

DSA

REPORT FINAL PROJECT



TEACHER

Samira Khodabandehlou

TEACHER ASSISTANT

Fatemeh Jafari Afshar

PROJECT PERFORMER

Muhammad Mahdi Motahari

Fall 1400

مقدمه

بعد از گذراندن درس ساختمان داده مانند باقی دروس تخصصی پروژه پایانی ای را باید می نوشتیم و تحويل می دادیم که موعود آن فرا رسیده است.

در این پروژه قصد آن است که کوتاه ترین مسیر دو سرباز که مقصدی یکسان دارند مشخص شود و بر اساس طول مسیر و امتیاز هایی که در آن مسیر وجود دارد سرباز برنده مشخص شود. (ما خیلی هم جدی هستیم:))

دایرکتوری پروژه

```

.
├── build
│   └── app
├── data
│   ├── map3.txt
│   ├── screen1.png
│   ├── screen2.png
│   └── screenOfGameProject.jpg
├── include
│   ├── node.hpp
│   └── printPlan.hpp
└── README.md
src
└── main.cpp
    └── printPlan.cpp
4 directories, 10 files

```

همانطور که مشخص است طبق استاندارد پوشه بندی در دایرکتوری src دو فایل main , printPlan وجود دارند که در ادامه بیشتر در موردنظر می بینید.

پوشه دارای include فایل های هدر می باشند که تعریف node در آن موجود است.

پوشه data دارای تصاویری از پروژه و نقشه های مورد استفاده در بازی می باشد.

پوشه build هم فایل اجرایی برنامه در آن است که اگر خواستید پروژه را کامپایل کنید مستقیما آن را اجرا کنید (که البته در سیستم عامل لینوکس کامپایل شده و خداروشکر قابل اجرا در سیستم عامل دوست و برادر ویندوز نمی باشد).

فرضیات

جدولی از مقادیر ممکنی که در خانه های یک نقشه می توانند قرار بگیرند را در زیر مشاهده می کنید.

	راه آسفالت	1
	چمن	3
	صحرا خشک	17
	درخت	2
	انبار مهمات	4
	معدن طلای گروه A	5
	بوته زار گروه A	6
	خانه مسکونی گروه A	7
	پادگان نظامی گروه A	8
	شخص عادی از گروه A	9
	سرباز از گروه	11
	شخص عادی از گروه B	10
	سرباز از گروه B	12
	معدن طلای گروه B	13
	بوته زار گروه B	14
	خانه مسکونی گروه B	15
	پادگان نظامی گروه B	16
	غیره	...

سرباز ها فقط از خانه های ۱، ۳، ۵، ۸، ۱۱، ۱۲، ۱۳، ۱۶ و ۱۷ می توانند عبور کنند و مابقی خانه ها غیر قابل عبور هستند.

امتیاز دهی

اگر سرباز A از خانه ۵ (معدن طلا) عبور کند، ۱۰۰۰ امتیاز مثبت ۱۶ (پادگان B) عبور کند، ۵۰۰ امتیاز منفی ۱۲ (سرباز B) عبور کند، ۱۰ امتیاز منفی برایش ثبت می شود.

اگر سرباز B از خانه ۱۳ (معدن طلا B) عبور کند، ۱۰۰۰ امتیاز مثبت ۸ (پادگان A) عبور کند، ۵۰۰ امتیاز منفی ۱۱ (سرباز A) عبور کند، ۱۰ امتیاز منفی برایش ثبت می شود.

پ.ن: همچنین فرض گرفته می شود که هنگام شروع بازی سرباز ها در یکی از موقعیت های ۱، ۳ و ۱۷ قرار بگیرند.

	A سرباز
	B سرباز
	مسیر طی شده سرباز A
	مسیر طی شده سرباز B

برنامه نویسی

این پروژه با زبان برنامه نویسی C++ نوشته شده و از رابط گرافیکی استفاده نشده و رابط کنسولی دارد.(که مقداری با ایموجی ظاهر زیبایی داده شده تا خسته کننده نباشد)

Function main

```
24 int main(int argc, char* argv[]){
```

برای اجرای بازی هنگام وارد کردن کامند باز کردن بازی بعد از اسم فایل بازی باید آدرس نقشه و سایز نقشه را وارد کنید.

