최종프로젝트 보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20211522 이름: 김정환

**1. 프로젝트 목표, 실험 환경**

.................

프로젝트는 이미 실습에서 진행하였던 Maze Project에 이전에 받아온 미로를 인터넷 창에서의 뒤로가기 버튼처럼 불러올 수 있는 기능을 추가하는 것을 목표로 하였다. 또한 사용자 로그인을 구현하여 각각의 사용자별로 연 미로의 데이터를 각각 저장할 수 있는 시스템을 구현하고자 하였다.

실험 환경으로는 visual studio 2022에서 OpenFramework 프로젝트를 생성하여 구현하도록 하였고, 미로 data는 실습에서 주어진 것과 마찬가지로 .maz 파일을 이용하도록 하였다.

................

**2. 변수 설명**

.......................

linked\* head : 미로의 데이터를 저장하기 위한 linked list의 head

int maze\_col : 미로의 column을 저장하는 변수

int maze\_row : 미로의 row를 저장하는 변수

int isOpen : 파일이 열렸는지 저장하는 변수(0이면 안 열림, 1이면 열림)

bool isdfs : true면 DFS가 성공했음을 저장하는 변수

int isBFS : BFS가 성공했을 경우 1, 아닐 경우 0으로 저장하는 변수

temp\_node before[31][31] : BFS 탐색 시 이전 위치 저장하는 배열

int\* row\_stack : DFS 시에 row를 저장하는 stack

int\* col\_stack : DFS 시에 col을 저장하는 stack

int top : DFS 시 사용하는 stack에서 top의 index를 저장하는 변수

int\* row\_q : BFS 시 사용할 queue의 row 배열

int\* col\_q : BFS 시 사용할 queue의 col 배열

int front, rear : queue에서의 삽입과 삭제 시 사용하는 index

linked\_stack\* before\_top : 이전 미로를 넣는 stack 변수

user\_list\* user : user의 리스트의 head

int logined : user 로그인 여부 저장하는 변수

char cur\_user[21] : 현재 user의 이름 저장

.........................

**3. 함수 설명**

.......................

void push(int row, int col) : DFS 시에 stack에 노드를 push해주는 함수

void pop(int\* row, int\* col) : DFS 시에 stack에서 노드를 pop해주는 함수

void addq(int row, int col) : BFS 시에 queue에 노드를 넣는 함수

void delq(int\* row, int\* col) : BFS 시에 queue에서 노드를 빼는 함수

void insert(linked\* item) : 미로 데이터의 linked list에 노드를 추가하는 함수

linked\* search(int row, int col) : linked list 내에서 row와 col을 가진 노드를 찾는 함수

void before\_push(int row, int col) : maze 데이터의 head 및 row, col을 stack에 넣어주는 함수

linked\_stack\* before\_pop() : maze 데이터들을 stack에서 불러오는 함수

user\_list\* user\_search(char\* name) : user\_list에서 user의 노드를 찾는 함수

void setup() : 초기 설정 및 초기화를 하는 함수

void appMenuFunction(string title, bool bChecked) : 눌려진 버튼 별로의 수행을 실행하는 함수

void draw() : 화면에 그리는 모든 부분을 처리하는 함수

void freeMemory() : Open 클릭 시 head를 NULL로 초기화하고 나머지 배열들을 메모리 해제 및 필요 변수 초기화하는 함수

void freeMemory\_list() : 현재 미로의 list의 메모리 해제도 추가된 함수

void freeMemory\_all() : 현재 있는 미로의 stack의 메모리 해제도 추가된 함수

void freeMemory\_all\_data() : 할당한 모든 메모리의 해제를 담당하는 함수

bool readFile() : .maz 파일을 읽어와 미로를 저장하는 함수

bool DFS() : DFS 탐색을 실시하는 함수

bool BFS() : BFS 탐색을 실시하는 함수

void dfsdraw() : DFS의 결과를 그리는 함수

void bfsdraw() : BFS의 결과를 그리는 함수

.........................

