|  |
| --- |
| 1. 프로젝트 주제 |

* 카카오톡이나 라인과 같은 메신저 프로그램의 자료관리 방법들을 분석하여 장점을 취하고, 단점을 극복할 수 있는 메신저 프로그램의 자료관리 시스템을 설계하고 구현한다.
* 클라이언트와 서버 클래스를 만들어 정말 서버와 클라이언트를 분리하여 생각하여 프로그램을 디자인한다..

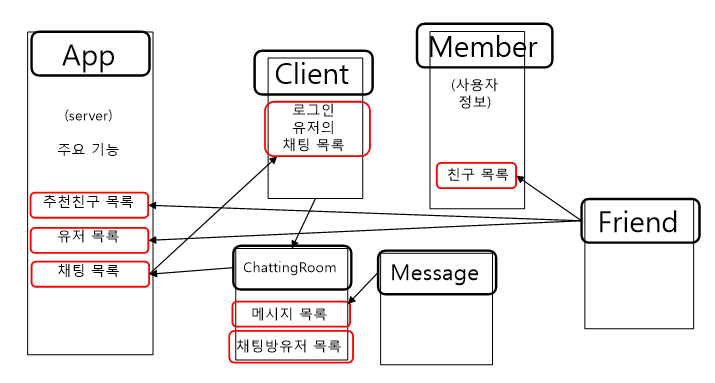
|  |
| --- |
| 1. 프로젝트 목표 |

* 1. 회원 리스트와 회원 개인의 정보(유저 목록, 채팅방 목록, 메시지 내용, 친구 목록 등) 각각의 특성에 맞게 최적의 자료구조를 선택하여 목록 관리와 정렬하는 것이 주된 목적이다.
  2. 일반적인 메신저 프로그램에 요구될 수 있는 기능을 포함한다.
     + 회원 가입 및 탈퇴
     + 로그인 및 로그아웃
     + 친구 리스트, 추천 친구 리스트
     + 회원 정보 수정
     + 친구와의 1:1 채팅
     + 친구 검색 추가 및 삭제
     + 단체 채팅
     + 채팅방 메시지 검색, 복사, 삭제, 전달
  3. 모든 리스트는 template class로 구현한다.
  4. 다양한 자료구조(리스트, 큐, 스택, 트리, 해시)를 실제로 구현해보며 이론과 다르게 애로사항이나 각각 장점이나 단점을 경험 하므로써 실제 코딩 실력을 늘리며 프로그램의 입체적인 시각을 기를 수 있게 한다.
  5. 자료구조의 다양한 사용법을 바탕으로 응용하여 새로운 자료구조를 창작하거나 오픈소스를 이해한다.
  6. 프로그램 데모 실행 시, 원활한 진행을 위하여 입력만 하며 출력은 생략하고 프로그램 완성도를 더욱 높인다.
  7. 서버와 클라이언트의 분리를 위해 서버는 싱글톤 패턴으로, 데이터 송수신(메시지 파일)은 파일 입출력으로 한다.