سپس برنامه فایل را باز میکند و محتواهی نقشه را در یک پوینتر دو بعدی ذخیره میکند

```
39     int** plan = new int*[height];
40     for(int i = 0; i < height; i++){
41         plan[i] = new int[width];
42     }
43
44     for(int i = 0; i < height; i++){
45         for(int j = 0; j < width; j++){
46             file >> *(*(plan+i)+j);
47         }
48     }
49 }
```

و با استفاده از تابع printPlan() (که در فایل src/printPlan.cpp پیاده سازی شده) نقشه خام را در کنسول چاپ میکند.

در دو node ریشه می سازد که موقعیت فعلی سرباز b, a به ترتیب در انها می باشد و با استفاده از تابع `findPath` تمامی مسیر های ممکن را بصورت یک درخت که ریشه آن موقعیت اولیه سرباز ها است می سازد و همچنین یک وکتور که در پارامتر این تابع بصورت پوینتر داده می شود تمامی برگ ها (مسیر های که به مقصد ختم می شوند) را در خود ذخیره می کند و با استفاده از تابع `sort` این وکتور را بر اساس میزان طول مسیر مرتب می کند و اولین خانه از این وکتور کوتاه ترین مسیر ممکن می شود.

و در ادامه تابع `main` با استفاده از شرط های تو در تو نتیجه نهایی را بر اساس زیر چاپ می کند.

```

115     if(leafsA.begin()->second < leafsB.begin()->second){
116         cout << "\t\t****Soldier A won****" << endl;
117     }else if(leafsA.begin()->second > leafsB.begin()->second){
118         cout << "\t\t****Soldier B won****" << endl;
119     }else{
120         if(totalPointA > totalPointB){
121             cout << "\t\t****Soldier A won****" ;
122             cout << "\t=>\tDirection length of A and B is equals but total point of A greater than B" << endl;
123         }else if(totalPointA < totalPointB){
124             cout << "\t\t****Soldier B won****" ;
125             cout << "\t=>\tDirection length of B and A is equals but total point of B greater than A" << endl;
126         }else{
127             cout << "\t\t****Point and direction length of soldier A and B is equals****" << endl;
128         }
129     }

```

تابع `findPath`

الگوریتم اصلی پیدا کردن کوتاه ترین راه در این تابع پیاده سازی شده.

در واقع ما به این تابع پوینتری از گره ریشه (که همان گره موقعیت اولیه سرباز است) را می دهیم و وظیفه این تابع این است که این درخت را تکمیل کند، از ریشه شروع می کند و با استفاده از تابع `checkCell` بررسی می کند که به کدام خانه های اطراف میتواند برود و اگر رفتن به هر کدام از ان خانه ها مقدور باشد یک برگ برای آن ریشه ساخته می شود و بصورت بازگشتی دوباره راه های مقدور را برای برگ بررسی می کند و ...

در ورودی های این تابع یک وکتور جفت نود و عدد وجود داره که در تابع این وکتور تکمیل می شه و اطلاعات برگ های انتهایی (خانه مقصد) در ان ذخیره می شه که شامل یک پوینتر از نود و میزان اتفاع اون نود هستش که در تابع `main` بر اساس همین وکتور کوتاه ترین راه (برگی که ارتفاع کمتری داشته باشد) را پیدا می کنیم.

همچنین امتیازات هر سرباز در متغیرهای بنام totalPointA , totalPointB ذخیره میشوند (که با استفاده از تابع totalPointCalc محاسبه شده اند)، که در صورت نیاز (هنگامی که مسافت کوتاهترین مسیر A,B یکسان شود) برای تشخیص برندگان سرباز ها از انها استفاده شود.