전산 수학

팀 프로젝트

3조

201110932 구건모

201411231 이지연

201411253 홍민지

201411254 황세연

|  |
| --- |
| 문제 정의 및 요구사항 |

서울 지하철 1호선부터 9호선까지의 지하철 노선 정보를 활용하여, 출발역에서 도착역에 이르는 최단 거리(통과하는 정거장의 수가 최소가 되는 거리)를 찾는 프로그램을 작성하시오. 사용자로부터 출발역과 도착역을 받은 다음, 최단 거리에 해당하는 역 정보를 출력한다. 최단 거리 경로상에 있는 모든 역을 나열하며, 환승이 발생하는 경우에는 환승에 관한 정보도 함께 표현한다. 다음의 기본 요구사항을 만족하도록 구현한다.

1. 프로그램은 사용자 입력 및 결과 출력에 있어서 이해하기 쉽도록 적절한 안내 문구가 포함되어야 한다.
2. 사용자의 입력이 잘못된 경우 또는 해당 역이 없을 경우에 다시 입력 받도록 한다.
3. 결과는 출발역에서 도착역에 이르는 모든 정거장을 순서대로 보기 좋게 나열하고, 총 통과하는 정거장의 수도 함께 알려준다.
4. 위 프로그램은 사용자가 실행을 종료할 때까지, 반복적으로 수행될 수 있어야 한다.

이 외의, 필요하다고 생각되는 추가 기능 및 사용 편의성을 위한 사용자 인터페이스는 자유롭게 구현한다

|  |
| --- |
| 시스템 환경 및 사용 언어 |

Visual studio 2008 ~ 2013 ( cpp )

|  |
| --- |
| 그래프를 이용한 모델링 방법 및 구현 방법 |

먼저 미리 텍스트 파일로 저장된 역 고유 넘버, 역간 시간, 어떤 역과 인접해 있는지를 파일에서 불러와 인접 리스트를 만듭니다. 다익스트라 알고리즘은 배열에서 구현하는 게 가장 직관적으로

편하게 보이지만 지하철과 같이 많은 데이터를 다룰 때에는 배열을 사용하면 가독하기 힘들고

제어하기도 쉽지 않습니다. 따라서 인접리스트 구조를 사용하여 현재에 해당하는 역에서

다음으로 갈 수 있는 모든 역을 연결하려고 하였습니다. 각 역마다 움직일 수 있는 역이 여러 개

이기 때문에 다수의 포인터를 가질 수도 있지만 그렇게 하지 않고 각 역은 포인터를 가지고 있으므로 다음 역이 다시 다음 역을 가리킬 수 있도록 지하철의 모든 노선을 연결하였습니다.

각 엣지의 값이 같을 때 최소 경로는 새로 발견한 노드까지의 경로 값이 현재 경로의 값+1 보다

크다면 새로 갱신을 해주는 것으로 구현할 있습니다.

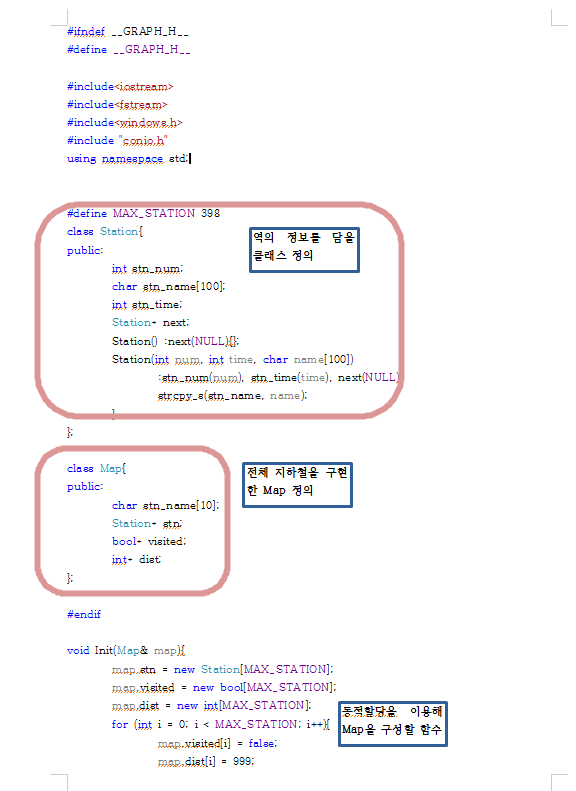
다음으로 경로를 출력하는 알고리즘에는 배열을 사용하여 보다 가볍게 코딩을 하였는데

