Data Structure Lab. Project #1

2022년 9월 16일

Due date : 2022년 10월 13일 (목) 23:59:58

본 프로젝트에서는 이진 탐색 트리와 연결리스트, 그리고 큐 자료구조를 이용하여 간단한 사진 파일 편집 프로그램을 구현한다. 이 프로그램은 특정 경로에 저장된 사진 파일에 대한 정보를 링크드 리스트로 저장한 후, 검색에 용이하게 트리 자료 구조로 저장한다.

이진트리 탐색 알고리즘을 통해 사용자의 입력에 적합한 이미지 파일을 찾아내고, 해당 파일을 읽어온다. 이후 큐를 활용하여 각각의 이미지 파일을 읽어오고, 사용자에 입력에 적합하게 이미지를 변형하는 기능을 추가하여 이미지 변형 프로그램을 구현한다. (*본 프로젝트에서 사용하는 이미지 파일의 크기[Width, Height]는 고정적이며, 동일하다)

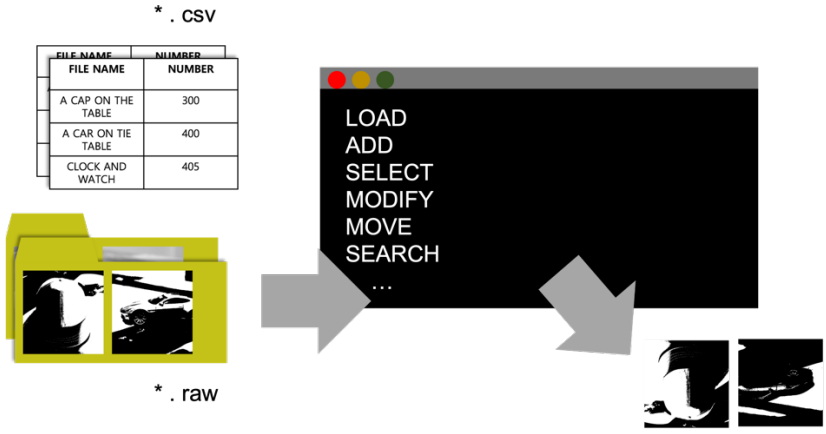

그림 1. 프로그램의 구조

☐ Program implementation

프로그램의 LOAD 명령어를 통해 프로그램이 존재하는 디렉토리 안에 'img_files'라는 디렉토리의 이미지 목록을 링크드 리스트 구조로 저장한다. 디렉토리에 존재하는 이미지 목록은 'filesnumbers.csv' 파일에 저장되어 있으며, 파일의 이름과 고유번호에 대한 정보를 담고 있다.

LOAD 명령어를 통해 생성된 리스트는 'Loaded_LIST'라고 정의하며, 리스트의 각 노드는 파일의 이름, 디렉토리 명, 파일의 고유번호에 대한 정보를 갖는다. Loaded_LIST는 노드가 입력되는 순서대로 링크드 리스트를 구성한다. 고유번호는 파일의 이름에 따라 할당되는 파일 고유의 숫자로 중복되지 않는다. 링크드 리스트로 저장된 Loaded_LIST 구조는 아래와 같다.

