



MARIO ESCAPE

DR.FANINAN & DR.DELDAR

ALI MOAZED

MAHROKH MOUSAVI

AMIR MOHAMMAD BARARI

SHAFAGH SEPEHR

YASIN DANESH

DEADLINE: 1403/11/08

مقدمه

سلام به همگی دوستان! رسیدیم به هیجان‌انگیزترین بخش این ترم، یعنی پروژه مبانی! این بار قراره یه قدم فراتر بریم و پروژه‌ای طراحی کنیم که هم خلاقیت شما رو به چالش بکشه و هم در نهایت از بازی کردنش کلی لذت ببرید. هدف پروژه اینه که یه بازی جذاب خلق کنید که روح بازی‌های کلاسیک مثل ماریو رو زنده کنه، اما با امضای خودتون! بازی باید شامل گرافیک ساده، حداقل دو مرحله با سختی تدریجی و یک سیستم امتیازدهی باشد. همچنین یک سری قابلیت در مپ بازی وجود داره که هر کدام ویژگی‌های مخصوص خودشون رو دارن، با جزئیاتش در ادامه آشنا میشید.

صفحه ورود

Sign up

در این صفحه کاربر باید با وارد کردن نام کاربری، رمز عبور و ایمیل، اکانت خود را ایجاد کند. از این پس می‌تواند با استفاده از نام کاربری و رمز عبور، وارد حسابش شود. رمز عبور باید حداقل دارای 8 کاراکتر باشد و برنامه از کاربر بخواهد که یک بار آن را تکرار کند. در صورتی که رمز وارد شده و تکرار آن یکسان نبود، برنامه خطا بدهد.

نام کاربری در بازی منحصر به فرد است، یعنی در هر ثبت‌نام باید چک شود که شخص دیگری با این نام قبلاً ثبت‌نام نکرده باشد و در صورتی که نام کاربری موجود بود، با پیغام خطا کاربر را مطلع سازد.

Sign in

با انتخاب این گزینه کاربر نام کاربری و رمز عبور خود را وارد می‌کند، در صورتی که درست بود، وارد اکانت خود می‌شود. اگر غلط بود، برنامه او را با یک پیغام مطلع سازد.

توجه: رمز ورود هنگام وارد شدن باید به صورت ستاره نمایش داده شود. همچنین در این بخش، کاربر می‌تواند در صورت فراموش کردن رمز عبور خود، با استفاده از گزینه [Forgot password](#) رمز عبور خود را بازیابی کند.

Forgot password : در این قسمت نام کاربری و ایمیل از کاربر دریافت شده و پس از بررسی درستی آن، کاربر می‌تواند رمز جدید خود را وارد کند.

Exit

از برنامه خارج می‌شود.

منوی بازی

تغییر اطلاعات کاربری

در این قسمت کاربر می‌تواند اطلاعات خود شامل نام کاربری، ایمیل و رمز عبور را تغییر دهد. در صورت تغییر نام کاربری، بررسی کنید که این نام کاربری توسط شخص دیگری انتخاب نشده باشد.

برای تغییر رمز عبور باید از کاربر بخواهید یک بار رمز جدید خود را تکرار کند و در صورت مطابقت، رمز آپدیت شود.

تاریخچه

کاربر می‌تواند تعداد برد و باخت و تاریخچه تمام بازی‌های خود را با نمایش امتیازات و سکه‌های کسب شده مشاهده کند. این اطلاعات باید در جدولی به ترتیب اطلاعات بازی‌های گذشته‌اش را نشان دهد. توجه کنید که برای این بخش نیاز به ذخیره سازی در فایل دارید تا در صورت بسته شدن بازی اطلاعات پاک نشوند.

شروع

با انتخاب این گزینه، نقشه بازی آورده شده و بازیکن بازی را آغاز می‌کند.

ذخیره سازی اطلاعات

برنامه‌ی شما باید قابلیت ذخیره اطلاعات تمام افرادی که ثبت نام کرده‌اند را داشته باشد و شما بتوانید پس از بستن کامل بازی، دوباره وارد بازی شوید و به تمام اطلاعات خود از جمله اطلاعات کاربری و تاریخچه بازی‌ها دسترسی داشته باشید.

قابلیت‌های زمین

بلوک تخریب‌ناپذیر: این بلوک نمیتواند توسط ماریو تخریب شود. آن را با  نشان میدهیم.

بلوک تخریب‌پذیر: اگر ماریو با سایز بزرگ بپرد و به آنها برخورد کند، تخریب شده و پرش ماریو متوقف میشود. این بلوک‌ها با احتمال 20 درصد دارای سکه هستند که پس از تخریب به سکه‌های ماریو اضافه میشود. آن را با  نشان میدهیم.

