



# Data Structure Programming Report

Project 01

Professor	Sangho Choi
Department	Computer engineering
Student ID	2020202090
Name	Minseok Choi
Submission Date	2023. 10. 12

# 1. Introduction

## - 프로젝트 개요

이번 데이터구조설계 1차 프로젝트의 내용은 이진 탐색 트리(Binary Search Tree), 연결 리스트(Linked List), 큐(Queue) 데이터 구조를 이용해 간단한 개인정보 관리 프로그램을 구현하는 것이다.

## - 조건: 자료구조

프로그램은 데이터 파일 (data.txt)로부터 각 회원의 이름, 나이, 개인정보수집일자, 가입약관종류 정보를 읽어 1차적으로 MemberQueue라는 큐를 구축한다.

가입 약관은 A, B, C, D 4종류가 존재한다. 각각 A는 6개월, B는 12개월, C는 24개월, D는 36개월의 개인정보 보관 유효기간을 갖는다.

Queue에서 pop 명령어를 통해 데이터를 방출하여 가입약관종류에 따라 나뉘지고 정렬되는 TermsLIST와 TermsBST, 그리고 이름에 따라 정렬되는 NameBST에 저장한다.

TermsLIST는 연결 리스트로, 가입 약관에 따라 그에 대응하는 TermsBST의 주소 정보를 저장한다. Queue에서 정보가 방출되면 해당하는 TermsLIST와 TermsBST에 정보가 저장되고, 해당 약관 정보를 가진 TermsLIST의 회원 수를 증가시킨다.

TermsBST와 NameBST는 이진 탐색 트리로, TermsBST에는 같은 약관인 회원들의 정보가, NameBST에는 이름순서로 모든 회원들의 정보가 저장된다. 각 노드에는 회원의 이름, 나이, 개인정보수집일자, 개인정보 만료일자 정보를 갖고 NameBST는 추가로 가입약관종류 정보도 포함한다.

## - 조건: 명령어

회원정보가 든 파일 data.txt를 읽어와 MemberQueue를 구축하는 LOAD 명령어,

임의의 회원 정보를 MemberQueue에 추가하는 ADD 명령어,

MemberQueue의 pop을 통해 TermsLIST, TermsBST, NameBST를 구축하는 QPOP 명령어,

특정 이름을 가진 회원을 검색해 정보를 출력하는 SEARCH 명령어,

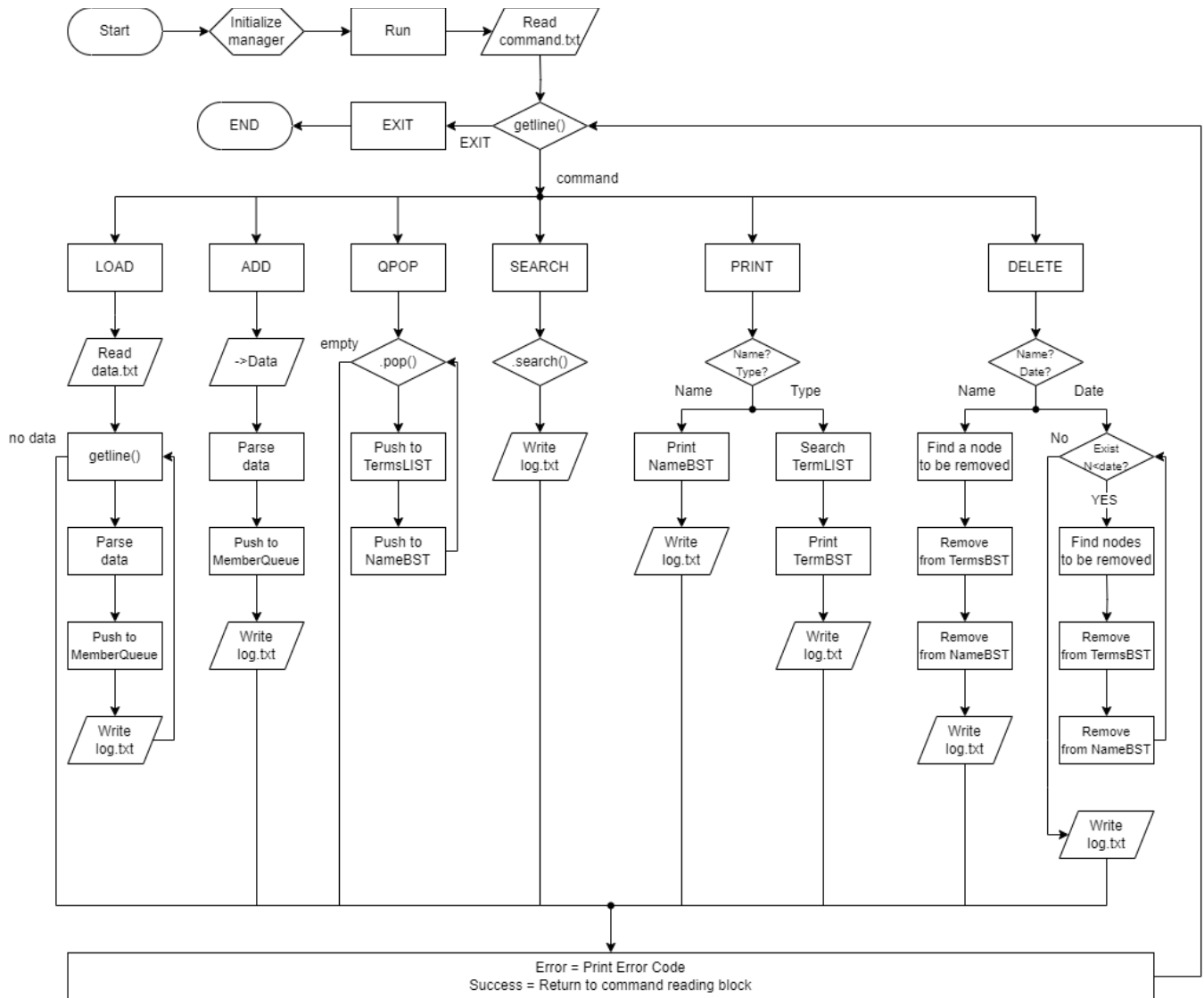
특정 가입약관종류를 가진 TermsBST 혹은 NameBST를 출력하는 PRINT 명령어,