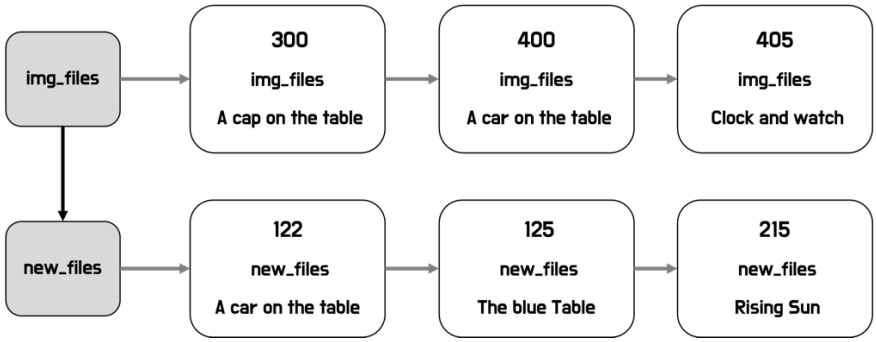

그림 2. 'Loaded_LIST'의 예

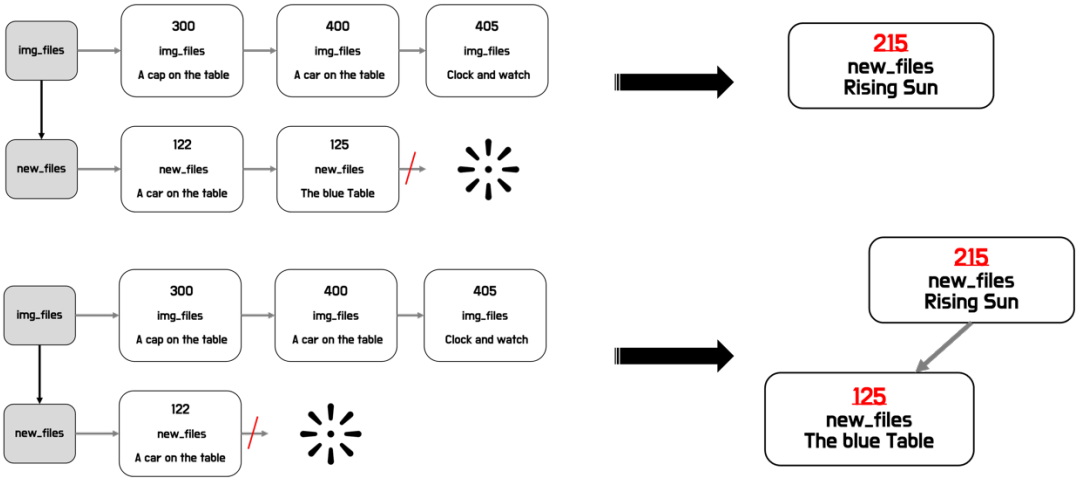

그림 3. 'Loaded_LIST'의 BST로의 변환

Loaded_LIST는 ADD 명령어를 통해 새로운 디렉토리에 대한 정보를 추가할 수 있다. ADD 명령어를 통해 탐색된 디렉토리는 새로운 리스트를 생성하여 Loaded_LIST에 2차원으로 연결한다.

MODIFY 명령어를 통해 Loaded_LIST에 존재하는 노드의 고유번호를 수정할 수 있다. 노드의 고유번호를 수정할 때는 기존 노드를 삭제하고 새로운 노드를 추가하는 방식으로 노드의 고유번호를 수정한다. 즉, 한번 생성된 노드 내부의 파일의 이름, 디렉토리, 고유번호는 절대 수정하지 않고 노드 자체를 삭제하고 새로이 연결하는 식으로 동작한다.

Loaded_LIST의 파일 노드는 링크드 리스트 형식으로 연결되어 있어야 하며, 각각의 파일 선행 노드(그림에서 'img_files'와 'new_files')의 경우 일반적인 리스트 형식으로 연결되도록 한다. Loaded_LIST에 존재하는 디렉토리에 대해 LOAD 혹은 ADD가 실행되면 기존에 Loaded_LIST에 존재하는 디렉토리에 대한 리스트를 삭제하고 새로 리스트를 추가하는 방법으로 Loaded_LIST를 업데이트 한다.

MOVE 명령어가 입력되면 Loaded_LIST에 존재하는 노드들의 데이터를 트리구조 노드의 데이터로 입력한다. 이때 생성된 트리구조의 자료구조를 'Database_BST'라고 한다. 즉, Loaded_LIST의 각각의 노드를 트리에 다시 저장하게 되며, 이때 트리에 저장된 노드는 Loaded_LIST에서 삭제(delete)된다.

트리에 노드를 저장할 때에는 Loaded_LIST의 마지막 노드(tail)부터 차례대로 입력한다. Loaded_LIST는 Database_BST에 들어가기 전 간단한 수정(노드의 고유번호 수정)을 위한 일종의 버퍼 개념의 역할을 하며 따라서 최대 100개의 데이터만 존재하도록 한다. 100개 데이터가 넘을 경우 먼저 들어왔던 데이터에 대한 노드를 제거 후 새로운 데이터를 추가한다.