بلوک سکه: این بلوک‌ها دارای یک سکه هستند که ماریو با پریدن و ضربه زدن به زیر بلوک سکه را دریافت میکند، بلوک سکه حداقل 5 سکه خواهد داشت و ماریو برای دریافت هر چند سکه باید به همان تعداد به بلوک ضربه بزند. آن را با  نشان میدهیم.

بلوک قارچ: با ضربه زدن به این بلوک قارچ ظاهر شده و با سرعتی کم به یک جهت رندوم حرکت میکند و به پایین می‌افتد.

توجه کنید که بلوک سکه و قارچ از لحاظ ظاهری عیناً یکسان هستند. () در نقشه به دلخواه بلوک سکه و قارچ را انتخاب کنید.

سکه: سکه‌ها در هوا معلق هستند و ماریو با برخورد به آنها دریافت‌شان میکند. آن را با  نشان میدهیم.

گل‌های درنده: لوله‌ها میتوانند دارای گلی درنده باشند که هر 2 ثانیه از لوله بیرون آمد و در صورت برخورد به ماریو او را می‌درند. این گل‌ها نمیتوانند با پرش ماریو به روی آنها نابود شوند. آن را با  نشان میدهیم. همچنین میتوانید از خلاقیت خودتان استفاده کنید و بالا آمدن گل را به کمک چند کاراکتر شبیه سازی کنید.

لوله: لوله ها با قطر 3 کاراکتر و در سطح لوله با 5 کاراکتر، نشان داده میشوند(با ).

قابلیت جابجا کردن ماریو به سر لوله ای دیگر را دارند. زمانی که ماریو روی کرکتر سوم از سطح قرار میگیرد، جا به جایی انجام میشود. شکل لوله در مپ به این شکل میشود:



بعضی لوله ها نیز قابلیت جابه جایی ندارند و مانند یک بلوک تخریب ناپذیرند. این لوله ها به رنگ سفید میباشند(با همان کاراکتر قبلی ولی رنگ سفید):



در بعضی مواقع تعداد کاراکترهای لوله فرق میکند که در مپ مشخص است.

دشمن عادی: به سمت چپ حرکت کرده و در صورتی که ماریو رویش بپرد از بین میرود و اگر از چپ یا راست با ماریو برخورد کند، ماریو آسیب میبیند. (اندازه یک کاراکتر) آن را با  نشان میدهیم. در صورت برخورد با سایر آیتم ها جهت حرکتش برعکس میشود.

دشمن خاردار: این دشمن ثابت است و برخورد از همه جهات با ماریو به او آسیب میزند. (اندازه یک کاراکتر) آن را با  میدهیم.

قارچ قرمز: این قارچ که با ضربه زدن به بلوک قارچ ظاهر میشود، باعث افزایش اندازه ماریو از 1 کاراکتر به 2 کاراکتر به صورت عمودی میشود. (پس از خوردن آن، دیگر ظاهر نمیشود، مگر اینکه ماریو با مانعی برخورد کرده و کوچک شود، آنگاه باز این قارچ قابل ظاهر شدن است). آن را با  میدهیم.

شیلد: باعث میشود که ماریو آسیب ناپذیر بشود و در صورت برخورد با دشمن یا تله یا تیر دشمن، باعث از بین رفتن آنها میشود ولی پس از برخورد با حتی یک دشمن این قابلیتش را از دست میدهد. آن را با  میدهیم.

نکته: ماریو را با کاراکتر دلخواه خودتان نمایش بدهید. برای مثال میتوانید از استفاده کنید و زمان افزایش اندازه به صورت عمودی کاراکتر را تکرار کنید. ماریو میتواند تا 4 کاراکتر به بالا بپردازد و هنگام پرش به جلو و عقب برود.

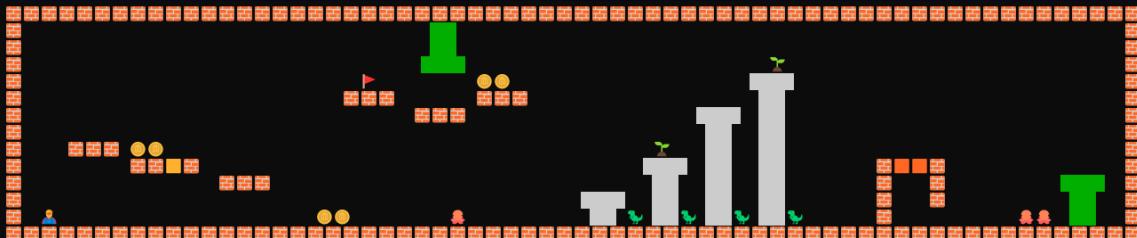
حرکت کردن در زمین بازی

همانطور که انتظار دارید، ماریو با d و a به چپ و راست حرکت میکند و با w یا space به بالا میپردازد. همچنین اگر ماریو اندازه اش بزرگ باشد با s مینشیند ولی در این حالت توانایی حرکت ندارد.

