**FILE PROCESSING (CSE4095)**

**Project 2**

학과 : 컴퓨터공학과

학번 : 20181593

이름 : 계인혜

**<목차>**

1. **프로젝트 설정상의 가정**
2. **클래스 다이어그램(standard UML specification)**
3. **프로그래밍 문제 요구 사항**
   1. **chapter7 13**
   2. **chapter7 16**
   3. **chapter7 17**
   4. **chapter7 18**
   5. **관리자 모드**
   6. **인덱스**
4. **파일 기능 설명**
5. **프로그램 흐름도 및 메뉴 이용 방법**
6. **프로젝트 설정상의 가정**

1.1 사용자가 GamePurchaseSystem을 처음으로 실행하면 memberTest, GameTest, PurchaseTest 함수로 인해 처음 레코드 10개가 출력된다.

1.2 사용자는 레코드의 검색, 삭제, 수정, 삽입 시 정해진 멤버변수의 타입과 길이에 맞는 값을 입력한다. (ex : GameID는 8글자의 숫자, PurchaseID는 12글자의 숫자)

1.3 새로 추가하는 Member는 level 9로 일반 사용자이다.

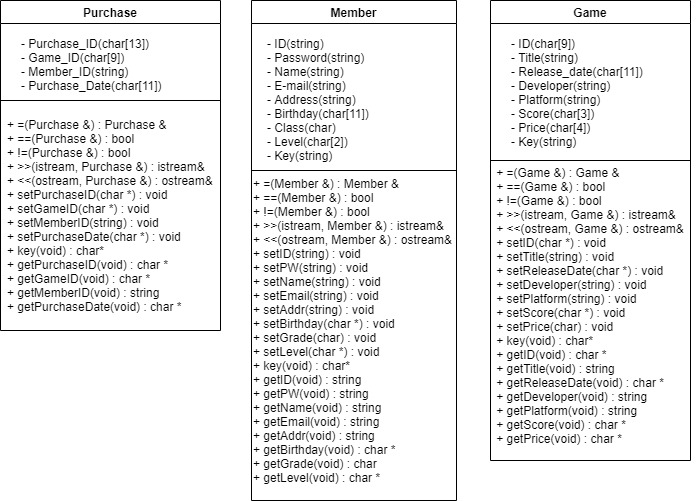
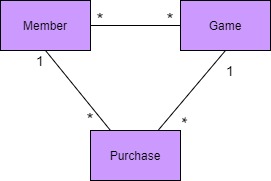
1.4 txt파일을 새로 만드는 경우 listOfMember.txt에 admin, TestUser에 대한 레코드를 따로 추가해야 한다. (기존에 들어있는 파일은 해당 레코드가 포함되어 있다.)

1.5 GamePurchaseSystem에서 관리자 모드와 일반 회원 모드의 메뉴가 같지만, 작동은 명세서의 모든 사항을 지킨다.

1.6 레코드 삭제 시, 어떤 클래스(Member, Game, Purchase)의 레코드를 지울 지 묻는 메뉴가 없다. 삭제하고자 하는 레코드 타입의 ID를 입력하면 의도대로 프로그램이 작동한다.

1.7 입력 받는 레코드에는 공백이 포함되지 않는다.

1. **클래스 다이어그램(standard UML specification)**



클래스 다이어그램은 위와 같다. 이때, Member와 Game은 다대다 관계를 가지며 Member와 Purchase는 1대다 관계를, Game과 Purchase는 1대다 관계를 갖는다. 이때 Purchase 클래스의 GameID와 MemberID는 dependancy가 존재하기 때문에 레코드 삭제 시 참조 무결성을 유지해야 한다. 즉, MemberID 또는 GameID가 삭제되면 해당 아이디를 갖는 Purchase 레코드 역시 삭제된다. 마찬가지로 레코드를 추가할 때도 참조 무결성이 유지된다. Purchase 레코드를 추가할 때 Member 레코드와 Game 레코드에 없는 ID를 입력할 수 없다.