**4. flow chart 및 자료구조와 알고리즘 설명**

.......................

<flow chart>

login 파트 flow chart

start

name

temp\_user = user\_search()

!temp\_user

make new temp\_user

logined

1

freeMemory\_all()

save stack to cur\_user’s node

freeMemory\_list()

각종 변수 초기화 및 stack이나 cur\_name 변경

end

logout 파트 flow chart

start

기존 stack을 cur\_user의 node에 넣어준다

freeMemory\_list()

필요한 변수들 초기화

end

Open 파트 flow chart

start

isOpen == 1

before\_push()

freeMemory()

readFile()

end

Before 파트 flow chart

start

before\_top

print error

freeMemory\_list()

temp = before\_pop()

head = temp->head

maze\_col = temp->col, maze\_row = temp->row

free(temp)

end

exit 파트 flow chart

start

freeMemory\_all\_data()

ofExit()

end

<자료구조>

실습에서 이미 진행한 DFS와 BFS는 각각 모두 array로 구현한 stack과 queue를 사용하였다. DFS의 경우에는 stack을 사용하므로 row\_stack과 col\_stack을 선언하여 해당 배열들은 top 변수의 값 변화에 따라 값 접근이 stack에 맞춰 동작한다. BFS의 경우에는 queue를 사용하므로 각각 row\_q와 col\_q 배열을 이용하고 front와 rear 변수의 값 조절을 통해서 queue 동작을 구현했다.

Before 버튼을 통해서 이전 미로를 불러오는 기능은 stack을 이용하여 구현하였다. 구조체 내에 row, col에 각각 maze\_row, maze\_col에 저장되어 있던 값들을 저장한다. head에 기존의 linked list가 저장되던 전역 변수 head에 있던 값을 넣고, 각각을 linked list로 만든 stack으로 구현하였다. Open으로 새로운 파일을 열 때 stack에 before\_push()로 넣어주고, Before로 이전 미로를 불러올 때 before\_pop()으로 미로를 불러온다. 이 경우에 각각 시간 복잡도와 공간 복잡도는 우선 각각의 push와 pop의 경우에는 시간 복잡도는 O(1)이고, 공간 복잡도는 미로의 최대 행과 열을 각각 row, col이라 하고, 미로의 개수를 n개라 할 때 O(n \* row \* col)이다. 그러나 pop의 경우에는 freeMemory\_list()가 함께 실행되고 이는 현재 미로의 메모리 해제 과정이 포함되어 현재 미로의 행, 열을 row, col이라 할 때 O(row\*col)의 시간 복잡도를 가지게 된다. push는 freeMemory()에서는 반복문이 없으므로 그대로 O(1)이다.

다음으로 로그인 및 user별 데이터 저장은 linked list를 이용하여 구현하였다. user의 name을 저장할 name 변수와 user별 저장된 stack을 저장할 user\_top, 그리고 linked list로 연결하기 위한 link가 구조체로 묶여 있는 구조이다. 이는 login 버튼을 눌렀을 때 입력받은 name을 바탕으로 우선 user\_search()로 탐색을 한 후에 존재 여부에 따라서 새로운 node를 추가하거나 기존 data를 불러온다. 또한 불러오면서 기존에 있던 이전 미로들을 저장하던 stack이나 다른 변수들에 대해서 free 처리를 하는 함수도 달라진다. 최악의 경우는 user\_search() 결과 끝까지 탐색해도 존재하지 않고, 이전에 로그인 하지 않아 메모리를 모두 해제하고 불러오는 경우가 최악의 수이다. 이 경우에 저장된 user의 개수를 n, 모든 미로의 행, 열이 row, col이라 하고, 현재 guest로 저장한 미로의 수를 m이라 하면, 시간 복잡도는 n보다는 row\*col\*m의 차수가 더 높으므로 O(row\*col \* m)이다. 공간 복잡도는 user당 k개의 미로를 저장하고 있다고 하면 O(n \* row\*col\*k)이다.