|  |
| --- |
| 1. 클래스 목록 |

|  |  |
| --- | --- |
| 클래스명 | 기능 |
| App | 프로그램의 기능을 총괄한다. 회원가입 기능과 로그인 기능만 있으며 앱에서 호출 할 수 있는 다른 함수는 없다. 로그인 시 클라이언트 클래스가 동적할당되어 활성화되며 오직 클라이언트에서 요청 시에만 서버의 함수를 실행 시킬 수 있다. 프로그램이 실행 될 때 파일 입력을 하며 유지되는 동안 싱글톤 패턴을 이용하여 유일한 인스턴스로 존재하게 한다. |
| Message | 메시지의 정보를 담는 클래스이다. 보낸사람 ID, 내용이 멤버변수로 선언되며, 유저가 채팅방에서 메시지를 작성될 때 호출된다. |
| Member | 사용자의 정보가 저장될 클래스이다. 이름, 전화번호, 카카오 계정ID, 비밀번호, 프로필 사진, 상태메세지, 메시지 함이 멤버변수로 선언되며, 이 때, Primary Key를 기준은 ID의 해시함수 값이다. 연산자 오버로딩을 구현하여 정렬이 수행되도록 한다. |
| Client | App에서 로그인한 정보를 포인터로 갖으며 로그인 유저의 채팅리스트를 모두 받아온다. 유저가 실제로 실행하는 화면이며 직관적인 UI구조로 유저의 선택을 호출하여 클라이언트가 직접 실행할 수 있으면 직접, 서버가 관여하여하 한다면 서버의 함수를 호출하여 클라이언트의 기능의 본질을 구현한다. 채팅, 친구 관리 등 구현한 대부분의 함수가 내장 되어있으며 카카오톡 시스템이다. |
| SortedlinkedList | Class로 작성하여 사용자가 원하는 자료형을 List로 관리할 수 있으며, Generic Class를 구현하기 위해 List 자료형에 종속적인 메소드를 사용하지 않으며, 자료형 내부의 연산자 오버로딩을 통해 연산이 수행되도록 한다. 정렬이 필요하면서 자료양이 적은 곳에서 필요한다. 이 프로그램에서는 친구 목록, 추천 친구 목록 |
| DoublyLinkedList | Class로 작성하여 사용자가 원하는 자료형을 List로 관리할 수 있으며, Generic Class를 구현하기 위해 List 자료형에 종속적인 메소드를 사용하지 않으며, 자료형 내부의 연산자 오버로딩을 통해 연산이 수행되도록 한다. 정렬이 필요하지 않으며 중간 삽입 삭제가 잦거나 검색 그리고 자료의 양이 적당할 때(500개 미만) 사용한다. 이 프로그램에서는 채팅 내용, 임시 채팅 목록 |
| RedBlackTree | Class로 작성하여 사용자가 원하는 자료형을 List로 관리할 수 있으며, Generic Class를 구현하기 위해 List 자료형에 종속적인 메소드를 사용하지 않으며, 자료형 내부의 연산자 오버로딩을 통해 연산이 수행되도록 한다. **BST의 발전 버전인 자가균형트리이며 검색 삽입 삭제등 일정한 속도를 요구하면서 자료형이 어느정도 있어도 괜찮은 자료구조가 필요한 곳에 사용한다. 이 프로그램에서는 유저리스트** |
| HASH | Class로 작성하여 사용자가 원하는 자료형을 List로 관리할 수 있으며, Generic Class를 구현하기 위해 List 자료형에 종속적인 메소드를 사용하지 않으며, 자료형 내부의 연산자 오버로딩을 통해 연산이 수행되도록 한다. **해시 함수는 각 자료형에서 각자의 원하는 바에 오버로딩 하며 자료가 방대하여 다이렉트 액세스가 필요한 경우에 사용한다. 이 프로그램에서는 채팅 목록** |

|  |
| --- |
| 1. 프로그램 구조 및 자료구조 선택 이유 |



총 6가지의 자료구조를 채택하였으며, 앞에 명시 한 것과 같이 다양한 자료구조를 직접 사용해봄으로 장점, 단점 코딩 상 애로사항 및 점을 체험하였다. 데이터 량은 대한민국 내의 카카오톡 시스템과 비슷하다고 가정(유저 1천만명, 개인이 쥐고 있는 카카오톡 방 20개)하였으며, 자료구조의 선택 사유는 다음과 같다.

App 클래스

* 1. 유저 리스트 : 유저리스트는 1천만명이 사용하기 때문에 기존의 링크드리스트와 같은 선형 자료구조로는 유저의 검색, 삭제, 추가에 대해 불균형이 심하고(추가가 빠르면 검색이 느리거나, 검색이 효율적이여도 추가가 느리거나) 방대한 자료구조에는 적합하지 안하고 생각하였다. 때문에 트리구조를 생각하였으며 트리 구조도 심각한 비대칭 트리가 된다면 선형자료구조와 다를 바가 없기 때문에 자가균형트리인 레드블랙트리를 사용하였다. 때문에 삽입, 삭제, 검색 속도가 logn으로 안정적인 효율을 낼 수 있도록 하였다.
  2. 채팅목록 리스트 : 채팅목록 리스트는 유저의 양보다 훨씬 많기 때문에(유저가 다수의 채팅방을 가지고 있음) 서버에서 로그인 실행 시 클라이언트에게 채팅방 목록을 빠르게 넘겨주려면 트리 구조도 불편하다고 생각하였다. 예를들어 유저가 1천만명이고 유저 1명당 채팅방을 20개씩 갖고 있다면 2억개의 검색을 해야하는데 logn도 너무 비효율적이라고 생각하였다. 따라서 log 1인 해시 자료구조를 선택하였다. 해시 테이블의 인덱스는 유저의 아이디를 해시함수에 통과시킨 값이며, 해시 테이블 인덱스 위에는 유저의 채팅 방 목록이 있으며, 유저 아이디의 각 자리수를 아스키코드화 하여 충분히 큰 소수를 곱하여 가중치를 달리해 중복을 최소화 하였다. 해시테이블위에 접근하면 테이블에는 채팅방이 링크드리스트로 결합되어있어 검색 범위를 2억에서 단 1로 줄일 수 있었다.
  3. 추천친구 리스트 : 친구의 친구를 추천하는 알고리즘이므로 양이 0~1,000명이라 생각하였고 이름 순으로 정렬하기 위해 sortedlinkedList를 사용하였다. 1000개의 데이터는 충분히 선형자료구조로 삽입,추가,삭제 연산을 빨리 할 수 있기 때문에 사용하였다.

