자료구조 과제 보고서

[학생정보 관리 프로그램 설계]

12141579 윤찬미
010-9287-1679
cmmcme0604@gmail.com

I. 개요

• 설계의 목적
– 학생 정보 관리 프로그램을 Hash Table을 이용해 설계한다.

• 요구 사항
– 새로운 과목을 수강하면 학점에 대한 정보를 변경한다.
– 요청에 따라 이메일주소를 변경할 수 있다.
– 초기에 파일에 저장된 학생들의 정보를 입력받아 이중 해싱을 이용하여 Hash Table에 입력한다.
– 새로운 학생을 추가 할 수 있다.

• 개발 환경
– Microsoft Visual Studio 2013
– 언어 : C++

II.자료구조 및 기능

•Hash Table을 구현하기 위한 자료구조

1. Hash Node

```
typedef struct Hash  //기본적 Hash 구조체 선언
{
    int keyval;  //학번
    int TotalS;   //학점
    string EM;  //이메일
    Hash() : keyval(0), TotalS(0) {};  //각 값을 초기화 한다.

}Hash;
```

– 구조체를 이용하여 Hash를 생성해 주었다.
– Hash 안에는 key값이 될 학번 keyval, 학점 TotalS, 이메일 EM을 선언해 주었고, 각 값을 초기화 하였다.

• 프로그램의 기능

1-1. insert_FileData

-> 파일 입 출력을 통해 들어온 데이터 값을 저장한다. 이중 해싱을 통해 비어있는 key의 위치를 찾고 해당 학번과 이메일을 삽입 해준다.
-> 1차 해싱을 하였을 때 비어 있는 공간을 찾지 못하면 2차 해싱 값을 더해주며 비어있는 공간을 찾는다

1-2. void insert_stu(Hash *h, int Num, String Email);

-> Num이라는 학번을 가지고 있는 데이터를 삽입해주는 함수이다. 이중 해싱을 통해 비어있는 key의 위치를 찾고 해당 학번과 이메일을 삽입 해준다.
-> 1차 해싱을 하였을 때 Probe를 1로 초기화 한 후, 비어 있는 공간을 찾지 못하면 2차 해싱 값을 더해주며 비어있는 공간을 찾는다. 2차 해싱 값을 더해줄 때, Probe의 횟수를 1 증가 시키고, 비어있는 공간을 찾은 후 데이터의 갯수를 1 증가 시킨다.

2. void Search (Hash *h, int Num);

-> Num이라는 학번을 가지고 있는 데이터를 찾는 함수이다. insert 비슷한 알고리즘으로 진행된다. Num의 1차 해싱 한 후 키 값을 찾은 후, Probe를 1로 초기화 한다. 1차 해싱을 하였을 때, 해쉬 테이블에서 key번째 위치에 해당 학번이 존재하지 않는다면 Not Found 를 출력하고,
해당 학번이 아니라면 2차 해싱 값을 key값에 더해주며 학번의 위치를 찾은 후, 반복문을 빠져나간다. 이 때, 해당 위치를 해쉬형 포인터 변수 Position에 저장한다.

3. void insert_score(Hash *h, int Num, int S);

-> 2의 함수 Search를 통해 해당 학번의 위치를 찾고 S학점을 더한 후 총 학점이 24학점을 초과하면 학점을 더하지않고 Probe 횟수와 Exceeded를 출력하고, 초과하지 않았을 경우 Probe 횟수와 S학점을 더한 총 학점을 출력한다.

4. void Trans(Hash *h, int Num, String Email);

-> 2의 함수 Search를 통해 해당 학번의 위치를 찾고 해당 학번이 존재 하였을 경우 이메일을 변경 해 준 후, Probe횟수를 출력한다.

5. void Print(Hash *h, int Num);

-> 2의 함수 Search를 통해 해당 학번의 위치를 찾고 해당 학번이 존재 하는 경우 학번, 학점, 이메일주소를 출력한다.

