과제 6: Progressive Overflow Hashing 구현

1. 개요

Texbook 11장의 progressive overflow hashing 기법을 다음과 같이 구현한다.

(1) 학생 레코드 파일은 주어진 student.h를 이용하여 만든 (fixed length record + fixed lengh field) 방식으로 저장되며, 레코드의 크기는 120바이트이다 (필요하면 과제 5의 코드를 수정하여 활용하기 바람). 헤더는 존재하지 않으며, 학번 필드의 크기는 10바이트이다.

(2) Hashing에서 사용하는 키는 학생 레코드의 '학번'이다.

(3) Hash function은 Chapter 11(1)의 3페이지에 나와 있는 예시와 비슷하게 동작하도록 정의해서 사용한다. 예시에서는 hash value(주소값)을 구하기 위해 킷값의 첫 두 개의 문자에 대한 ascii code값을 사용하였지만, 과제 6에서는 중간값을 구하기 위해 맨마지막 두 문자를 이용한다. 또한 예시는 hash table의 크기 N = 1,000을 가정한 것이며, 이번 과제에서는 N을 프로그램을 실행시킬 때 인자로 받아서 사용한다. 따라서 예시의 home address를 이에 맞게 계산하여야 한다 (N에 대한 modulo 연산처리).

(4) Hash file(hash table)의 크기는 위 (3)의 N과 동일하며, 각 레코드는 학번 킷값과 이것을 갖는 학생 레코드 번호의 필드를 갖는다. 킷값이 존재하지 않는 레코드는 NULL값으로 채운다. Hash file의 맨앞에 4 바이트의 헤더를 두며, 여기에는 hash table의 크기 N을 4바이트 할당하여 저장한다. 그 이유는 search()나 (필요하면) delete()를 처리하기 위해서는 hash table의 크기가 필요하기 때문이다.

(4) Hash table의 수정은 킷값의 삭제만을 통해 이루어지며, 킷값의 수정은 킷값의 첫 번째 바이트에 *를 저장하는 것을 의미한다.

2. 프로그램 기능

프로그램은 아래와 같이 세 개의 기능을 갖는다. 여기서 학생 레코드 파일명은 student.dat, 해시 파일명은 student.hsh로 고정해서 사용한다 (student.h 참조).

(1) Hash table(hash file)의 생성

같은 디렉토리에 존재하는 학생 레코드 파일에서 각각의 학생 레코드를 읽어서 학번 킷값을 추출한 후 이것을 hash function을 이용하여 hash file에 저장한다.

```
a.out -c hash_table_size
```

hash_table_size 크기의 hash file을 생성하며, 학생 레코드 파일 student.dat로부터 학번 킷값을 추출하여 progressive overflow 방식으로 저장한다. 그 결과로서 student.hsh 파일이 생성된다.

(2) Hash table을 이용한 키 검색

위의 (1)에서 만든 student.hsh를 이용하여 학번 킷값을 검색한다. 그 결과는 주어진 학번 킷값이 존재하는 hash file에서의 주소(레코드 번호)와 search length이다 (student.c

참조).

a.out -s “sid”

(3) Hash table에서 킷값 삭제
위의 (1)에서 만든 student.hsh에서 주어진 학번 킷값을 삭제한다.

a.out -d “sid”

3. 개발 환경

- OS : Linux 우분투 버전 16.04
- 컴파일러 : gcc 5.04
- 반드시 Linux 우분투와 gcc 환경을 준수해야 하며, 이를 따르지 않아서 발생하는 불이익은 학생이 책임져야 함!!

4. 제출물

- 작성한 student.c를 하위폴더 없이(최상위 위치에) zip파일로 압축하여 myclass.ssu.ac.kr 과제 게시판에 제출 (소스파일, 헤더파일, zip파일은 반드시 소문자로 작성)
- 압축한 파일은 반드시 학번_6.zip (예시 20061084_6.zip)과 같이 작성하며, 여기서 6은 세 번째 과제임을 의미함
- 채점 프로그램상 오류가 날 수 있으니 꼭 위 사항을 준수하기 바람!