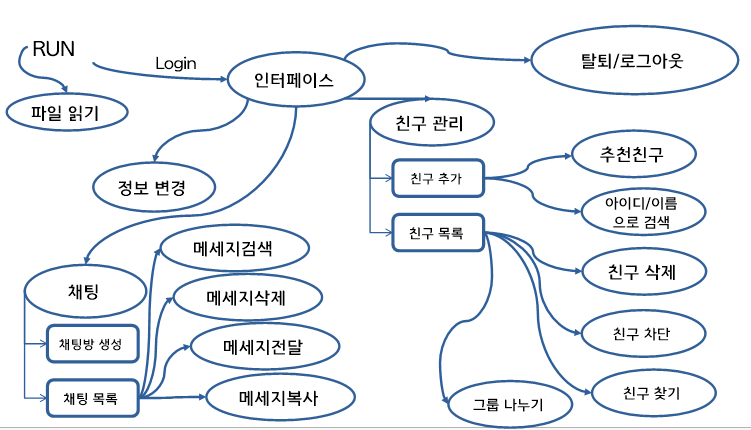
Client 클래스

1. 임시 채팅 목록 리스트 : 서버에서로부터 받아온 임시 채팅목록이며 자료의 양이 0~100이며, 충분히 100개 내외는 자료 구조에 따라 속도 영향을 받기 어렵기 때문에 안정성이 높은 더블리 링크드리스트를 사용하였다. 링크드리스트보다 접근이 중간에서도 뒤로갈수 있고 링크를 하나 잃어버려도 다시 자료를 살릴수 있는 안정적인 자료구조이므로 선택하였다. 새로운 메시지가 생기면 채팅방 목록이 다시 맨위로 올라오므로 채팅방목록에 적합한 자료구조이다.

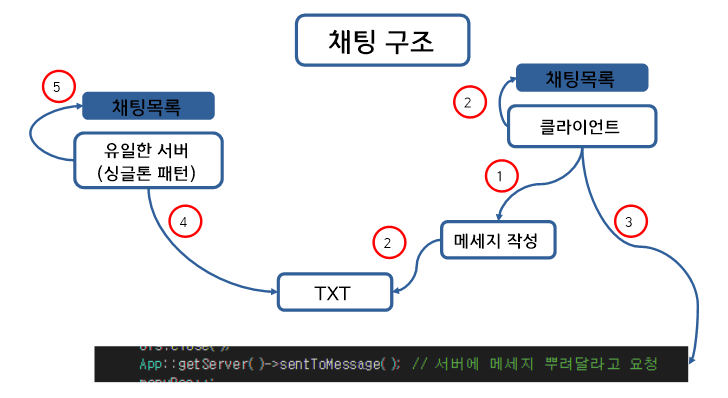
ChattingRoom 클래스

1. 채팅 내용 : 클라이언트클래스의 임시채팅 목록리스트 채택이유와 같다. 자료를 잃어버리지 않도록 안정성이 높은 더블리링크드 리스트를 사용하였고 선형구조를 선택하여 직관적인 데이터 관리를 쉽게 하도록 하였다.
2. 참여자 리스트 : 자료형이 방대하지 않고 이름 순으로 정렬 하여 채팅 참여자 목록을 쉽게 알 수 있도록 구현하기 위해 sortedLinkedList 사용.

