박정진

2015104027

2019.05.03

자료구조 프로젝트1 결과보고서

1. 목적

현재 우리 일상 속 컴퓨터의 운영체제 안에 구현 되어있는 폴더 관리 시스템을 비슷하게 구현하고 그 와 같은 구조로 후에 있을 음원 및 파일 관리도 할 수 있게 끔 하는 것을 계획한 프로그램을 구상한다. 폴더, 파일, 음원 등 이런 아이템들을 효율적으로 검색하고 다룰 수 있는 자료구조를 이용하여 나만의 스타일을 갖춘 폴더 관리 시스템을 만든다.

2. 기능

(macOs운영체제 이므로 macOs 기준에 맞춰서 구현한다)

폴더와 파일은 매우 유사한 구조 및 정보를 가지고 동작한다. 하지만 그 둘의 차이점이 몇가지가 있다.

1. 폴더만이 서브 아이템을 가진다.

2. 폴더 실행은 change directory이고 파일 실행은 확장자에 맞는 파일 어플리케이션을 실행하는 것이다.

기능 구현

1. Interface는 Command Line Interface로 구성

2. 폴더, 파일 생성 (중복 이름 자동 \_숫자 로 처리)

3. 폴더, 파일 삭제 및 이름 수정

4. 폴더, 파일 정보 관리 및 출력

5. 폴더 내 서브 폴더 및 파일 들 리스트 및 상세 정보 출력

6. Change Directory ( PathProcessor 함수를 통한 Path 접근 허용)

7. History를 이용한 앞으로가기, 뒤로가기

8. Recently 항목을 통한 최근 Access 한 폴더 및 파일 정리

9. 서브 폴더 및 파일 이름 순, 생성 날짜 순, Access time 순으로 내림차순, 오름차순 정렬

10. Cut, Copy, Duplicate, Paste 기능 구현

11. TextApplication, JPGApplication 기능을 통한 확장자에 따른 파일 실행

3. 기능에 대한 설명

1. 폴더 와 파일 은 아이템이고, 공통점이 많은 점을 이용해 공통 기능은 ItemType에 구현하고 FolderType과 FileType은 ItemType을 상속받아 고유의 속성 및 함수를 구현함.

2. 모든 동작이 현재 working directory 기준이다. Working directory 안에서 다양한 기능을 제공하는 BaseApplication에서 폴더를 관리(BaseApplication의 현재 폴더를 가리키는 FolderType\* 포인터로 관리) 하는 구조로 구현함

3. interface 는 Command Line 으로 구성, [Operation] [Option / Target] [Value] 형식으로 명령에 대한 처리

4. 폴더, 파일 의 정렬은 이름으로 정렬한 것을 기본으로 하고, 이후 sort 로 정렬 기준을 바꾼다. 정렬은 sort로 어떤 기준으로 정렬하라고 명령할 때만 수행되므로 기준이 바뀌면 이름으로 정렬된 리스트는 그 순간 정렬되지 않은 리스트가 되므로 O(nlogn)을 보장하는 merge sort로 구현한다.

5. Cut, paste, copy는 ItemType\* heap이라는 BaseApplication 의 멤버 변수로

지정을 하여 polymorphism으로 구현한다.(폴더, 파일 모두 사용해야 한다)