مپ‌های بازی

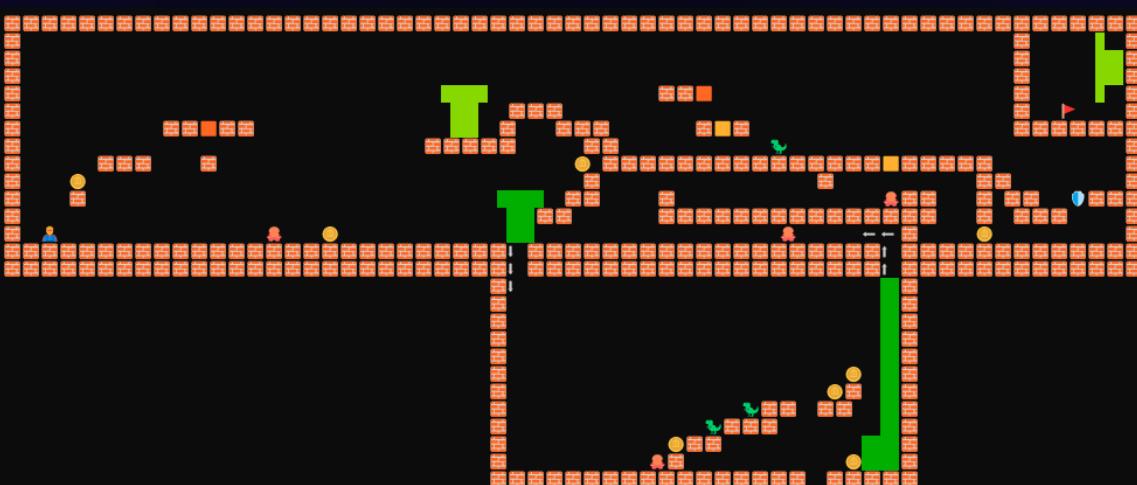
شما در این بازی باید 2 مپ طراحی کنید که تصاویر آن‌ها در ادامه آمده است:

مپ اول:



در این مپ لوله سبز رنگ سمت راست ماریو را به لوله سبز بالا منتقل میکند.

مپ دوم:



در این مپ مسیر انتقال از طریق لوله‌ها با فلش مشخص شده. دو لوله بالایی به رنگ سبز روشن به یکدیگر راه دارند.

این که مپ‌های شما به صورت اسکرولینگ مپ باشند یا به طور کامل در یک صفحه نمایش داده شوند، اختیاری است، اسکرولینگ مپ امتیاز دارد.

در مپ دوم اگر بتوانید قسمت پایین را به حالت مخفی در بیاورید و تنها پس از جابه‌جایی به کمک لوله‌ی مربوطه‌اش نمایش داده شود، نمره امتیازی کسب می‌کنید.

نحوه بازی

در ابتدا پس از ورود به بازی مپ اول نمایش داده می‌شود، پس از رسیدن به پرچم به مپ دوم می‌رویم. با رسیدن به پرچم مپ دوم بازی تمام می‌شود. پس از انتهای هر مرحله (مپ)، صفحه‌ای نمایش داده می‌شود که در آن امتیاز و سکه‌های کسب شده آن مرحله مشخص است. در اینجا امکان تکرار دوباره‌ی این مرحله یا ادامه دادن وجود دارد. پس از انتهای هر 2 مپ، تاریخچه مربوط به دو مرحله به history اضافه می‌شود. در صورت باخت در هر یک از مراحل آن مرحله از ابتدا تکرار می‌شود ولی باید توجه کرد که تعداد باخت‌ها یا تکرار دوباره‌ی مراحل نیز باید در تاریخچه نشان داده شود.

پایان بازی و امتیازدهی

ماریو در صورت نابود کردن هر جسم آسیب زننده 100 امتیاز می‌گیرد. اگر آنها را با فاصله کمتر از 5 ثانیه نابود کند، امتیاز دریافتی با هر نابودی دو برابر می‌شود. (برای مثال ماریو 3 دشمن ساده را پشت سر هم نابود می‌کند و به ترتیب 100، 200 و 400 امتیاز می‌گیرد) امتیاز با سکه متفاوت است.