특정 일자보다 개인정보만료일자가 이전인 모든 노드를 제거하거나 특정한 이름을 가진 회원의 정보가 든 노드를 제거하는 DELETE 명령어,

프로그램의 메모리를 할당 해제하며 프로그램을 종료하는 EXIT 명령어

각 명령어에서 에러가 발생한 경우 알맞은 번호의 에러 코드를 출력하고 모든 출력은 정해진 출력 형식을 통해 log.txt 파일에 기록한다.

## 2. Flowchart



### 3. Algorithm

#### 1. MemberQueue

입력된 데이터가 들어와 처음으로 생성되는 자료구조이다. LOAD혹은 ADD 명령어를 통하여 회원의 개인정보가 들어오면 MemberQueue에 전달해 큐를 구축한다. 이후 QPOP 명령어를 통해 큐의 모든 데이터를 방출하여 TermsLIST와 TermsBST, NameBST를 구축한다.

생성자에선 큐에서 사용할 헤드와 테일 포인터를 초기화해주고 사이즈는 항상 100으로 사용하므로 현재 큐에 들어있는 노드의 개수를 표시할 사이즈 변수도 초기화해준다.

##### - MemberQueueNode class

MemberQueue를 구성하는 노드를 정의한 클래스이다. 이번 프로젝트에서 MemberQueue는 연결리스트로 구현했으므로, 각 노드는 저장하는 회원의 이름, 나이, 개인정보수집일자, 가입약관유형 정보 및 다음 노드의 주소를 가지는 링크필드로 구성된다. 멤버 함수로는 입력이 있는 생성자와 정보를 반환 set함수, 링크필드를 수정하는 setNext함수, 출력을 위한 ostream 연산자 오버로딩 함수 등을 구현했다.

##### - MemberQueue( )

MemberQueue의 생성자에선 노드의 삭제, 삽입이 이루어지는 head 포인터와 tail 포인터를 nullptr로 초기화해준다. 이 프로젝트의 큐는 최대 용량이 항상 100으로 고정되어 있으므로, 현재 큐 내부 노드의 개수를 나타내는 size 변수도 0으로 초기화 해준다.

##### - ~MemberQueue( )

MemberQueue의 삽입에선 new를 이용한 동적할당을 통해 이루어지므로 힙 메모리 영역의 데이터를 메모리 해제해주기 위해 소멸자에선 큐에 대한 선형 순회를 통해 모든 노드를 방문하여 각각의 노드의 메모리를 delete해준다.

##### - empty( ), full ( )

empty함수에선 현재 큐의 size가 0일 때 true를, 아닐 때 false를 반환한다.

full함수에선 현재 큐의 사이즈가 100보다 크거나 작을 때 true를, 아닐 때 false를 반환한다.

##### - push(string& name, int age, string& date, Type type)

push함수에선 입력된 정보를 바탕으로 새로운 노드를 동적할당해 생성하여 MemberQueue에 추가한다. 만약 큐가 꽉 찬 경우 노드를 새로 추가하지 않고 예외를 발생시켜 반환한다.

큐가 비었을 경우 새로 생성한 노드를 head와 tail 포인터에 연결해준다. 큐가 비어 있지 않으면 새로 생성한 노드를 tail의 next로 설정하고 tail을 새로 생성한 노드로 설정해준다.

push함수에서 큐에 새로운 노드의 추가는 tail 포인터에서 이루어진다. 노드를 방출하는 pop 함수는 head 포인터에서 작업해주면 나중에 추가된 노드는 tail쪽에, 먼저 추가된 노드는 head쪽에 위치하고 각각 삽입과 방출이 이루어지므로 선입선출의 원리를 만족하는 자료구조로서 동작할 수 있게 된다.