.........................

**5. 창의적 구현 항목**

.......................

이 프로젝트에서 추가한 부분으로는 사용자 로그인을 구현한 부분과 이전에 불러왔던 미로를 저장해 두었다가 이전 버튼으로 불러오는 기능이 있다. 사용자 로그인은 name을 입력받고, 이를 통해서 새로운 node를 만들거나 기존의 node를 불러오는 방식을 통해서 사용자별로 저장해둔 미로를 이용할 수 있도록 한다. 또한 밑에 현재 어떤 유저가 사용 중인지 표시하여 현 유저도 알 수 있도록 하였다. 그리고 View 버튼에 하위로 만든 Before 버튼으로 이전에 불러온 미로를 저장해뒀다가 불러올 수 있는 기능은 stack을 활용하였다. Open 버튼으로 새로운 파일을 열 때 before\_top 변수에 새롭게 link하여 미로를 저장하고 불러온다. 그 후 Before를 누를 때 pop하며 현재 띄우고 있는 미로는 freeMemory\_list()로 메모리를 해제하며 이전 미로를 불러온다. 해당 stack은 사용자에 따라 저장하고, guest인 경우, 즉 로그인 하지 않은 경우에는 사용자가 바뀔 때 저장하지 않는 방식으로 구현하였다.

.........................

**6. 프로젝트 실행 결과**

.......................

텍스트, 폰트, 스크린샷, 블랙이(가) 표시된 사진

자동 생성된 설명

-로그인 시 입력 받는 화면

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

-로그인으로 cur user가 kim으로 바뀐 것을 확인할 수 있다.

텍스트, 폰트, 스크린샷, 블랙이(가) 표시된 사진

자동 생성된 설명

-로그인 후에 5\_5.maz를 열고 12\_8.maz를 열고 난 후의 cmd 화면

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

-before를 눌렀을 때 다시 5\_5.maz의 미로를 보여주는 것을 확인 가능한 화면

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

-로그 아웃으로 cur user가 guest로 바뀐 것을 확인 가능한 화면

패턴, 스크린샷, 스티치이(가) 표시된 사진

자동 생성된 설명

-로그인과 로그 아웃 이후에도 BFS를 보여주는 기능이 정상작동함을 확인 가능한 화면

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

-exit 버튼으로 정상 종료됨을 볼 수 있다.

.........................

**7. 느낀 점 및 개선 사항**

.......................

이번 프로젝트를 진행하면서 Before 버튼은 인터넷 브라우저에서 이전 버튼이 stack의 원리로 작동한다는 점에 착안하여서 구현했다. 이를 통해서 기존에 배열을 통해서 구현하는 stack 외에도 linked list를 활용하여 구현하는 stack의 활용에 대해서 더 익숙해질 수 있었다. 또한 다양한 구조체들을 만들어보면서 구조체의 활용에 있어서도 내가 직접 구상하고 구현하는 과정을 통해서 구조체에 더 익숙해질 수 있었다. OpenFramework에서는 window 버튼을 추가하는 부분이나 string을 밑 화면에 출력하는 기능을 추가하는 과정을 거쳐보면서 어떤 기능이 있는지 알아볼 수 있는 부분도 있었다.

프로젝트를 진행하면서 아쉬웠던 점으로는 화면을 전환하여 현재 저장되어 있는 user의 리스트를 보여주는 부분이나, 플레이어가 움직이면서 미로를 탈출해볼 수 있는 기능에 대한 구현을 해보지 못한 부분에 대한 아쉬움이 있다. 또한 자료구조를 활용함에 있어서 linked list나 stack, queue와 같은 구조만 사용하여 구현함에 있어서 좀 더 다양한 자료구조를 이용해보는 것에 대해서 아쉬움이 있었다.

.........................