در صورت رسیدن ماریو به پرچم باوزر (دشمن ماریو که عکسشم پایین هست :)) پرچم پایین آمد و آن مرحله به اتمام میرسد. پرچم را با  نشان میدهیم.



زمان باقی مانده به همراه جان های باقی مانده به امتیاز تبدیل میشوند. (هر 10 ثانیه 100 امتیاز و هر جان 1000 امتیاز دارد)

سپس امتیاز و سکه این مرحله در تاریخچه ذخیره میشود.

نکته مهم: لازم نیست میپهاتون دقیقاً مثل نمونههای بالا باشه، اما باید یه شباهت نسبی داشته باشه. اگه نمیتوانید ایموجیها رو نشون بدید، میتوانید از کاراکترهای دیگه استفاده کنید، فقط حواستون باشه که پروژهتون یه حداقل زیبایی بصری داشته باشه. حتماً به راهنمایها هم توجه کنید!

موارد امتیازی

اگر تعدادی از موارد زیر را به درستی انجام بدید، نمره امتیازی را به طور کامل دریافت میکنید:

1- استفاده از گیت و گیتهاپ از ابتدا تا انتهای پیاده سازی پروژه همراه با کامیت های متعدد و با معنی.

یک فایل readme مناسب نیز اضافه کنید) کوتاه و مختصر و در عین حال (کامل)

2- ظاهر گرافیکی زیبا و جذاب

3- صدا گذاری بازی

4- راکت لانچر: یک جسم ثابت بوده که در جهت ماریو روی یک خط افقی شلیک میکند و راکت با سرعت کم حرکت میکند و تنها بعد از بین رفتنش، راکت بعدی را شلیک میکند. راکت عرض سه کاراکتر و ارتفاع یک کاراکتر را داراست و در صورت پریدن ماریو به روی آن از بین رفته و به ماریو امتیاز مبدهد (ماریو مانند پرش عادی به بالا پرت میشود). آن را با [=] نشان میدهیم.

5- طراحی میپ سوم

6- اسکرولینگ میپ

7- مخفی کردن قسمت پایین در میپ 2 (توضیحات بالاتر داده شده)

8- امتیازی آزاد: هرچیز پیچیده ای که ارزش پروژه را بالا ببرد میتواند به اختیار خودتان پیاده شود و مقدار محدودی امتیاز دریافت کند.

راهنمایی‌ها

چطوری در کنسول ایموجی نمایش بدیم؟

نمایش ایموجی در کنسول به عوامل مختلفی بستگی دارد. یکی از مهمترین عوامل فونت کنسوله، یعنی فونت باید از کاراکترهای یونیکد و ایموجی‌ها پشتیبانی کنے. عامل بعدی سیستم‌عامله که باید به درستی رندر کاراکترهای ایموجی رو پشتیبانی کنے. در نهایت، نرم‌افزاری که کنسول رو اجرا می‌کنه هم نقش مهمی دارد.

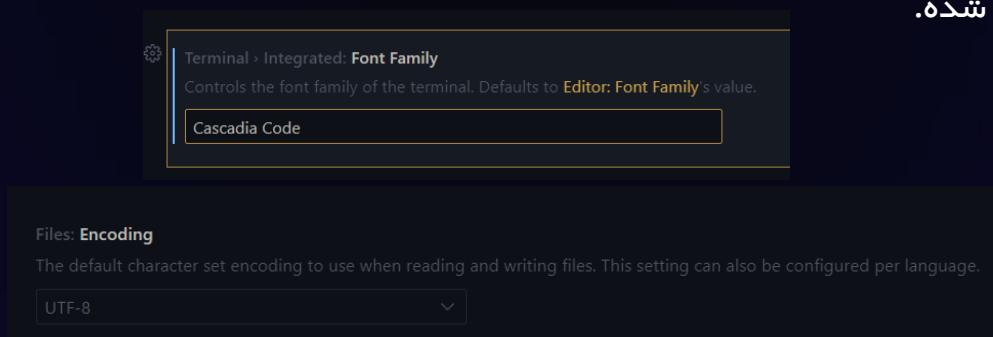
:VSCode (Visual Studio Code)

تو و VSCode، اگر از ترمینال داخلیش استفاده کنی و یک فونت مناسب مثل Fira Code یا Cascadia Code انتخاب کنی، ایموجی‌ها نمایش داده می‌شن.

چطوری فعالش کنیم:
به تنظیمات برو.

تو بخش Cascadia Code Terminal > Integrated > Font Family اسم فونت مثل Cascadia Code را وارد کن.