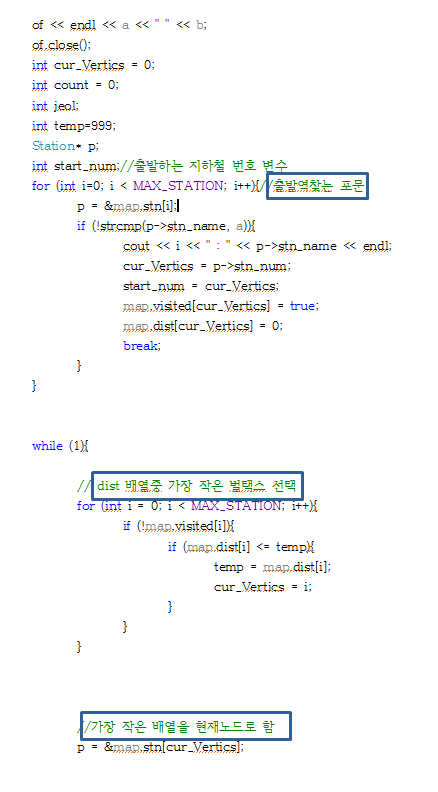
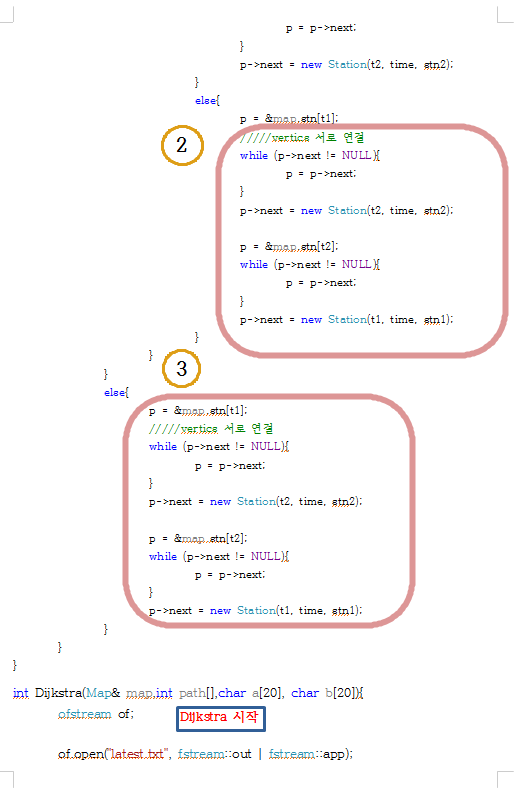
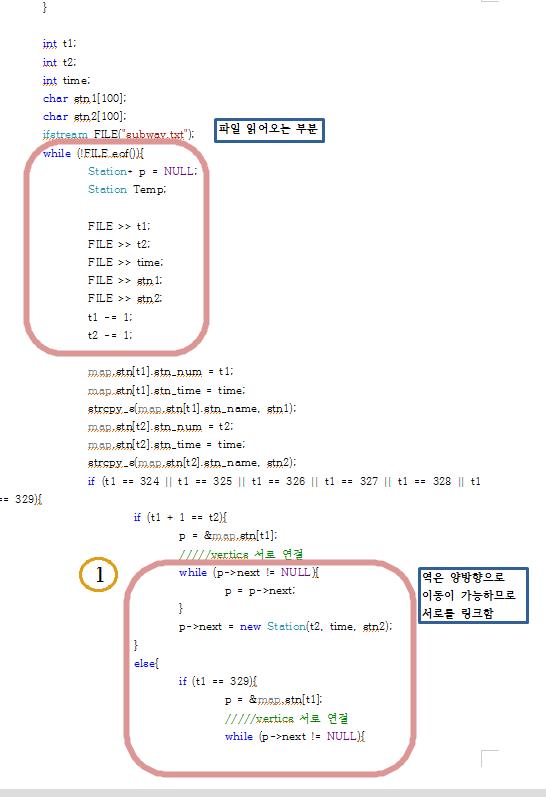
path[역이름(END)]에는 부모(A)의 역이름이 들어있습니다. 다시 path[부모(A)]를 한다면 값은

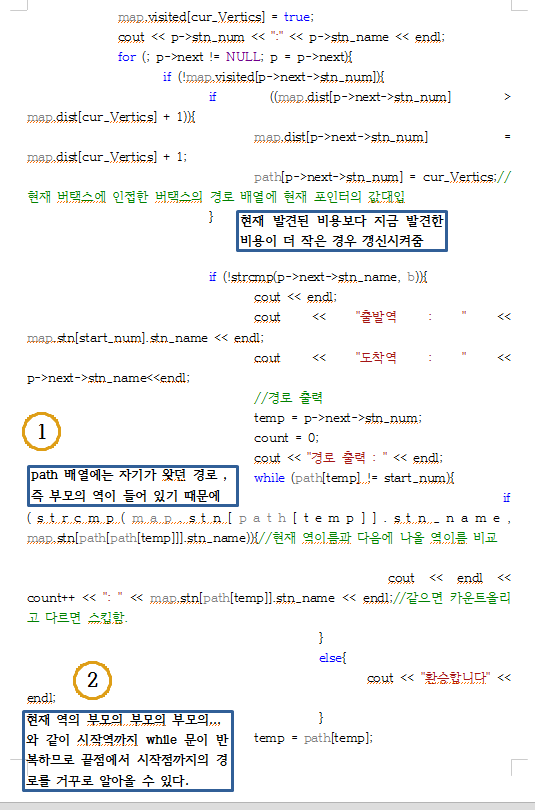
부모(A)의 부모이기도 하지만 END 의 할아버지이기도 합니다 다음과 같이 시작했던 역의 이름까지 거슬러 올라가면 시작점과 끝점 사이의 경로를 찾아낼 수 있습니다.