4. 사용한 자료 구조

1. Sorted Linked List

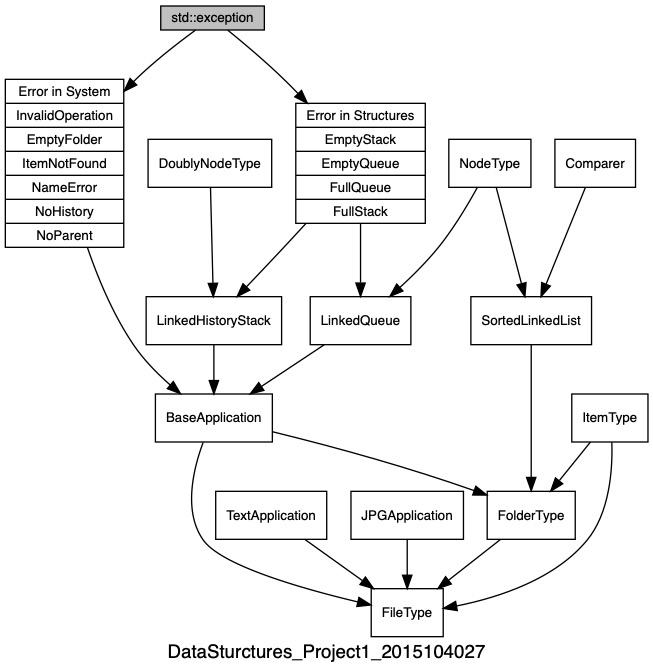
* 스택이나 큐는 자료의 접근이 제한
* 반면 리스트는 중간에 새로운 객체를 삽입하거나 삭제 할 수 있다. 즉 임의의 위치에 있는 항목에 대한 연산을 허용
* 유저가 얼마나 많은 정보(아이템)들을 생성할지 모르기 때문에 메모리를 효율적이고 또한 제약을 받지 않는 연결리스트가 적절한 구조라고 생각
* 단순 리스트는 크기를 키우려면 리스트 안에 있는 배열의 정보들을 모두 복사하고 새로운 배열에 넣고 삭제해야하는 큰 연산을 진행 정렬을 위해서는 비효율적인 많은 자료의 이동이 필요하므로 리스트 보다 linked list가 적합.

2. Linked History Stack, Linked Queue

* History는 가장 최근 것이 먼저 나와야 한다는 점에서 Stack 구조가 적합
* Recent Item과 Path Processor는 가장 처음에 들어간 것을 가장 먼저 접근 해야한다는 점에서 적합
* Stack과 Queue 둘다 Linked 형태로 구현했는데 이 또한 List에서의 이유 처럼 유저가 얼마나 많은 연산을 할지 모르기 때문에 메모리를 효율적으로 사용할 수 있는 Linked 스타일의 구조가 적합하다고 판단

이후 이중연결리스트 (Binary Search는 아니지만 중간에서 비교 먼저 하고 양방향 Search가 가능하므로 검색 속도 증진에 도움이 될 것) 나 AVL트리(Binary Search 가능) 구조가 더 효율적으로 판단이 된다면 사용, 해당 구조와 목적에 맞는 검색, 정렬 방법 중 시간복잡도와 구조적 특성을 고려하여 더욱 효율적인 알고리즘을 사용한다.

5. Class ADT



6. 한계 및 보완점

1. 파일 기록, 즉 원래 저장되어있는 정보 를 저장하고 불러오는 시스템 미구현

- 프로그램 종료되면 저장되어있던 기록 모두 삭제됨

- 데이터 파일에 기록하고 불러올 수 있게끔 구현(프로그램 실행되면 자동으로 데이터 불러옴)

- 현재 운영체제가 macOS이므로 C++ 라이브러리에 있는 디렉토리 처리 라이브러인 Windows.h로 는 실제 디렉토리와 연관지어서 구현을 하지 못함

2. help 미구현 (명령어에 대한 설명 미구현 – 유저가 사용할 때 사용법을 모를 수도 있다.)

3. TextApplication에 대한 기능 및 실현성 부족 JpgApplication에 대한 기능 및 실현성 부족

4. 리스트에 대한 구조 개선 필요 (Doubly Linked List나 트리 구조)

5. 검색의 효율성이 많이 떨어져 Recently, 와 자주 가는 즐겨찾기 기능에서 먼저 검색하게 끔 하여 검색의 효율성을 높인다.

6. 가장 많이 엑세스한 아이템 Top10 을 정렬한 리스트를 만들어야함(바로 엑세스 가능하게끔 한다. 또한 검색 최우선순위로 두어 검색한다, Pin기능)

7. Help 함수 구현

8. JPGApplication에 대한 다양한 기능 추가

9. Jpg를 관리하는 PhotoList 와 EventList 생성

10. 음악파일을 관리하는 Application 구현