اگه فونت مناسب نباشه، ایموجی‌ها ممکنه به صورت مربع یا علامت سؤال دیده بشن. مطمئن شو که زبان پیش‌فرض UTF-8 هست. در قسمت جستجوی تنظیمات، عبارت Files: Encoding را تایپ کنید. مطمئن بشید که گزینه‌ی encoding روی UTF-8 تنظیم شده.



حال بریم سراغ افرادی که از IDE‌هایی غیر از vscode استفاده می‌کنن ☺ ویندوز 10 و 11:

ویندوز 10 و 11 از ایموجی‌ها پشتیبانی می‌کنن، اما بستگی داره از کدام کنسول استفاده کنی:

Windows Terminal بهترین انتخابه. این برنامه روی هر دو ویندوز 10 و 11 قابل نصب هست، ولی تو ویندوز 11 به صورت پیش‌فرض نصب شده فقط کافیه به تنظیمات ترمینال ببرید و اون رو روی windows terminal قرار بدید:



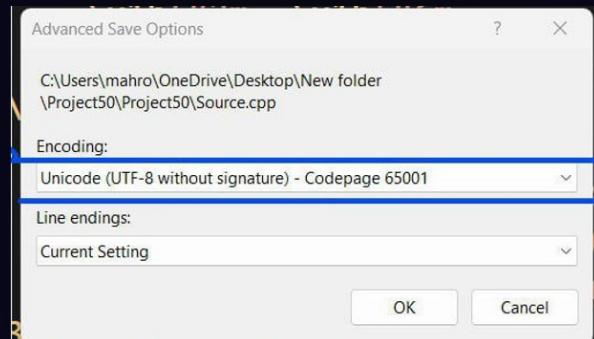
CMD هنوز از یونیکد و ایموجی‌ها به صورت کامل پشتیبانی نمی‌کنه. حتی با تغییر کدپیچ به UTF-8 (chcp 65001)، باز هم مشکلاتی در نمایش ایموجی‌ها وجود دارد. ایموجی‌ها در CMD معمولاً به صورت مربع یا علامت سؤال دیده می‌شون. windows پس پیشنهاد ما بهتون اینه که اگر ویندوز 10 دارید و به طور پیش فرض terminal روی سیستمتوں ندارید، اون رو نصبش کنید (〇). نصبش هم خیلی راحته، میتوانید ازین [لينك](#) استفاده کنید.

:Visual Studio

خب تا به اینجای کار پیش او مدیم ولی ویژوال استودیو هنوزم ایموجی‌هارو با؟ نشون میده. راه حل سادست. اول از همه که windows.h رو اینکلود کن. بعد کافیه این خط رو به کدت اضافه کنی:

SetConsoleOutputCP(CP_UTF8);

در نهایت وقتی کد رو ران میکنی ازت یه سوال میپرسه:



باید encoding رو به چیزی که بالا نشون داده شده تغییر بدی، اون وقت میتوانی به راحتی هر ایموجی ای که میخوای رو نمایش بدی: این هم یه نمونه:

```
#include<iostream>
#include<windows.h>
int main() {
    SetConsoleOutputCP(CP_UTF8);
    std::cout << "\x1b[49;36m " << "\x1b";
}
```

لینوکس:

ترمینال‌های مدرن لینوکس مثل konsole و gnome-terminal از ایموجی‌ها پشتیبانی می‌کنند، البته اگه:

از فونت‌های مناسب مثل DejaVu Sans Mono یا Noto Color Emoji استفاده کنی و یونیکد در تنظیمات سیستم فعال باشه. چطوری فعالش کنیم؟ فونت مناسب رو نصب کن. مثلاً تو اوبوتو با این دستور:

```
sudo apt install fonts-noto-color-emoji
```

بعد تو تنظیمات ترمینال، فونت رو تغییر بد.

خب این از این، اگه تو کتگوری‌های بالا هم قرار نمی‌گرفتید که خودتون چاره بیندیشین: همونطور هم که قبل تر گفته شده اجباری تو استفاده از ایموجی‌های گفته شده نیست و می‌توانید از جایگزین استفاده کنید برای مثال کاراکترهای ANSI که برای نمایششون تو CMD به مشکل نمی‌خورید.

چطوری کاراکترهارو رنگی کنیم؟

اون ANSI بود الان گفتیم، اینجا هم به کار می‌آید. اصلاً ANSI چیه؟ یه استاندارده که برای یکپارچه‌سازی نمایش متن و گرافیک در محیط‌های متنی مثل کنسول‌ها استفاده می‌شده. این استاندارد کدهایی برای رنگی کردن متن، تغییر مکان cursor و نمایش کاراکترهای خاص ارائه می‌دهد. حالا چطوری ازش می‌شده و اسه رنگی کردن استفاده کرد؟ کافیه قبل و بعد اون قسمتی که قراره رنگی بشه از کدهای ansi استفاده کنین. یکم برباد بالاتر یه مثال ازش تو یه عکس بود 😊. جزئیات بیشتر هم می‌توانید از این [لينك](#) مطالعه کنین.