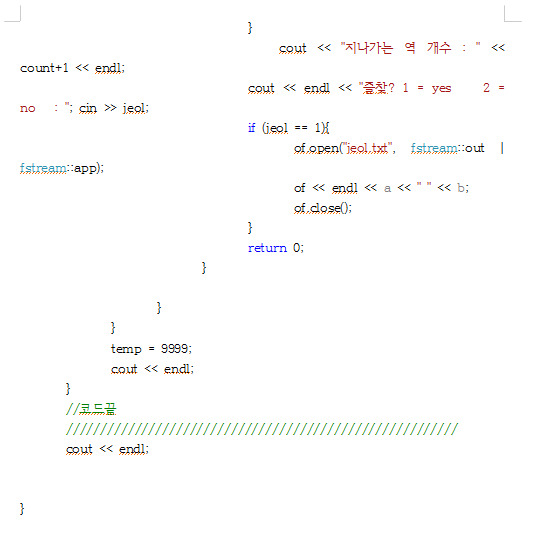
|  |
| --- |
| 주요 기능 및 기능별 구현 |

**< 기본 기능 >**

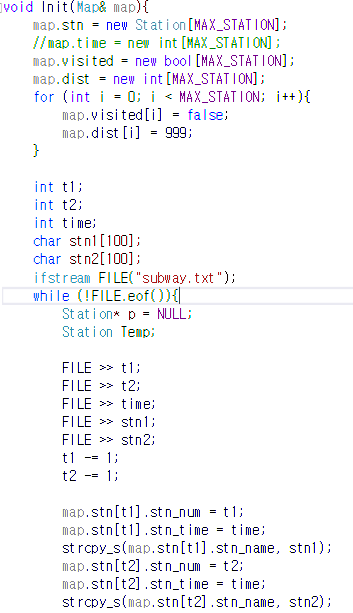
* **Dijkstra()**



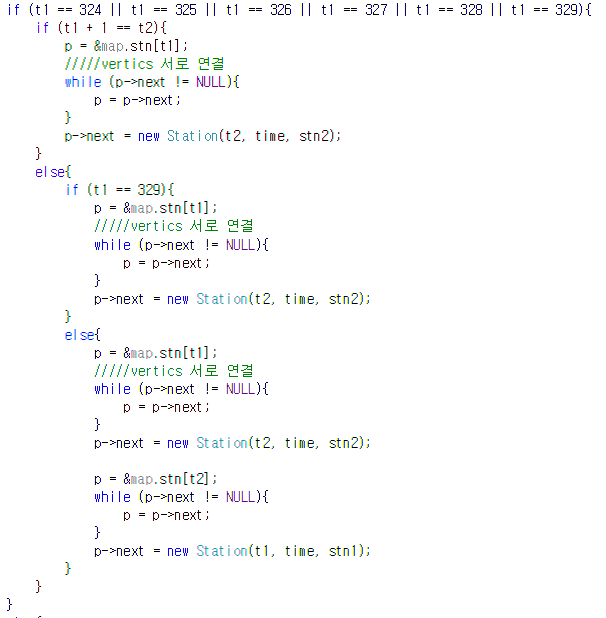




* **Init함수**



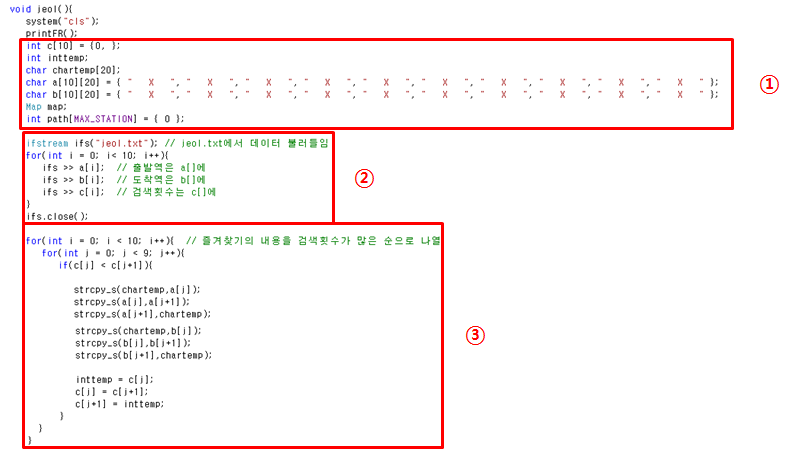
텍스트 파일에 역 번호와 시간 그리고 역 이름을 저장해서 배열에 넣었다. 두 개의 역 정보를 한 줄에 적어 두 역이 연결될 수 있도록 했다.

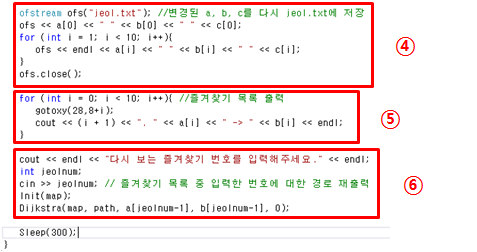


초기화 함수에서 회차하는 지하철역의 정보를 따로 예외 처리를 했다. 앞에서 말했듯이 두 역 정보를 한 줄에 표기하여 서로 연결하는 방식인데 서로 연결하는 것이 아니라 한쪽에서만 연결시킨다. 방향성 있으므로 앞에서 뒤로만 갈 수 있게 된다.(324~329 불광역~)

