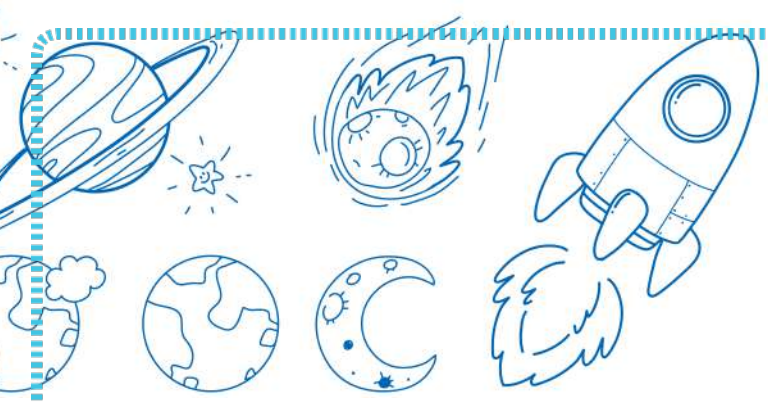
دانشگاه بوعلی سینا

مینی پروژه

starwars

مبانی کامپیوتر و برنامه‌نویسی | پاییز ۱۴۰۲

استاد: دکتر حسن بشیری
طرح و تهیه: تیم طراحی پروژه
مهلت تحویل: ۲۳ دی ساعت ۲۳:۵۹

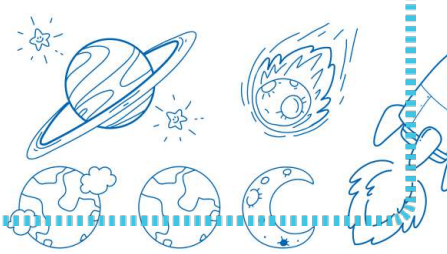


جنگ ستارگان چیه؟

سلام! به بازه‌ی طلایی پیاده‌سازی تمام اطلاعاتی که تا اینجا یادگرفتید، خوش اومدید!
توی سه هفته‌ی پیش رو، قرار بر اینه که هرچیزی که تا الان از کدزنی یادگرفتیم رو مروری کنیم و باهاش یه بازی بنویسیم! داستان از این قراره که ما یه سفینه‌ی سردرگم داریم و یه سری دشمن... باید بتونیم با استفاده از حرکات مختلف، شلیک و فرار کردن و هرکاری که مجوز انجامش تو برنامه بهتون داده شده، این دشمنای فضایی رو بکشید و با سفینه‌تون جون سالم از این نقشه‌ی مربعی به در ببرید!
هدف از این بازی، علاوه بر نجات دادن سفینه‌ی فضایی گیر افتاده توی نقشه‌ی مربعی، یاد گرفتن کارگروهی و کنار اومدن و کد زنی با هم‌گروهی‌هاتونه! شما باید با تشکیل تیم‌های دو نفره، انتخاب کنید که مورد اعتمادترین فرد برای هم‌گروهی شدن با شما و نجات سفینه، کیه؟!(اصلا آیا خودتون هم فرد مورداعتمادی هستید؟!)
دقت کنید که توی این بازی، هر دو نفر باید بتونن طی مسیر کدزنی و تعبیه کردن ابزارهایی برای نجات سفینه، با هم هماهنگ باشن و به خوبی بدونن که چه کسی، کدوم راه و وسیله‌ی نجات رو طراحی کرده! از طرف دیگه، لطفا دقت کنید که تقسیم‌بندی کد زنی و طراحی نقشه و ابزارها، به درستی و عدالت بین‌تون اتفاق بیفته!
یادتون نره که باید از هر چیزی که تا اینجا یادگرفتید، کمک بگیرید که بتونید خیلی ساده و سریع‌تر فرار کنید؛ پس تلاش کنید که توابع و کارکردهایی که حتی از منطق یکسانی برخوردار هستن ولی می‌شه با سینتکس‌های متفاوتی پیاده‌سازی‌شون کرد رو، هرجایی با روش‌های متفاوت پیاده کنید(مثل پیاده‌سازی حلقه‌ها یا شرط‌ها)، این عمل نقش به‌سزایی در تثبیت آموخته‌های مختلف شما داره و یکی از بزرگ‌ترین هدف‌های طرح و پیاده‌سازی مینی‌پروژ رو شامل می‌شه.
در آخر باید اضافه کنیم که اگه دوست داشتید یا به نمره‌ی بیشتری نیاز داشتید، از پیاده‌سازی ویژگی‌های امتیازی بازی، غافل نشید.