چطوری مپ رو تعریف کنیم؟

منطقاً برای چاپ زمین به یک آرایه‌ی دو بعدی نیاز داریم. اگر ایموجی‌ها رو مستقیماً تو آرایه بذاری، ممکنه کار با مپ سخت‌تر بشه. دلیلش اینه که اندازه ایموجی‌ها در مقایسه با کاراکترهای معمولی یکسان نیست، چون ایموجی‌ها در یونیکد می‌توانند از چند بایت تشکیل بشن و این می‌شده که محاسبه مختصات برای ساخت می‌شده. پس می‌توانی از کاراکترهای جایگزین استفاده کنی و موقع چاپ به جای اونها ایموجی مربوطه رو چاپ کنی. البته این صرفاً یه راهکار بود: میدونیم که به تعداد آدم راه هست و اسه رسیدن به خدا، پس خلاق باشید. احتمالاً به راههای بھتری هم برسید.

چجوری در کنسول حرکت کنیم؟

در این بخش یک راهنمایی برای چگونگی حرکت ماریو در بازی برای شما قرار گرفته شده است که بد نیست از آن راهنمایی بگیرید. در ادامه برخی از جزئیات آن برای شما تعریف شده است و شما می‌توانید با جست و جو اطلاعات بیشتری درباره آن کسب کنید.

COORD

این یک ساختار در ویندوز است که برای نشان دادن یک نقطه در کنسول با استفاده از مختصات X (افقی) و Y (عمودی) استفاده می‌شود.

```
COORD coordination;  
coordination.X = 5;  
coordination.Y = 10;
```

GetAsyncKeyState

این تابع وضعیت بالا یا پایین بودن یک کلید خاص را روی صفحه کلید در زمان فرآخوانی بررسی می‌کند. این تابع عموماً در برنامه‌هایی استفاده می‌شود که به بررسی واکنش‌های آنی به فشردن کلیدها نیاز دارند، مانند بازی‌ها.

```
SHORT retval = GetAsyncKeyState(int key);
```

آرگومان ورودی این تابع کلید مدنظر برای بررسی و MSB خروجی این تابع نشان‌دهنده بالا یا پایین بودن کلید مدنظر(ورودی تابع) می‌باشد.

می‌توانید برای بررسی بیشتر به این [لينک](#) مراجعه کنید.

GetStdHandle

ازین تابع می‌توانید برای هندل کردن stdin, stdout, stderr استفاده کنید. خروجی این تابع Handle ای به Standard device مدنظر است.

```
HANDLE hout = GetStdHandle(STD_OUTPUT_HANDLE);
```

ورودی این تابع Standard device(stderr or stdin or stdout) مدنظر است.

برای اطلاعات بیشتر باید سرچ کنید.

WriteConsoleOutputCharacterA

با استفاده از این تابع می توانید کاراکترهای متنی را مستقیماً و بدون نیاز به تابع در جای مشخصی روی صفحه کنسول بنویسید.

```
WriteConsoleOutputCharacterA(hout, map[y], 40, place, &bytes_written);
```

همانطور که متوجه شدید در این تابع از Handle ای که از تابع قبل گرفتیم کمک می‌گیریم و کرکترهای دلخواه و تعداد آنها و مکانی که می‌خواهیم و یک آدرس برای ذخیر تعداد کرکترهای نوشته شده به این تابع پاس داده و این تابع این کرکترهارا در place می‌نویسد.

نکته مهم :

اصراری بر استفاده ازین توابع در کد نیست صرفاً یک راه برای رسیدن به هدف استفاده ازین توابعه. پس برای پیدا کردن راههای دیگه و شاید ساده‌تر از استفاده ازین توابع حتماً حتماً سرچ کنید.

Threads

یکی دیگه از چیزایی که می‌توانید ازش توانی این برنامه استفاده کنید، تردد.

