과제 4: 레코드 저장 및 삭제

1. 개요

Chap6의 레코드 삭제 관리 및 재사용에 관한 프로그램을 작성한다. 아래의 조건을 반드시 준수하여야 한다.

- 파일 I/O 연산은 system call 또는 C 라이브러리만을 사용한다.
- 제공되는 person.h와 person.c를 이용하여 아래의 (2) (3)의 기능을 완성한다.

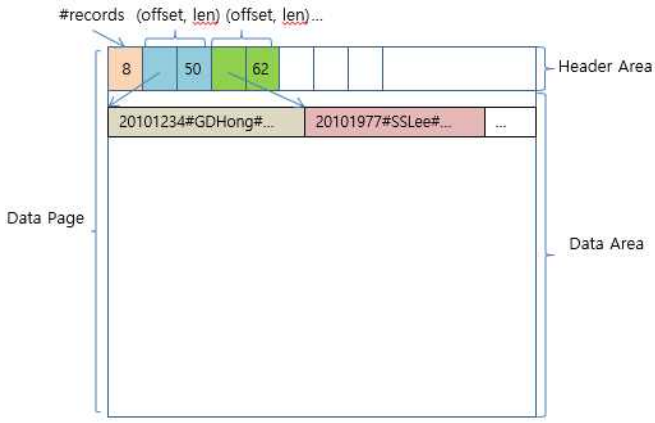
(1) 'Person' 레코드 파일의 구조

■ 레코드 파일
- 레코드 파일은 크게 헤더 영역(header area)과 데이터 영역(data area)으로 나뉜다.
- 헤더 영역에는 헤더 레코드가 존재하고, 데이터 영역에는 여러 개의 데이터 페이지가 존재한다.

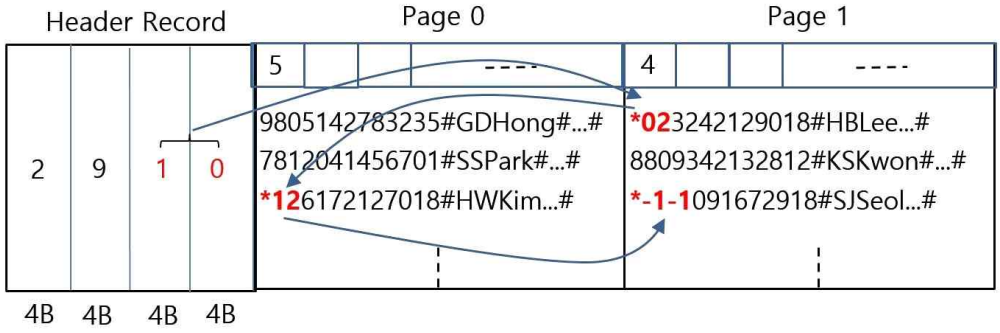
■ 헤더 레코드
- 헤더 레코드는 세 종류의 정보를 저장하며, 첫 4바이트에는 전체 데이터 페이지 수(헤더 레코드 불포함)를, 그 다음 4바이트에는 레코드 파일에 존재하는 모든 레코드(삭제 레코드 포함)의 수를, 그 다음 8바이트에는 페이지 번호와 레코드 번호를 각각 4바이트씩 할당하여 저장한다. 페이지 번호와 레코드 번호는 삭제 레코드 리스트를 관리할 때 사용하며, 항상 가장 최근에 삭제된 레코드를 가리킨다. 페이지 번호와 레코드 번호의 초기값은 각각 -1이며, 삭제 레코드가 더 이상 존재하지 않을 경우에도 각각 -1의 값을 가진다. 헤더 레코드의 데이터는 항상 최신의 정보로 유지해야 한다.
- 헤더 레코드는 레코드 파일을 생성할 때 공간을 할당 받으며, 동시에 메타데이터 값을 초기화한다.

■ 데이터 페이지
- 데이터 페이지는 header area와 data area로 구분되며, header area에는 페이지에 저장되어 있는 레코드의 수(#records), 각 레코드의 (offset, length)가 저장되고, data area에는 레코드가 저장된다.
- Header area의 #records를 위해 4바이트, offset을 위해 4바이트, length를 위해 4바이트를 할당하며, 첫 번째 offset의 값은 항상 '0'이다.
- 데이터 페이지 번호는 0, 1, 2, ...을 부여한다.

■ 레코드
- 레코드는"variable-length record" 방식으로 저장하며, 레코드 안의 필드는 "delimiter" 방식으로 구분된다.
- 새로운 레코드는 파일의 마지막 페이지의 마지막 레코드 다음 위치에 저장되며, 이때 공간이 부족하면 파일에 새로운 데이터 페이지 하나를 할당한 후 이 페이지에 저장한다.
- Delimiter로 지정된 특수문자는 '#'을 사용한다.



- 아래의 레코드 파일은 헤더 레코드와 두 개의 데이터 페이지로 구성되어 있으며, 전체 레코드(삭제 레코드 포함)는 9 개, 삭제 레코드는 3 개를 갖고 있다. 삭제 레코드는 linked list로 관리되며, 가장 최근에 삭제된 레코드는 (1, 0)이 가리키는 곳에 위치하며 화살표의 역방향 순서대로 삭제가 이루어졌음을 알 수 있다.