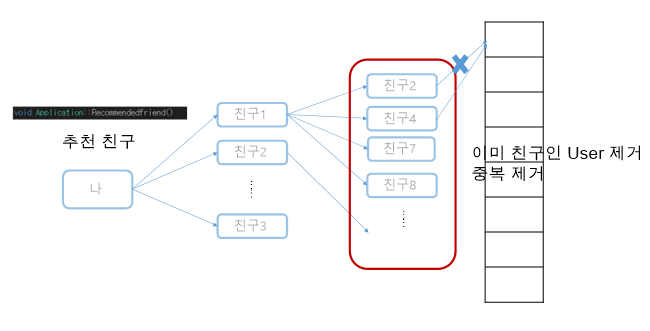
|  |
| --- |
| 1. 기능 요약도 |



전체 기능 요약도



채팅구조 요약도



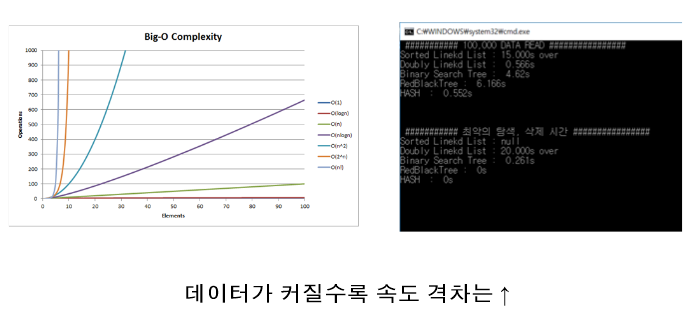
추천친구 기능 요약도

|  |
| --- |
| 1. 체크리스트 |

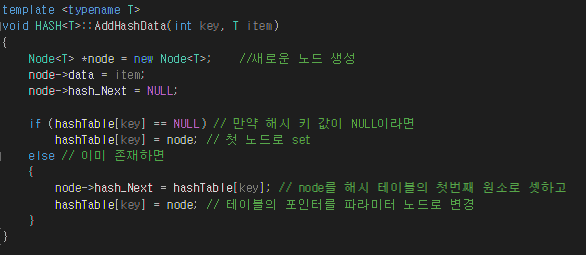


다른 점 : 친구리스트를 서버가 직접 갖고 있게 하려면 해시함수를 유저 아이디로 하여 친구 리스트를 다이렉트 엑세스 할 수 있게 구현 할 수 있으나, 효율적인 측면에서 멤버가 쥐고있는게 낫다고 판단하여 멤버에 친구 목록을 넣음

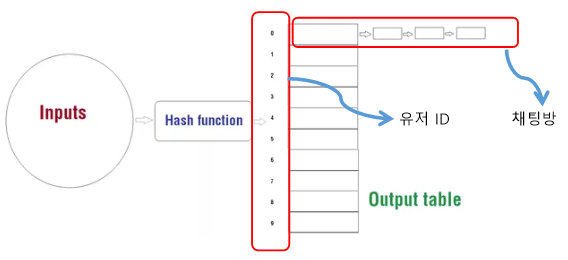
|  |
| --- |
| 1. 나만의 기능 |



자료 형 간 데이터 속도 측정(100,000개의 데이터 난수 스트링)

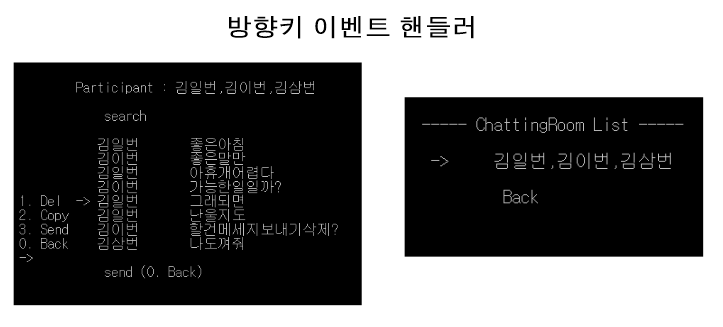


해시 테이블에 노드를 추가(링크드리스트 결합)으로



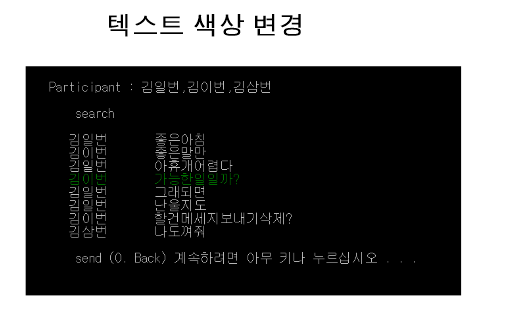
해시 테이블의 인덱스를 유저 ID로 채팅방을 해당 테이블에 링크드리스트로 결합함으로써

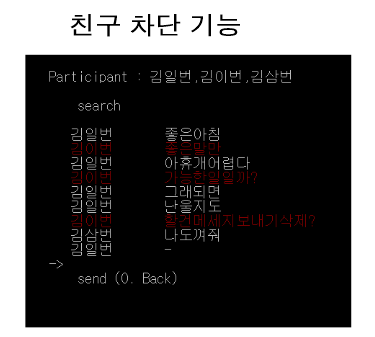
채팅방 접근 효율을 극대화

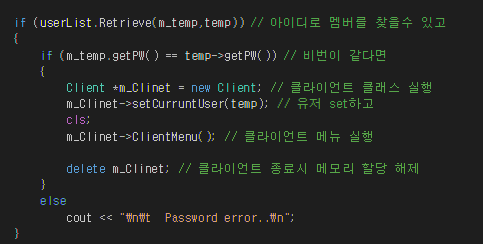


방향키 이벤트 속에 또 방향키 이벤트로 유저 친화적 UI 구현

메시지를 방향키로 선택하면 서브메뉴(삭제, 복사, 전달) 출력







로그인 시 클라이언트 객체 생성 후 메뉴 출력만 한 후 서버는 아무 동작 안함.

서버와 클라이언트 텍스트와 싱글톤패턴으로 이원화.

|  |
| --- |
| 1. 개발 환경 |

* Microsoft Visual Studio 2017 C++