تلاش شده که حتی ویژگی‌های امتیازی بازی هم به گونه‌ای انتخاب بشن که شما با بسترهای مهم و کارآمد برنامه‌نویسی، بیشتر و هدفمندتر آشنا بشید.

تیم طراحی پروژه هرجایی از مسیر نوشتن بازی شما، آماده ست که بهتون کمک کنه و کاری کنه که بازی جذاب‌تری داشته باشید؛ اما سرچ کردن اولویت بیشتری نسبت به سوال پرسیدن از تیم طراحی پروژه داره و شما رو برای ترم‌های بعدی‌تون آماده‌تر می‌کنه. خلاصه که... بازی‌سازی خوبی رو براتون آرزومندیم ؛)



شرح مینی پروژه:

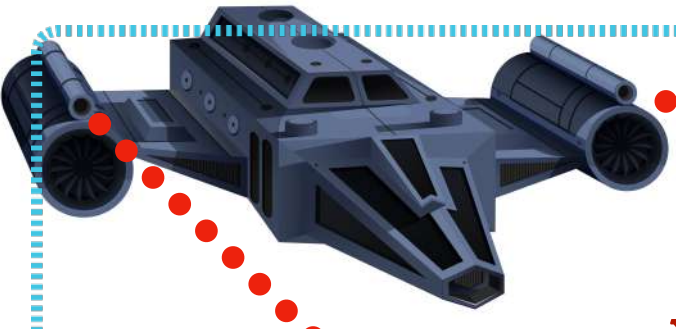
مقدمه

در این بازی قراره که شما یک سفینه رو هدایت کنین تا بدون اینکه آسیبی ببینه دشمنان کهکشان رو نابود کنه. تو این پروژه قراره که از موارد مختلفی توی برنامه نویسی استفاده کنین . سفینه به کمک شما نیاز داره تا بتونه از این مرحله عبور کنه . پس دانش برنامه نویسی تون رو افزایش بدین تا راحت تر بتونین به سفینه کمک کنین . قراره که دستورات و الگوریتم های مختلفی رو طی روند این مینی پروژه یاد بگیرین . پس بیاین که خودمون رو به چالش بکشیم!!!!

```cpp
#include <iostream>
// تعریف توابع
void initializeGame();
void updateGame();
void renderGame();
void handleInput(char input);
void gameOver();
// تابع اصلی برنامه
int main() {
    run();
    return 0;
}
// تابع اجرای بازی
void run() {
    initializeGame();
    char userInput;
    bool isGameRunning = true;
    while (isGameRunning) {
        renderGame();
        std::cout << "Enter your move (w/a/s/d): ";
        std::cin >> userInput;
        handleInput(userInput);
        updateGame();
        // اگر شرایط بازی به پایان رسیده باشد
        if (/* شرایط پایان بازی */) {
            isGameRunning = false;
            gameOver();
        }
    }
}
// تابع مقداتی بازی
void initializeGame() {
    // کد مربوط به شروع بازی
}
// تابع بروزرسانی وضعیت بازی
void updateGame() {
    // کد مربوط به بروزرسانی بازی
}
// تابع نمایش وضعیت فعلی بازی
void renderGame() {
    // کد مربوط به نمایش بازی
}
// تابع مدیریت ورودی کاربر
void handleInput(char input) {
    // کد مربوط به مدیریت ورودی
}
// تابع پایان بازی
void gameOver() {
    // کد مربوط به پایان بازی
}
```

ساختار پروژه

برنامه به صورت ماژولار و از تعدادی تابع جداگانه استفاده می‌کند تا بهبود خوانایی و قابلیت توسعه داشته باشد. یک تابع و یک تابع اصلی وجود دارد.