- pop( )

pop함수에선 큐의 노드 중 가장 먼저 삽입된 노드를 방출한다. 만약 큐가 비었는데 pop이 호출됐다면 예외를 발생시키며 큐의 사이즈를 1 감소시키고 방출할 큐의 정보를 복사해 남겨놓은 뒤 head포인터를 temp에 남겨놓고 복사한 다음 head를 앞으로 한 칸 이동시킨 후 temp를 삭제해 메모리를 해제해주고 남겨놓은 노드를 반환한다.

- front( )

front함수에선 가장 앞에 있는 노드를 반환한다. 만약 큐가 꽉 찼는데 front가 호출됐다면 예외를 발생시킨다.

- printqueue( )

printqueue함수에선 큐를 head부터 선형 순회해 모든 노드의 정보를 출력한다. 데이터를 클래스 외부로 넘겨주기 위해 stringstream참조 반환형을 통해 정보를 전달해준다. 전달된 정보를 manager 내부에서 받아 log.txt에 입력해준다.

## 2. TermsLIST

MemberQueue에서 방출된 노드의 정보를 이용해 구축하는 자료구조 중 하나이다. 리스트의 노드가 직접 가지고 있는 정보는 약관유형밖에 없지만 해당하는 약관 유형의 회원 정보를 저장하는 TermsBST의 포인터 정보도 함께 가진다. 이번 프로젝트에서 가입약관종류에 따라 정렬되는 TermsBST를 분류하기 용이하게 해주는 자료구조이다.

### - TermsListNode class

TermsLIST를 구성하는 노드를 정의한 클래스이다. 직접 가지는 정보는 가입약관유형 뿐이므로 입력을 받는 생성자에선 입력 받은 가입약관유형에 따라 초기화해주고, next 링크필드는 nullptr로 설정해주고 BSTlinker에 동적할당을 통해 새로운 BST를 생성한다. 해당 가입약관유형의 회원 수는 0으로 초기화해준다. 소멸자에선 반드시 담당하는 TermsBST를 메모리 해제해 메모리 누수를 방지한다.

주요 함수는 정보를 반환하는 get함수와, 링크필드를 설정하는 setNext함수, 정보를 입력 받아서 연결된 TermsBST에 멤버를 추가하는 insertmember(string& name, int age, Date& date) 함수, 연결된 TermsBST에 print함수를 호출해주는 printBST( )함수, 연결된 TermsBST에 remove 함수를 전달하고 이를 확인해 회원 수를 1만큼 감소시키는 remove(string& name)함수 등이 있다.

각 노드는 담당하는 TermsBST에 명령을 전달해주는 기능을 포함하고 있다.

### - TermsLIST( )

생성자에선 head 포인터를 nullptr로 초기화해준다.

### - ~TermsLIST( )

소멸자에선 MemberQueue와 마찬가지로 선형 순회를 통해 모든 노드의 메모리를 해제한다. 노드의 소멸자에서 할당된 TermsBST도 할당해제 되므로 메모리 누수의 가능성은 없다.

### - insert(string& name, int age, Date& date, Type type)

insert함수에선 입력 받은 정보를 바탕으로 새로운 TermsBST 노드를 동적할당으로 생성하여 BST에 삽입한다. 만약 리스트가 비어 있는 경우 head포인터에 새로이 노드를 동적할당한 후 거기의 BST에 멤버를 삽입해준다. 만약 입력 받은 회원의 가입약관유형을 담당하는 노드가 이미 존재한다면 해당 노드의 BST에 멤버를 삽입해준다. 만약 리스트 내에 해당하는 가입약관유형의 노드가 존재하지 않는다면 새로운 노드를 생성하고 거기의 BST에 회원 정보를 삽입해준다.

### - print(Type type)

print함수에선 입력 받은 가입약관유형의 노드를 탐색한다. 만약 노드가 존재한다면 해당 노드의 BST를 출력해주는 함수를 호출한다. 노드가 존재하지 않는 경우 예외를 발생시킨다.

### - remove(string& name)

remove함수에선 각 BST에 입력 받은 이름에 대한 삭제 함수를 호출해준다. 만약 삭제의 결과로 리스트 노드의 회원수가 0명이 되면 해당 리스트 노드를 삭제한다.

### 3. TermsBST

MemberQueue에서 방출된 노드의 정보를 이용해 구축하는 자료구조 중 하나이다. TermsBST는 이진탐색트리 유형의 자료구조로, TermsLIST의 노드에 종속되어 인스턴스화 된다. TermsBST는 가입약관유형이 같은 회원의 정보만 저장되므로 A, B, C, D 최대 4개의 TermsBST가 존재할 수 있다. BST 내부에선 개인정보만료일자에 따라 정렬된 형태로 회원 정보가 저장된다.

