



بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

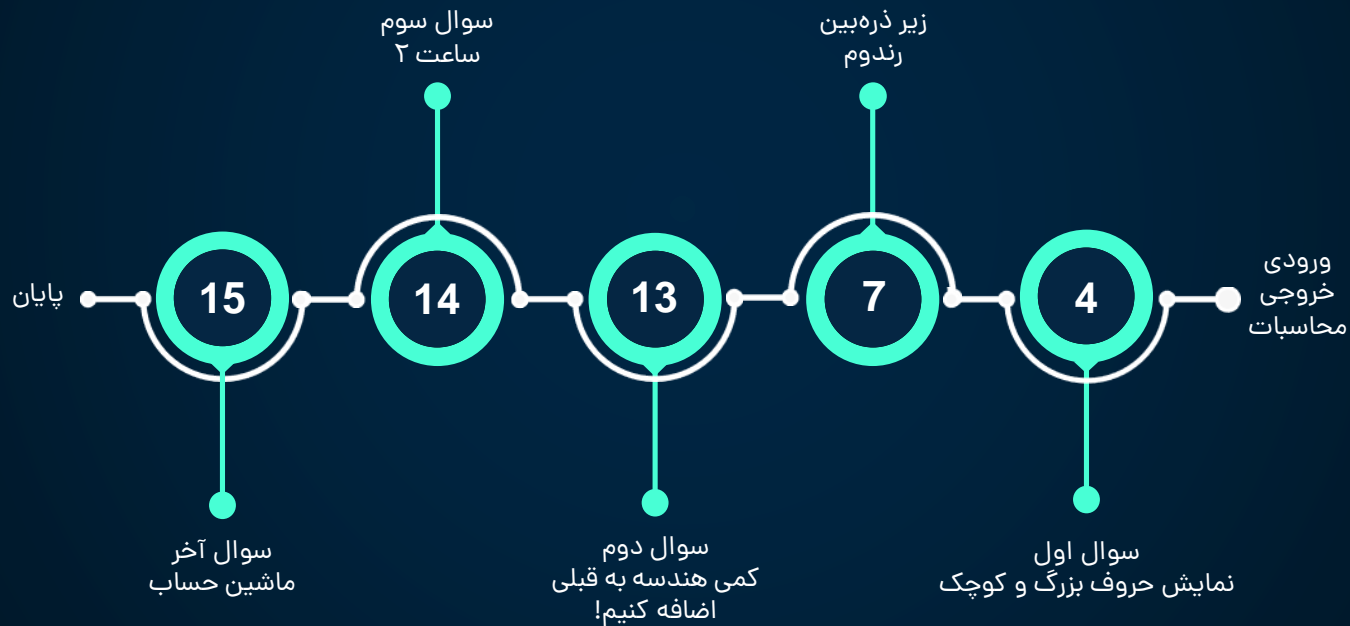
ورودی و خروجی
محاسبات

جلسه سوم


در دنیای کامپیوتر، تکنولوژی و برنامه‌نویسی، اهمیت محاسبات و برنامه‌هایی که در این زمینه به کمک انسان‌ها می‌آیند کاملاً مشهود است. اگر کامپیوترها نبودند، انسان هرگز قادر به انجام برخی از این محاسبات نبود. بخش بزرگی از پیشرفتی که امروزه در تمام زمینه‌ها به دست آمده، مدیون همین محاسبات پیچیده‌ایست که با برنامه‌های کامپیوتری انجام می‌شود. به همین دلیل این جلسه از کارگاه را به انجام محاسبات و گرفتن ورودی از کاربر و نشان دادن خروجی توسط برنامه اختصاص داده‌ایم.


برای قدم گذاشتن در دنیای برنامه‌نویسی لازم است تا بتوانیم با کامپیوتر **ارتباط** برقرار کنیم. در این مسیر باید بدانیم چگونه می‌توانیم اطلاعات خود را در کامپیوتر وارد و ذخیره کنیم و همچنین با چگونگی انجام **عملیات‌های ریاضی** و رسیدن به یک خروجی درست آشنا شویم.

فهرست



سوال اول: نمایش حروف بزرگ و کوچک

این جمله را بارها و بارها شنیده‌ایم که کامپیوترها تنها با اعداد ۰ و ۱ کار می‌کنند. ۰ به معنای خاموش و ۱ به معنای روشن. در درس فهمیدیم که کامپیوتر برای تشخیص اعداد صحیح، اعشاری و غیره باید هرکدام را به صورت ۰ و ۱ در بیاورد و شیوه‌های کد کردن اعداد نیز با هم متفاوت است. 

اما حروف را چگونه باید کدگذاری کرد؟ آیا شیوه‌ی فهماندن حروف به کامپیوتر را به یاد دارید؟ 

✓ برای حل مسئله‌ی پیش‌رو در صورت نیاز می‌توانید به جدول تبدیل کد اسکی^۱ به کاراکتر که در لینک زیر آمده‌است مراجعه کنید.



ASCII lookup table


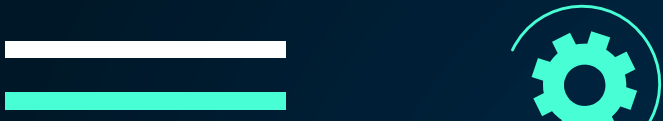
<https://b2n.ir/202623>





با هم‌گروهی خود برنامه‌ای بنویسید که ابتدا در ورودی یک عدد و یک کاراکتر الفبایی دریافت کند.