توضیحات ورودی و خروجی توابع

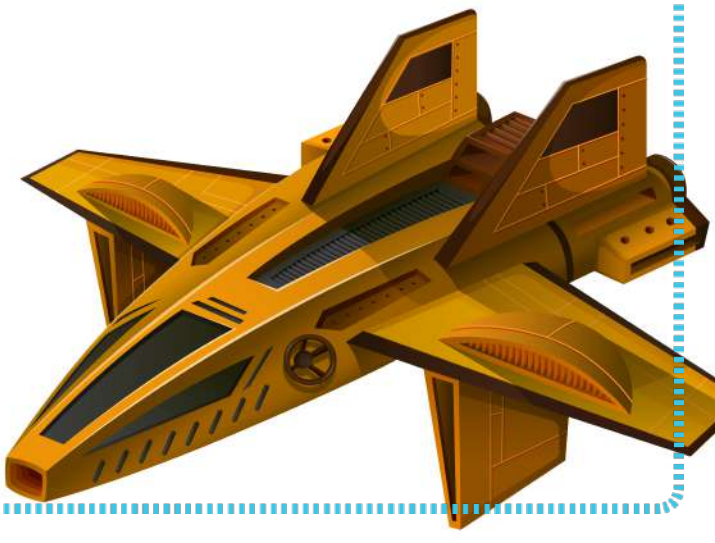
1. initializeGame: این تابع مسئول شروع بازی و مقدمات آن است. هیچ ورودی ندارد و خروجی ندارد.
2. updateGame: این تابع مسئول بروزرسانی وضعیت بازی است. هیچ ورودی ندارد و خروجی ندارد.
3. renderGame: این تابع مسئول نمایش وضعیت فعلی بازی به کاربر است. هیچ ورودی ندارد و خروجی ندارد.
4. handleInput(char input): این تابع ورودی کاربر را مدیریت کرده و بر اساس آن حرکت‌های لازم را انجام می‌دهد. ورودی آن یک کاراکتر است و خروجی ندارد.
5. gameOver: این تابع در صورت پایان بازی فراخوانی می‌شود و نتیجه نهایی بازی را به کاربر نمایش می‌دهد. هیچ ورودی ندارد و خروجی ندارد.

قوانین بازی

قوانین داخل بازی به عنوان دستورالعملی عمل می کنن که بر بازی حاکمن و رقابت منصفانه و سازمان یافته رو تضمین می کنن. این قوانین اهداف، اقدامات و محدودیت‌های بازیکنان و همچنین مجازات‌های تخلفات را مشخص می‌کنن. اونها ساختار رو ارائه میدن. استراتژی و تجربه ای ثابت رو برای همه بازیکنان ارائه میدن . قوانین واضح و به خوبی تعریف شدن که برای گیم پلی بازی بسیار مهم هستن.

- **Map** : مپ باید مربعی (Square) باشه . به عنوان مثال اگه سایز مپ (map) رو 10 در نظر بگیریم , یه ماتریس 2 بعدی در نظر میگیریم که 10 تا سطر و 10 تا ستون داره .
- **Matrix** : ماتریس دو بعدی چیه ؟ آرایه دو بعدی به عنوان ماتریس (جدولی از سطرها و ستون ها) نیز شناخته می شود. شکل رو به رو یه ماتریس 5*5 که تو هر خونه ی اون index مربوطه هم نوشته شده.

[0][0]	[0][1]	[0][2]	[0][3]	[0][4]
[1][0]	[1][1]	[1][2]	[1][3]	[1][4]
[2][0]	[2][1]	[2][2]	[2][3]	[2][4]
[3][0]	[3][1]	[3][2]	[3][3]	[3][4]
[4][0]	[4][1]	[4][2]	[4][3]	[4][4]



- **Spaceship** : فقط یه سفینه خودی وجود داره که توسط کاربر حرکت داده میشه و باید تو یه موقعیت از مپ به صورت random قرار بگیره . بقیه سفینه ها دشمن (Enemy) حساب میشن . همچنین برای جایگذاری سفینه در map باید حواستون باشه که اون خونه خالی باشه . سفینه دشمن یا .... تو اون index نباشه . سفینه خودی 3 تا heal داره که در صورت برخورد با سفینه های دشمن از heal سفینه ما یکی کم میشه و سفینه به موقعیت قبلی معتبر خودش برمیگرده .

- **Enemy Spaceship** : تعدادی سفینه دشمن تو مپ وجود داره که موقعیت این سفینه ها هم به صورت تصادفی (random) انتخاب میشه . تعداد سفینه ها حداقل باید به اندازه size مپ باشه (اگه سایز مپ برابر 10 باشه , حداقل باید 10 سفینه دشمن در مپ ایجاد بشه البته به صورت موقعیت های تصادفی) . در نظر داشته باشین که تو یه سطر حداکثر به اندازه size – 1 میتونیم سفینه داشته باشیم (به عنوان مثال اگه size برابر 10 بود , تو یه سطر نهایت میتونیم 9 سفینه دشمن داشته باشیم .