1. **프로그래밍 문제 요구 사항**
   1. **chapter7 13**

textind.cpp에 구현하였다. Int FirstRecAddr() 함수는 가장 작은 키에 대한 참조를 리턴하며, int NextRecAddr() 함수는 다음키에 대한 참조를 리턴한다. 만약 다음 키가 존재하지 않는다면 -1을 리턴한다.

* 1. **chapter7 16**

textind.cpp의 search method를 수정하여 이진 탐색으로 변경하였다. L, h인덱스를 사용하여 (각각 low, high index를 의미) key값이 mid인덱스의 값보다 작으면 h 인덱스를 mid -1로, key 값이 mid 인덱스의 값보다 크면 l 인덱스를 mid+1로 설정하였다. 만약 key값과 mid 인덱스의 값이 일치하면 해당하는 키에 대한 참조를 리턴한다. 이 과정에서 key 값을 찾지 못하면 -1을 리턴한다.

* 1. **chapter7 17**

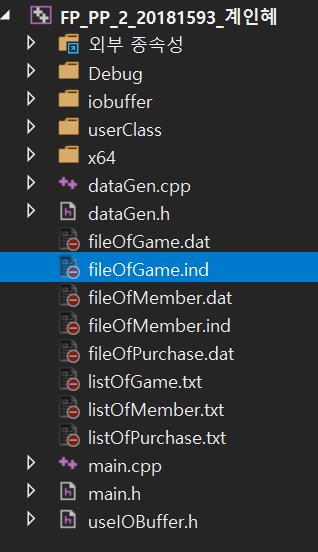
indfile.h의 int Remove(Rectype &) 함수를 구현하였다. 레코드의 키를 찾아 해당하는 키를 인덱스 파일에서 지운다. 성공하면 1을 리턴하고, 그렇지 않으면 0을 리턴한다.

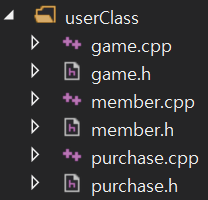
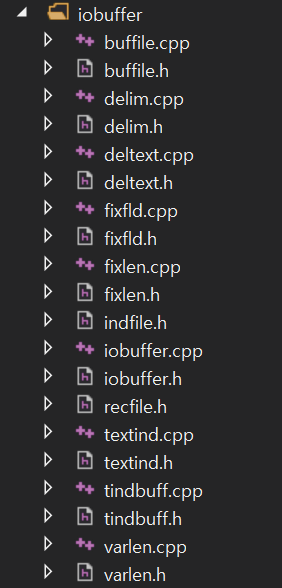
* 1. **chapter7 18**

main.cpp에 void renewIdx() 함수를 구현하였다. Member, Game, Purchase 클래스에 해당하는 .dat 파일을 읽어 이 레코드의 키를 .ind 파일에 저장한다. 이 함수를 사용하여 존재하는 데이터 파일을 열고 읽고 파일 내의 레코드로부터 인덱스를 생성하여 TextIndexedFile 객체를 초기화 할 수 있다.

