typedef struct point{

int x;

int y;

}Point;

typedef struct vertex\_{

Point p;//vertex 번호 -> 미로에서 (x,y)좌표로 변환

struct vertex\_\*next;//각각의 출발 vertex간 연결

struct arcNode\_\*degree; //arc 노드의 포인터(하나의 출발점에 연결되어 있는 노드들 서로 연결)

}vertex;

typedef struct arcNode\_{

Point p;//좌표

int arc;//arc 가중치(갈림길 노드까지의 거리)

struct arcNode\_\*nextArc;//출발 vertex에 연결되어 있는 각각의 노드들 연결

}arcNode;

typedef struct graph\_ {

struct vertex\_\* start; //출발 graph하나로 쭉 연결된 그래프

}graph;

typedef struct dijkstra{ //다익스트라 테이블 구조체

int found;//(k)최단거리를 찾았는가? 1=yes, 0=no

int distance;//최단거리(미로의 한 갈림길에서 다음 갈림길까지의 거리)

Point prev\_vertex;//바로 직전에 어느 vertex를 거쳤는가(x,y좌표)

}Dijkstra;

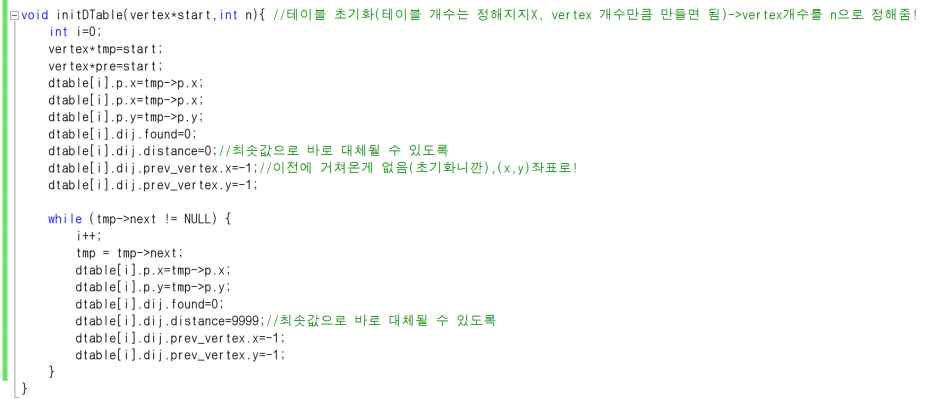
typedef struct \_dtable{

Point p;//x,y좌표

Dijkstra dij;//다익스트라 구조체

}Dtable;

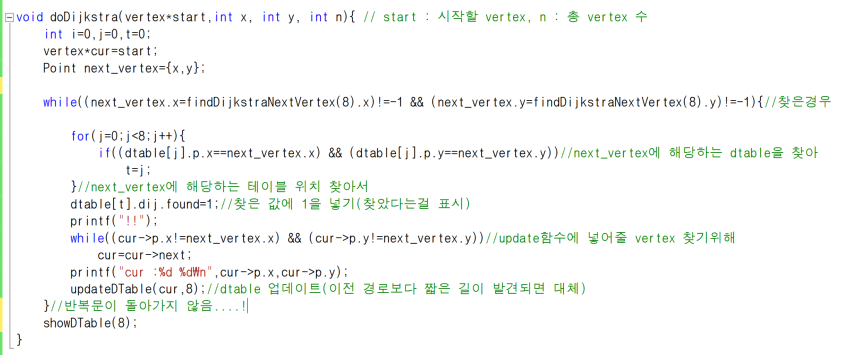
Dtable dtable[NUM\_VERTEX];



Initdtable : start vertex와 총 vertex의 개수를 입력으로 넣고

Table은 (x,y)좌표, found(찾았는지 여부), distance(최단 거리), prev vertex(이전 vertex의 좌표)로 초기화할때 현재 있는 vertex의 x,y는 각각 넣어주고 found는 0으로, distance는 출발하는 vertex에 해당하는 표만 0, 나머지와의 거리는 9999로 설정(계속 최단거리가 업데이트 되도록 하는 알고리즘이라 처음의 거리를 거의 무한으로 설정)