**< 추가 기능 >**

* **jeol() 함수**



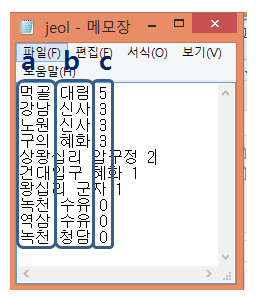


① 변수

- a : jeol.txt에 저장되어있는 즐겨 찾기의 출발 역을 담은 배열

- b: jeol.txt에 저장되어있는 즐겨 찾기의 도착 역을 담은 배열

- c : jeol.txt에 저장되어있는 즐겨 찾기의 검색 횟수를 담은 배열

ex) jeol.txt 

- inttemp, chartemp : 즐겨 찾기의 목록을 검색 횟수가 많은 순으로 나열하기 위해 배열의 순서를 바꿀 때 변수를 임시적으로 담아두는 변수

- map, path : 최소 역 경로 출력 시 Dijkstra()함수를 사용하기 위한 변수

② ifstream을 이용해 jeol.txt으로부터 데이터를 읽어 a. b, c 배열(a-출발 역, b-도착 역, c- 검색 횟수)에 하나씩 저장

③ 배열의 내림차순 정렬을 이용하여 a, b, c를 c에 담긴 검색 횟수 기준으로 정렬

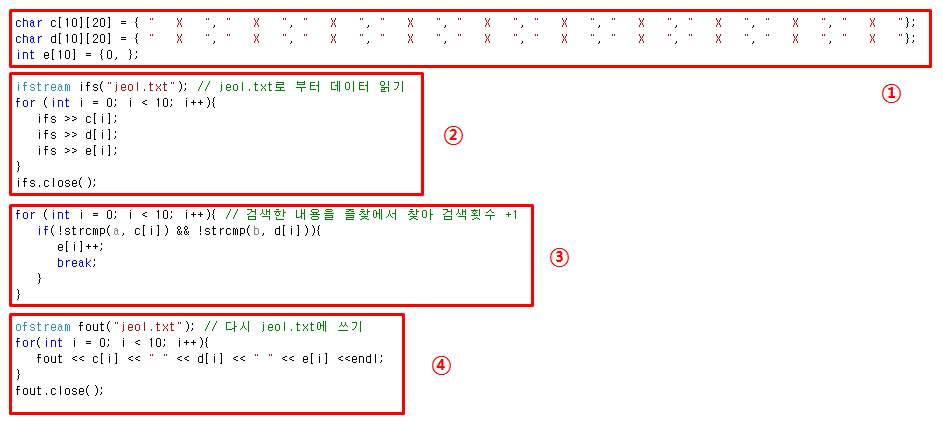
④ ofstream을 이용해 ②와 반대로 jeol.txt에 데이터 쓰기

⑤ 현재 a, b, c에 저장되어있는 내용이 jeol.txt로부터 읽어 들인 데이터를 정렬한 내용이므로 그대로 출력하여 **즐겨 찾기 목록을 검색 횟수가 많은 순**으로 보여줌

⑥ 즐겨 찾기에 있는 목록 중 다시 경로를 알고 싶을 때 해당 번호에 출발 역, 도착 역을 배열에서 찾아 Dijkstra()함수로 최소 역 경로 출력

* **Dijkstra() – 즐겨 찾기 기능 추가 위한 부분**

1.



① jeol함수의 변수 a, b, c와 같은 역할

( - a : jeol.txt에 저장되어있는 즐겨 찾기의 출발 역을 담은 배열

- b: jeol.txt에 저장되어있는 즐겨 찾기의 도착 역을 담은 배열

- c : jeol.txt에 저장되어있는 즐겨 찾기의 검색 횟수를 담은 배열 )

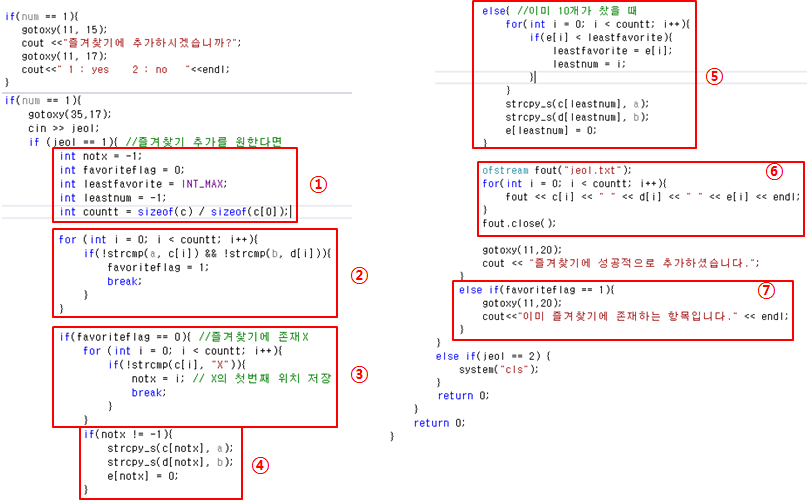
② jeol함수의 ②와 동일한 방법으로 jeol.txt로부터 데이터를 읽는다.

③ jeol.txt에 저장되어 있는 즐겨 찾기 중 입력한 출발 역, 도착 역과 같은 것이 있는지 비교해,

있다면 해당 즐겨 찾기의 검색 횟수를 1증가시킨다.