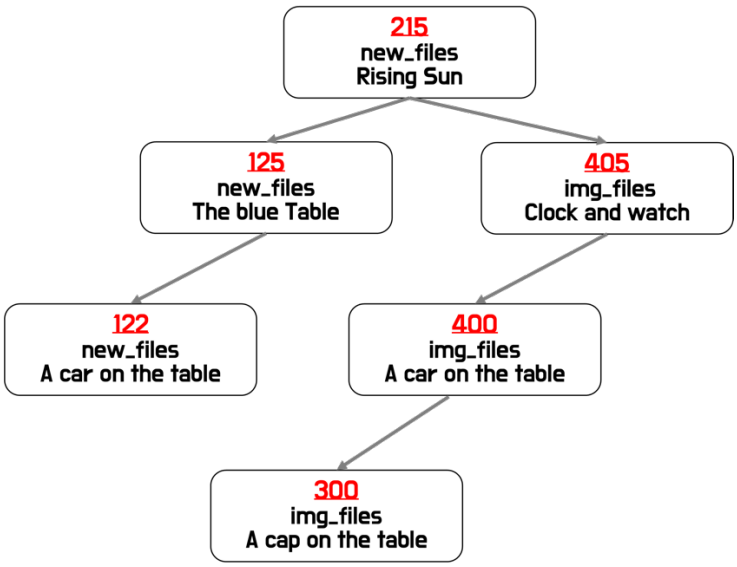

그림 4. 'Database_BST' 예

Database_BST에는 링크드 리스트와 동일하게 파일의 이름, 디렉토리 명, 고유번호로 기록된다. Database_BST를 구성하는 각각의 노드들은 아래 연결 규칙에 따라 연결된다.

1. 부모 노드보다 **파일 고유번호**가 작은 노드는 왼쪽, 큰 노드는 오른쪽 서브 트리에 위치하도록 한다.
2. 노드를 제거할 때, 양쪽 자식 노드가 모두 존재할 경우에는 왼쪽 자식 노드 중 가장 큰 노드를 제거되는 노드 위치로 이동시킨다.

Database_BST는 이러한 연결 규칙을 통해 보다 빠른 검색을 가능하도록 한다. 즉, 파일 고유번호 기반의 검색 속도를 강화한다. Database_BST의 최대 크기는 300개 노드이며, 이 이상의 노드가 배치되었을 때는 **트리내 고유 번호가 낮은 노드부터** 순차적으로 삭제하도록 한다.

이렇게 트리 구조에 저장된 노드에 대해서 사용자가 편집할 파일을 지정하게 하는 SELECT 명령어를 구현한다. SELECT 명령어는 파일의 고유 번호를 바탕으로 파일을 전위 순회 알고리즘을 통해 탐색하여 찾아내고, 편집을 위해 선택하도록 한다.

파일의 검색을 위해 SEARCH 명령어 또한 구현한다. 파일의 이름이 3글자 이상의 문장으로 구현되어 있음으로, **보이어 무어 알고리즘**을 통해 Database_BST 트리에 존재하는 파일 이름에 대해 탐색한다. 탐색 방법은 아래와 같다.

1. 파일의 이름기반 검색을 위해 트리를 Iterative post-order(반복문을 이용한 post-order, 재귀함수 사용금지) 방식으로 순회하며 큐에 트리 노드 안에 존재하는 파일이름과 고유번호에 대한 정보를 담는다.
2. 모든 노드에 대한 정보가 저장되었다면 순차적으로 POP을 진행하며 사용자가 SEARCH 명령어에 검색한 단어가 파일 명에 있는지 검색한다.
3. 단어가 존재하는 경우 해당 노드의 파일명 전체와 파일 고유번호를 출력한다.
4. 검색이 완료된 이후의 큐는 반드시 비어 있어야 한다.