استفاده از برنامه سازی همروند (thread)

برنامه سازی همروند یا همزمان به معنای اجرای برنامه هاست به گونه ای که تعدادی از عملیات به طور همزمان اجرا شوند. اگر قسمت هایی از برنامه به صورت همزمان اجرا شوند، زمان پاسخ (یا تأخیر) و توان عملیاتی برنامه بهبود می یابد. در کد نویسی که تا کنون انجام میدادید، برنامه فقط یک رشته داشت و همزمان یک بخش از کد اجرا میشد، اما با برنامه سازی همروند (استفاده از thread) (میتوان همزمان بیش از یک بخش از کد را اجرا کرد، مثلًا بخش محاسبات برنامه، بخش ورودی گرفتن و بخش نمایش برنامه جدا باشند. مدیریت thread کار نسبتاً سختی است پس تا میتوانید تعداد کمتری از آن را استفاده کنید، حتی می توانید اصلاً از آن استفاده نکنید! فایل های کمکی برای کار با thread بهتون ارائه شده.

این خط تابعی که ورودی و خروجی اش void است را به صورت همرون در یک thread با نام thread1 اجرا میکند و برنامه مستقیم به خط بعد میرود حتی اگر تابع تمام نشده باشد. (مثلا در تابع حلقه‌ی بی‌نهایت زده باشید.)

```
HANDLE thread1 = start_listening(my_thread_function);
```

این خط صبر میکند تا 1 تا تمام شده و سپس ادامه میدهد، میتوانید به جای زمان را به میلی ثانیه وارد کنید و بعد از آن زمان thread به اجبار پایان می‌یابد؛ بهتر است تابع thread تان پایان نیابد که مجبور به فراخوانی مکرر آن نباشید، به طور مثال کد آن در حلقه‌ی بی‌نهایت اجرا شود.

```
WaitForSingleObject(thread1, INFINITE);
```

نمونه کد:

```
#include <stdio.h>
#include "helper_windows.h"

int i = 0;
void my_thread_function() {

    while (1) {
        system("cls");
        printf("%d\n", i);
        Sleep(100);
    }
}

int main() {
    HANDLE thread1 = start_listening(my_thread_function);
    while (1) {
        char ch = _getch();
        if (ch == 'q')
            break;
        i++;
    }
    return 0;
    //WaitForSingleObject(thread1, INFINITE);
}
```

توابع کمکی:

_kbhit(void) : این تابع در کتابخانه‌ی conio.h قرار دارد. ورودی ای نمیگیرد، اگر کلیدی در کیبورد فشرده شده باشد یک خروجی میدهد، در غیر اینصورت خروجی آن صفر است. تا وقتی آن کاراکتر از کیبورد خوانده نشود، مقدار یک خروجی داده میشود وقته مقدار وارد شده مثلا با تابع _getch(void) خوانده شود و بافر ورودی خالی شود، مقدار صفر خروجی داده می‌شود. (مناسب برای جدا کردن بخش خواندن از ورودی و محاسبات)

نمونه کد:

```
#include <stdio.h>
#include <conio.h>
int main() {
    while (1) {
        if (_kbhit()) {
            printf("data read\n");
            char ch = _getch();
            //do stuff with ch
        }
        // run part of the game that is indepedend of input
    }
    return 0;
}
```

چطوری اسکرولینگ مپ رو پیاده سازی کنم؟

اصلًا منظور از اسکرولینگ چیه؟ یعنی اینکه مپ تون به صورت کامل نمایش داده نشه توی صفحه ترمینال تون و وقتی ماریو به سمت جلو یا عقب حرکت میکنه بر اساس موقعیتش بخش جدیدی از مپ نمایش داده بشه. در این خصوص باید بگیم که خیلی کار چالش بر انگیزی نیست! (برای این کار شما باید عرض ترمینالتون رو به یه سایز مشخصی تغییر بدین که مپ تون فقط به اندازه عرض ترمینال معلوم باشه. برای پیاده سازی این بخش شاید پیش خودتون فکر کنید که اگه یه کدی باشه که بشه باهاش موقعیت اسکرول رو تغییر داد (سرچ کنید پیدا میشه) میتوانید وقتی ماریو به نزدیکای آخرای صفحه رسید صفحه رو اسکرول کنید جلو... ولی شاید بشه یه جوره دیگه ای هم پیاده سازیش کرد که نیازی به استفاده از کد آماده نباشه.

فرض کنید صفحه ترمینالتون رو اسکرول کردین رفتهین یه جایی که کرسر (cursor) دیگه توی دید نیست، حالا اگه بیایید یه چیزی تایپ کنید صفحه ترمینال تون اسکرول میشه به جایی که کرسر الان هست پس هروقت کرسر جا به جا بشه صفحه ترمینال به صورت خودکار اسکرول میشه به سمت موقعیت کرسر. فرض کنید ماریو رسیده به نزدیکای آخر صفحه و شما باید صفحه رو اسکرول کنید به سمت جلو با توجه به این توضیحات شما باید با کرسر چیکار کنید که صفحه به جلو اسکرول شه؟ پس میتوانید این رو از همون اول در نظرش داشته باشید تا به راحتی نمره امتیازی کسب کنین:)

البته که این فقط یه راهنماییه، ممکنه شما ایدههای خلاقانهتری پیدا کنید، پس خودتون رو محدود نکنید!

