((به نام خدا))

استاد اشراقی

گروه سوم

نام اعضا:

عرفان رادفر

بابک تاریویردی لوی اصل

حسین عباسی

محمد جواد احمدی

محمد سجادی

ابوالفضل وطن خواه

نام پروژه: بازی **2048**

فهرست مطالب

شرح بازی

بازی **2048** یک بازی ریاضی می‌باشد که در آن صفحه ای ۴*۴ متشکل از تعدادی از توان های عدد ۲ و تعدادی صفر(خان های خالی) است و شما باید با انتقال این اعداد به چپ یا راست یا بالا یا پایین سعی بر امتیاز گیری داشته باشید

به این صورت که با هر بار تغییر ارایش بازی عدد دلخواه ۲(در مراحل بالاتر اعداد ۴ و ۸) به جای یکی از صفر ها جایگزین می‌شود و نیز در صورت برخورد اعداد یکسان با هم یکی از آنها (با توجه به حرکت صورت گرفته) صفر شده و دیگری ۲ برابر میشود و به اندازه ۲ برابر یکی به امتیاز افزوده میشود (خانه ها روی هم افتاده و جمع اعداد آن ها به امتیاز شما اضافه می شود.)

اگر یکی از خانه های جدول شما به عدد ۲۰۴۸ برسد شما برنده می شوید و می توانید خارج شوید یا به رکورد خود اضافه کنید ولی اگر قبل از این که یکی از خانه هایتان به ۲۰۴۸ برسد و وضعیت صفحه بازی شما به گونه ای باشد که هیچ دو عدد مشابهی در کنار هم نباشند و شما امکان هیچ حرکتی نداشته باشید شما می بازید.

تابع pr()

هیچ گونه وروردی و خروجی ندارد و صرفا در ابتدای تابع main فراخوانی می‌شود تا چند جمله را که راهنمایی هایی درباره بازی است را چاپ کند.

تابع notfilled ()

دو عدد به عنوان سایز آرایه و نیز یک آرایه دو بعدی به همان سایز ها دریافت می‌کند. سپس با استفاده از دو حلقه تو در تو ، تک تک درایه های آن آرایه را بررسی می‌کند. اگر در بین درایه‌ها صفر وجود داشته باشد عدد ۱ را برمی گرداند و اگر هیچ صفری وجود نداشته باشد و همه درایه ها غیرصفر باشد عدد صفر را برمی‌گرداند.

تابع initia0()

دو عدد به عنوان سایز آرایه و نیز یک آرایه دو بعدی به همان سایز ها دریافت می‌کند. سپس با استفاده از دو حلقه تو در تو تک تک درایه های آن آرایه را صفر می‌کند.

تابع printboard()

تعدادی از دستورعمل ها را نمایش می دهد و دو عدد به عنوان سایز آرایه و نیز یک آرایه دو بعدی به همان سایزها را به عنوان ورودی دریافت می‌کند. در ابتدا با استفاده از دستور system (“cls”) محتویات کنسول را پاک می‌کند تا صفحه شلوغ نشده و هر بار برد جدیدی چاپ نشود. سپس چند راهنمایی برای ادامه بازی را چاپ می‌کند. در ادامه با استفاده از ۴ حلقه (که ۲ تای آنها تودرتو است) آرایه مربوط به برد بازی را به همراه خطوط تیره عمودی و افقی (به منظور تفکیک خانه های جدول) چاپ می‌کند. سپس با توجه به مقدار متغیر گلوبال signal که مقدار آن متناسب با هر بار دریافت ورودی از صفحه کلید تغییر می‌کند با یک دستور شرط if اقدام به چاپ عبارت مناسب می‌نماید (به این صورت که اگر مقدار متغیر signal صفر باشد یعنی ما دریافت مناسب از صفحه کلید داشته ایم و کلید مناسب فشار داده شده در غیر این صورت یعنی دکمه اشتباهی زده شده و عبارت

Your move is not possible try something else

چاپ می‌شود.)

بعد از این دستور مقدار signal صفر می شود تا حرکت بعدی را با فرض درست بودن بررسی کرده و در صورت ممکن نبودن دوباره مقدار ۱ را به آن منتصب کند.