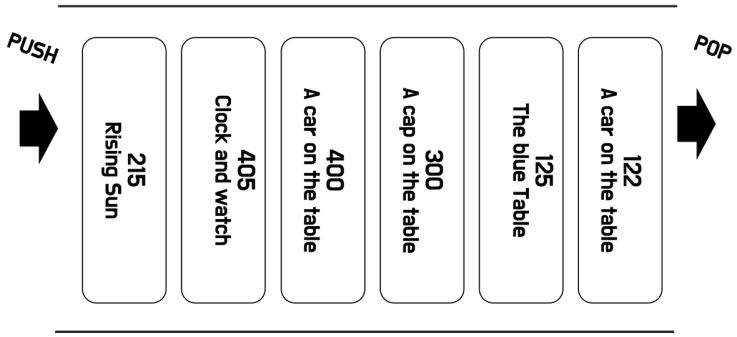

그림 5. SEARCH명령어의 구현을 위한 큐의 예

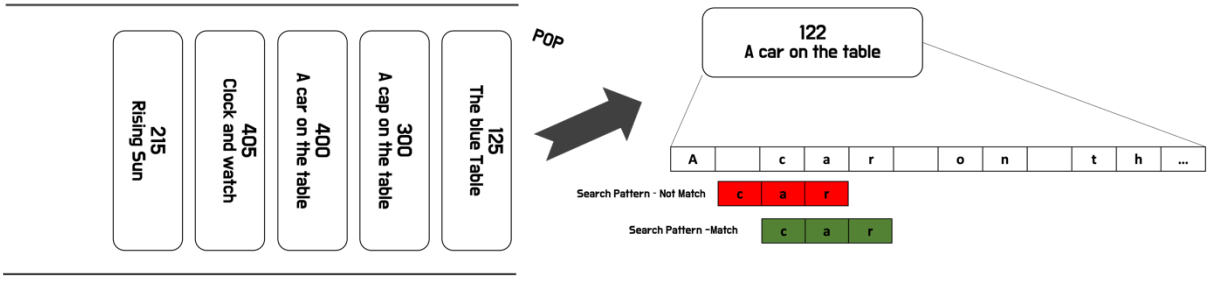

그림 6. 보이어 무어 알고리즘을 활용한 검색

보이어 무어 알고리즘은 특정 문자열에서 원하는 패턴을 찾는 알고리즘으로 문장과 같은 character를 가진 문장 패턴만을 비교해가며 본래 문장에서 찾고자 하는 패턴을 찾아낸다. 해당 방식을 통한 탐색은 탐색 속도를 개선할 수 있다.

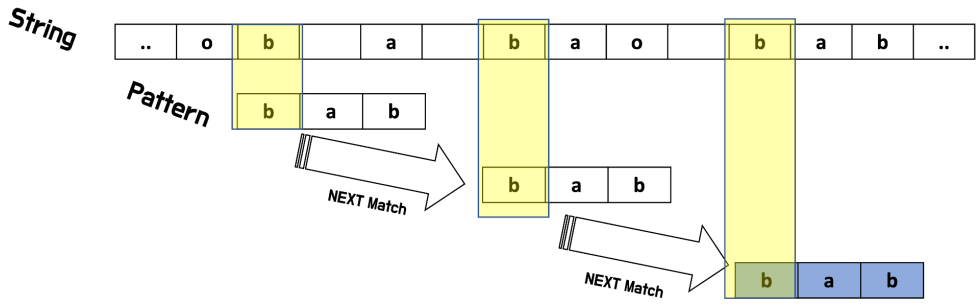

그림 7. 보이어 무어 알고리즘의 예

최종적으로 SELECT 명령어를 통해 선택된 파일에 대해서 주어진 코드를 이용해 이미지 파일을 읽어와서 아래와 같은 이미지 편집 프로그램을 구현한다.