- TermsBSTNode class

TermsBST를 구성하는 노드를 정의한 클래스이다. 생성자에선 저장할 회원의 정보를 입력 받아 해당하는 멤버 변수를 초기화해주고, 링크필드도 nullptr로 초기화해준다. 이진트리 유형이므로 링크필드는 left, right 2개가 존재한다.

주요 함수로는 멤버 변수를 반환하는 get함수, 멤버 변수를 수정하는 set함수, 출력을 위한 ostream 연산자 오버로딩 함수 등을 구현했다.

- TermsBST( )

생성자에선 루트 포인터를 nullptr로 초기화해준다.

- ~TermsBST( )

소멸자에선 모든 노드가 해제될 때까지 아래의 재귀 삭제함수를 호출해 모든 노드의 메모리를 해제해준다.

- recursivedelete(TermsBSTNode\* root)

재귀 삭제 함수는 중위 순회를 이용해 모든 노드를 방문하도록 하는 함수이다. 입력 받은 루트의 좌측을 방문한 후, 현재 노드에 대한 메모리 해제를 진행한다. 다만 우측 노드로 이동할 포인터를 남겨놓지 않고 노드를 삭제하면 우측 노드를 방문할 수 없으므로 이에 유의해야 한다.

- recursiveprint(TermsBSTNode\* root)

재귀 출력 함수는 마찬가지로 중위 순회를 이용해 모든 노드를 방문한 후 해당 노드의 정보를 출력하는 함수이다. 재귀 삭제 함수와는 달리 노드의 메모리가 해제되지 않으므로 우측 링크 필드를 미리 남겨놓지 않아도 된다.

- insert(string& name, int age, Date& date, Date& exdate)

insert함수는 입력 받은 데이터를 저장하는 TermsBSTNode를 새로 동적할당해 BST에 삽입한다. 만약 루트 포인터가 nullptr이라면 루트에 노드를 지정해주고, 아니라면 exdate 날짜 변수를 비교해 날짜가 빠르면 좌측으로, 느리면 우측으로 이동해 최종적으로 도착한 노드에 삽입을 진행한다.

- print( )

print함수에선 static stringstream 변수를 이용해 재귀 출력 함수를 반복한 후 결과를 전달해주는 함수이다. 재귀 출력 함수를 통해 트리 내부로 중위 순회가 이루어지고, 결과적으로 트리의 모든 노드를 출력한다.

- `remove(TermsBSTNode* root, TermsBSTNode* parent, string& name)`

`remove`함수는 해당 트리(서브트리)에 대해 입력 받은 이름과 이름이 일치하는 노드를 삭제하는 함수이다. 삭제할 노드를 찾기 위해 내부에서 이름의 사전 순서를 비교하며 반복적으로 재귀 호출하여 노드로 이동한다. 만약 트리가 비어 있다면 `false`를 반환한다.

삭제할 노드에 도착한 경우 해당 노드의 차수에 따라 적절한 삭제 과정을 진행한다. 만약 해당 노드의 차수가 2일 경우 해당 노드의 자손 중 가장 작은 노드를 탐색한 후 (계승자) 계승자의 정보를 얻은 뒤 삭제할 노드의 정보를 계승자의 정보로 덮어씌운다. 이후 계승자에 대한 삭제 명령을 재귀 호출한다.

삭제할 노드의 차수가 1 혹은 0일 경우 남은 노드 `child`에 대한 정보를 얻은 후 만약 삭제할 노드가 루트 노드라면 `child`를 루트 노드로 설정하고, 아닌 경우 조상 노드의 적절한 방향에 `child` 노드를 삽입해준다. 이후 삭제할 노드의 메모리를 해제해주고 `true`를 반환한다.

- `remove(string& name)`

외부에서 `remove`를 호출하려면 루트 포인터를 호출하는 등 번거로운 점이 많은데 외부에서 `remove`명령을 편하게 호출하기 위해 만든 함수이다. 이름을 전달받아 위 `remove`함수에 루트 포인터와 조상으로 `nullptr`을 넣어줘 삭제 시퀀스가 진행되도록 해주는 함수이다.



#### 4. NameBST

MemberQueue에서 방출된 노드의 정보를 이용해 구축하는 자료구조 중 하나이다. TermsBST와 동일한 이진탐색트리 유형의 자료구조이다. 다만 차이점은 TermsBST는 입력되는 회원의 가입약관유형에 따라 트리 자체가 나뉘어 정렬되어 저장되는데, NameBST에는 모든 가입약관유형의 회원이 이름의 사전적 순서에 따라 정렬되어 저장된다는 것이다.