* 1. **관리자 모드** 
     1. GamePurchaseSystem에 들어가면 로그인 화면이 뜨고, 로그인에 성공하면 멤버의 레벨에 맞는 기능을 수행할 수 있다. 일반 회원과 관리자의 메뉴를 따로 구성하지 않았지만, 일반 회원의 접근 권한을 벗어나는 명령은 수행할 수 없도록 코드를 구성하였다.
     2. 프로젝트1에서 레코드의 삽입, 삭제, 수정, 검색을 각각 하나의 함수로 해결하였다. 그러나 이번 프로젝트에서는 명세서의 요구조건(참조 무결성)을 맞추기 위해 레코드의 삭제 함수를 세분화하였다. DeleteRecord에서 삭제하고자 하는 클래스의 ID를 입력받은 후 그 레코드가 Member, Game, Purchase 인지 결정한다. 만약 Member 레코드를 삭제한다면 참조 무결성을 해결하기 위해 해당하는 Purchase 레코드도 삭제해야 한다. 마찬가지로 Game 레코드를 삭제한다면 해당하는 Purchase 레코드도 삭제해야 한다. 시간 부족의 관계로 Delete의 경우 어떤 레코드를 삭제하고자 하는지 묻는 메뉴를 구성하지 못했다.
     3. 일반 회원의 레코드 삭제 시 수행 권한을 체크해야 한다. 일반 회원은 Game 레코드를 삭제할 수 없고, 자기 자신의 Member 레코드와 자신의 Purchase 레코드만 삭제할 수 있다. 이를 checkAccess라는 함수를 구현하여 해결하였다.
  2. **인덱스**
     1. main.cpp에 renewIdx 함수를 구현하여 .ind 파일이 존재하지 않는 경우 각각의 레코드 파일에 대한 .ind 파일을 생성하였다.
     2. TextIndex와 TextIndexBuffer 클래스를 사용하여 레코드의 삽입, 삭제, 수정의 경우 참조 무결성을 유지할 수 있었다. 예를 들어 Member의 삽입 시, 먼저 .dat 파일에 해당 레코드를 추가하고 .ind 파일에도 추가하는 방식으로 구현하였다. 레코드의 삭제와 수정의 경우에도 같은 방식으로 진행된다.

1. **파일 기능 설명**

****파일 기능 설명에 앞서 먼저 프로젝트의 파일 구성은 다음과 같다. 각각은 순서대로 기본 폴더, iobuffer 폴더, UserClass 폴더의 파일 구성을 나타낸다.

****

* 1. **기본 폴더**

4.1.1 dataGen 파일은 각 클래스의 필드에 맞는 타입과 최대 길이등을 고려해 txt 파일을 만들어주는 역할을 한다. 만약 저장된 txt 파일을 사용하지 않고 새로 txt 파일을 만들게 된다면, listOfMember.txt 파일에 admin, testUser의 레코드를 직접 추가해야한다.

4.1.2 프로젝트1의 메뉴 형식을 그대로 사용하였다. 그러나, GamePurchaseSystem 메뉴의 형식은 수정하여 이용의 편의성을 높였다.

4.1.3 레코드 처리 방식을 RecFile에서 TextIndexdFile을 이용하도록 변경하였다.

4.1.4 GamePurchaseSystem을 처음으로 실행하면 저장된 .dat 파일을 사용하여 .ind 파일을 만든다. 이 과정에서 memberTest, gameTest, purchaseTest 함수가 수행되기 때문에 프로젝트 1에서 구현한대로 처음 레코드 10개가 출력된다.

4.1.5 GamePurchaseSystem을 실행하면 로그인 화면이 뜨도록 기능을 추가했다.

4.1.6 레코드 삭제 시 참조 무결성을 유지하기 위해 delete 함수를 여러개로 쪼갰다. (deleteRecord, deleteMemer, deleteGame, deletePurchase)

4.1.7 checkAccess 함수를 추가하여 로그인 한 유저가 명령을 수행할 수 있는 권한이 있는지 level을 통해 확인한다.