- 이미지의 점대칭
  - 각 이미지 픽셀을 알맞게 Stack 구조에 순서대로 입력하고(Push) 후입선출(POP)하여 이미지를 점대칭으로 뒤집는다. (단, 각 픽셀에 대한 정보 할당은 반드시 처음 PUSH한 픽셀부터 다시 할당한다)

- 이미지의 밝기 조정
  - 각 이미지 픽셀을 알맞게 Queue에 순서대로 입력하고(Push) 선입선출(POP) 하며 각 픽셀의 값에 특정 숫자를 더해 이미지를 밝게 만든다. (단, 이미 밝기가 최대인 픽셀에 대해서는 최대 값(255)을 할당한다. 특정 숫자만큼 더하지 못할 경우 더할 수 있는 만큼 최대한 더한다.)

- 이미지의 크기 조정
  - 입력된 이미지의 크기를 4분의 1로 축소하는 구현을 진행한다. 4개의 셀의 평균값을 구해 하나의 셀에 입력하는 방식으로 구현한다. (제공되는 이미지의 [Width, Height]는 짝수이다.)

그림 8. 구현된 기능들의 최종 출력 예

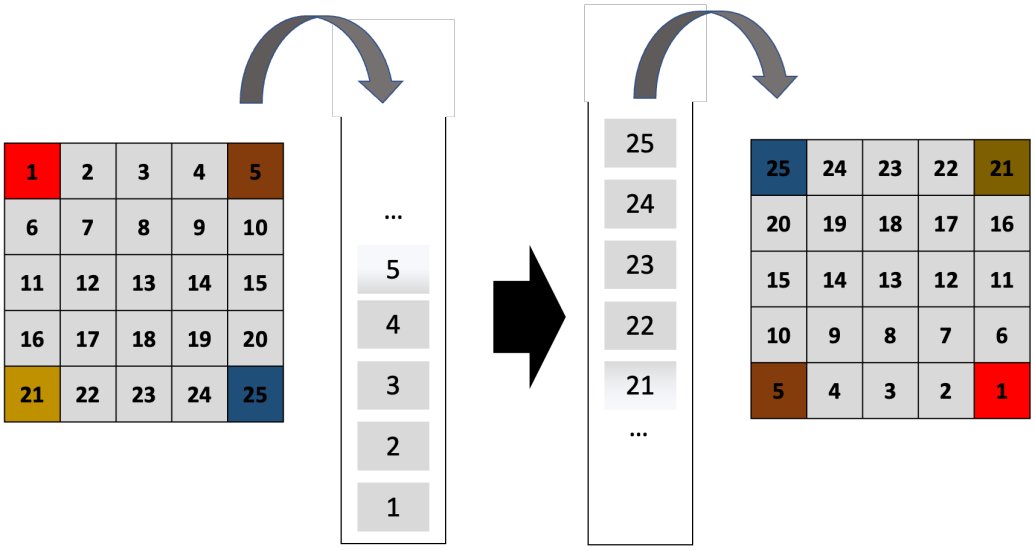

그림 9. Stack 구조를 이용한 이미지 점대칭의 예

작업된 결과는 반드시 이미지 파일로 지정한 디렉토리(Result 디렉토리)에 주어진 코드를 이용하여 저장되어야 하며, EXIT 명령어를 통해 프로그램을 종료할 수 있도록 한다. 프로그램 종료 이후 모든 메모리는 free 되어야 한다.