- NameBSTNode class

NameBST를 구성하는 노드를 정의한 클래스이다. 생성자에선 저장할 회원의 정보를 입력 받아 해당하는 멤버 변수를 초기화해주고, 링크필드도 nullptr로 초기화해준다. 이진트리 유형이므로 링크필드는 left, right 2개가 존재한다. TermsBSTNode와의 차이점은 TermsBST는 TermsLIST에 종속되어 생성되어 각 회원의 가입약관유형 정보는 따로 저장하지 않지만, NameBST에선 모든 유형의 회원이 삽입되므로 각 노드는 가입약관유형 정보를 저장하는 멤버 변수를 가진다.

주요 함수로는 멤버 변수를 반환하는 get함수, 멤버 변수를 수정하는 set함수, 출력을 위한 ostream 연산자 오버로딩 함수 등을 구현했다.

- NameBST( )

생성자에선 루트 포인터를 nullptr로 초기화해준다.

- ~TermsBST( )

소멸자에선 모든 노드가 해제될 때까지 아래의 재귀 삭제함수를 호출해 모든 노드의 메모리를 해제해준다.

- recursivedelete(NameBSTNode\* root)

재귀 삭제 함수는 중위 순회를 이용해 모든 노드를 방문하도록 하는 함수이다. 입력 받은 루트의 좌측을 방문한 후, 현재 노드에 대한 메모리 해제를 진행한다. 다만 우측 노드로 이동할 포인터를 남겨놓지 않고 노드를 삭제하면 우측 노드를 방문할 수 없으므로 이에 유의해야 한다.

- recursiveprint(NameBSTNode\* root)

재귀 출력 함수는 마찬가지로 중위 순회를 이용해 모든 노드를 방문한 후 해당 노드의 정보를 출력하는 함수이다. 재귀 삭제 함수와는 달리 노드의 메모리가 해제되지 않으므로 우측 링크 필드를 미리 남겨놓지 않아도 된다.

- insert(string& name, int age, Date& date, Date& exdate)

insert함수는 입력 받은 데이터를 저장하는 NameBSTNode를 새로 동적할당해 BST에 삽입한다. 만약 루트 포인터가 nullptr이라면 루트에 노드를 지정해주고, 아니라면 exdate 날짜 변수를 비교해 날짜가 빠르면 좌측으로, 느리면 우측으로 이동해 최종적으로 도착한 노드에 삽입을 진행한다.

- print( )

print함수에선 static stringstream 변수를 이용해 재귀 출력 함수를 반복한 후 결과를 전달해주는 함수이다. 재귀 출력 함수를 통해 트리 내부로 중위 순회가 이루어지고, 결과적으로 트리

의 모든 노드를 출력한다.

- `remove(NameBSTNode* root, NameBSTNode* parent, string& name)`

`remove`함수는 해당 트리(서브트리)에 대해 입력 받은 이름과 이름이 일치하는 노드를 삭제하는 함수이다. 삭제할 노드를 찾기 위해 내부에서 이름의 사전 순서를 비교하며 반복적으로 재귀 호출하여 노드로 이동한다. 만약 트리가 비어 있다면 `false`를 반환한다.

삭제할 노드에 도착한 경우 해당 노드의 차수에 따라 적절한 삭제 과정을 진행한다. 만약 해당 노드의 차수가 2일 경우 해당 노드의 자손 중 가장 작은 노드를 탐색한 후 (계승자) 계승자의 정보를 얻은 뒤 삭제할 노드의 정보를 계승자의 정보로 덮어씌운다. 이후 계승자에 대한 삭제 명령을 재귀 호출한다.

삭제할 노드의 차수가 1 혹은 0일 경우 남은 노드 `child`에 대한 정보를 얻은 후 만약 삭제할 노드가 루트 노드라면 `child`를 루트 노드로 설정하고, 아닌 경우 조상 노드의 적절한 방향에 `child` 노드를 삽입해준다. 이후 삭제할 노드의 메모리를 해제해주고 `true`를 반환한다.

- `remove(string& name)`

외부에서 `remove`를 호출하려면 루트 포인터를 호출하는 등 번거로운 점이 많은데 외부에서 `remove`명령을 편하게 호출하기 위해 만든 함수이다. 이름을 전달받아 위 `remove`함수에 루트 포인터와 조상으로 `nullptr`을 넣어줘 삭제 시퀀스가 진행되도록 해주는 함수이다.

## 5. Date

Date 클래스는 입력으로 주어지는 개인정보수집일자 정보를 편하게 처리하기 위한 클래스이다. 스켈레톤 코드에 제공되진 않지만 출력 및 개인정보만료일자 연산을 string으로만 처리하기는 불편하기에 이 클래스를 추가했다.

멤버 변수로는 연도, 달, 일 정보를 포함한다. 가입약관에 따른 개인정보만료일자의 계산은 개월수로만 연산이 이루어지는데, 개월수만 더하여 수정한 Date 객체를 반환하는 멤버 함수를 통해 개인정보만료일자를 계산하기 편하게 만들었다. 생성자는 연도, 달, 일 정보를 각각 받는 생성자와 string입력을 구문 분석하여 구축하는 생성자를 구성했다.