(2) 레코드 삽입(add)

- 터미널에서 person 정보를 입력받고 이를 레코드 파일에 저장한다. 레코드 파일에 삭제 레코드가 존재하면 삭제 레코드 공간에 우선적으로 저장해야 한다. 그렇지 않으면 맨 마지막 데이터 페이지에 append 형식으로 저장한다.
- 레코드 파일에 삭제 레코드가 존재하면 가장 최근에 삭제된 레코드부터 “right size”를 검사하여 이를 만족하는 첫 번째 삭제 레코드 공간에 새로운 레코드를 저장한다 (first fit 전략). 만약 right size를 만족하는 삭제 레코드가 존재하지 않는 경우 맨 마지막 데이터 페이지에 append 형식으로 저장한다. 한 가지 고려할 것은, right size를 검사할 때 삭제된 레코드의 길이만을 고려하면 된다. 예를 들어, 어떤 페이지의 마지막 레코드가 삭제된 상태에서 이 레코드에 대해 right size를 검사할 때 이 레코드의 길이뿐만 아니라 이 페이지에서 쓰지 않고 남아 있는 공간까지 계산에 넣을 필요는 없다.
- 레코드를 append 형식으로 저장할 때 데이터 페이지에 공간이 부족하면 새로운 데이터 페이지를 할당받아 그 레코드를 저장한다 (두 페이지에 걸쳐서 레코드를 저장하지 않는다).
- 선택된 삭제 레코드 공간에 새로운 레코드를 저장할 때 앞쪽부터 채워나가며, 남는 공간이 존재하면 이 공간은 재사용하지 않고 그대로 둔다 (즉, internal fragmentation).

따라서 새로운 레코드의 크기(length)는 선택된 삭제 레코드의 크기와 동일하다.
- 당연히 필요한 경우, 헤더 레코드의 정보와 삭제 레코드 리스트 등을 수정한다. 참고로, 헤더 레코드에서 전체 레코드의 수 (#records)는 정상적인 레코드와 삭제 레코드의 수의 합이다.
- Person 정보는 6 개의 필드로 구성되며, 사용자는 주민번호, 이름, 나이, 주소, 전화번호, 이메일주소 순서대로 필드값을 입력한다. 필드값은 큰따옴표로 묶어서 입력하며, 필드값은 영문자, 숫자, 특수문자(-, @, 스페이스 등)만으로 구성된다고 가정한다.

a.out a <record file name><field values list>

<예시>
a.out a person.dat "8811032129018" "GD Hong" "23" "Seoul" "02-820-0924" "gdhong@ssu.ac.kr"

옵션으로 a를 사용하며 입력받은 필드값들을 packing한 후, 즉 8811032129018#GD Hong#23#Seoul#02-820-0924#gdhong@ssu.ac.kr# 형태를 만들어서 이것을 person.dat 레코드 파일에 저장한다. 명령의 수행 후 출력은 없다.

** 주의: 나이를 포함한 모든 필드값은 문자열로 레코드에 저장한다. 하지만 헤더 레코드, 데이터 페이지의 header 영역과 삭제 레코드에서의 모든 메타 데이터는 이진 정수(binary integer)로 저장한다.

(3) 레코드 삭제(delete)
- 터미널에서 person 정보 중 키인 "주민번호"의 값을 입력받고 레코드 파일에서 이것과 일치하는 레코드가 존재하면 해당 레코드를 삭제한다.
- 삭제 레코드는 Chap6에서 배운 방식대로 관리한다. 삭제 레코드의 첫 번째 바이트는 delete mark *를, 그 다음 4바이트에는 페이지 번호를, 그 다음 4바이트에는 레코드 번호를 저장한다.
- 페이지 방식으로 레코드를 저장하기 때문에 약간의 수정이 필요하다. Chap6에서는 liked list에서 'link'에 해당하는 값으로 바로 직전에 삭제된 레코드 번호를 사용하였으나 이 과제에서는 레코드 번호 대신에 (페이지 번호, 레코드 번호)의 조합을 사용하며, 여기서 페이지 번호는 바로 직전에 삭제된 레코드가 존재하는 페이지를, 레코드 번호는 이 페이지에서의 레코드 번호를 의미한다. 각 데이터 페이지에서의 레코드 번호는 0, 1, 2, ... 순서의 값을 가지며, 모든 데이터 페이지에서의 첫 번째 레코드 번호는 0이다.
- 직전에 삭제된 레코드가 존재하지 않는 경우 페이지 번호와 레코드 번호 모두 -1의 값을 갖는다.
- 페이지에서 레코드가 삭제되더라도 그 페이지의 header 영역에 저장되어 있는 삭제 레코드의 (offset, len)의 정보는 수정할 필요가 없으며, 또한 header 영역의 #records의 값도 변경할 필요가 없다.
- 키 검색 등에서 필요한 삭제 레코드 판별 여부는 delete mark '*'를 사용해서 해결한다.
- 당연히 필요한 경우, 헤더 레코드의 정보와 삭제 레코드 리스트 등을 수정한다.

a.out d <record file name> <field value>

<예제>
a.out d person.dat “8811032129018”

옵션은 d를 사용하며, person.dat 레코드 파일에서 주민번호 8811032129018과 일치하는 레코드를 찾아서 삭제한다. 명령의 수행 결과에 대한 출력은 없다.

2. 개발 환경

- OS: Linux 우분투 버전 18.0.4
- 컴파일러: gcc 7.5

** 반드시 이 환경을 준수해야 하며, 이를 따르지 않아서 발생하는 불이익은 본인이 책임져야 함

3. 제출물

- 프로그래밍한 소스파일 person.c를 하위폴더 없이(최상위 위치에) zip파일로 압축하여 myclass.ssu.ac.kr 과제 게시판에 제출한다 (모든 제출 파일들의 파일명은 반드시 소문자로 작성하며, 제공된 person.h는 제출할 필요가 없음).
- 압축한 파일은 반드시 학번_4.zip (예시 20061084_4.zip)과 같이 작성하며, 여기서 4는 네 번째 과제임을 의미함

** 채점 프로그램상 오류가 날 수 있으니 꼭 위 사항을 준수하기 바라며, 이를 따르지 않아서 발행하는 불이익은 본인이 책임져야 함