تابع assign_array()

دو عدد به عنوان سایز آرایه و دو آرایه دو بعدی به همان سایز ها را به عنوان ورودی می‌گیرد و با استفاده از دو حلقه تو در تو تک تک درایه های آرایه دوم را در اولی منتصب می کند.

تابع is _array_same()

دو عدد به عنوان سایز آرایه و دو آرایه دو بعدی به همان سایز ها را به عنوان ورودی می‌گیرد و با استفاده دو حلقه تو در تو تک تک درایه های این دو آرایه را با یکدیگر مقایسه کرده و در صورت یکسان بودن تمام درایه های آرایه ها مقدار ۱ را برمی‌گرداند و در غیر این صورت عدد صفر را برمی‌گرداند.

تابع save_scores()

کد این تابع به این صورت است که یک فایل با نام scores * * تعریف می‌شود که در آن * * همان ابعاد برد بازی است که که توسط کاربر وارد می‌شود. Mode این فایل به صورت a می باشد و اطلاعات قبلی موجود در فایل پاک نمی‌شود بلکه تنها به انتهای آن افزوده می‌شود. در این تابع نام بازیکن نیز دریافت می‌شود و در فایل متنی نام بازیکن به همراه امتیاز او چاپ می‌شود. با

توجه به این که mode فایل a است می توان نام و امتیاز همه بازیکنانی که در یک ابعاد خاص ، بازی را انجام داده اند را در فایل مشاهده نمود.(در صورت عدم وجود فایل با ابعاد مورد نظر ، این فایل تشکیل می شود.)

تابع top_scores()

دو عدد که همان ابعاد آرایه بازی هستند را به عنوان ورودی می‌گیرد و سپس کل محتویات کنسول را با دستور system (“cls”) پاک کرده و به صفحه جدید می‌رود. سپس فایل متنی که با تابع save_scores() نوشتیم (و امتیاز ها در آن بر اساس ترتیب ورود آن ها است) را با استفاده از mode r می‌خواند و سپس با یک حلقه (تا زمانی که کرسر به انتهای فایل نرسیده باشد (تابع feof)) نام کاربران را در آرایه‌ای به نام user به صورت رشته و امتیاز آن ها را در آرایه‌ای به نام scores ذخیره می‌کند و به هر فرد یک شماره از ۱ تا تعداد نفرات شرکت کننده را نسبت می‌دهد (که در آرایه number ذخیره می شود) و با استفاده از یک شمارنده مناسب در حلقه شمار افراد شرکت کننده را می‌شمارد . در ادامه با استفاده از تابع bubble که در ادامه توضیح داده می‌شود این داده ها مرتب می‌شوند و در محیط کنسول رتبه هر فرد به همراه نام و امتیاز او نمایش داده می‌شود. سپس با گرفتن ورودی مناسب(‘c’) به صفحه بازی بازمی‌گردد. لازم به ذکر است که در صورتی که نتواند فایل متنی را پیدا کند عدد صفر را برمی‌گرداند و در صورت موفقیت عدد ۱ را برمی‌گرداند.

با توجه به این که اسم شخص آخر ، کاراکتر('\n') می شود و مورد نظر ما نیست، در تمام فرایند های شمارش بعدی ، از آن صرف نظر می کنیم.

تابع bubble()

در عمل همان الگوریتم bubble sort را دارد ولی درواقع برعکس آن است و با استفاده از دو حلقه تودرتو و یک دستور شرط به همراه یک متغیر کمکی امتیاز های افراد را از بزرگ به کوچک مرتب می‌کند. همین طور متناسب با امتیاز ها شماره های افراد (که همان رتبه آنهاست(آرایه number)) نیز تغییر میکند و رتبه ها از ۱ تا نفر آخر و امتیاز ها از بزرگ به کوچک مرتب می‌شوند.

تابع organize()

این تابع به صورت کلی برای مدیریت ورودی هایی می باشد که از طریق صفحه کلید دریافت می‌شود و فرایند حرکت بازی از طریق آن پیش میرود. ورودی های این تابع شامل یک کاراکتر، دو عدد که همان ابعاد برد بازی هستند و نیز یک آرایه دو بعدی به همان سایز ها است.

