

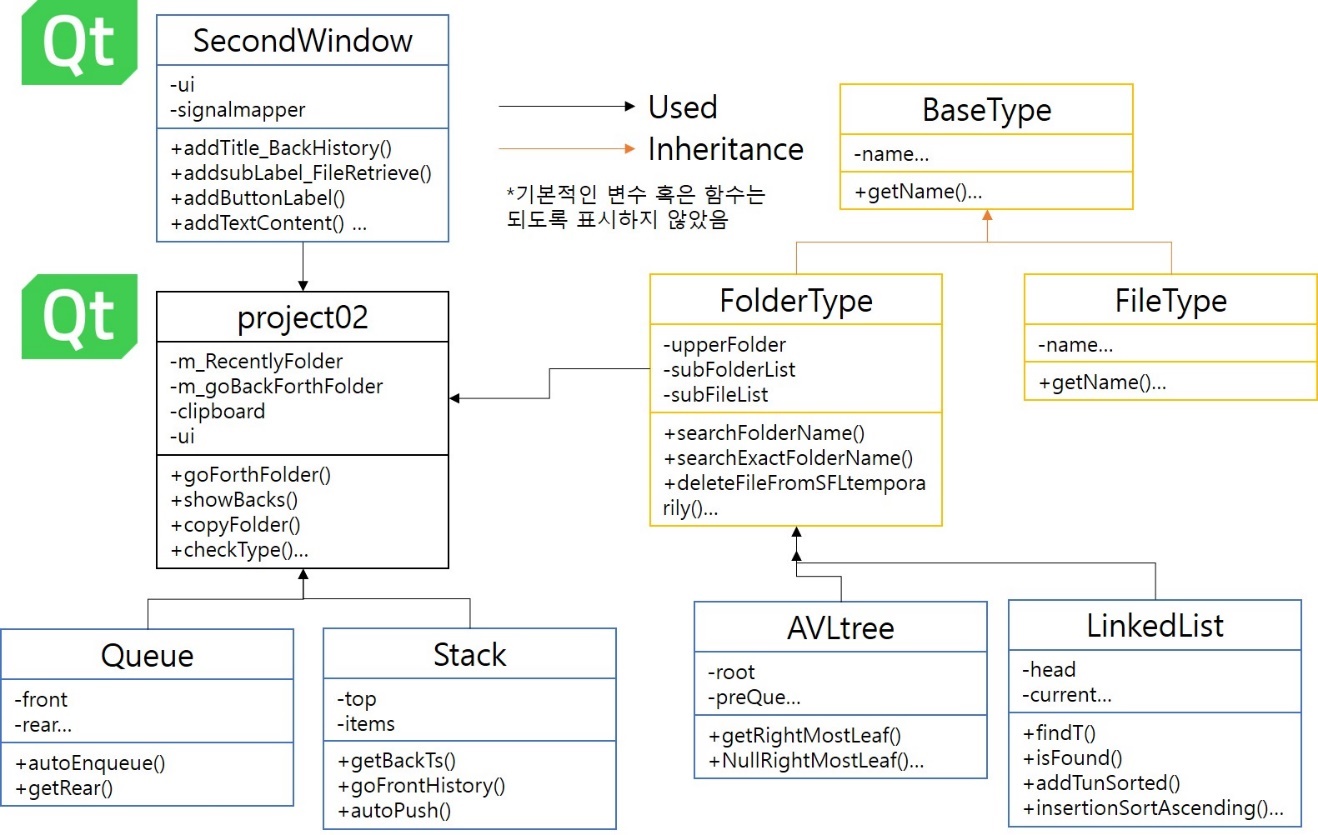
Folder & File Management Report

Data structure Project2

2013100988 | 전자공학과 | 최범순

# 목차

1. ADT(Abstract Data Type) & 자료 구조
2. 구현한 기능
3. 미구현 기능(제한점)
4. 어려웠던 점 & 해결방안
5. 느낀점
6. ADT(Abstract Data Type) & 자료 구조



전체적인 구조는 위와 같다. 기본적으로 project02 가 사용자와 가장 가까운 클래스이고, 따라서 결국 project02 는 모든 class의 헤더파일을 포함하고 있다. 사용한 자료 구조는 Circular Queue, Stack, AVLtree, Linkedlist와 Heterogeneous list로 총 다섯 개의 자료 구조를 사용하였으며, 모든 자료 구조는 멤버 함수와 멤버 변수를 사용하지 않고 template을 사용한 generic 클래스이다. GUI라는 클래스는 콘솔 내에서 사용자가 프로그램을 더 편리하게 사용할 수 있도록 구현한 클래스이다.