-dtable은 출력결과 이상 없습니다



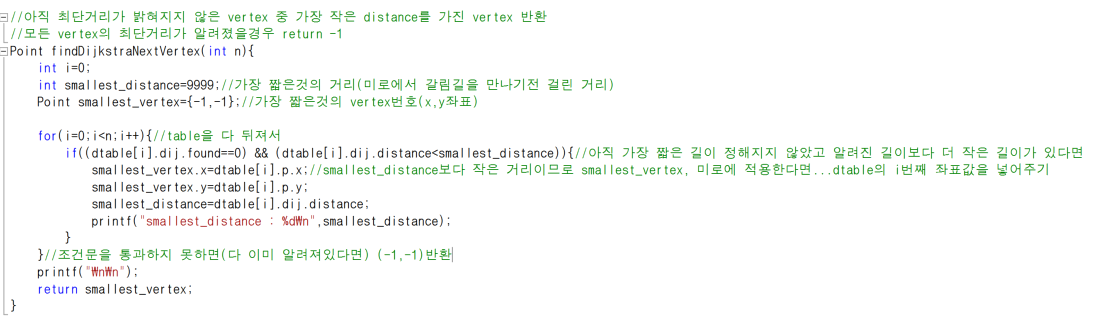
이게 doDijkstra로 실제로 main에서 실행시킬 함수입니다. 현재 dtable은 위에서 보다시피 전역변수로 Dtable dtable[NUM\_VERTEX]; 이렇게 선언해 놓은 상태고 매개변수로는 출발 vertex와 출발 vertex의 x,y좌표를 입력으로 넣어서 Point next\_vertex에 출발 x,y좌표를 넣어주고

그 다음 while((next\_vertex.x=findDijkstraNextVertex(8).x)!=-1 && (next\_vertex.y=findDijkstraNextVertex(8).y)!=-1) 이것을 보면 지금 findDijkstraNextVertex(8)이 vertex 테이블에서 최단거리가 밝혀지지 않은것의 vertex를 찾아서 그 중 가장 작은 vertex의 x,y좌표를 반환하는 함수인데 (아래에서 다시 설명할게요! )

그렇게 찾은 후 조건문에 들어가서 그 아래의 for문은 그 x,y 좌표에 해당하는 dtable의 위치를 찾아가서 t에 넣고 가장 짧은 거리를 가진 vertex를 발견했으므로 dtable[t].dij.found=1;로 찾았음을 저장해준 후에

이제 그 아래 while문은 또 update함수에 넣어줄, 그니깐 t에 해당하는 vertex의 x,y좌표를 가진 실제 vertex를 찾는 과정입니다.

이렇게 찾은 ‘cur’ vertex를 updateDTable함수에 넣어서 update를 해주는데 이는 아래서 다시 설명하겠습니다.

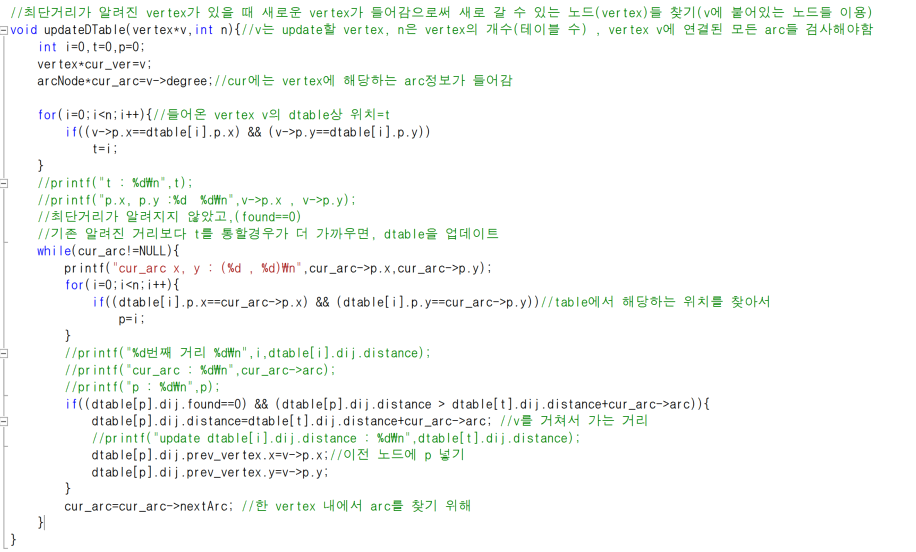


이게위에서 말한 findDijkstra함수인데 우선 smallest 거리를 9999로 해놨고 smallest vertex를 9999로 설정해 두었습니다