در ابتدای این تابع آرایه به نام clone تعریف می‌شود و سپس به وسیله تابع assign_array ، آرایه board در آن منتصب می شود.

دریافت z به عنوان ورودی(برای ادامه بازی ذخیره شده): یک فایل متنی دریافت کرده و با mode r آن را می‌خواند.

محتویات این فایل شامل دو عدد در ابتدا است که همان ابعاد برد بازی است و سپس آرایه ای به همان سایز و در انتها امتیاز بازی می باشد.

این ورودی برای ادامه دادن بازی ای می‌باشد که قبلا به صورت فایل متنی ذخیره شده است.

این قسمت دو عدد اولیه را به n و m منتصب می کند و سپس با دو حلقه تودرتو آرایه را خوانده و در آرایه ای به نام a ذخیره می‌کند و در انتها عدد امتیاز بازی را در متغیری ذخیره می‌کند.

سپس مقدار متغیر signal را برابر با ۳ قرار میدهد که در قسمت های بعدی از آن استفاده می‌کنیم.(برای این که عدد ۲ رندومی را به آرایه ذخیره شده اضافه نکند.)

در انتهای این قسمت تابع startgame اجرا می‌شود تا بازیکن بتواند بازی را با ابعاد و اعداد جدید ادامه دهد.

دریافت u به عنوان ورودی(برای ذخیره بورد بازی): یک فایل متنی با استفاده از مُد w ایجاد کرده و اقدام به نوشتن در داخل این فایل می‌نماید. (با توجه با mode این فایل اگر اطلاعات قبلی وجود داشته باشد حذف میشود و تنها امکان ذخیره یک بازی وجود دارد.)

به این صورت که دو عددی که در ابتدای فایل متنی درج می‌شود همان ابعاد بازی هستند و سپس به کمک دو حلقه تودرتو آرایه صفحه بازی چاپ می‌شود و سپس اقدام به درج امتیاز کنونی در فایل می‌کند.

دریافت d به عنوان ورودی(حرکت به راست): نحوه عملکرد آن به این گونه است که درایه های غیر صفر را آنقدر به سمت راست می‌برد تا سمت راستی ترین درایه به انتها برسد و نیز در صورتی که دو درایه یکسان بوده و در سمت راست و چپ هم باشند ، درایه سمت راستی دوبرابر شده و درایه سمت چپی صفر می‌شود.

نحوه عملکرد آن به این گونه است:

با استفاده از دو حلقه تودرتو تک تک درایه ها از بالاترین و سمت راستی ترین نقطه بررسی میکند و اگر درایه ای صفر باشد یک واحد به شمارنده می‌افزاید(البته این شمارنده در اجرای حلقه بعدی صفر می‌شود و تنها می‌تواند مقادیر صفر و یک داشته باشد) . اگر درایه صفر نباشد با استفاده از یک متغیر کمکی آن درایه را صفر می‌کند و مقدار آن را به درایه سمت راستی منتقل می‌کند.

مثلا آرایه ۲۴۲۰۴ ۰۲۴۲۴

۰۲۲۰۲ را به ۰۰۲۲۲ تبدیل می‌کند.

۲۲۰۴۰ ۰۰۲۲۴

سپس باردیگر با دو حلقه تودرتو از بالاترین و سمت راستی ترین نقطه شروع می‌کند و تک تک درایه ها را با درایه سمت چپی خود مقایسه می‌کند.

اگر این دو درایه برابر باشند درایه سمت چپ را صفر کرده و مقدار درایه سمت راست را دو برابر می کند. در این مرحله به اندازه مقدار درایه سمت راست جدید به امتیاز افزوده می‌شود.

مثلا آرایه ۰ ۲ ۴ ۲ ۴ ۰ ۲ ۴ ۲ ۴

۰ ۰ ۲ ۲ ۲ را به ۰ ۰ ۲ ۰ ۴ تبدیل می کند.