그리고 가입약관유형을 관리하는 enum 유형의 변수를 정의해 가입약관유형의 정보를 편리하게 관리할 수 있도록 하였다.

또한 이 클래스의 헤더는 모든 자료구조의 헤더에도 포함된다. 따라서 이곳에 구현에 필요한 모든 라이브러리를 포함했다.

## 6. Manager

Manager에선 모든 자료 구조의 인스턴스를 가지고 파일을 읽고 쓰며 전체적으로 프로그램을 관장하는 클래스이다. 데이터 파일을 읽어와 큐에 집어넣는 것, 특정한 회원 정보를 추가하는 것, 큐를 방출하여 TermsLIST와 TermsBST, NameBST를 구축하는 것, 특정한 회원을 검색하는 것, 특정한 회원의 정보를 삭제하거나 개인정보만료일자가 특정 날짜보다 이전인 모든 노드를 삭제하는 등 구현한 자료구조를 이용해 직접적으로 개인정보를 관리하는 프로그램을 구축한다.

Manager의 멤버 변수는 자료구조인 MemberQueue, TermsLIST, NameBST이다. TermsBST는 TermsLIST에 종속되어 생성되므로 굳이 Manager 클래스에서 멤버로 가지고 있을 필요는 없다.

run에서는 command.txt에 있는 명령어를 읽고 해당하는 명령어 함수를 출력한다. 만약 읽은 명령어가 잘못됐을 경우, 에러코드 1000번을 출력한다.

다음 함수의 구현에 대한 정보는 각 명령어에서 사용되는 함수를 설명한 것이다.

### - Parser(string& data)

Parser함수는 회원의 이름, 나이, 개인정보수집일자, 가입약관유형의 정보가 포함된 한 줄의 string 데이터를 공백을 기준으로 스플릿 해주는 함수이다. 구문분석이라는 단어에 맞게 각 정보에 대한 string변수를 생성하고 stringstream을 통해 공백으로 나누고, int형 자료인 age는 stoi를, 나머지 자료는 그대로 아래의 PUSH 함수에 전달해 구문분석한 데이터가 MemberQueue에 저장되도록 한다.

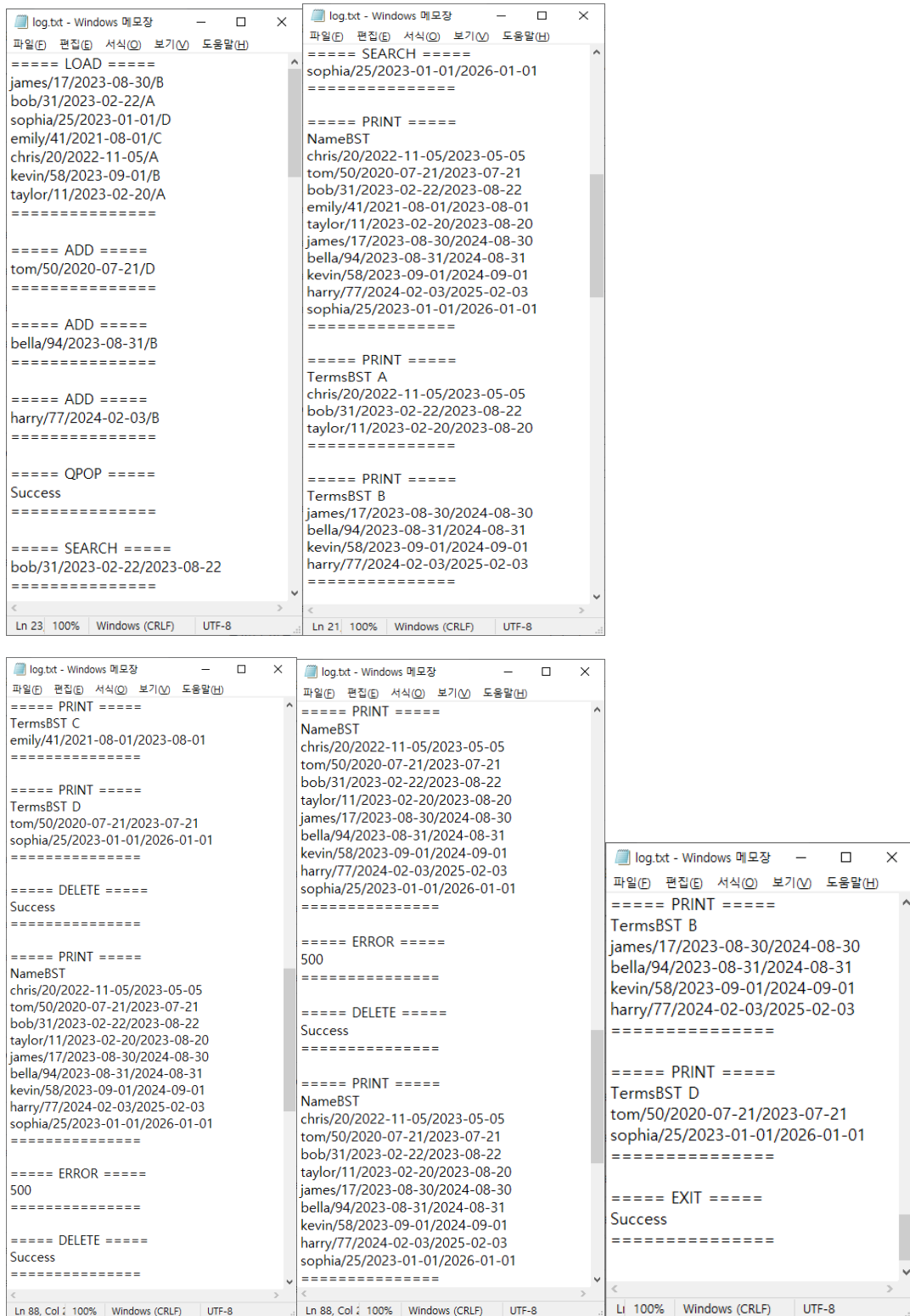
### - PUSH(string& name, int age, string& date, Type type)

