행된 경우 Terms_List에서 해당하는 가입약관 노드의 회원수를 감소시키며, 보유한 회원수가 0이 되는 노드는 삭제한다.

3) Terms_BST

- List의 가입약관 별 노드 구성 후 Queue에서 방출된 데이터를 이용하여 구축한다.
- BST의 노드는 TermsBSTNode class로 구현되며, 회원이름, 나이, 개인정보수집일자, 개인정보만료일자 정보를 갖는다.
- 개인정보만료일자는 개인정보수집일자에 가입약관 별 개인정보 보관 유효기간이 더해져 계산된다. (A: 6개월, B: 12개월, C: 24개월, D: 36개월)
- 가입약관 종류 별로 BST가 생성된다. (ex. 총 4종류의 가입약관(A, B, C, D) → 총 4개의 BST)
- BST에서 지원하는 연산은 삽입, 삭제, 출력(중위 순회)이며, 각 연산에 대해서는 **Functional Requirements**에서 자세히 설명한다.
- DELETE 명령어를 통해 Terms_BST 노드의 삭제 연산이 수행된 경우 Name_BST에서도 해당 회원의 노드를 삭제한다.
- BST 연결 규칙은 다음과 같다.
  ① 부모 노드보다 개인정보만료일자가 이전인 노드는 왼쪽, 같거나 이후인 노드는 오른쪽 서브 트리에 위치한다.
  ② 노드를 제거할 때, 양쪽 자식 노드가 모두 존재할 경우에는 오른쪽 자식 노드 중 가장 작은 노드를 제거되는 노드 위치로 이동한다.

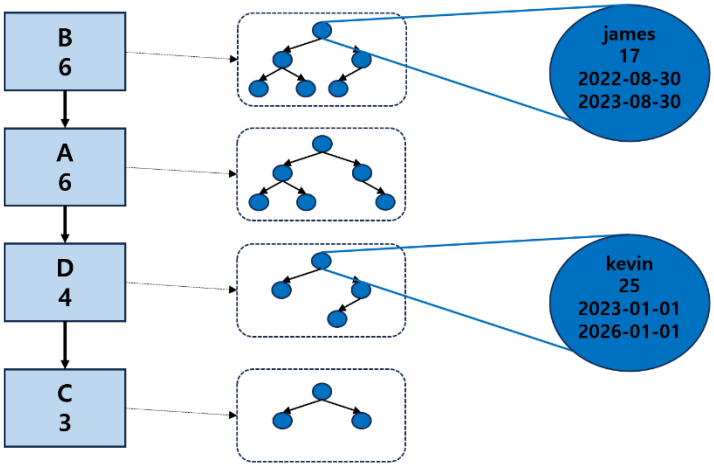

그림 3. Terms_List & Terms_BST 예시

4) Name_BST

- Queue에서 방출된 데이터를 이용하여 구축한다.
- BST의 노드는 NameBSTNode class로 구현되며, 회원이름, 나이, 개인정보수집일자, 개인정보만료일자, 가입약관종류 정보를 갖는다.
- 개인정보만료일자는 개인정보수집일자에 가입약관 별 개인정보 보관 유효기간이 더해져 계산된다. (A: 6개월, B: 12개월, C: 24개월, D: 36개월)
- BST에서 지원하는 연산은 삽입, 삭제, 검색, 출력(중위 순회)이며, 각 연산에 대해서는 **Functional Requirements**에서 자세히 설명한다.
- DELETE 명령어를 통해 Name_BST 노드의 삭제 연산이 수행된 경우 Terms_BST에서도 해당 회원의 노드를 삭제한다.
- BST 연결 규칙은 다음과 같다.
  ① 부모 노드보다 회원이름의 사전적 순서가 이전인 노드는 왼쪽, 같거나 이후인 노드는 오른쪽 서브 트리에 위치한다.
  ② 노드를 제거할 때, 양쪽 자식 노드가 모두 존재할 경우에는 오른쪽 자식 노드 중 가장 작은 노드를 제거되는 노드 위치로 이동한다.