۰ ۰ ۲ ۲ ۴ ۰ ۰ ۰ ۴ ۴

سپس مثل همین مرحله اول درایه های غیر صفر را به سمت راست منتقل می‌کند.

مثلا آرایه ۰ ۲ ۴ ۲ ۴ ۰ ۲ ۴ ۲ ۴

۰ ۰ ۲ ۰ ۴ را به ۰ ۰ ۰ ۲ ۴ تبدیل می کند.

۰ ۰ ۰ ۴ ۴ ۰ ۰ ۰ ۴ ۴

دریافت a به عنوان ورودی(حرکت به چپ): نحوه عملکرد آن به این گونه است که درایه های غیر صفر را آنقدر به سمت چپ می‌برد تا سمت چپی ترین درایه به انتها برسد و نیز در صورتی که دو درایه یکسان بوده و در سمت راست و چپ هم باشند درایه سمت چپی دوبرابر شده و درایه سمت راستی صفر می‌شود.

نحوه عملکرد آن به این گونه است:با استفاده از دو حلقه تودرتو تک تک درایه ها از بالاترین و سمت چپی ترین نقطه بررسی میکند و اگر درایه ای صفر باشد یک واحد به شمارنده می‌افزاید(البته این شمارنده در اجرای حلقه بعدی صفر می‌شود و تنها می‌تواند مقادیر صفر و یک داشته باشد) . اگر درایه صفر نباشد با استفاده از یک متغیر کمکی آن درایه را صفر می‌کند و مقدار آن را به درایه سمت چپی منتقل می‌کند.

مثلا آرایه ۴۰۲۴۲ ۴۲۴۲۰

۲۰۲۲۰ را به ۲۲۲۰۰ تبدیل می‌کند.

۰۴۰۲۲ ۴۲۲۰۰

سپس باردیگر با دو حلقه تودرتو از بالاترین و سمت چپی ترین نقطه شروع می‌کند و تک تک درایه ها را با درایه سمت راستی خود مقایسه می‌کند.

اگر این دو درایه برابر باشند درایه سمت راست را صفر کرده و مقدار درایه سمت چپ را دو برابر می کند. در این مرحله به اندازه مقدار درایه سمت چپ جدید به امتیاز افزوده می‌شود.

مثلا آرایه ۴۲۴۲۰ ۴۲۴۲۰

۲۲۲۰۰ را به ۴۰۲۰۰ تبدیل می کند.

۴۲۲۰۰ ۴۴۰۰۰

سپس مثل همین مرحله اول درایه های غیر صفر را به سمت چپ منتقل می‌کند.

مثلا آرایه ۰ ۴ ۲ ۴ ۲ ۴ ۲ ۴ ۲ ۰

۰ ۴ ۰ ۲ ۰ ۰ را به ۴ ۲ ۰ ۰ ۰ تبدیل می کند.

۴ ۴ ۰ ۰ ۰ ۴ ۴ ۰ ۰ ۰

دریافت W به عنوان ورودی(حرکت به بالا): نحوه عملکرد آن به این گونه است که درایه های غیر صفر را آنقدر به سمت بالا می‌برد تا بالا ترین درایه به انتها برسد و نیز در صورتی که دو درایه یکسان بوده و در سمت بالا و پایین هم باشند درایه سمت بالا دوبرابر شده و درایه سمت پایین صفر می‌شود.

نحوه عملکرد آن به این گونه است:

با استفاده از دو حلقه تودرتو تک تک درایه ها از بالاترین و سمت چپی ترین نقطه بررسی میکند و اگر درایه ای صفر باشد یک واحد به شمارنده می‌افزاید(البته این شمارنده در اجرای حلقه بعدی صفر می‌شود و تنها می‌تواند مقادیر صفر و یک داشته باشد) . اگر درایه صفر نباشد با استفاده از یک متغیر کمکی آن درایه را صفر می‌کند و مقدار آن را به درایه بالایی منتقل می‌کند.

مثلا آرایه ۰ ۲ ۴ ۴ ۲ ۴

۰ ۰ ۴ را به ۲ ۲ ۲ تبدیل می‌کند.