사용한 다섯 가지 자료 구조는 다음과 같은 때에 이용하였다.

1. AVLtree

① 사용 : ArrayList는 **하나의 폴더(FolderType)가 갖고 있는 subFolderList 와 subFileList, 즉 포함하는 하위 폴더와 하위 파일들을 관리하는 데에 사용**되었다.

② 사용한 이유 : 사용자가 어떤 폴더나 파일을 찾을 때, AVL tree로 폴더와 파일을 관리하면 Binary Search의 검색 속도로 빠른 검색을 실행할 수 있다. 폴더나 파일을 추가하거나 삭제할 때에는 단순히 Binary Search Tree에 노드를 추가하고, Big O의 상수 스케일에 해당하는 Balance만 해주면 되기 때문에 속도가 빠르다. 즉, 검색, 추가, 삭제 시 속도가 빠른 AVL tree의 장점을 이용하였다.

③ 특징 : AVL tree는 왼쪽 노드와 오른 쪽 노드를 가리키는 노드를 멤버 변수로 갖고 있으며, height와 Balance Factor를 통해 항상 Balance 를 유지한다.

1. Circular Queue

① 사용 : Circular Queue는 **최근 열어본 폴더**를 저장하는 데 사용되었다. 또, **복사한 폴더와 파일을 저장하는 공간인 Clipboard**에 사용되었다.

② 사용한 이유 : 사용자가 최근 열어본 폴더를 보고 싶다면 폴더들의 정보를 모두 저장하기 보다는 개수를 제한하고 필요한 만큼의 목록을 보여준다. 만약 Queue가 꽉 찼을 때에는 가장 오래된 폴더를 지운다.

③ 특징 : Circular Queue 안에 이미 있는 폴더를 열었을 때, 그 전에 Queue에 저장되어 있던 폴더의 정보가 Queue에서 지워지고 가장 최근 공간에 폴더가 저장된다. Queue가 꽉 찼을 때엔 Queue에 특성에 따라 오래된 폴더를 지우고 새로운 폴더를 넣는다.

1. Stack

① 사용 : Stack은 **앞, 뒤로 이동할 폴더를 저장**하는 데 사용되었다.

② 사용한 이유 : 뒤로 여러 번 이동하였다가 앞, 뒤 이동이 아닌 다른 선택으로 폴더를 열면, Stack의 가장 최근 것이 없어져야 하기 때문에 Stack을 사용하였다.

③ 특징 : Stack은 pure한 Stack이 아닌 Circular Queue의 특성이 합쳐진 modified Stack이라고 할 수 있다. 리스트가 꽉 찼을 때는 가장 오래된 것을 먼저 지우고 가장 최근의 것을 추가하며, 뒤로 이동할 때에는 가장 최근 것부터 없어진다.

1. LinkedList

① 사용 : LinkedList는 사용자가 **폴더나 파일을 검색했을 때, 그 검색 결과를 잠시 저장하고 보여주는 데**에 사용하였다. 또, **뒤로 가기, 앞으로 가기 목록을 보여줄 때**에 사용되었다.

② 사용한 이유 : 사용자가 검색할 때에는 Binaray Search가 빠르지만, 그 검색 결과에서 또 다시 검색하는 경우는 드물다. 그래서 이 때에는 LinkedList를 사용하여 검색 결과에 찾은 폴더나 파일을 추가하는 데 필요한 작업량을 줄이고자 LinkedList를 사용하였다.

③ 특징 : 검색 결과에서 사용한 Linked List는 그 폴더나 파일의 주소를 포함하고 있어서, 사용자가 작업을 끝낸 뒤에 이 List를 지워주어 메모리 낭비를 없앴다.

1. Heterogeneous List

① 사용 : Heterogeneous List는 LinkedList로 사용되었고, 사용자가 **폴더나 파일을 검색했을 때, 그 검색 결과를 잠시 저장하고 보여주는 데**에 사용하였다.

② 사용한 이유 : 폴더나 파일을 한 번에 검색하고, 그 결과를 저장하고 보여주며, 또 폴더면 폴더 열기, 파일이면 파일 열기 기능을 제공하기 위해 Heterogeneous List를 사용하였다. ClipBoard 역시 Heterogeneous List를 이용하여 폴더가 복사되거나 잘라내졌을 때도, 파일이 복사되거나 잘라내졌을 때도 ClipBoard에 추가된다.