پرش ماریو به چه صورت‌ه؟

فرض کنید ماریو سر جای خودش ایستاده و کاربر اسپیس رو فشار میده، با فشار دادن اسپیس ماریو باید با یه سرعت معقولی و به اندازه تعداد کاراکتر مشخصی به سمت بالا حرکت کنه و دوباره به همون تعداد کاراکتری که بالا رفته برگرده به پایین. ممکنه کاربر موقع پریدن ماریو کلید های چپ و راست رو هم فشار بده و ماریو باید بتونه توی هوا به چپ و راست حرکت کنه.

چیزهای دیگه ای هم که خوبه برای پیاده سازی پرش بدونین اینه که حواستون به مانع هایی که بالای سر ماریو هست باشه و اینکه ماریو نمیتونه وقتی روی زمین نیست بپره. بقیش با خودتون 

راهنمایی بیشتر:

به ویدیویی که از دموی بازی بر اتون [لينک](#) شده خوب دقت کنید؛) طبیعتاً قرار نیست خروجی بازی شما به این شکل ابتدایی باشه، بلکه این فقط یه دمو مختصر برای نشون دادن نحوه پریدن ماریو و اسکرول شدنه.



سخن آخر

قبل از هرچیز، پیشنهاد ما اینه که لپتاپ‌هاتون رو خاموش کنید، کاغذ و قلم بردارید و کمی وقت بذارید فکر کنید که قراره برنامتون رو چطور بنویسید. چه بخش‌هایی نیاز دارید؟ ساختار کلی برنامه‌تون رو چطور می‌خواهید بچینید؟ اصلاً نگران قسمت‌هایی که فکر می‌کنید سخت هستن نباشد، موقتاً ازشون بگذرید. وقتی شروع کنید به نوشتن کد، می‌بینید که اون قسمت‌ها هم به مرور برآتون راحت‌تر می‌شن. مطمئن باشید اون قسمت‌ها هم برآتون آسون می‌شه و برمی‌گردید روشون فقط کافیه قدم به قدم پیش برید و اجازه بدید که تجربه بهتون راه حل‌ها رو نشون بد.

وقتی ساختار کلی برنامتون رو مشخص کردید و شروع به نوشتن کد کردید، یادتون باشه که زیاد بودن خطوط کد همیشه به معنی بهتر بودن برنامه نیست! برنامه خوب برنامه‌ایه که هم درست کار کنه، هم تمیز، خوانا و مرتب نوشته بشه. پس حتماً به کامنت‌گذاری توی کدها توجه کنید و برای متغیرها و توابع از اسم‌های معنادار استفاده کنید. اینجوری هم خودتون راحت‌تر می‌فهمید چی به چی هست، هم کسایی که کد رو می‌بینن، می‌تونن راحت‌تر متوجه بشن.

یکی از چیزهایی که توی این پروژه تمرین می‌کنید، طراحی توابع خوبه. هر تابع باید یک کار مشخص انجام بده و با بقیه توابع خوب ارتباط برقرار کنه. اینجوری برنامتون هم زیباتر می‌شه، هم خودتون راحت‌تر می‌توانید روش کار کنید. سعی کنید تابع main رو کوتاه و مختصر بنویسید و هر بخش از برنامه رو به یه تابع اختصاص بدهید. به این پروژه فقط به عنوان یک تکلیف نگاه نکنید. این یه فرصت خوبه که از کدنویسی لذت ببرید و خیلی بیشتر از چیزی که فکر می‌کنید یاد بگیرید. مطمئن باشید که این پروژه یکی از بهترین خاطرات ترمنتون می‌شه! پس با انگیزه و انرژی جلو ببرید.

یه نکته مهم دیگه هم اینه که باید حتماً به ذخیره‌سازی داده‌ها فکر کنید. برنامه باید طوری باشه که وقتی بستیدش و دوباره بازش کردید، اطلاعات قبلی از دست نرن. بنابراین از همون اول به ذخیره‌سازی فایل‌ها توجه کنید.

در نهایت، یادتون باشه که این پروژه یه مسیر یادگیریه. با دقت و انگیزه کد بنویسید و از این تجربه لذت ببرید، چون قطعاً تیجه‌اش خیلی بیشتر از اون چیزی که فکر می‌کنید می‌ارزه.

مثل قارچ باش، هر جا رشد کن!