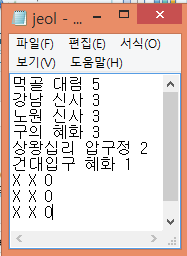
④ jeol함수의 ④와 동일한 방법으로 변경된 검색 횟수를 반영한 c, d, e배열을 다시 jeol.txt에 쓴다.

2.



① 변수

- notx : 즐겨 찾기에 등록된 사항이 10개 미만일 때, 역의 초기값은 x, 검색 횟수의 초기값은 0이다. 이 때 즐겨 찾기가 처음 x인 배열 번호를 저장한다.

ex )  -> notx = 7

- favoriteflag : 즐겨 찾기에 추가하고자 하는 내용이 이미 즐겨 찾기에 들어있는지 안 들어 있는지 여부를 판단하기 위해 설정한 변수이다. (1이면 이미 즐겨 찾기에 존재)

- leastfavorite, leastnum : 즐겨 찾기에 이미 10개가 모두 등록되어 있을 때, 검색 횟수가 가장 작은 것이 가장 덜 즐겨 찾는다는 가정아래 제거한다. 이 때 사용되는 변수이다.

- countt : c배열의 행의 수를 저장한다. ( sizeof(c)/sizeof(c[0]) 이므로)

② 현재 윗부분에 의해 c, d, e에는 각각 jeol.txt의 즐겨 찾기 항목들의 출발 역, 도착 역, 검색 횟수가 들어있는 상태이다. 이 배열을 검사하여 입력한 출발 역, 도착 역이 이미 즐겨 찾기에 있는지 검사하고, 있다면 favoriteflag = 1로 바꾼다.

③ favoriteflag = 0일 때, 즉 즐겨 찾기에 입력한 출발 역과 도착 역이 존재하지 않을 때에는 즐겨찾기에 추가가 가능하다. 이 때 먼저 즐겨 찾기 목록의 수가 10개 미만인지 검사하기 위해 c배열에 x가 존재하는 지 검사하고, 있다면 notx에 그 배열의 번호를 저장한다.

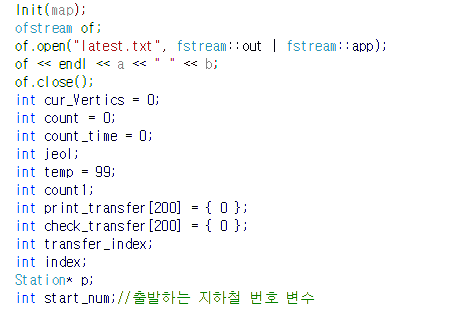
④ notx의 값이 처음에 설정한 -1이 아닐 경우에는 c배열에 x가 존재한다는 뜻이므로 아직 즐겨 찾기로 등록된 개수가 10개 미만이라는 뜻이다. 따라서 notx에 해당하는 즉, 비어있는 즐겨 찾기 중 첫 번째에 입력한 출발 역과 도착 역을 입력하고, 검색 횟수는 0을 넣는다.

⑤ notx의 값이 처음에 설정한 -1일 경우는 c배열에 x가 존재하지 않는다, 즉 이미 즐겨 찾기가 10개가 등록되었다는 뜻이다. 따라서 검색 횟수가 가장 적게 즐겨 찾는다는 알고리즘 하에 검색 횟수가 가장 적은 즐겨 찾기를 고른다. 이를 위해 e배열 중 최솟값을 골라 그 해당 번호에 입력한 출발 역과 도착 역을 입력한다.

⑥ ④,⑤로 인해 변경된 c, d, e배열을 다시 jeol.txt에 넣는다.

⑦ favoriteflag = 1인 경우에는 이미 즐겨 찾기 목록에 입력한 출발 역과 도착 역이 존재하는 것이므로 이미 등록 되어있다는 문구를 출력한다.