③ 특징 : 검색 결과(Linked List)가 BaseType의 포인터를 저장하고 있다.

1. 구현한 기능 (핵심은 진하게, 추가 기능은 빨간색으로 표시하였음)
2. (파일 및 폴더) 생성, 삭제, 이름 바꾸기, 속성 보기

① 기본적으로 폴더 생성, 삭제, 이름 바꾸기, 속성 보기가 가능하며, 폴더 이름 바꾸기를 했을 때엔 하위 폴더는 보존된 채 원하는 특정 폴더의 이름 바꾸기가 가능하다.

② 파일 생성, 삭제, 이름 바꾸기, 속성 보기가 가능하며, 실제로 프로젝트 디렉토리에 해당 파일이 생성되고, 삭제되며, 이름 바꾸기가 가능하다. **파일이 삭제 되었을 때에는 해당 파일을 포함하고 있는 모든 상위 폴더들의 크기도 갱신된다.**

1. (파일 및 폴더) 복사, 잘라내기, 붙여넣기

① 폴더를 복사 했을 때, **재귀 함수로 해당 폴더의 Subfolder들 모두 복사가 된다**. 잘라내기 했을 때에는 해당 폴더의 포인터를 하위 폴더 리스트에서 삭제하고 ClipBoard에 추가하여 잘라내기를 구현하였다. 붙여넣기는 해당 폴더의 포인터가 가리키는 폴더를 붙여넣기한 폴더에 추가하였다. 이 기능은 Queue의 getRear()라는 함수로 구현하였다.

② 파일 복사도 복사 생성하여 Clipboard에 저장하고, 잘라내기는 포인터 자체를 추가하였다. 붙여넣기는 해당 파일의 포인터가 가리키는 파일을 붙여넣기한 폴더에 추가하였다. 이 기능은 Queue의 getRear()라는 함수로 구현하였다.

1. (파일 및 폴더) 검색 , 현재 폴더에서, 전체 시스템에서 선택하여 검색 가능

파일이나 폴더를 검색하면, **현재 폴더에서 모든 하위 폴더들까지 폴더와 파일을 검색하여 사용자에게 보여준다.** 검색 결과에는 Type, Location과 Name이 보여지며, 타입이 폴더면 Folder, 파일이면 File으로 보여진다. 검색 결과는 기본적으로 타입은 폴더에서 파일 순으로, 이름을 기준으로 오름차순으로 정렬되어있다.

1. 파일 읽기, 쓰기

텍스트 상자를 이용해서, 사용자에게 메모장과 유사한 파일 수정 기능을 제공하고,

1. 폴더 이동

폴더의 이동은 Application 클래스의 FolderType\* m\_curFolder의 변경으로 이루어진다.

① 상위 폴더, 하위 폴더로 이동할 수 있다.

상위 폴더는 폴더가 추가될 때 정해진다.

② 뒤로 가기, 앞으로 가기가 가능하다.

뒤로 가기, 앞으로 가기가 가능하고, 앞, 뒤 이동이 아닌 행위로 폴더가 이동되었을 때는 Stack에 이동된 폴더를 넣고 그 위의 모든 데이터는 NULL로 만든다. 이 기능은 autoPush()라는 함수로 구현하였다.

③ Stack의 내용을 임시로 가져와서, 뒤로 목록 보기, 앞으로 목록 보기가 가능하고, 선택하여 이동도 가능하다. Stack의 getBackTs()와 getFrontTs() 함수로 구현하였다.

1. 기록 지우기

기록 지우기를 하면 뒤로 가기, 앞으로 가기 목록과 최근 열어본 폴더 목록이 전부 초기화된다.

1. GUI (Graphic User Interface)

QT라는 프레임워크와 Visual studio를 이용하여 사용자가 이용하기 편리한 창인 GUI를 구성하였다.

프로그램을 실행하면 정확한 크기로 프로그램이 실행되며, 상단에 이름도 File and Folder Management로 나타난다.