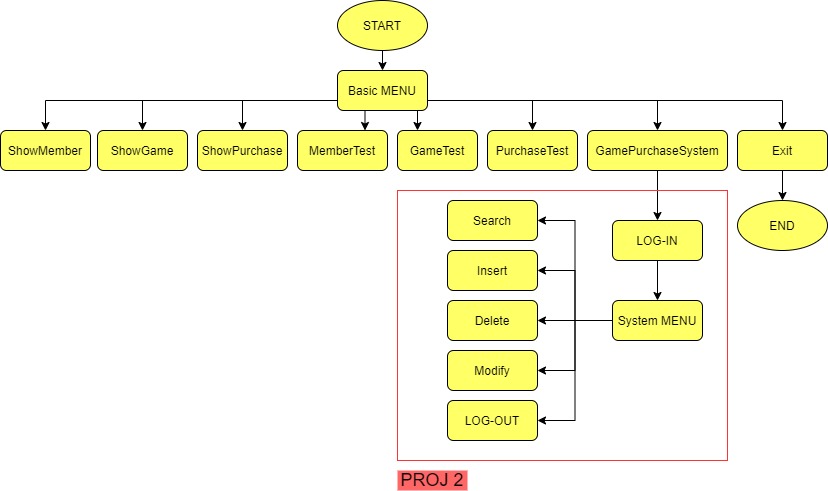
4.1.8 레코드 탐색 알고리즘을 순차탐색에서 이진탐색으로 변경하였다.

4.1.9 searchRecordPW 함수를 추가하여 아이디와 비밀번호가 일치하는지 확인하고, 일치하면 해당 레코드의 레벨을 리턴하여 관리자 모드와 일반 유저 모드를 구분하였다.

4.1.10 GamePurchaseSystem을 실행하고 나가게 되면 자동으로 로그아웃이 된다.

4.1.11 레코드 삭제 시 클래스에 관계없이 삭제하고자 하는 클래스의 ID를 입력한다. 이때, 해당하는 클래스만 삭제하는 것이 아니라 참조 무결성을 유지하기 위해 deleteMember, deleteGame, deletePurchase 함수를 모두 수행한다.

* 1. **userClass 폴더** 
     1. 각 클래스에 key() 메소드를 추가하였다. .ind 파일에 사용되는 키를 리턴하는 함수이다.
     2. Member 클래스에 level이라는 변수를 추가하였다. 관리자 모드와 일반 회원 모드의 구분을 위한 멤버 변수이다.
     3. 각 클래스에 getLength 메소드를 추가하였다. 클래스에 속한 멤버 변수의 각 길이의 합을 리턴한다.
  2. **iobuffer 폴더**
     1. 이 폴더 내에는 조교님께서 제공해주신 기존의 소스 및 헤더 파일이 담겨있다.
     2. Textind 파일에는 TextIndex 클래스가 구현되어 있다. 인덱스 파일을 구현하기 위한 char\*\* keys와 이에 해당하는 주소를 저장하고 있는 int \* 배열 recaddrs을 기본으로 갖는다.
     3. Tindbuff 파일에는 FixedFieldBuffer를 상속받은 TextIndexBuffer 클래스가 구현되어 있으며 인덱스 파일의 내용을 담는 버퍼로 사용된다.
     4. Indfile 파일에는 TextIndexedFile 클래스가 구현되어 있으며, 이 클래스는 클래스의 종류에 따라 해당 타입에 맞게 함수를 수행할 수 있는 템플릿 클래스이다. 레코드 내용을 갖는 recordFile과 인덱스 파일에 접근할 수 있다.

1. **프로그램 흐름도 및 메뉴 이용 방법**

GamePurchaseSystem에 처음 접속하면 dataTest 함수를 통해 .dat 파일을 생성하고, renewIdx 함수에서 이 파일을 기준으로 다시 .ind 파일을 생성한다. 이 과정에서 초기 레코드가 10개씩 출력된다.

이후, 로그인 화면에서 성공적으로 로그인을 마치면 proj2를 수행하기 위한 메뉴가 출력된다. 원하는 작업을 선택하고 콘솔 창의 안내를 따라 입력하면 작업을 수행한다. 이때, 로그인 한 유저의 level에 따라 허용되지 않은 접근 시 경고 메시지가 뜨며 작업을 수행하지 않을 수도 있다.

기본적으로 메뉴에 뜨지 않은 숫자를 입력하면 이전 화면으로 돌아간다. GamePurchase System 함수를 종료하면 자동으로 로그아웃이 되며 이후에 GamePurhaseSystem 함수를 수행한다면 로그인을 다시 해야 한다.