그리고 이제 table을 for문으로 뒤지는데 if문에는 found==0(아직 최단거리를 발견하지 못했고), dtable[i].dij.distance(i번째 dtable의 거리)가 smallest\_distance(처음엔 9999)보다 작으면 이제 조건문을 들어갑니다.

이렇게 들어온 뒤에는 smallest\_vertex의 x,y좌표를 조건문에 해당하는 dtable[i].p.x, p.y를 넣어주고 smallest\_distance에는 거리를 넣어줍니다. Smallest거리가 갱신됐으므로 반복문을 돌면서 if문안에 맞지 않는 경우가 생기겠죠. 그러면 이제 반복문이 끝나고 smallest\_vertex를 출력합니다. 이 때 조건문을 통과하지 못한다면 smallest\_vertex는 -1,-1을 출력합니다(이래서 위에 doDijkstra함수에 while((next\_vertex.x=findDijkstraNextVertex(8).x)!=-1 && (next\_vertex.y=findDijkstraNextVertex(8).y)!=-1){//찾은경우

이부분이 있는건데 이해 안가시면 말씀해주세요..!)



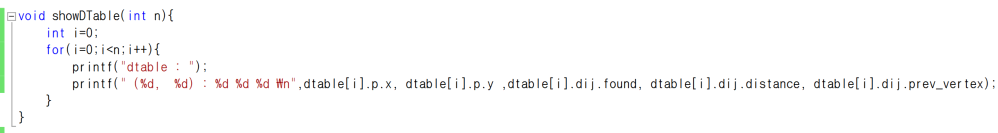
다음은 이제 table을 update하기 위한 함수입니다. 매개변수 vertex\*v는 doDijkstra함수에서 볼 수 있듯이 현재 업데이트 할 vertex를 넣어주고 arcNode\*cur\_arc=v->degree를 보면 vertex에서 arc들을 찾아 나갈 수 있게 cur\_arc를 선언해주었습니다