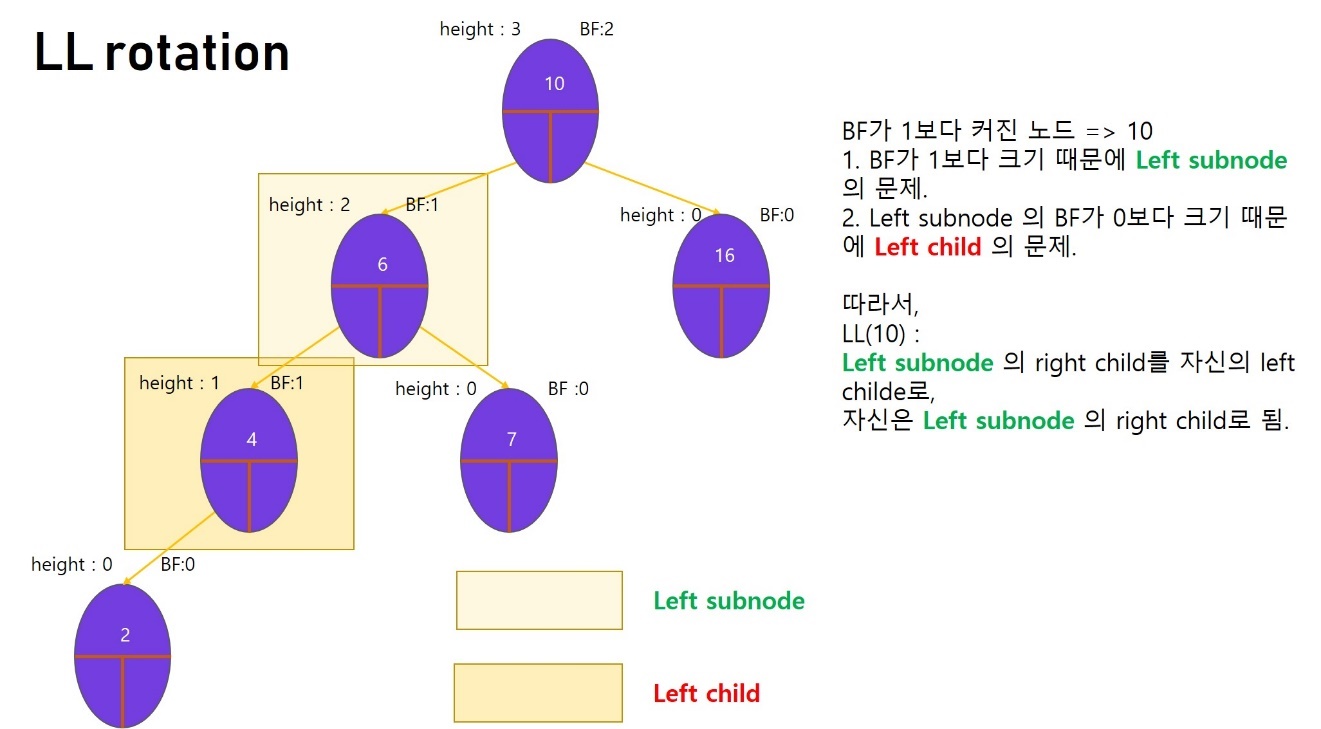
폴더나 파일을 관리하기 위해 명령창을 6개의 구역으로 나누었고, 메뉴를 제외한 나머지 구역은 계속해서 갱신되는 방식으로 구성하였다. 커서의 이동은 gotoWorkSpace(), gotoMessageSpace() 와 같은 함수들로 구성하였으며, 내용을 지우는 것은 clearWorkSpace()와 같은 함수들로 구성하였다. 폴더 생성, 삭제, 이동과 파일 생성, 삭제, 이동 시 Current folder Property space와 Current Path space가 갱신되며, 한 번의 명령이 끝나면 명령 창과 Work Space, Message Space도 갱신된다.