PUSH함수는 전달받은 데이터를 MemberQueue에 삽입해준다. 만약 큐가 꽉 찼다면 이 함수에서 예외를 발생시켜, ADD 혹은 LOAD에서 예외를 잡아내도록 한다. 이 함수는 큐에 회원 정보를 삽입하지만 외부 출력이 존재하지 않기 때문에 LOAD와 ADD 모두 사용한다.

각 명령어에 대한 설명은 다음 표와 같다.

LOAD	<p>LOAD는 프로젝트 디렉토리 내부의 data.txt 파일을 읽고 구문 분석하여 사용자의 정보를 얻은 다음 MemberQueue에 삽입하는 명령어이다.</p> <p>파일을 읽고 string 변수에 저장한 다음 구문 분석한 후 큐에 삽입한다. 이 과정을 파일에 데이터가 없을 때까지 반복한다. (PUSH 사용)</p> <p>LOAD에 성공했다면 읽은 회원의 정보를 log.txt 파일에 모두 기록한다. 만약 파일을 읽는데 실패하거나 입력 형식이 올바르지 않을 경우 에러코드 100을 출력하고 프로그램을 종료한다.</p>
ADD	<p>ADD는 data.txt에 존재하는 회원 정보 이외에도 사용자가 직접 입력하여 프로그램에 회원 정보를 추가할 수 있는 명령어이다. 입력 받은 string 데이터를 구문 분석한 후 PUSH를 호출해 MemberQueue에 추가해준다. 큐가 꽉 찼는데 ADD 명령어가 들어오면 에러코드 200을 출력한다.</p>
QPOP	<p>QPOP은 MemberQueue의 모든 노드를 방출해 정보를 TermsLIST(와 연계되는 TermsBST 포함), NameBST에 삽입한다. TermsLIST로 들어간 노드는 가입약관종류에 따라 정렬된 형태로 저장되고 NameBST에는 이름에 따라 정렬되어 저장된다. 만약 큐가 비어 있는데 방출 명령이 들어올 경우 에러코드 300을 출력한다.</p>
SEARCH	<p>SEARCH는 특정한 이름을 입력 받아 NameBST에서 해당 이름을 가진 노드를 탐색해 찾은 경우 해당 노드의 정보를 출력하고, 존재하지 않을 경우 에러코드 400을 출력한다.</p>
PRINT	<p>PRINT는 입력으로 가입약관종류가 들어온 경우 리스트에 전달해 해당하는 노드를 찾은 이후, 해당 노드의 BST를 출력한다. 이름이 입력으로 들어온 경우 NameBST를 출력한다.</p> <p>출력하려는 회원의 정보를 가진 노드가 존재하지 않거나, 자료구조에 데이터가 존재하지 않을 경우 에러코드 500을 출력한다.</p>
DELETE	<p>DELETE는 이름 혹은 날짜를 입력 받아 삭제를 수행하는 명령어이다. 이름을 입력 받은 경우 NameBST와 TermsLIST에서 해당 이름을 가진 노드를 삭제한다. 날짜를 입력 받은 경우 해당 날짜보다 개인정보만료일자가 이전인 모든 노드를 삭제한다.</p> <p>삭제하려는 회원의 정보를 가진 노드가 존재하지 않거나, 자료구조에 데이터가 존재하지 않을 경우 에러코드 600을 출력한다.</p>
EXIT	<p>프로그램의 모든 메모리를 해제하며 프로그램을 종료한다. 모든 메모리는 각 자료구조의 소멸자에서 해제되므로 exit(0)을 호출한다.</p>