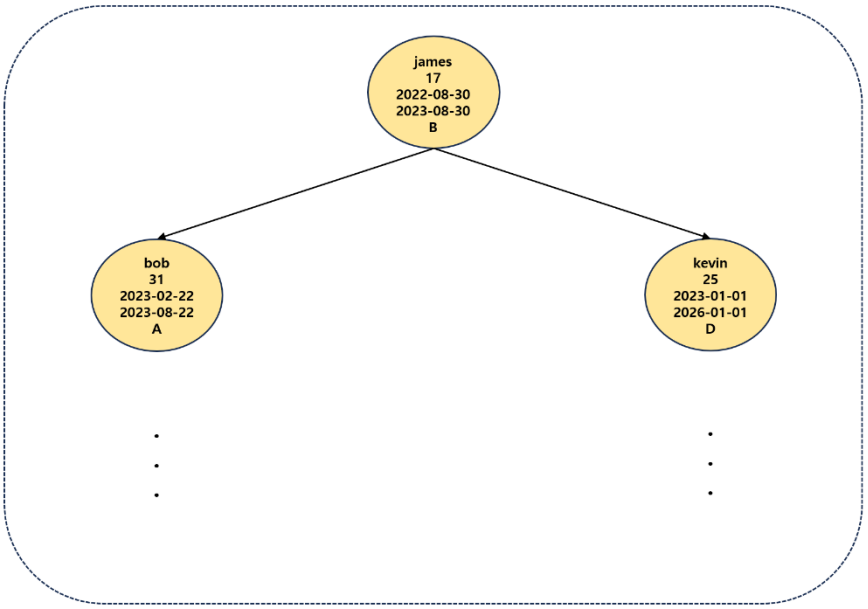

그림 4. Name_BST 예시

☐ Functional Requirements

* 출력 포맷은 포맷에 대한 예시일 뿐 실제 출력되는 데이터들과는 차이가 있을 수 있습니다.

명령어	명령어 사용 예시 및 기능 설명
LOAD	사용 예시) LOAD 텍스트 파일의 데이터 정보를 불러오는 명령어로, 텍스트 파일에 데이터 정보가 존재할 경우 텍스트 파일을 읽어 Member_Queue 자료구조에 모두 저장한다. 성공적으로 데이터를 불러온 경우 읽은 데이터를 출력하며, 출력 형태는 (회원이름)/(나이)/(개인정보수집일자)/(가입약관종류)이다. 만약 텍스트 파일이 존재하지 않거나 자료구조에 이미 데이터가 들어가 있으면 에러 코드를 출력한다. 사용되는 텍스트 파일의 이름은 아래와 같으며 파일 이름을 수정하지 않는다. 텍스트 파일: data.txt 데이터 조건 - 회원이름은 공백문자 없이 소문자로 구성되며, 20자 이하의 길이를 갖는다. 또한 중복되는 회원이름은 없다. - 나이는 자연수 형태로 입력되며, 10~99세 범위로 한정한다. - 개인정보수집일자는 XXXX-XX-XX의 형식으로 입력된다. - 가입약관종류는 대문자 A, B, C, D 중 하나로 입력된다. - 각 데이터의 인자들은 모두 공백문자로 구분된다 - 각 회원정보는 라인 별로 작성되며, 개행문자로 구분된다. 출력 포맷 예시) ===== LOAD ===== james/17/2023-08-30/B bob/31/2023-02-22/A kevin/25/2023-01-01/D emily/41/2021-08-01/C . . . ========== ===== ERROR ===== 100 ==========
ADD	사용 예시) ADD tom 50 2020-07-21 D

	Member_Queue에 데이터를 직접 추가하기 위한 명령어로, 총 4개의 인자를 추가로 입력한다. 첫 번째 인자부터 회원이름, 나이, 개인정보수집일자, 가입약관종류를 나타내며 하나라도 존재하지 않을 시 에러 코드를 출력한다. 성공적으로 데이터를 추가하였다면 해당 데이터를 출력하며, 출력 형태는 (회원이름)/(나이)/(개인정보수집일자)/(가입약관종류)이다. 출력 포맷 예시) ===== ADD ===== tom/50/2020-07-21/D ========== ===== ERROR ===== 200 ==========
QPOP	사용 예시) QPOP Member_Queue에서 데이터를 POP하여 Terms_List & Terms_BST와 Name_BST를 구성하는 명령어이다. Member_Queue의 front부터 모든 노드를 POP하여 해당 노드의 데이터로 자료구조를 구축한다. Member_Queue에 데이터가 존재하지 않을 시 에러 코드를 출력한다. 출력 포맷 예시) ===== QPOP ===== Success ========== ===== ERROR ===== 300 ==========
SEARCH	사용 예시) SEARCH bob Name_BST에 저장된 회원정보를 찾아 출력하는 명령어이다. 인자로 탐색하고자 하는 회원이름을 입력 받으며, Name_BST에서 입력 받은 회원이름의 노드를 찾아 정보를 출력한다. 출력 형태는 (회원이름)/(나이)/(개인정보수집일자)/(개인정보만료일자)이다. 찾고자 하는 회원정보가 Name_BST에 존재하지 않는 경우 에러 코드를 출력한다.