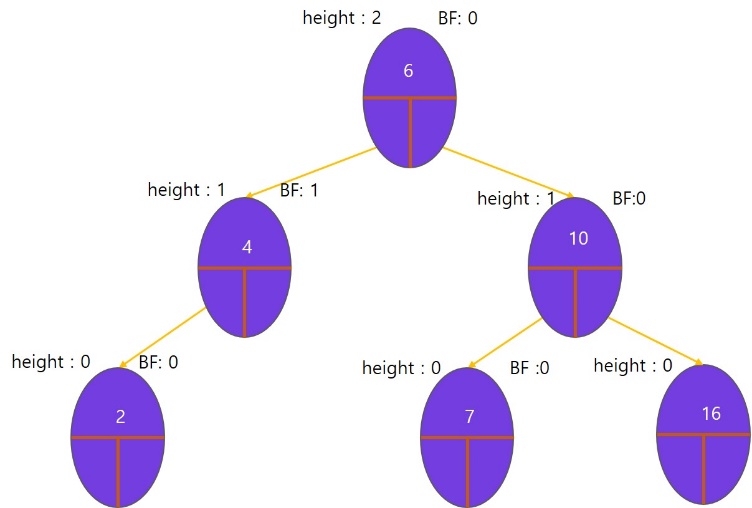
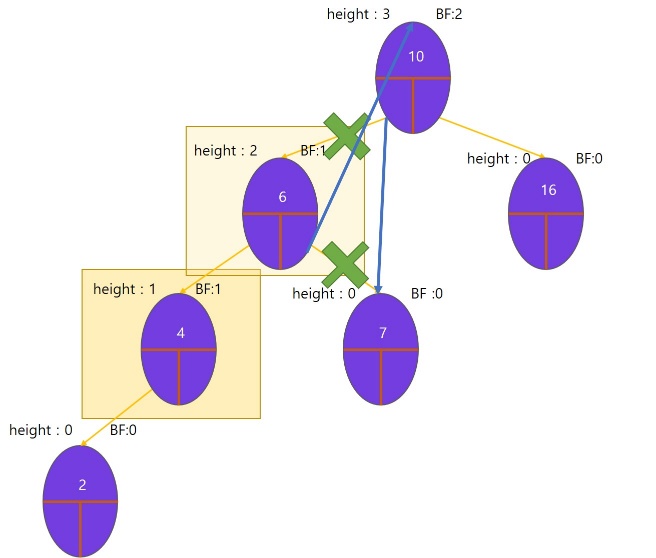
1. 미구현 기능(제한점)
2. 포함하는 폴더와 파일 수가, 해당 폴더의 하위 폴더의 개수와 하위 파일만의 개수만 갖고 있다. (포함하는 모든 하위 폴더가 아닌) 이에 대한 해결 방안으로 폴더와 파일을 추가 혹은 삭제할 때에, 모든 상위 폴더들도 갱신하는 방식으로 해결할 예정이다.
3. 파일이 같은 디렉토리(프로젝트 디렉토리)에 생성되기 때문에, 프로그램 내에서 다른 폴더에 같은 이름의 폴더를 생성하게 되면 예상치 못한 결과가 일어난다. 이에 대한 해결 방안으로, 프로젝트 디렉토리에 실제로 폴더와 파일을 만들어, 파일 이름이 겹치지 않게 할 예정이다.
4. 어려웠던 점 & 해결 방안
5. AVL tree에서 Rename 했을 때 Order가 무너지는 현상이 발생한다.

정렬된 tree에서 중간의 key값 만이 바뀌므로 Order가 Binary Search Tree의 Order를 만족하지 않는다. 그래서, 임시적으로 해당 노드의 연결을 끊고 Balance한 후, 노드를 AVL tree에 추가하고 다시 Balance하는 방법을 이용하여 재정렬 하였다.

1. AVL tree의 Rotation 이해와 구현

Rotation은 회전이라는 뜻만 바라보고, 강박적으로 Balance가 무너지면 회전을 해야한다는 생각을 가지고 있어 이해하기 힘들었다. 그러나 Rotation은 단지 노드 간의 연결을 바꾸는 한 과정이었음을 이해하고 구현하였다. 다음은 나름대로 이해한 LL Rotation의 예제를 기록하여 첨부한 내용이다.





1. GUI programming의 시도와, Visual studio c++로 GUI 구현

그저 QT 프로그램을 이용하여 프로그래밍할 수도 있었지만, Visual Studio와의 추후 연동 문제가 생길 것 같아 Visual Studio내에서 QT designer를 연동하여 코드를 작성하였다. 그러면서 QT에서 지원하는 기능을 전부 이용하지 못하고 우회하는 방법을 이용해야 해서 불편함이 있었지만, Google과 Youtube외 여러 페이지에서 정보의 도움을 받아 구현하였다.

1. 느낀점

매번 Visual studio의 c++로만 프로그램을 작성하다가 GUI 프로그램을 작성하려고 하니까 막막했다. 그러나 차근차근 찾아보면서 시각적인 완성도가 높아지는 걸 보면서 GUI 프로그래밍의 필요성을 느꼈다. 내가 사용자라면 우선적으로 더 보기 좋고 사용하기 편리한 프로그램을 사용하고 싶을 것이고, 그런 의미에서 이번 과제는 매우 의미 있었다고 생각한다. 비록 내가 구현하고 싶은 모든 기능에서 절반도 채 미치지 못하고 기본 기능에다 적은 기능만이 추가로 구현되었으며 중간 프로젝트보다 추가 기능이 적어졌지만, 시각적인 프로그램을 구성하는데 시간을 투자하여 더 편리한 프로그램이 되었고 할 수 있는 만큼 했으므로 후회하지 않는다.