۲ ۲ ۰ ۴ ۲ ۰

۴ ۲ ۲ ۰ ۰ ۰

سپس باردیگر با دو حلقه تودرتو از بالاترین و سمت چپی ترین نقطه شروع می‌کند و تک تک درایه ها را با درایه پایینی خود مقایسه می‌کند.

اگر این دو درایه برابر باشند درایه پایین را صفر کرده و مقدار درایه بالایی را دو برابر می کند. در این مرحله به اندازه مقدار درایه بالایی جدید به امتیاز افزوده می‌شود.

مثلا آرایه ۲ ۲ ۴ ۴ ۴ ۴

۲ ۲ ۲ را به ۲ ۰ ۰ تبدیل می کند.

۴ ۲ ۰ ۴ ۲ ۰

۰ ۰ ۰ ۰ ۰ ۰

سپس مثل همین مرحله اول درایه های غیر صفر را به سمت بالا منتقل می‌کند.

مثلا آرایه ۴ ۴ ۴ ۴ ۴ ۴

۲ ۰ ۰ را به ۴ ۲ ۲ تبدیل می کند.

۴ ۲ ۰ ۰ ۰ ۰

۰ ۰ ۰ ۰ ۰ ۰

دریافت s به عنوان ورودی(حرکت به پایین): نحوه عملکرد آن به این گونه است که درایه های غیر صفر را آنقدر به سمت پایین می‌برد تا پایینی ترین درایه به انتها برسد و نیز در صورتی که دو درایه یکسان بوده و در بالا و پایین هم باشند درایه پایینی دوبرابر شده و درایه بالایی صفر می‌شود.

نحوه عملکرد آن به این گونه است:

با استفاده از دو حلقه تودرتو تک تک درایه ها از پایین ترین و سمت چپی ترین نقطه بررسی میکند و اگر درایه ای صفر باشد یک واحد به شمارنده می‌افزاید(البته این شمارنده در اجرای حلقه بعدی صفر می‌شود و تنها می‌تواند مقادیر صفر و یک داشته باشد) . اگر درایه صفر نباشد با استفاده از یک متغیر کمکی آن درایه را صفر می‌کند و مقدار آن را به درایه سمت پایینی منتقل می‌کند.

		۴ ۲ ۰	۰ ۰ ۰	
مثلا آرایه	۴ ۰ ۰	را به	۴ ۲ ۰	تبدیل می‌کند.
		۰ ۲ ۲	۴ ۲ ۲	
		۲ ۲ ۰	۴ ۲ ۲	

سپس باردیگر با دو حلقه تودرتو از پایین ترین و سمت چپی ترین نقطه شروع می‌کند و تک تک درایه ها را با درایه بالایی خود مقایسه می‌کند.

اگر این دو درایه برابر باشند درایه بالایی را صفر کرده و مقدار درایه پایینی را دو برابر می کند. در این مرحله به اندازه مقدار درایه پایینی جدید به امتیاز افزوده می‌شود.

مثلا آرایه

۰	۰	۰		۰	۰	۰
۰	۲	۴	را به	۰	۲	۴
۴	۲	۲		۴	۰	۲
۲	۲	۴		۲	۴	۴

تبدیل میکند.

سپس مثل همین مرحله اول درایه های غیر صفر را به سمت پایین منتقل می‌کند.

مثلا آرایه

۰	۰	۰		۰	۰	۰
۰	۲	۴	را به	۰	۰	۴
۴	۰	۲		۲	۲	۴
۲	۴	۴		۴	۴	۲

تبدیل میکند.

نکته : در تمام حرکت ها ، اگر هر یک از درایه های بازی به مقدار رسید که از بقیه درایه ها بزرگتر باشد، مقدار آن در max_array ذخیره می شود.

دریافت m به عنوان ورودی(ذخیره امتیاز): این حرکت ، کنترل را از تابع organize خارج می کند و اگر direction برابر ’z‘ نباشد(یعنی در حال خارج شدن از یک بازی ذخیره شده نباشیم که سبب شود بازی ابتدایی تمام شود) وارد save_scores می کند و از ما اسممان دریافت کرده و امتیاز ما در فایل متنی متناسب با ابعاد بازی ذخیره می شود و اعلام می کند که آیا بازی را برده ایم یا نه. در نهایت از startgame نیز خارج شده و در main برنامه بازی به انتها می رسد.