* **Dijkstra\_time 함수 – 최단 시간**



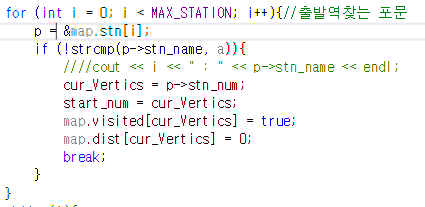
주요 변수 설명

Cur\_vertics : 현재 위치에 있는 노드의 번호 , 역의 아이디값

Count\_time : 걸리는 숫자의 합

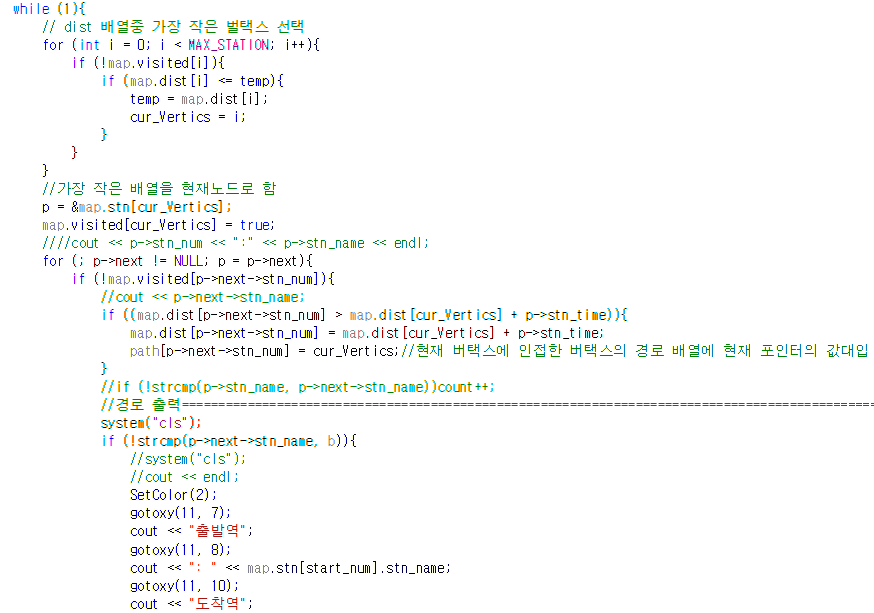
Print\_transfer[] : 역정보 역순 출력

Check\_transfer[] : 환승역 저장 배열

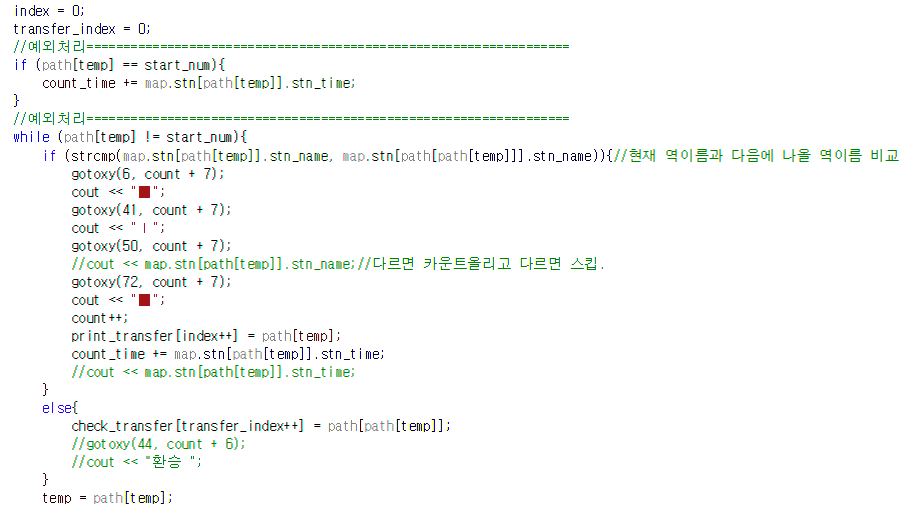


변수 a 는 출발역이고 b는 도착역이다

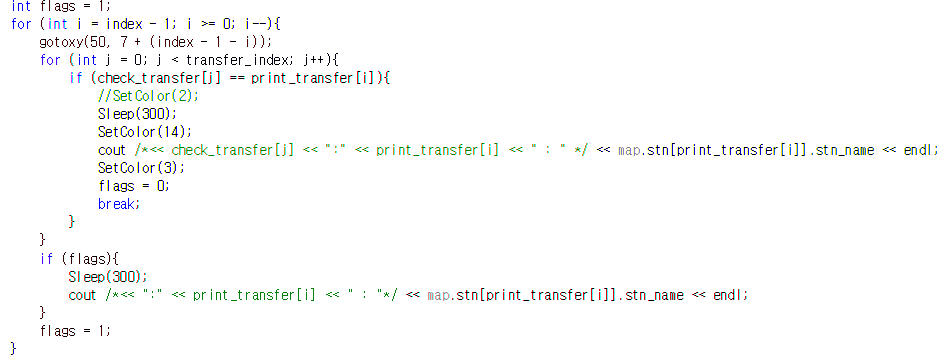
사용자가 입력한 역이름을 기준으로 출발역을 찾고 cur\_Vertics에 그 역의 고유번호값을 저장한다.



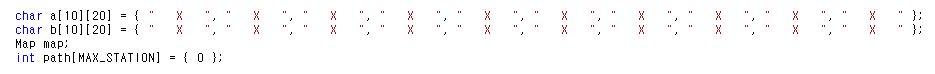
다익스트라 알고리즘 구현부분에서 거리를 1로해줬던 부분을 역사이의 시간값으로 대체하여 가중치를 두었다. 노드에 연결된 노드중에 찾고자하는 노드가 있으면 이동하고 또이동하는 식으로해서 도착역까지 간다.



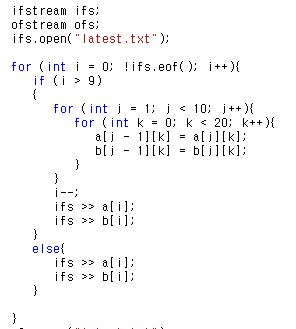
경로를 찾을때는 path라는 배열을이용하여 지나온 노드들을 저장해둔 배열을 이용한다.

출력해 줄 때는 따로 저장한 환승역들과 지나온 노선 경로의 역들을 비교하여 환승역일 때를 다르게 출력해준다. 역들이 순차적으로 보일 수 있도록 Sleep함수를 써주었다.