☐ Functional Requirements

명령어	명령어 사용 예 및 기능
LOAD	사용 예) LOAD CSV 파일의 데이터 정보를 불러오는 명령어로, CSV 파일에 데이터 정보가 존재할 경우 CSV 파일을 읽어 링크드 리스트 자료구조에 모두 저장하고, 링크드 리스트에 저장된 순서대로 파일 이름과 고유 번호를 출력한다. 노드 개수가 100개가 넘을 경우 먼저 들어왔던 노드를 삭제 후 새로운 데이터를 추가한다. 만약 CSV 파일이 존재하지 않으면 에러 코드를 출력한다. 사용되는 텍스트 파일의 이름은 아래와 같으며 파일 이름을 수정하지 않는다. CSV 파일: img_files/filesnumbers.csv 출력 포맷 예시) =======LOAD======== A CUP ON THE TABLE/300 ... ==================== ========ERROR======== 100 ====================
ADD	사용 예) ADD new_files new_filesnumbers.csv 새로운 디렉토리를 탐색하여 데이터를 추가하는 명령어로 Loaded_LIST에 새로운 디렉토리의 파일 데이터들을 추가한다. 새로운 디렉토리에서 추가되는 노드에 대한 연결은 2차원 링크드 리스트로 구성한다. 노드 개수가 100개가 넘을 경우 먼저 들어왔던 노드를 삭제 후 새로운 데이터를 추가한다. 첫 번째 인자부터 디렉토리 이름, CSV 파일 이름을 나타내며 인자가 하나라도 존재하지 않거나 Loaded_LIST가 존재하지 않는 경우 에러 코드를 출력한다. 출력 포맷 예시) =======ADD======== SUCCESS ==================== ========ERROR======== 200 ====================

MODIFY	사용 예) MODIFY img_files “A CUP ON THE TABLE” 303 Loaded_LIST에 존재하는 노드의 파일 고유 번호를 수정하기위한 명령어로, Loaded_LIST에 존재하는 파일 이름에 대한 노드를 삭제하고 새로운 노드를 추가하는 방식으로 고유 번호를 수정한다. 파일 고유번호는 모든 데이터셋에 대해 중복되지 않는다. 인자가 하나라도 없거나 파일 이름에 대한 노드가 존재하지 않거나 중복되는 고유 번호로 할당할 경우 에러코드를 출력한다. 출력 포맷 예시) =======MODIFY======== SUCCESS ==================== ========ERROR======== 300 ====================
MOVE	사용 예) MOVE Loaded_LIST의 정보들을 Database_BST로 옮기는 명령어이다. 파일 고유 번호와 파일 이름을 이용하여 BST의 노드를 생성하며 다음과 같은 기준으로 BST를 연결한다. BST의 최대 노드 개수는 300개이며 300개를 초과할 경우 트리 내 고유번호가 낮은 순서부터 차례로 제거한다. Loaded_LIST에 노드가 존재하지 않을 경우 에러코드를 출력한다. BST 연결 규칙 1. 부모 노드보다 **파일 고유번호**가 작은 노드는 왼쪽, 큰 노드는 오른쪽 서브 트리에 위치하도록 한다. 2. 노드를 제거할 때, 양쪽 자식 노드가 모두 존재할 경우에는 왼쪽 자식 노드 중 가장 큰 노드를 제거되는 노드 위치로 이동시킨다. 출력 포맷 예시) =======MOVE======== SUCCESS ==================== ========ERROR======== 400 ====================
PRINT	사용 예) PRINT

	Database_BST에 저장되어 있는 정보를 출력하는 명령어이다. 트리 중위 순회(In-order) 방식의 탐색을 이용하여 정보를 출력한다. Database_BST가 존재하지 않을 경우 에러를 출력한다. 출력 포맷 예시) =======PRINT========== img_files / "A CUP ON THE TABLE" / 303 … ========== ========ERROR======== 500 ==========
SEARCH	사용 예) SEARCH "CUP" 특정 단어가 포함된 파일의 이름을 탐색하여 출력하기 위한 명령어로, 인자로 하나 이상의 단어가 포함된다. 보이어 무어 알고리즘을 이용하여 다음 규칙에 따라 파일을 탐색하여 출력한다. 인자가 없거나 Database_BST가 비어있는 경우 에러를 출력한다. 1. 파일의 이름기반 검색을 위해 트리를 Iterative post-order(반복문을 이용한 post-order, 재귀함수 사용금지) 방식으로 순회하며 큐에 트리 노드 안에 존재하는 파일이름과 고유번호에 대한 정보를 담는다. 2. 모든 노드에 대한 정보가 저장되었다면 순차적으로 POP을 진행하며 사용자가 SEARCH 명령어에 검색한 단어가 파일 명에 있는지 검색한다. 3. 단어가 존재하는 경우 해당 노드의 파일명 전체와 파일 고유번호를 출력한다. 4. 검색이 완료된 이후의 큐는 반드시 비어 있어야 한다. 출력 포맷 예시) =======SEARCH========== "A CUP ON THE TABLE" / 303 "A BLUE CUP" / 305 … ========== ========ERROR======== 600 ==========