	출력 포맷 예시) ===== SEARCH ===== bob/31/2023-02-22/2023-08-22 ========== ===== ERROR ===== 400 ==========
PRINT	사용 예시1) PRINT D 사용 예시2) PRINT NAME BST에 저장된 데이터들을 출력하는 명령어이다. 인자로 가입약관종류(A, B, C, D) 중 하나가 입력되는 경우 해당 가입약관 Terms_BST의 저장된 데이터들을 출력한다. 인자로 NAME이 입력되는 경우 Name_BST의 저장된 데이터들을 출력한다. 중위 순회(in-order) 방식으로 BST를 탐색하여 데이터를 출력하며, 출력 형태는 (회원이름)/(나이)/(개인정보수집일자)/(개인정보만료일자)이다. BST에 데이터가 존재하지 않을 시 에러 코드를 출력한다. 출력 포맷 예시) 1) PRINT D ===== PRINT ===== Terms_BST D . . . harry/16/2022-04-25/2025-04-25 kevin/25/2023-01-01/2026-01-01 . . . ========== 2) PRINT NAME ===== PRINT ===== Name_BST . . . bob/31/2023-02-22/2023-08-22 emily/41/2021-08-01/2023-08-01 james/17/2023-08-30/2024-08-30 kevin/25/2023-01-01/2026-01-01 . . .

	========== ===== ERROR ===== 500 ==========
DELETE	사용 예시) DELETE DATE 2023-09-20 사용 예시) DELETE NAME emily List와 BST에 저장된 데이터를 제거하는 명령어이다. 인자로 DATE "특정 일자"가 입력되는 경우 모든 Terms_BST에서 해당 일자보다 개인정보만료일자가 이전인 모든 노드들을 제거한다. 각 Terms_BST에서 노드 삭제가 수행된 경우 Terms_List에서 해당하는 가입약관 노드의 회원수를 삭제 노드 수만큼 감소시키며, 보유한 회원수가 0이 되는 노드는 삭제한다. 또한 Terms_BST에서 노드 삭제가 수행된 경우 Name_BST에서도 해당하는 회원의 노드를 삭제한다. 인자로 NAME "특정 회원이름"이 입력되는 경우 Name_BST에서 해당 회원이름을 가진 노드를 제거한다. Name_BST에서 노드 삭제가 수행된 경우 Terms_BST와 Terms_List에서도 해당 회원의 노드를 삭제 연산을 수행한다. 삭제하고자 하는 회원 정보가 존재하지 않거나, 자료구조에 데이터가 존재하지 않을 시 에러 코드를 출력한다. 출력 포맷 예시) ===== LOAD ===== Success ========== ===== ERROR ===== 600 ==========
EXIT	사용 예시) EXIT 프로그램 상의 메모리를 해제하며, 프로그램을 종료한다. 출력 포맷 예시) ===== EXIT ===== Success ==========

□ 명령어 별 에러 코드

명령어	에러 코드
LOAD	100
ADD	200
QPOP	300
SEARCH	400
PRINT	500
DELETE	600
잘못된 명령어	1000

□ Requirements in Implementation

- ✓ 모든 명령어는 command.txt에 작성되며 순차적으로 읽고 처리한다.
- ✓ 모든 명령어는 반드시 대문자로 입력한다.
- ✓ 명령어에 인자가 모자라거나 필요 이상으로 입력 받을 경우 에러 코드를 출력한다.
- ✓ 개인정보 관리 프로그램에는 중복된 환자 이름이 존재하지 않는다.
- ✓ 예외처리에 대해 반드시 에러 코드를 출력한다.
- ✓ 출력은 "출력 포맷"을 반드시 따라한다.
- ✓ log.txt 파일에 출력 결과를 반드시 저장한다.