III.기능별 알고리즘 명세

1-1. Algorithm insert_FileData
1-2. Algorithm insert_stu(Hash *h, int Num, String Email)

//filedata 입력과 새로운 학생 입력의 알고리즘이 유사하기에 1개만 정의

	시간복잡도
key <- Num%n;	1
jump <- q - (Num%q);	1
Probe <- 1	1
if h[key].keyval = 0	1
h[key].keyval <- Num;	1
h[key].Em <- Email;	1
else then	1
begin	
key <- key+jump;	1
key <- key%n;	1
Probe <- Probe+1;	1
if ¬ h[key].keyval = 0; goto begin	n
h[key].keyval <- Num;	1
h[key].Em <- Email;	1
m <- m+1;	1
	O(n)

2. Algorithm Search (Hash *h, int Num)

	시간복잡도
key <- Num%n;	1
jump <- q - (Num%q);	1
Probe <- 1	1
begin	1
if h[key].keyval = 0	1
print(Probe, “Not Found”)	1
return ;	1
key <- key+jump;	1
key <- key%n;	1
Probe <- Probe+1;	1
if ㄱ h[key].keyval = Num; goto begin	n
Position <- &h[key]	1

O(n)

3-1. Algorithm print(Hash *h, int Num)
3-2. Algorithm Trans(Hash *h, int Num, String Email)
3-3. Algorithm insert_score(Hash *h, int Num, int S)

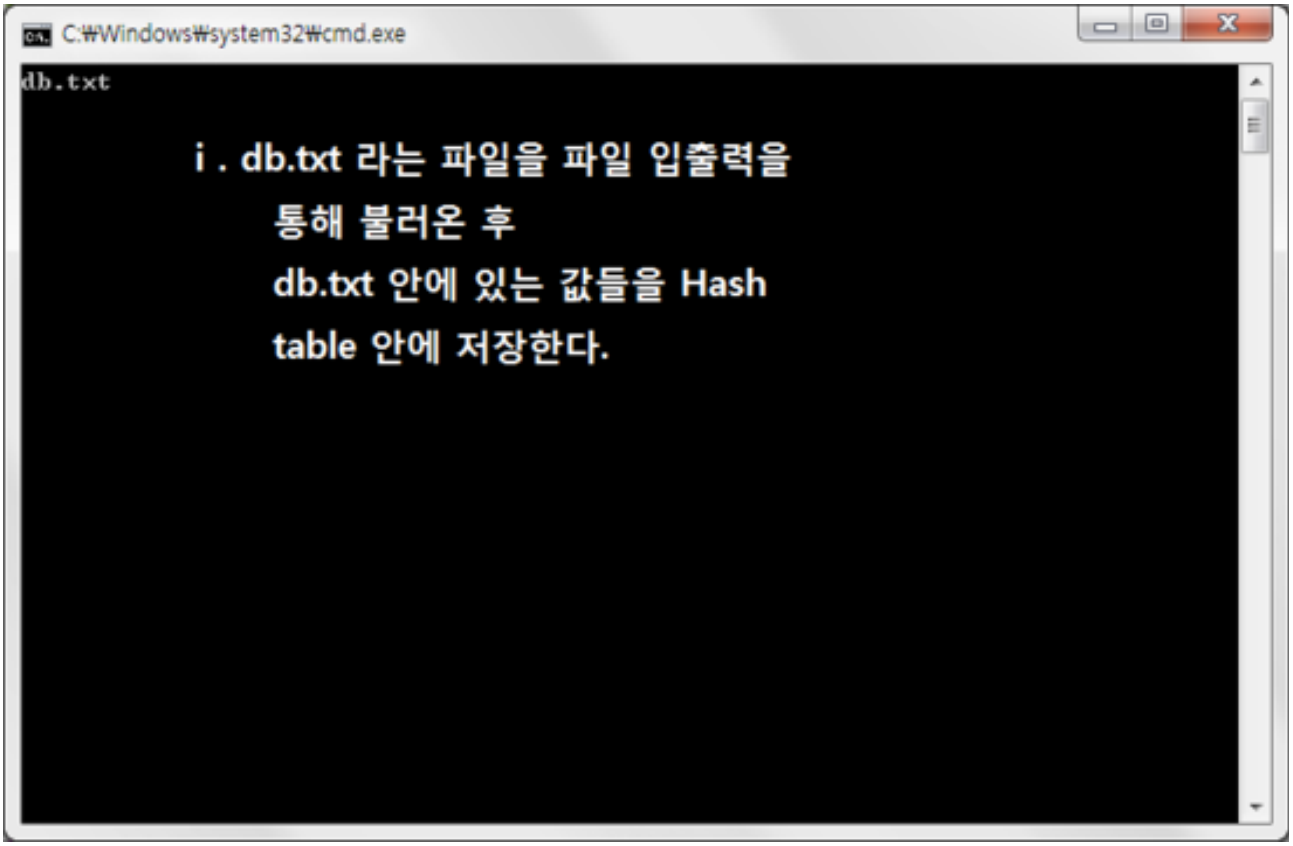
	시간복잡도
Search(h,Num)	n
if(ㄱ Position = NULL)	1
Print(Probe,Num,…)	1
Position -> EM = Email	1
if(Position->TotalS + S>24) than	1
Print(Probe,”Exceeded”)	1
else than	
Position->TotalS = Position->TotalS+ S;	1