<table>
<tr><td>SELECT</td><td>사용 예) SELECT 303

EDIT 명령어를 위해 Database_BST에서 파일 고유 번호를 기반으로 전위 순회(pre-order)를 통해 이미지의 경로를 찾아서 이미지를 불러오는 명령어로, 인자가 없거나 파일 고유번호가 존재하지 않을 경우 에러를 출력한다.

출력 포맷 예시)
==========SELECT==========
SUCCESS
==========

========ERROR========
700
==========</td></tr>
<tr><td>EDIT</td><td>사용 예) EDIT -f
사용 예) EDIT -l 22
사용 예) EDIT -r

SELECT 명령어를 통해 가져온 이미지를 편집하는 명령어로 주어진 코드를 이용하여 이미지 파일을 읽어와서 수정하고 저장한다. 인자로 세 가지의 옵션 -f, -l, 그리고 -r이 존재한다. 아래의 조건에 따라 -f 옵션은 점대칭 동작을 수행하고, -l 옵션은 밝기 조정 동작을 수행하며, -r 옵션은 크기 조정 동작을 수행한다. 이때 밝기 조정 동작을 수행하는 -l 옵션은 조정하는 밝기에 대한 정보가 추가 인자로 필요하다 편집된 이미지는 프로그램이 존재하는 디렉토리의 “Result” 디렉토리에 저장한다. 인자가 존재하지 않는 경우 에러를 출력한다.

이미지 편집 동작
1. 이미지의 점대칭
A. 각 이미지 픽셀을 Stack 구조에 순서대로 입력하고(Push) 후입선출(POP)하여 이미지를 점대칭으로 뒤집는다. (단 각 픽셀에 대한 정보 할당은 반드시 처음 PUSH한 픽셀부터 다시 할당한다)
B. 이미지 파일 저장 (“파일이름_flipped”)
2. 이미지의 밝기 조정
A. 각 이미지 픽셀을 알맞게 Queue에 순서대로 입력하고(Push) 선입선출(POP) 하며 각 픽셀의 값에 특정 숫자를 더해 이미지를 밝게 만든다. (단, 이미 밝기가 최대인 픽셀에 대해서는 최대 값(255)을 할당한다. 특정 숫자만큼 더하지 못할 경우 더할 수 있는 만큼 최대한 더한다.)
B. 이미지 파일 저장 (“파일이름_adjusted”)</td></tr>
</table>

	3. 이미지의 크기 조정 A. 입력된 이미지의 크기를 4분의 1로 축소하는 동작을 진행한다. 인접한 4개 셀의 평균값을 구해 하나의 셀에 입력하는 방식으로 구현한다. B. 이미지 파일 저장 ("파일이름_resized") 출력 포맷 예시) ==========EDIT========== SUCCESS ========== ========ERROR======== 800 ==========
EXIT	사용 예) EXIT 프로그램 상의 모든 메모리를 해제하며, 프로그램을 종료한다. 출력 포맷 예시) =======EXIT======== SUCCESS ==========

☐ Requirements in implementation

- 모든 명령어는 반드시 대문자로 입력한다.
- 명령어에 인자(Parameter)가 모자라거나 필요 이상으로 입력 받을 경우 에러 코드를 출력한다.
- 예외처리에 반드시 에러코드를 출력한다.
- 출력은 "출력 포맷"을 반드시 따른다.
- 읽어야할 파일이 존재하지 않을 경우 해당 파일을 생성한 뒤 진행하도록 한다.

