|  |
| --- |
|  |
| 과제5: 레코드 삭제 및 추가 |
| 1. 개요 |
| 주어진 student.h에 정의되어 있는 학생 레코드를 삭제(delete)하고 추가(insert)할 수 있는 프로그램을 작성한다. 과제 4에서 작성한 프로그램을 기반으로 한다.   * 레코드 삭제(delete) * 사용자가 삭제 명령어를 통해 레코드를 삭제하려고 할 때 학생 레코드의 primary key인 ‘학번’ 키만을 사용한다. (0) * ‘학번’ 키값을 갖는 학생 레코드를 검색한 후 해당 레코드 자리에 “\*Page\_numRecord\_num…”와 같이 내용을 수정한다. 여기서 ‘\*’는 delete mark이며, Page\_num은 바로 직전에 삭제된 레코드가 존재하는 페이지 번호이고 Record\_num는 바로 직전에 삭제된 레코드가 존재하는 페이지(Page\_num)에서 그 레코드의 번호를 의미한다. 즉, Page\_num과 Record\_num은 바로 직전에 삭제된 레코드의 주소를 나타낸다. 여기서 각 페이지에서의 레코드 번호는 0, 1, 2, 가 된다. (0) * 만약 바로 직전에 삭제된 레코드가 존재하지 않으면 Page\_num과 Record\_num는 모두 -1의 값을 가진다. (0) * 레코드가 삭제되더라도 해당 페이지의 header에 저장되어 있는 레코드 수(#records)에는 변함이 없다. 예를 들어, 레코드가 삭제되기 전 #records=8이었다면, 레코드가 삭제되어도 여전히 #records=8을 유지한다. 이것은 header의 offset list에 변화를 주지 않고 그대로 사용하기 위함이다. 또한 freespace의 값에도 변함이 없다. * Page\_num과 Record\_num을 저장할 때 각각 2B를 할당한다. * 삭제 레코드 관리   + 삭제 레코드는 교재의 linked\_list 개념을 사용하며, 삭제 레코드 리스트는 스택(stack) 기법을 사용한다.   + 스택 기법을 적용하므로 새로운 삭제 레코드는 삭제 레코드 리스트의 맨앞에 저장한다. (0)   + 위에서 설명한 Page\_num과 Record\_num는 삭제 레코드 리스트에서 ‘링크’를 의미한다.   + 삭제 레코드 리스트의 헤더(head)는 맨마지막으로 삭제된 레코드를 가리키므로 이 레코드의 Page\_num과 Record\_num를 저장한다. 이를 위해 레코드 파일의 헤더 영역의 reserved space의 맨처음 2B씩을 할당하여 사용한다. (0)   + 삭제 레코드가 존재하지 않을 때는 당연히 레코드 파일의 헤더 영역의 Page\_num과 Record\_num의 값은 각각 -1을 저장한다. (0) * 레코드 추가(insert) * 과제 4에서 구현한 insert() 함수를 업그레이드한다. 즉, 레코드를 추가할 때 삭제 레코드 리스트에서 ‘right size’를 만족하는 삭제 레코드를 찾아서 여기에 새로운 레코드를 저장한다. right size를 만족하는 레코드를 선택할 때 교재의 first-fit 전략을 사용한다. * right size를 만족하는 삭제 레코드가 없으면 레코드 파일의 맨 마지막 페이지에 새로운 레코드를 append한다. * 당연히 삭제 레코드 리스트에 변화가 있으면 이를 레코드 파일에 반영되어야 한다. * 프로그램 기능  1. 레코드 삭제: 사용자로부터 ‘학번’ 키값를 입력을 받고 이를 만족하는 학생의 레코드를 찾은 후 학생 레코드 파일에서 이를 삭제한다. 2. 레코드 추가: 과제 4의 insert() 함수를 새롭게 수정한다.   \*\*\* 과제 5에서는 ‘student.c’ 파일의 다음 함수를 수정하거나 추가하면 된다.   * + int searchByID(FILE \*fp, char \*keyval, char \*recordbuf, int \*pagenum, int \*recordnum);   + void delete(FILE \*fp, char \*keyval);   + void insert(FILE \*fp, const STUDENT \*s); |
| 2. 레코드 삭제 |
| 사용자로부터 레코드 파일 이름과 “ID=필드값”을 입력받는다. 출력은 student.c에 정의되어 있는 포맷을 따른다.   |  | | --- | | $ a.out –d record\_file\_name “ID=field\_value” |   실행파일과 같은 디렉토리에 있는 'students.dat' 파일에 대해서, ‘ID=20091234'를 만족하는 레코드를 삭제하고 그 결과를 출력하면 다음과 같다.   |  | | --- | | $ a.out –d students.dat “ID=20091234”  #Records = 1  20091234#Gildong Hong#Computer#3#Dongjak-gu, Seoul#820-0924#gdhong@ssu.ac.kr | |
| **3**. 레코드 추가  과제 4와 동일 |

4. 개발 환경

- OS: Linux 우분투 버전 22.04 LTS (Ubuntu 홈페이지에서 버전 확인 가능)

- 컴파일러: gcc 13.2

\* 과제 채점 환경은 위와 동일하며, 따라서 프로그램 개발 환경도 위의 환경에 맞추길 권장하며 이를 따르지 않아서 발생하는 불이익은 본인이 책임져야 함

5. 제출물

- 프로그래밍한 소스파일 student.c student.h를 하위폴더 없이(최상위 위치에) zip파일로 압축하여 스마트캠퍼스 과제 게시판에 제출한다 (모든 제출 파일들의 파일명은 반드시 소문자로 작성).

- 압축한 파일은 반드시 학번\_5.zip (예시 20201084\_5.zip)과 같이 작성하며, 여기서 5는 다섯 번째 과제임을 의미함

\* 채점은 채점 프로그램을 통해 자동으로 처리하기 때문에 위의 사항들을 준수하지 않는 경우 채점 점수가 0이 될 수도 있기 때문에 반드시 준수하기 바라며, 이를 따르지 않아서 발생하는 불이익은 본인이 책임져야 함