
یکم راهنمایی:)


thread

# 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 تاخیر) 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، مثلا بخش محاسبات برنامه، بخش ورودی گرفتن و بخش نمایش برنامه جدا باشند.

فایل های کمکی برای کار با thread بهتون ارائه شده،

مدیریت thread کار نسبتا سختی است پس تا میتوانید تعداد

کمتری از آن را استفاده کنید، حتی می توانید اصلا از آن

استفاده نکنید.

در فایل کمکی توضیحات مربوط به استفاده از thread آمده است، فایل

کمکی را کنار کد اصلی‌تان قرار دهید و آن را در کد خودتان include

کنید(به جای <> از "" استفاده کنید).

این خط تابعتان که ورودی و خروجی اش void است را به صورت همروند 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 thread1 تمام شده و سپس ادامه میدهد، میتوانید به جای INFINITE زمان را به میلی ثانیه وارد کنید و بعد از آن زمان thread به اجبار پایان می یابد؛ بهتر است تابع thread تان پایان نیابد که مجبور به فراخوانی مکرر آن نباشید، به‌طور مثال کد آن در حلقه‌ی بی‌نهایت اجرا شود.

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

# نمونه کد:

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

int i = 0;
void my_thread_function() {

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

```c
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 کتابخانه ی 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 وقتی 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:

```c
#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 independed of input
    }
    return 0;
}
```

# راهنمای حرکت

## نمونه کد کمکی ویندوز

## نمونه کد کمکی لینوکس

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

در اینجا، یک متغیر از نوع ساختار COORD به نام coordination ایجاد می‌کنید و مختصات آن را به X=10 و Y=5 تنظیم می‌کنید.

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

# GetAsyncKeyState

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

مقدار عددی کد کلید را مشخص می‌کند که وضعیتش قرار است بررسی شود.

مقدار برگشتی این تابع که اگر بیت بالایی آن ۱ باشد، نشان‌دهنده‌ی این است که کلید مورد نظر فشرده شده است.

```
SHORT retVal = GetAsyncKeyState(int vKey);
```

# GetStdHandle

این تابع هندل پیش‌فرض ورودی، خروجی یا خطای استاندارد مربوط به کنسول را برمی‌گرداند که می‌توانید از آن برای نوشتن یا خواندن اطلاعات از کنسول استفاده کنید.

ثابتی است که برای دریافت هندل مرتبط با خروجی استاندارد (معمولاً صفحه کنسول) استفاده می‌شود.

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

# WriteConsoleOutputCharacterA

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

همانطور که در مثال می‌بینید، شما باید هندل کنسول، آرایه کاراکتر مورد نظر برای نوشتن، تعداد کاراکترهایی که قرار است نوشته شوند، مختصات شروع نوشتن و یک آدرس برای ذخیره تعداد واقعی کاراکترهای نوشته شده ارائه دهید.

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

در استفاده از این توابع اجباری وجود ندارد.
قطعا راه های دیگری برای رسیدن به این مقصود ها وجود دارد که می‌توانید با سرچ کردن به آن ها برسید.