O(n)

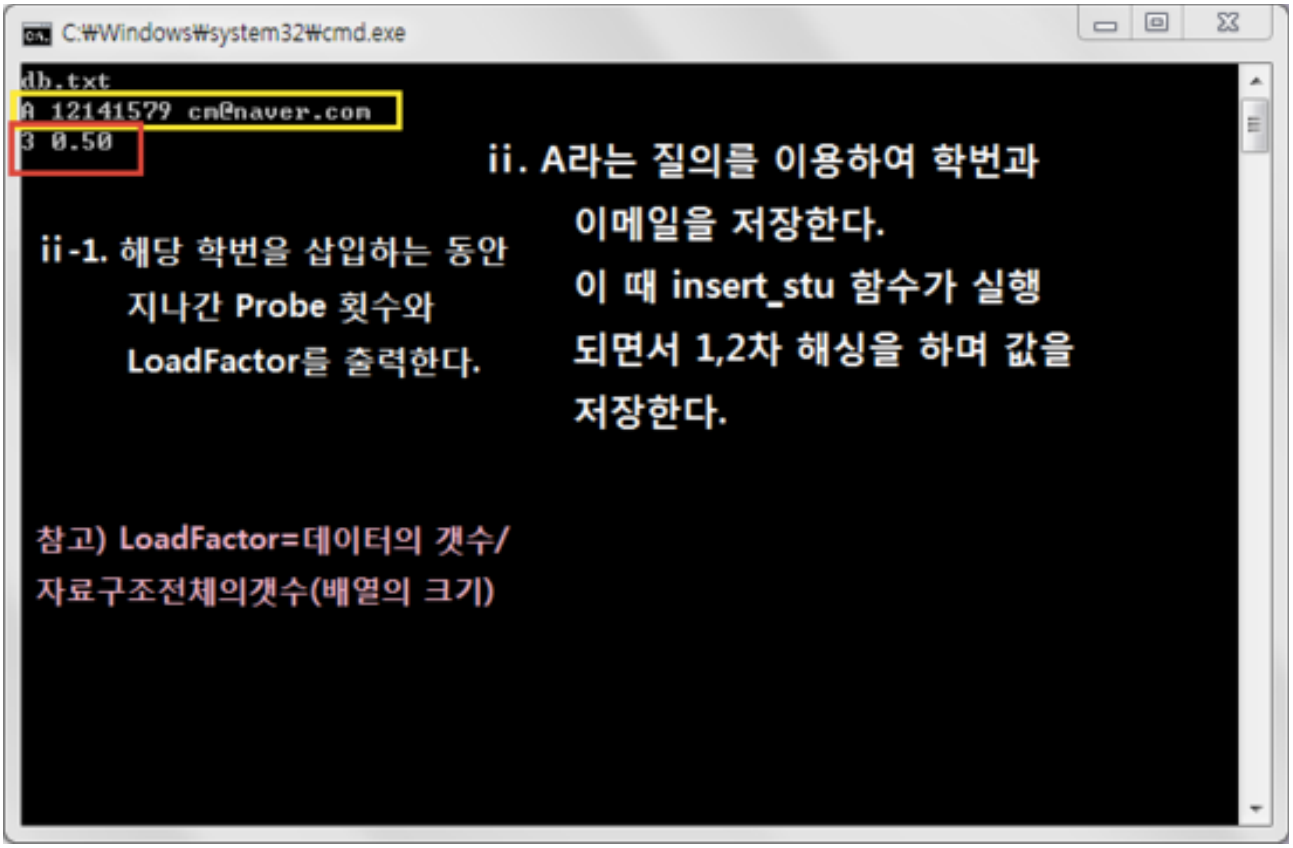
IV. 인터페이스 및 사용법

– 실행화면 스크린샷과 기능 설명

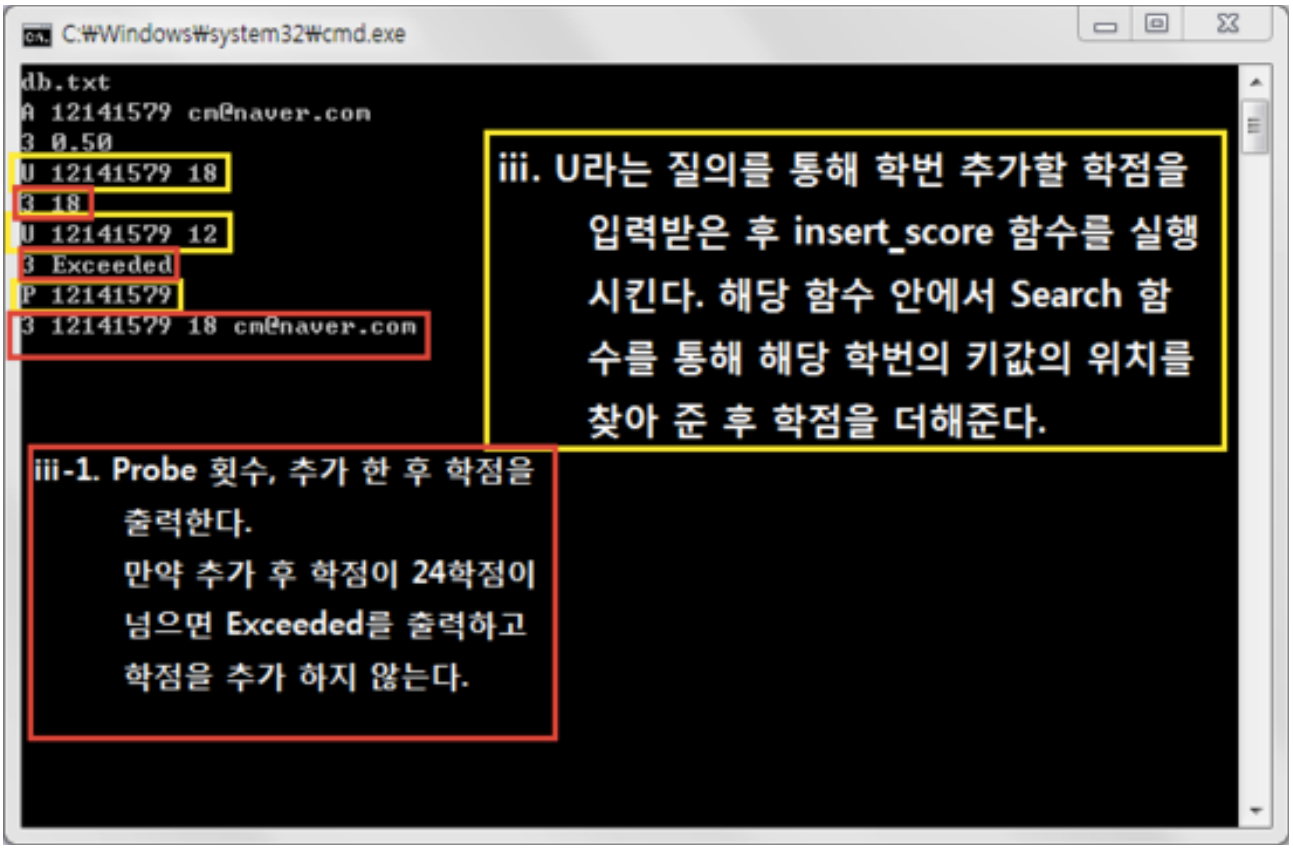
1) 파일 입 출력을 통한 파일 불러오기



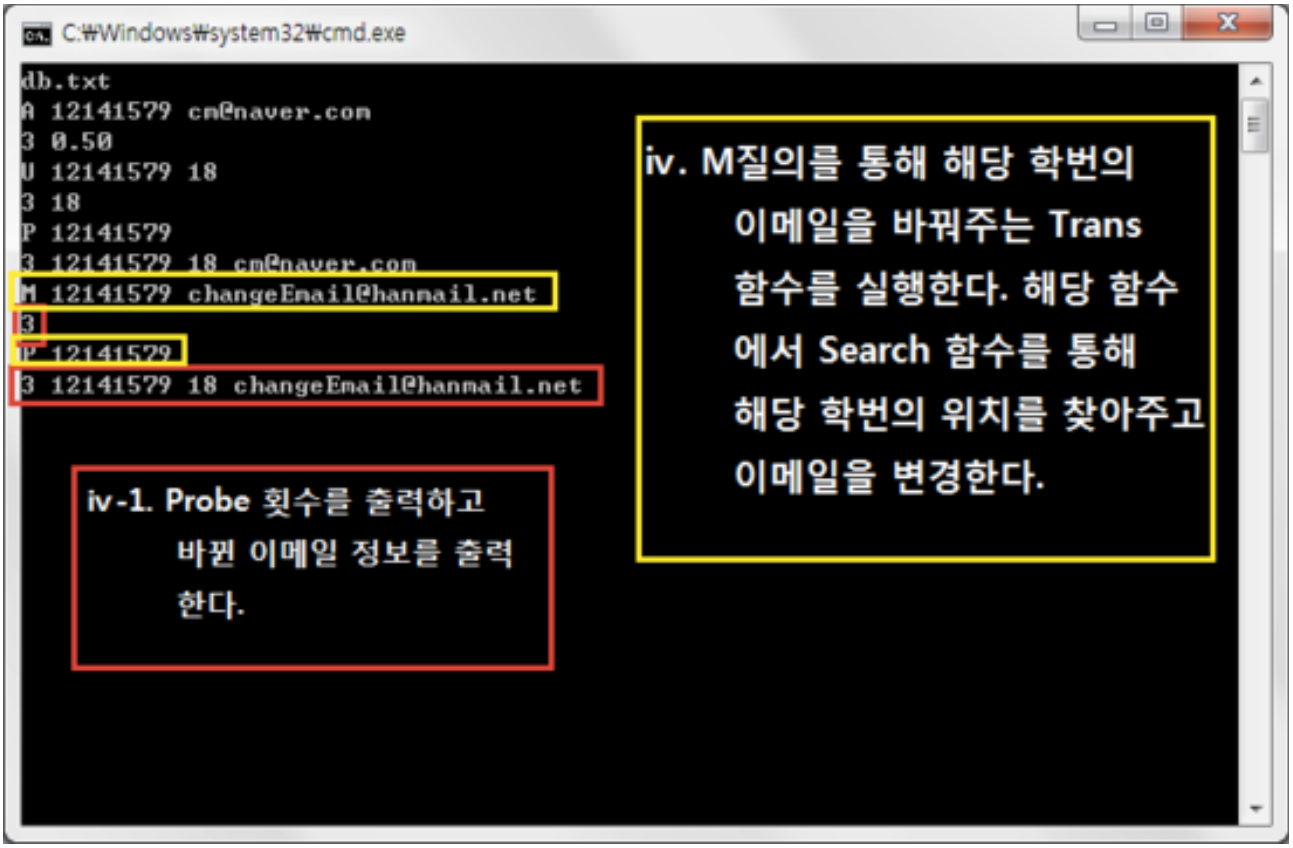