□ 명령어별 에러 코드

명령어	에러 코드
LOAD	100
ADD	200
MODIFY	300
MOVE	400
PRINT	500
SEARCH	600
SELECT	700
EDIT	800
잘못된 명령어	777

□ 채점 기준

● 코드

채점 기준	점수
LOAD 명령어가 정상 동작 하는가?	1
ADD 명령어가 정상 동작 하는가?	1
MODIFY 명령어가 정상 동작 하는가?	1
MOVE 명령어가 정상 동작 하는가?	1
PRINT 명령어가 정상 동작 하는가?	1
SEARCH 명령어가 정상 동작 하는가?	2
SELECT 명령어가 정상 동작 하는가?	1
EDIT 명령어가 정상 동작 하는가?	2
총합	10

* 채점 기준 이외에도 조건 미달시 감점 (linux 컴파일 에러, 파일 입출력 X, 주석미흡)

● **보고서**

채점 기준	점수
Introduction을 잘 작성하였는가?	1
Flowchart을 잘 작성하였는가?	2
Algorithm을 잘 작성하였는가?	3
Result Screen을 잘 작성하였는가?	3
Consideration을 잘 작성하였는가?	1
총합	10

- 최종 점수는 (코드 점수 × 보고서 점수) 로 계산됩니다.

□ 구현 시 반드시 정의해야하는 Class

- Loaded_LIST
- Loaded_LIST_Node
- Database_BST
- Database_BST_Node
- Manager : 다른 클래스들의 동작을 관리하여 프로그램 전체적으로 조정하는 역할 수행

□ 제한사항 및 구현 시 유의 사항

- 클래스의 함수 및 변수는 자유롭게 추가 구현이 가능하다.
- 제시된 Class를 각 기능에 알맞게 모두 사용한다.
- 프로그램 구조에 대한 디자인이 최대한 간결하도록 고려하여 설계한다.
- 채점시 코드를 수정해야하는 일이 없도록한다.
- 주석은 반드시 영어로 작성한다. (한글로 작성하거나 없으면 감점)
- 프로그램은 반드시 리눅스 (Ubuntu 18.04)에서 동작해야한다. (컴파일 에러 발생시 감점, 빌드 실패시 0점)
- **인터넷에서 공유된 코드나 다른 학생이 작성한 코드를 절대 카피하지 않도록 하**

며 적발시 전체 프로젝트 0점 처리됨을 명시

● 제공되는 Makefile을 사용하여 테스트하도록 한다.

□ 제출기한 및 제출 방법

● 제출기한

- 2022년 10월 13일 목요일 23:59:58 까지 제출

● 제출방법

- 소스코드(Makefile과 csv파일 제외) 와 보고서파일(학번_이름 DS_project1.pdf)을 함께 압축하여 제출
- 확장자가 .cpp, .h, .pdf가 아닌 파일은 제출하지 않음 (.csv파일 제출 X, 제출 시 감점)
- 보고서 파일 확장자가 pdf가 아닐 시 **감점**
- KLAS -> 과제 제출 -> tar.gz로 과제 제출

● 제출 형식

- 학번_DS_project1.tar.gz (ex. 2021202001_DS_project1.tar.gz)

*** 제출 형식 미준수시 감점 (보고서 및 압축파일 학번, 이름, 프로젝트 이름 반드시 준수)**

● 보고서 작성 형식 및 제출 방법

Introduction과 Consideration외의 모든 각 항목은 제목 외 최소 2장 이상 작성하여 제출하고 고찰은 반드시 1페이지 이상 작성할 것. 고찰에는 본 프로젝트 설계에 있어 힘들었던 점이나 성능 및 코드 구조를 개선하기 위해 작업한 내용, 프로젝트 및 코드관리를 위해 작업한 내용, 프로젝트를 설계하며 새로이 알게 되거나 참고한 내용이 될 수 있음.

- Introduction : 프로젝트 내용에 대한 설명
- Flowchart : 설계한 프로젝트의 플로우차트를 그리고 설명
- Algorithm : 프로젝트에서 사용한 알고리즘의 동작을 설명
- Result Screen : 모든 명령어에 대해 결과화면을 캡처하고 동작 설명
- Consideration : 고찰 작성

* 보고서 최소 분량은 10장이며 미달시 0점 처리될 수 있음

- 위의 각 항목을 모두 포함하여 작성하며 보고서에는 소스코드를 포함하지 않음