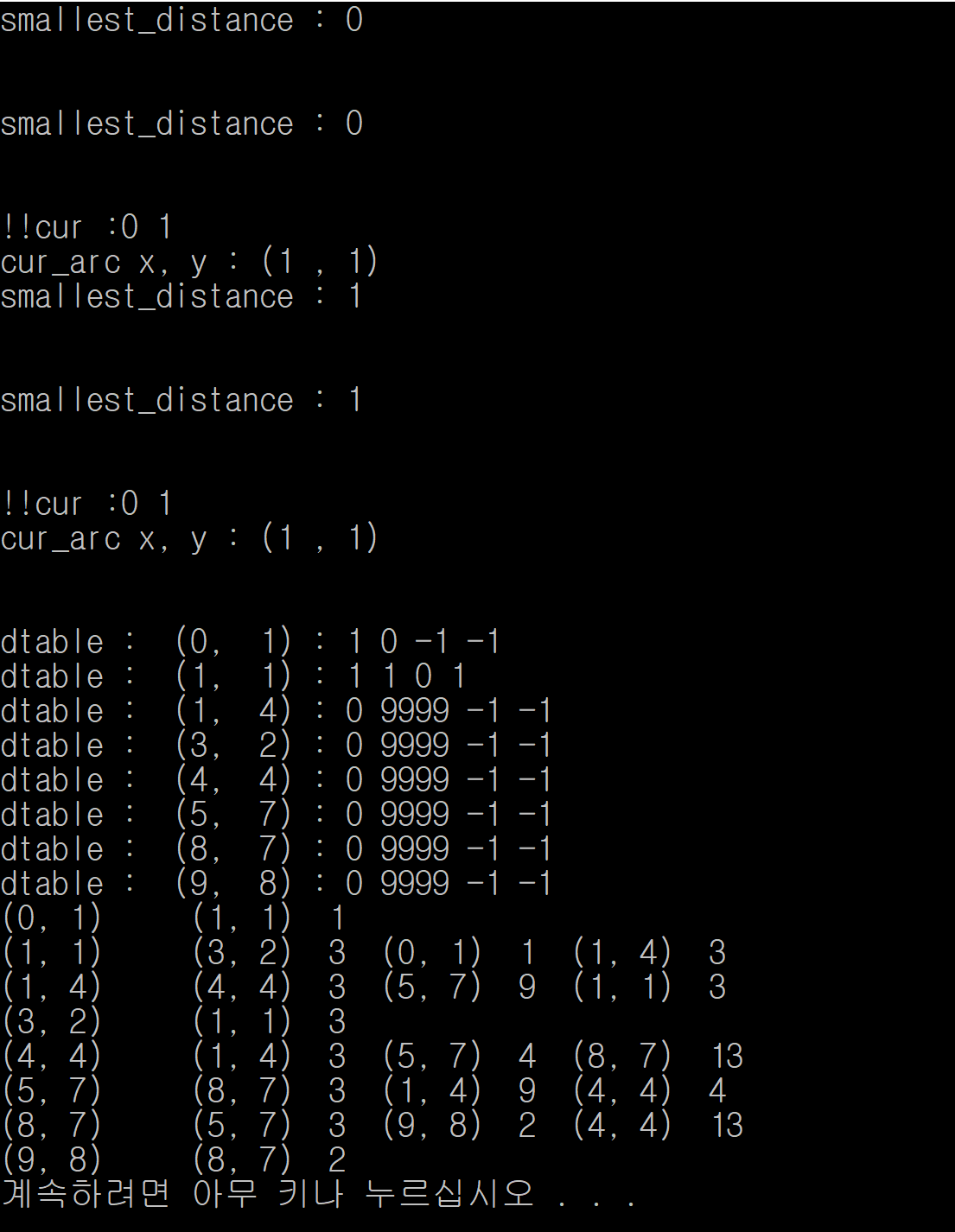
그 다음에 for문으로 역시 매개변수로 들어온 update할 vertex가 dtable에서 어디에 해당하는지 찾아서 t에 그 인덱스를 저장했고

While(cur\_arc!=NULL)은 arc끝까지 갈 때 까지 다시 for문에서 이제 그 arc가 table에 어디에 해당하는지 위치를 찾은 후 그 인덱스를 p에 저장했고

조건문을 들어가 dtable[p].dij.found (이 때 p는 들어온 vertex가 있는 table)이 0으로 아직 최단거리를 찾지 못했고, dtable[i].dij.distance(그 p의 거리가) dtable[t].distance(t는 while문 안에서 cur\_arc의 dtable상 위치)+cur\_arc->arc(해당하는 cur\_arc의 arc)를 합한거보다 크다면 즉, 기존의 거리가 새로운 거리+아크(가중치)값보다 컸다면

조건문을 들어가 업데이트 하는데 dtable[p].dij.distance=새로운 거리+아크(가중치)값으로 대체하고 이전 노드의 좌표 넣어주었습니다.





이 결과를 보면 처음 (0,1) : 1 0 -1 -1에서 처음 자기 자신은 찾았으므로 found에 1, 자기 자신과의 거리는 0, 이전 좌표는 -1, -1로 채워져있고 그 아래줄에 dtable : (1 ,1) : 1 1 0 1에서 found 1이고 거리는 1(이전 노드 (0,1) 에서 거리), 이전 노드의 좌표 0,1로 업데이트가 잘 되있는 결과를 볼 수 있는데 그 아래로는 아예 업데이트가 되지 않는군요… 테이블 위에는 디버깅을 위해 찍어본건데 역시나 반복문을 돌지 못해 죽어버리는걸 볼 수 있구요… 코드가 깔끔하지 않고 글이 길어서 죄송합니다…ㅠ… 하루 종일 해봤는데 해결책을 못찾고 있어서… 부탁드릴게요..!