دریافت h به عنوان ورودی: این ورودی برای نمایش نفرات برتر بازی به همراه امتیازات آن ها است. در صورت فشردن h از ما می‌خواهد تا ابعاد مدنظر خود را وارد نماییم تا نفرات برتر مربوط به آن ابعاد را از طریق تابع top_scores مشاهده کنیم.

همچنین در صورتی که تابع top_scores نتواند فایل متنی امتیازات را بخواند عبارت مناسب را چاپ خواهد کرد که چنین فایلی وجود ندارد.

همچنین در ادامه با فشردن کلید مناسب از طریق راهنمایی می توان به صفحه بازی بازگشت.

بعد از این دستور ها ، در انتهای تابع organize بررسی می شود که آیا خانه های بازی کاملا پر شده اند یا نه.اگر آرایه board پر نشده باشد و اگر دو آرایه board و clone یکی باشند (و دستورات منوm,u,z,h نباشند)به این معناست که حرکت بی نتیجه و بدون تغییر بوده پس signal برابر ۱ می شود تا بعدا در printboard پیام حرکت غیر ممکن چاپ شود و عدد رندُم جدیدی وارد خانه ها نشود. در غیر این صورت و زمانی که حرکت انجام شده قابل قبول باشد تابع fill_rand اجرا می شود تا یکی از خانه های خالی پر شود. برگشت مقدار ۱ به این معناست که بازی ادامه دارد.

اما اگر خانه ها پر شده باشند، آنگاه برنامه بررسی می کند که آیا هیچ حرکتی ممکن است انجام شود که بازی ادامه پیدا کند.اگر نه آنگاه با انجام هر یک از حرکات بالا ، پایین ، چپ و راست ، بازی پایان می یابد وگرنه بازی می گوید که حرکت ما غیر ممکن است(ولی امکان ادامه وجود دارد.)

روش بررسی حرکت های ممکنه به این صورت است که ابتدا board در آرایه filled منتصب شده و filled هر دفعه وارد یک organize مجزا می شود و به روش بازگشتی ۴ حرکت ممکنه چک می شود، اگر حرکت پیشنهادی ناموفق بود به متغیر گلوبال stuck_moves یکی اضافه شده تا برای بررسی بعدی ، سراغ حرکت بعدی برود. اگر یکی از حرکت ها ممکن بود مقدار ۱ برگردانده شده و stuck_moves صفر شده و برنامه به حلقه داخل startgame باز می گردد. اگر هم هیچ حرکتی ممکن نبود،

stuck_moves به ۵ رسیده و صفر برگردانده می شود که یعنی برنامه تمام شده است.

نکته : دلیل وجود

```
if(signal==2)
score=previous_score;
```

این است که در طی مراحل مختلف بررسی و در صورت ممکن بودن حرکت، حرکت بررسی شده توسط برنامه به عنوان حرکت کاربر شناسایی نشده و امتیاز اضافی به کاربر داده نشود.

تابع fill_rand()

این تابع وظیفه پر کردن خانه های خالی آرایه را دارد. برای این کار آدرس خانه های خالی را به صورت پوینتر در آرایه emp ذخیره می کند. سپس یکی از اعداد یک تا تعداد خانه های خالی(empty) را با تابع rand() که در main با srand(time(NULL)) دانه گذاری شده است را انتخاب کرده و با توجه به شروط بازی یکی از خانه های خالی را با عدد ۲ (بزرگترین درایه<۲۵۶) ، ۴ (۲۰۴۸>بزرگترین درایه> ۲۵۶) و ۸(بزرگترین درایه>۲۰۴۸) جایگذاری میکند. مقدار بازگشتی همان empty تعداد خانه های خالی است.

تابع maximum_array()

این تابع، اندازه بزرگترین درایه آرایه دو بعدی خود را باز می گرداند.