□ 구현 시 반드시 정의해야 하는 Class

- ✓ MemberQueue : 회원 정보 Queue 클래스
- ✓ MemberQueueNode : 회원 정보 Queue의 노드 클래스
- ✓ NameBST : 회원이름을 기준으로 정렬되는 BST 클래스
- ✓ NameBSTNode : 회원이름을 기준으로 정렬되는 BST의 노드 클래스
- ✓ TermsBST : 개인정보만료일자를 기준을 정렬되는 BST 클래스
- ✓ TermsBSTNode : 개인정보만료일자를 기준을 정렬되는 BST의 노드 클래스
- ✓ TermsList : 가입약관종류 별 노드로 연결되는 List 클래스
- ✓ TermsListNode : 가입약관종류 별 노드로 연결되는 List의 노드 클래스
- ✓ Manager : Manager 클래스
  - 다른 클래스들의 동작을 관리하여 프로그램을 전체적으로 조정하는 역할을 수행

□ Files

- ✓ data.txt : 프로그램에 추가할 회원 정보들이 저장되어 있는 파일
- ✓ command.txt : 프로그램을 동작시키는 명령어들이 저장되어 있는 파일
- ✓ log.txt : 프로그램 출력 결과를 모두 저장하고 있는 파일

□ 채점 기준

✓ 코드

채점 기준	점수
LOAD 명령어가 정상 동작 하는가?	1
ADD 명령어가 정상 동작 하는가?	1
QPOP 명령어가 정상 동작 하는가?	3
SEARCH 명령어가 정상 동작 하는가?	1
PRINT 명령어가 정상 동작 하는가?	1
DELETE 명령어가 정상 동작 하는가?	3
총합	**10**

– 채점 기준 이외에도 조건 미달 시 감점 (linux 컴파일 에러, 파일 입출력 X, 주석 미흡 등)

✓ 보고서

채점 기준	점수
Introduction을 잘 작성하였는가?	1
Flowchart을 잘 작성하였는가?	2
Algorithm을 잘 작성하였는가?	3
Result Screen을 잘 작성하였는가?	2
Consideration을 잘 작성하였는가?	1
총합	**10**

✓ 최종 점수는 (코드 점수 × 보고서 점수) 로 계산됩니다.

□ 제한사항 및 구현 시 유의사항

✓ 제공되는 코드(github 주소 참고)를 이용하여 구현하며 작성된 소스 파일과 클래스, 함수의 이름을 임의로 변경하지 않는다.
✓ 클래스의 함수 및 변수는 자유롭게 추가 구현이 가능하다.
✓ 제시된 Class를 각 기능에 알맞게 모두 사용한다.
✓ 프로그램 구조에 대한 디자인이 최대한 간결하도록 고려하여 설계한다.
✓ 채점 시 코드를 수정해야 하는 일이 없도록 한다.
✓ 주석은 반드시 영어로 작성한다. (한글로 작성하거나 없으면 감점)
✓ 프로그램은 반드시 리눅스(Ubuntu 18.04)에서 동작해야한다. (컴파일 에러 발생 시 감점)
– 제공되는 Makefile을 사용하여 테스트하도록 한다.
✓ 인터넷에서 공유된 코드나 다른 학생이 작성한 코드를 절대 카피하지 않도록 하며 적발 시 전체 프로젝트 0점 처리됨을 명시

□ 제출기한 및 제출방법

✓ 제출기한

- 2023년 10월 12일 목요일 23:59:59 까지 제출
  (추가 제출: 2023년 10월 13일 금요일 23:59:59 까지, 10% 감점)

✓ 제출방법

- 소스코드와 보고서 파일(**학번_DS_project1.pdf**)을 함께 압축하여 제출
- 확장자가 .cpp, .h, .pdf가 아닌 파일은 제출하지 않음 (Makefile과 텍스트 파일 제외)
- 보고서 파일 확장자가 pdf가 아닐 시 감점
- KLAS -> 과제 제출 -> tar.gz로 과제 제출

✓ 제출형식

- **학번_DS_project1.tar.gz** (ex. 2023123456_DS_project1.tar.gz)
- 제출 형식 미 준수 시 감점 (보고서 및 압축파일 학번, 프로젝트 이름 반드시 준수)

✓ 보고서 작성 형식 및 제출방법

- Introduction : 프로젝트 내용에 대한 설명
- Flowchart : 설계한 프로젝트의 플로우 차트를 그리고 설명
- Algorithm : 프로젝트에서 사용한 알고리즘의 동작을 설명
- Result Screen : 모든 명령어에 대해 결과화면을 캡처하고 동작을 설명
- Consideration : 고찰 작성

보고서는 위의 각 항목을 모두 포함하여 작성하며 보고서에는 소스코드를 포함하지 않음. Introduction과 Consideration 외의 모든 각 항목은 최소 2페이지 이상 작성하여 제출함. Consideration는 본 프로젝트 설계에 있어 힘들었던 점이나 성능 및 코드 구조를 개선하기 위해 작업한 내용, 프로젝트 및 코드관리를 위해 작업한 내용, 프로젝트를 설계하며 새로이 알게 되거나 참고한 내용이 될 수 있음.