- **Move** : سفینه ای که توسط کاربر کنترل میشه میتونه به چهار جهت حرکت کنه (بالا , پایین , چپ و راست) . در نظر داشته باشین که سفینه نباید از مپ خارج بشه و در هر مرحله فقط یه حرکت قابل انجامه . به عنوان مثال اگه W کلید بالا و S کلید پایین باشه . کاربر در هر مرحله باید بتونه یکی از کلید های W یا S رو انتخاب کنه و نمیتونه همزمان هردوتا کلید رو وارد کنه . اگه کلیدی غیر کلید های تعریف شده زده شد باید خطای مناسب نمایش داده بشه .

- **Shot** : سفینه ما میتونه به چپ و راست شلیک کنه و سفینه های دشمن رو نابود کنه . تنها در صورتی شلیک موثره که سفینه در جهت شلیک گلوله باشه . دو کلید به عنوان کلید های شلیک به چپ و راست در نظر بگیرید . در نظر داشته باشین که مثل Move نباید بیشتر از یک کلید وارد بشه و صرفا در هر مرحله یه شلیک انجام میشه . اگه کلیدی ورودی مخالف کلید های انتخابی ما بودن باید خطای مناسبی نمایش داده بشه.

نکات پیاده‌سازی

- از اسامی خوانا و معنی دار برای متغیرها, توابع و .... استفاده کنین.
- چون برنامه در ترمینال اجرا میشه, زیبایی برنامه خیلی مهمه. به عنوان مثال محیطی قابل قبول برای نمایش ماتریس پیاده سازی کنین.
- دقت داشته باشید که مینی‌پروژه **کاتاف ۴۰ درصدی** دارد.
- کدنویسی تمیز حتما رعایت بشه.
- سعی کنین تعداد خط کد هاتون توی توابع خیلی زیاد نشه.
- اعضای تیم باید به کل کد مسلط باشن.
- میتونین به صورت تک نفره یا تیم دو نفره روی پروژه کار کنین.
- فایل گزارش‌کار رو به صورت pdf به همراه فایل(ها) پروژه به صورت zip. در بخش مخصوص به مینی پروژه آپلود کنید.

نکات گزارش‌کار

موارد زیر باید حتما در گزارش کار شما وجود داشته باشد:

- کاور: شامل نام برنامه نویس(ها)، شماره دانشجویی برنامه نویس‌ها، نام دانشگاه، اسم برنامه/بازی، سال و نیم‌سال تحصیلی، نام استاد، لوگوی دانشگاه
- شماره صفحه
- کادر بندی
- فهرست دقیق با عناوین درست
- توضیحی کوتاه و جامع (پروژه در یک نگاه) از بازی/برنامه‌ی نوشته شده.
- توضیح کوتاهی از توابع اصلی و یا قطعه هایی از کد
- وجود تصاویری از قطعه هایی از کد که توضیح داده شده اند
- توضیحی از استثنائات (در صورت وجود)
- جمع بندی
- زیبایی حداقلی (مرتب بودن گزارش کار - فونت و سایز خوب نوشته ها و...)

برنامه نویسی ماژولار

برنامه‌نویسی ماژولار به معنای طراحی و توسعه‌ی برنامه‌ها به شکلی است که اجزای مختلف آن به عنوان ماژول‌ها قابل تفکیک و جداگانه قرار گیرند. هر ماژول مسئولیت مشخص و مستقلی را بر عهده دارد و می‌تواند به عنوان یک واحد منطقی مستقل از سایر ماژول‌ها استفاده شود. این ایده باعث افزایش خوانایی کد، قابلیت نگهداری، و قابلیت توسعه برنامه می‌شود.

تابع چیست؟

```
int add(int number1, int number2)
{
    sum = number1 + number2;
    return sum;
}
int sub(int number1, int number2)
{
    sub = number1 - number2;
    return sub;
}
int mul(int number1, int number2)
{
    mul = number1 * number2;
    return mul;
}
int div(int number1, int number2)
{
    div = number1 / number2;
    return div;
}
```

توابع قطعه کد های مستقلی هستند که پس از دریافت ورودی های لازم روی آنها عملیات هایی را انجام داده و سپس خروجی لازم را به دیگر توابع تحویل میدهند تا فرایند های دیگری روی آن اطلاعات انجام شود. برای مثال: در پیاده سازی برنامه ای با هدف انجام عملیات چهارگانه (جمع و تفریق و ضرب و تقسیم) میتوان برای هر عملیات تابع جداگانه ای ایجاد و از آن توابع استفاده کرد.