تابع main() و startgame()

تابع main بعد از دریافت ابعاد برد و چاپ یک سری از دستورعمل ها ، تابع startgame را اجرا می کند. در این تابع ابتدا بزرگ ترین درایه آرایه در max_array و در pre_max_array ذخیره می شود.

آرایه un که مربوط به یک حرکت قبل است با board مقداردهی اولیه می شود. و امتیازات حرکت کنونی و قبلی نیز مقدار دهی اولیه می شوند. اگر signal برابر ۳ نباشد و بازی جدیدی باشد، یکی از خانه ها به صورت رندوم پر می شود. سپس signal صفر می شود تا بازی به شرایط عادی بازگردد. سپس صفحه بازی و دستورعمل ها چاپ می شود و وارد حلقه بازی می شویم که isnotfin خروجی organize() است تا بر اساس آن بازی ادامه پیدا کند. ابتدا متغیر دستور direction اسکن شده و اگر ‘b’ باشد به این معناست که undo انجام میگیرد. و تمام متغیرها و آرایه های بازی از مقدار قبلی خود پر می شوند و اگر غیر از این باشد، تمام متغیرها و آرایه های بازی کنونی در مقادیر قبلی ذخیره می شوند و با ورود به organize ، آپدیت می شوند.

اگر دستور ’z‘ باشد ، صفحه بازی چاپ نمی شود چون کنترل وارد startgame() دیگری شده و بازی ذخیره شده را انجام داده است.

همین طور وقتی که از حلقه خارج می شویم در صفحه در صورتی که شرط ذکر شده برقرار نباشد، نتیجه بازی در جدول امتیازات ذخیره و گزارش آن داده می شود که شامل امتیاز نهایی و اینکه کاربر برنده شده است یا نه(برای برنده شدن باید بیشترین درایه بازی از ۲۰۴۸ عبور کرده باشد.)

نکته: با استفاده از دستور(HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE) ;)

مربوط به کتابخانه #include <windows.h> می‌توان نوشته ها را رنگی کرد.

مقایسه با سایر برنامه های موجود:

اکثر برنامه های این بازی تنها مربوط به حالت ۴*۴ بود در حالی که کد ما n*m می‌باشد.

عمده کد های مربوط به این بازی که در اینترنت وجود داشتند این مشکل را داشتند که برد بازی با هر حرکت تغییر نمی‌کرد و یک برد جدید باز می‌شد و پس از چند حرکت صفحه بازی شلوغ می‌شد.

هیچ کدام از کد های بازی دارای نوشته های رنگی نبودند و به صورت سیاه و سفید نوشته شده بودند.

همچنین هیچ یک از کد ها ویژگی هایی که در رابطه با این پروژه از ما خواسته شده بود (مثل : بازگشت به حرکت قبلی، دخیره بازی و توانایی ادامه آن، ذخیره امتیازات برتر و ...) را نداشتند.

توسعه بازی:

نکاتی که علاوه بر ویژگی های خواسته شده در کد ما پیاده شده است:

- صفحه بازی الزاما مربعی نیست و می تواند مستطیلی هم باشد یعنی برد بازی ما برخلاف آنچه خواسته شده بود(n*n) n*m می‌باشد.
- نوشته های بازی رنگی می‌باشد.
- پس از پایان بازی به منظور ذخیره شدن از فرد بازی کننده درخواست نام می‌شود تا امتیاز و رتبه او به همراه نام او چاپ شود.
- بازی ما توانایی ذخیره نفرات برتر و امتیازات آن ها را به صورت مجزا برای ابعاد متفاوت دارد (مثلا نفرات برتر برد ۵*۵ جدا از برد ۶*۴ ذخیره می‌شود.
- درصورتی که در ابتدای بازی برد k*l انتخاب کنید ولی در ادامه بخواهید یک بازی که قبلا ذخیره شده ولی ابعاد آن متفاوت است مثلا c*d است را ادامه دهید،صفحه جدید به صورت c*d تبدیل خواهد شد.
- با وارد کردن کلید های اشتباهی بازی به مشکل نمی‌خورد و از شما درخواست وارد کردن کلید های صحیح خواهد کرد.

منابع:

از منبع خاصی استفاده نشده است😅😅😅