* **latest() 함수 – 최근 검색**

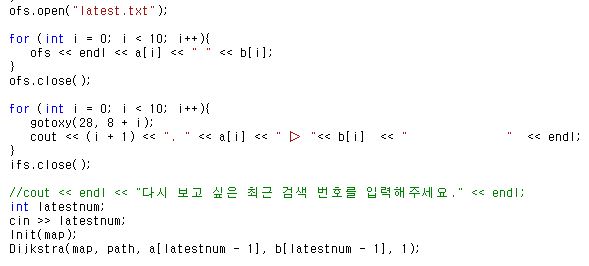


Map, Path와 최근 검색한 경로를 최대 10개까지 받기 위한 문자열들을 초기화 해주는 부분.



최근 검색한 경로들이 10개를 넘을 시, 맨 뒤 배열에서부터 한 칸 씩 앞 배열로 옮긴다.

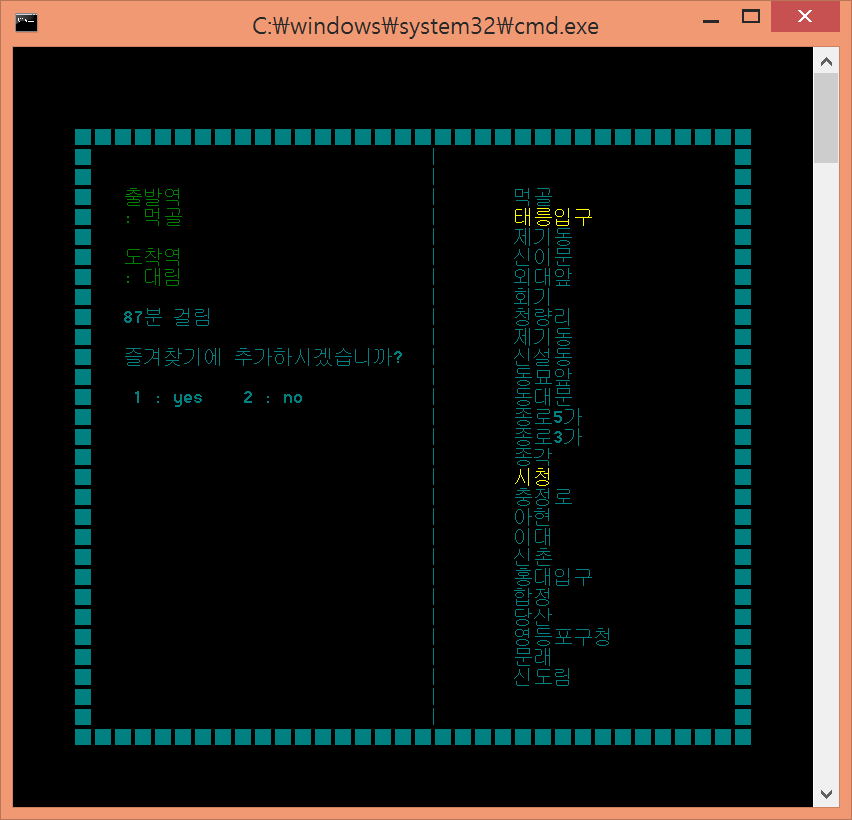
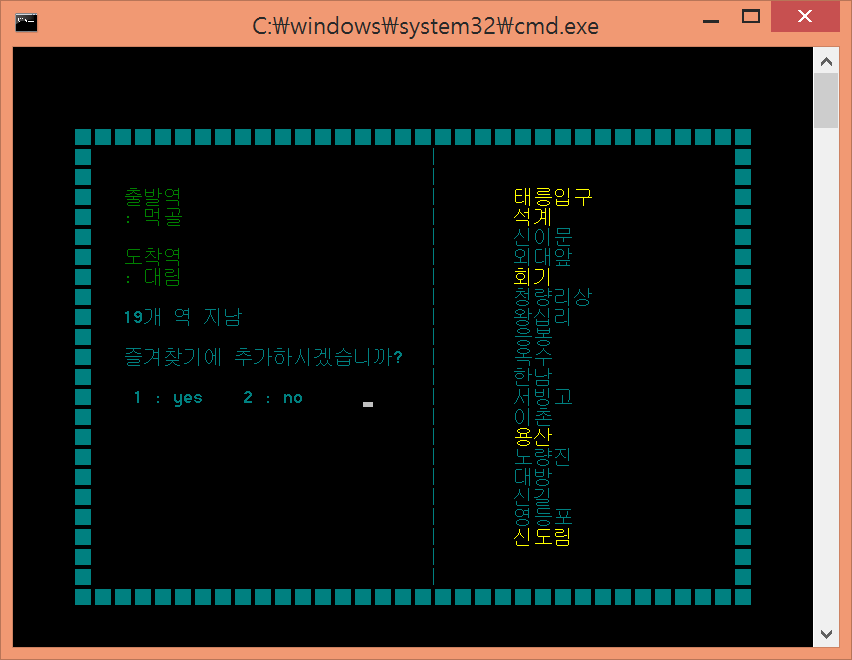
그 뒤 마지막 배열에 가장 최근 검색한 경로를 넣어준다.