2) 학생 추가 질의



3) 수강 과목 추가

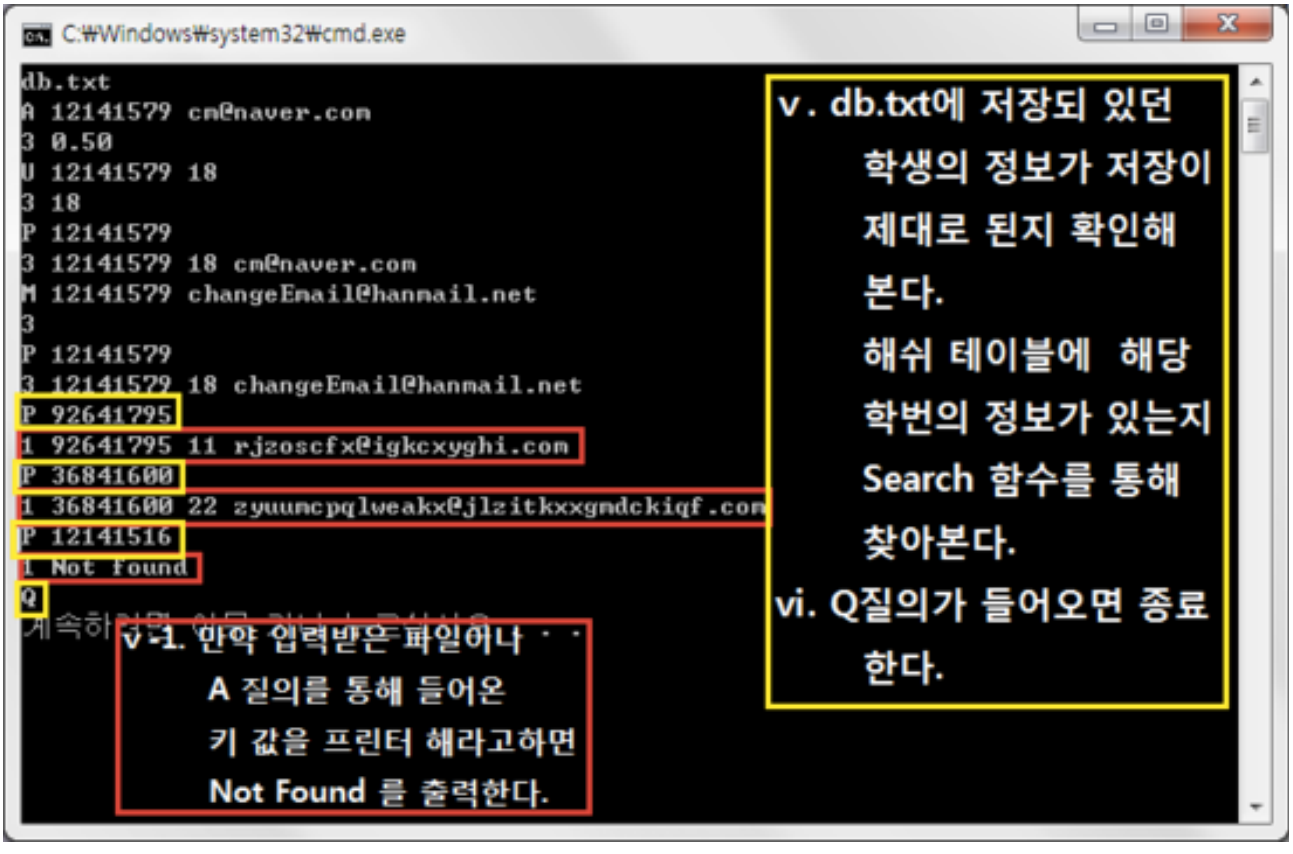


4) 이메일 주소 변경



6) 파일 입 출력을 통해 들어온 데이터 확인

7) 종료



V.평가 및 개선 방향

– 본 결과의 장점 및 단점

장점

-> 모든 질의를 만족한다.

Worst Case가 아닌 경우 빠른 시간에 기능이 수행 될 수 있다.

Search 함수를 만들었기에 같은 일을 하는 함수에서 사용 할 수 있기에 편리하다.

단점

-> Worst Case의 경우 O(n)의 시간복잡도를 가진다.

전역 변수를 많이 사용했기 때문에 해당 과제처럼 간단한 코드는 문제가 없지만 복잡한 코드에서 전역변수는 문제를 만들 수 있다.

파일 입 출력에서의 데이터를 main에서 받아서 main이 길다.

–향후 개선 방향

-> 전역변수로 선언된 변수를 지역변수를 사용하여 문제의 가능성을 줄인다.

-> main에서 받은 파일 입 출력 데이터를 함수를 통해 main을 간단하게 한다.