پروژه نهایی ساختمان داده MAZE

اسحاق موتابی

9331573

**توضیحات مربوط به کد:

ما یک کلاس تعریف کرده ایم به اسم maze که در آن نقطه شروع و نقطه پایان و نقشه بازی را نگه میداریم.

در این کلاس از 3 تابع اصلی و چند تابع کمکی استفاده شده است:

```
bool Readingfile();
void print();
vector<pair<int, int> > DFS(pair<int, int>, pair<int, int>);
vector<pair<int, int> > BFS(pair<int, int>, pair<int, int>);
vector<pair<int, int> > Dijkstra(pair<int, int>, pair<int, int>);
bool checkfor_finish_or_not(int, int);
bool check(int, int, vector<vector<int> >);
pair<int, int>find(int, vector<vector<int> >, pair<int, int>);
void setdes(pair<int, int>, pair<int, int>);
bool DFS_check(int, int, vector<vector<int> >, vector<vector<int> >);
  pair<int, int>find_optimized(int x, vector<vector<int> >, pair<int, int>);
```

- `bool Readingfile();`

در این تابع ما نقشه را به صورت فایل میخوانیم.

- `vector<pair<int, int> > DFS(pair<int, int>, pair<int, int>);`
- `vector<pair<int, int> > BFS(pair<int, int>, pair<int, int>);`
- `vector<pair<int, int> > Dijkstra(pair<int, int>, pair<int, int>);`

این 3 تابع اصلی ما هستند که نقطه شروع و نقطه پایان رو به صورتpairبه آنها میدهیم و در نهایت مسیر مورد نظر را در یک vectorمیریزیم.

- `bool checkfor_finish_or_not(int, int);`

این تابع چک میکند که در هر لحظه به هر نقطه ای که میرسیم آیا به نقطه پایانی رسیده ایم یا نه.

- `bool check(int, int, vector<vector<int> >);`

این تابع نیز valid بودن نقطه را چک میکند که برای مثال در ان نقطه دیوار نباشد یا از صفحه بیرون نزده باشیم.

- `pair<int, int>find(int, vector<vector<int> >, pair<int, int>);`

این نیز چک میکند که در هنگام رسیدن به مقصد چون به خانه ها عدد میدادیم ان عدد ها را به صورت وارونه و نزولی پیدا کند تا به خانه اول برسیم.

- `void setdes(pair<int, int>, pair<int, int>);`

این تابع نیز مختصات های ورودی را به صورتpair**به کلاس میدهد.**

- `pair<int, int>find_optimized(int x, vector<vector<int> >,pair<int,int>);`

این همان تابع find**است که البته جهت های حرکت بر اساس ارزش ها بهینه شده است.**