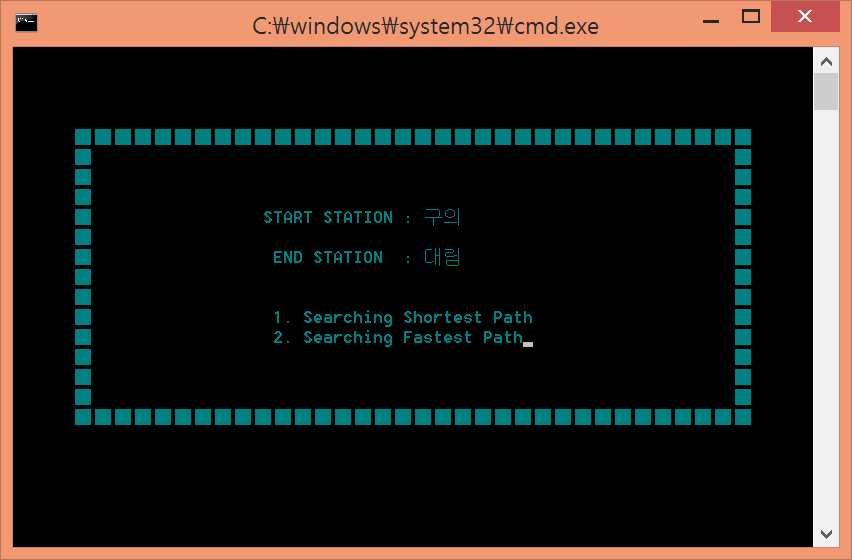
새롭게 받은 배열들을 latest 텍스트에 덮어씌워주고 닫는다.

순서에 따른 숫자를 입력하면 그 위치에 있는 경로를 다시 다익스트라 함수에 넣어서 보여준다.

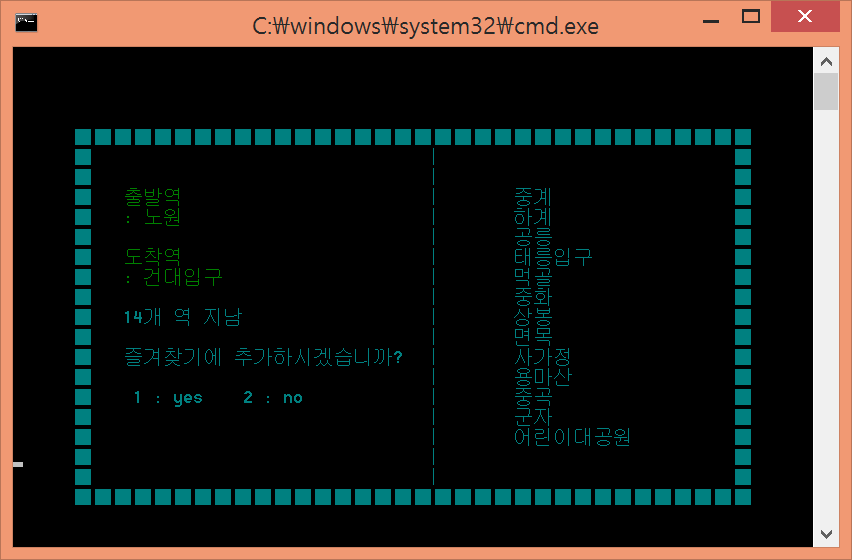
|  |
| --- |
| 실행 예 |



최소 역 / 최단 시간 선택 가능



즐겨 찾기 시



최근 검색 시



(즐겨 찾기, 최근 검색 모두 숫자를 누르면 해당 경로 다시 출력)

|  |
| --- |
| 토론 사항 |

처음에는 다익스트라 알고리즘을 배열애서 구현하려고 하였으나 배열을 이용하였을 때 제어가 쉽지 않고 너무 많은 시간이 소요될 것 같아 다른 방법을 고민해보았습니다. 미리 텍스트 파일에 역 고유 넘버, 역간 시간, 어떤 역과 인접해 있는지를 입력해 놓은 뒤 텍스트 파일에서 불러와 필요한 데이터 정보를 불러와 구현하는 방법으로 보완했습니다.

즐겨 찾기와 최근 검색을 구현 할 때 파일 입출력을 활용하였다. 이때 텍스트 파일에 추가할 수 있는 출발 역, 도착 역 개수를 10개로 한정시켜 놓았는데 지하철 검색이 많아지고 추가가 10개가 넘어가다 보면 오류가 나는 경우가 있었는데 파일 입출력으로 구현하며 이 문제를 해결하는 데 어려움을 겪었습니다. 이를 해결하기 위해 즐겨 찾기에서는 컴퓨터구조에서 배운 LRU 방법을 적용시키면 어떨까 싶어 가장 적게 검색한 것을 빼주고 그 부분에 새로운 경로를 추가하는 방법으로 하여 10개를 초과하는 점을 보완하였습니다. 최근 검색에서는 10개가 넘어가면 FIFO 방법을 적용해 제일 옛날에 등록된 최근 검색을 빼주는 방법으로 보완하였습니다.

|  |
| --- |
| 기여도 |

구건모 : 25% (다익스트라 함수 구현 및 다른 오류 수정)

이지연 : 25% (추가 기능 및 다른 오류 수정)

홍민지 : 25% (추가 기능 및 다른 오류 수정)

황세연 : 25% (최단 거리, 경로 출력, 디자인 및 다른 오류 수정)