## 4. Result Screen



```
log.txt - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
===== LOAD =====
james/17/2023-08-30/B
bob/31/2023-02-22/A
sophia/25/2023-01-01/D
emily/41/2021-08-01/C
chris/20/2022-11-05/A
kevin/58/2023-09-01/B
taylor/11/2023-02-20/A
=====

===== ADD =====
tom/50/2020-07-21/D
=====

===== ADD =====
bella/94/2023-08-31/B
=====

===== ADD =====
harry/77/2024-02-03/B
=====

===== QPOP =====
Success
=====

===== SEARCH =====
bob/31/2023-02-22/2023-08-22
=====

log.txt - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
===== SEARCH =====
sophia/25/2023-01-01/2026-01-01
=====

===== PRINT =====
NameBST
chris/20/2022-11-05/2023-05-05
tom/50/2020-07-21/2023-07-21
bob/31/2023-02-22/2023-08-22
emily/41/2021-08-01/2023-08-01
taylor/11/2023-02-20/2023-08-20
james/17/2023-08-30/2024-08-30
bella/94/2023-08-31/2024-08-31
kevin/58/2023-09-01/2024-09-01
harry/77/2024-02-03/2025-02-03
sophia/25/2023-01-01/2026-01-01
=====

===== PRINT =====
TermsBST A
chris/20/2022-11-05/2023-05-05
bob/31/2023-02-22/2023-08-22
taylor/11/2023-02-20/2023-08-20
=====

===== PRINT =====
TermsBST B
james/17/2023-08-30/2024-08-30
bella/94/2023-08-31/2024-08-31
kevin/58/2023-09-01/2024-09-01
harry/77/2024-02-03/2025-02-03
=====

log.txt - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
===== PRINT =====
TermsBST C
emily/41/2021-08-01/2023-08-01
=====

===== PRINT =====
TermsBST D
tom/50/2020-07-21/2023-07-21
sophia/25/2023-01-01/2026-01-01
=====

===== DELETE =====
Success
=====

===== PRINT =====
NameBST
chris/20/2022-11-05/2023-05-05
tom/50/2020-07-21/2023-07-21
bob/31/2023-02-22/2023-08-22
taylor/11/2023-02-20/2023-08-20
james/17/2023-08-30/2024-08-30
bella/94/2023-08-31/2024-08-31
kevin/58/2023-09-01/2024-09-01
harry/77/2024-02-03/2025-02-03
sophia/25/2023-01-01/2026-01-01
=====

===== ERROR =====
500
=====

===== DELETE =====
Success
=====

===== PRINT =====
NameBST
chris/20/2022-11-05/2023-05-05
tom/50/2020-07-21/2023-07-21
bob/31/2023-02-22/2023-08-22
taylor/11/2023-02-20/2023-08-20
james/17/2023-08-30/2024-08-30
bella/94/2023-08-31/2024-08-31
kevin/58/2023-09-01/2024-09-01
harry/77/2024-02-03/2025-02-03
sophia/25/2023-01-01/2026-01-01
=====

log.txt - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
===== PRINT =====
TermsBST B
james/17/2023-08-30/2024-08-30
bella/94/2023-08-31/2024-08-31
kevin/58/2023-09-01/2024-09-01
harry/77/2024-02-03/2025-02-03
=====

===== PRINT =====
TermsBST D
tom/50/2020-07-21/2023-07-21
sophia/25/2023-01-01/2026-01-01
=====

===== EXIT =====
Success
=====
```

4번째 화면의 ERROR는 유형이 C인 TermsBST가 삭제되어 사라졌기 때문이다

## 5. Consideration

- Date class

날짜를 관리하기 위해 직접 만든 자료형 클래스이다. 연, 월, 일 정보를 한 번에 전달하고, 약관 유형에 따라 정해지는 개인정보만료일자를 편하게 계산하려고 만들었는데, 다 구현하고 나서 이미 비슷한 클래스인 ctime이 라이브러리에서 제공된다는 것을 알게 되었다.

- DELETE

BST에서 노드를 하나 삭제하는 함수는 구현했지만 DELETE의 기능 중 하나인, 개인정보만료일자가 특정 날짜보다 이전인 모든 노드를 삭제하는 기능은 구현하지 못했다.

- 과제 전반

코드를 어느정도 구축하고 디버그를 진행할 때, syntax상으로는 문제가 없지만 데이터가 이동하며 발생한 오류를 찾았는데, 유형이 "this가 nullptr입니다."인 경우 직접적인 원인을 찾기 매우 어려웠고, manager에서 실행 중 한 명령어에서 오류가 난 경우 이후 명령어가 모두 실행되지 않는 문제점이 있었다. 이 오류는 노드의 함수에서 nullptr이 전달되었을 때 발생하는 오류로 판단하고 노드의 함수를 호출하는 모든 구역을 점검해 수정했다.

이러한 경우에는 사전에 고려한 예외 처리문을 통한 검출이 불가능해서 정말 모든 구현부를 다시 들여다봐야 하는 번거로움이 있었다. 앞으로는 순회를 위한 반복문을 구성할 때 해당 노드가 nullptr인지를 조건으로 하지 말고 노드가 존재하는지를 조건으로 설정하는 것이 좋겠다고 판단했다.