هر تابع عملیات لازم را بر روی دو ورودی دریافت شده انجام میدهد و نتیجه مد نظر را خروجی میدهد.

```
#include <iostream>

using namespace std;

int main()
{
    int num1, num2;
    cin >> num1 >> num2;

    int result;
    result = add(num1, num2);
    cout << result << endl;
}
```

برای استفاده از توابع فقط کافیست نام آن را صدا زده و ورودی های لازم را به آن نسبت دهیم.

نمونه اولیه تابع (function prototype)

```
#include <iostream>
using namespace std;

int main(){
    cout << add(1, 2) << endl;
}

int add(num1, num2){
    return num1 + num2;
}
```

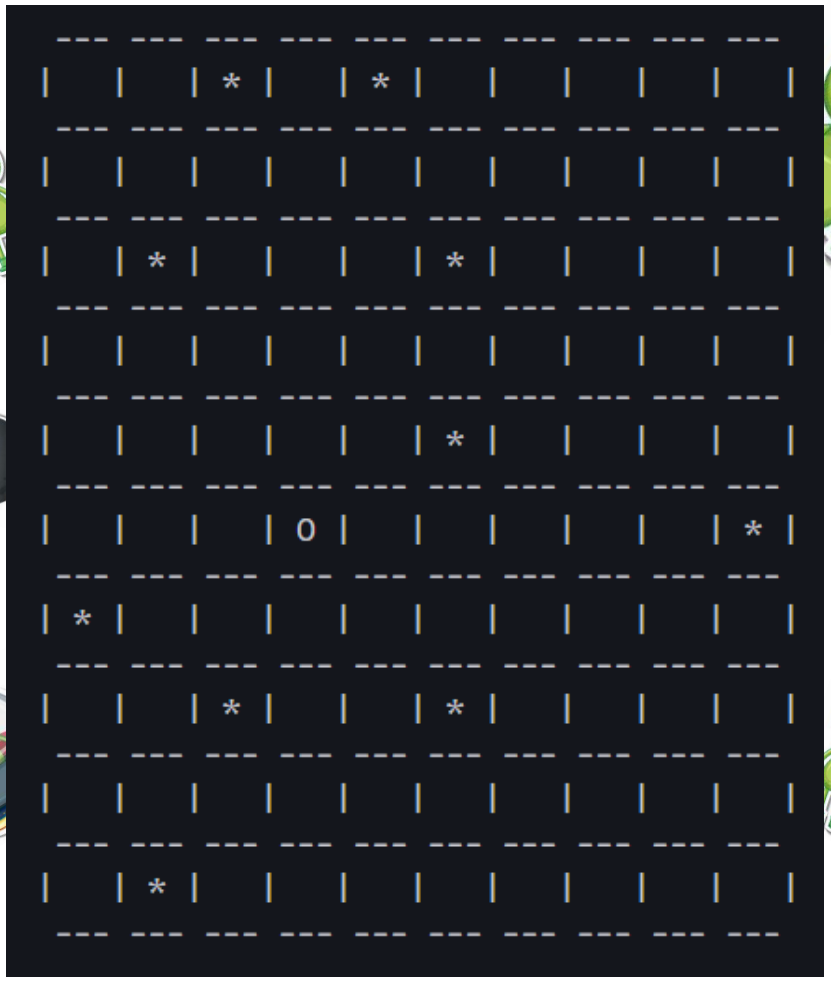
قابل ذکر است که درصورت استفاده از تابع قبل از تعریف برنامه با خطا مواجه میشود.

```
include <iostream>
using namespace std;

int add(int num1, int num2);

int main(){
    cout << add(1, 2) << endl;
}

int add(num1, num2){
    return num1 + num2;
}
```

یکی از راهکار های مناسب برای رفع خطا نوشتن function prototype در ابتدای فایل است.

با این کار به کامپایلر گوشزد میکنیم که در ادامه تابعی با این ویژگی ها تعریف خواهد شد.

نمونه خروجی بازی



موفق باشید :)