سپس کاراکتر الفبایی را به اندازه‌ی عدد شیفت دهد. خروجی برنامه هر دو صورت حرف بزرگ و حرف کوچک آن کاراکتر خواهد بود.

✓ مثال: اگر در ورودی کاراکتر B و عدد 3+ باشد، در خروجی E و e چاپ شود.



به عنوان یه برنامه‌نویس خوب باید حواسمون باشه که بتونیم ورودی‌هایی که اشتباه هستند و با اون‌ها نمی‌تونیم خروجی مناسب رو تولید کنیم، محدود کنیم. این کار می‌تونه با شرط گذاری‌های درست، بازه‌بندی کردن ورودی و ... انجام بشه ولی حالا که هنوز شرط رو با هم یاد نگرفتیم فرض می‌کنیم که ورودی‌ها همه درست و منطقی هستند و برنامه رو بدون محدودیت روی ورودی می‌نویسیم.



زیر ذره‌بین: رندوم

چرا اعداد رندوم مهم هستند؟ به نظر شما این اعداد چه کاربردی در دنیای کامپیوتر دارند؟
در برنامه‌نویسی خیلی وقت‌ها ما می‌خواهیم مسائل دنیای واقعی را شبیه‌سازی کنیم که بسیاری از آن‌ها حداقل از دید ما تصادفی هستند.

مثلا شکل ابرها یا بُر زدن دسته‌ای کارت و نحوه‌ی قرار گرفتن آن‌ها. ✓

به‌علاوه اعداد رندوم کاربرد زیادی در ساخت و پیاده‌سازی بازی‌ها دارند. طبیعتا شما دوست ندارید بازی‌ای را انجام دهید که در آن همیشه یک اتفاق می‌افتد و می‌خواهید هر بار در یک حالت مشابه اتفاقات متفاوتی رخ دهد مثلا در بازی شطرنج. بازی ریاضی جلسه‌ی قبل را به یاد دارید؟ اگر همیشه ترتیب کارت و شماره‌های بازی یکسان باشد آیا ادامه‌ی بازی معنایی دارد؟

یکی دیگر از کاربردهای خیلی مهم اعداد رندوم در آمار و احتمالات و حل و پیاده‌سازی مسائل مربوط به آن است که با آن در ترم‌های آینده به طور کامل آشنا خواهید شد. ✓



حال اعداد رندوم چگونه تولید می‌شوند؟ آیا واقعا کامپیوتر قدرت اینکه عدد کاملا تصادفی تولید کند را دارد؟ آیا اعدادی که تولید می‌شوند به شکل کامل تصادفی هستند؟

جواب این پرسش خیر است!

اعداد تصادفی‌ای که کامپیوتر تولید می‌کند به هیچ وجه کاملا تصادفی نیستند و کامپیوتر تنها بر اساس الگوریتم‌های عدد رندومی که کاملا بر اساس ریاضی هستند این اعداد را تولید می‌کند.

یعنی با کمک برخی توابع ریاضی از پیش تعریف شده این اعداد تولید می‌شوند و نه به شکل تصادفی. همین موضوع باعث می‌شود که اگر ما برخی قواعد تولید عدد رندوم را رعایت نکنیم و یا به نفع خودمان تغییر دهیم اعداد تولید شده حتی در ظاهر هم تصادفی نباشند و از یک الگوی مشخص پیروی کنند.

تا الان متوجه شدیم که اعداد تولید شده به هیچ عنوان تصادفی نیستند و به کمک برخی توابع و فرمول‌های ریاضی تولید می‌شوند حال لازم است تا با مفهومی به نام **seed** آشنا شوید.

در واقع می‌توان گفت **seed** نقطه‌ی شروع تابع تولید عدد رندوم است. بدیهی است در صورتی که این مقدار تغییر نکند تابع همواره رشته‌ای از اعداد ثابت تولید می‌کند و دیگر تصادفی نیست. برای مثال تکه کد زیر را اجرا کنید.

```
for(int i = 0; i < 5; i++)  
    printf("%d", rand());  
return 0;
```

مشاهده می‌کنید که در هر بار اجرای این شبه کد ۵ عدد ثابت تولید می‌شوند. دلیل آن هم واضح است، ما هیچ‌گاه **seed** را تغییر ندادیم.

برای تغییر **seed** چه باید کرد؟




در این مرحله تابع **srand** به کمک ما می‌آید. این تابع خروجی ندارد و کاربرد آن تنها این است که مقدار **seed** را برای تابع **rand** مشخص می‌کند. به این شکل که ورودی آن یک عدد صحیح می‌باشد که به عنوان **seed** تابع **rand** انتخاب می‌شود. (در حالت قبلی که از این تابع استفاده نکردیم مقدار **seed** به شکل پیش‌فرض ۱ قرار گرفت).




حال برای استفاده از این تابع هم چالش‌هایی وجود دارد، زیرا ما نیاز داریم برای اینکه عددمان هر بار تصادفی باشد، مقدار **seed** در هر بار اجرا تفاوت کند و ثابت نباشد. به نظر شما چگونه می‌توان این مشکل را حل کرد؟





چه چیزی را می‌توان به عنوان هسته یا **seed** رندوم انتخاب کرد که مدام در حال تغییر باشد و خروجی ثابت ایجاد نکند؟

یکی از راه‌حل‌های اصلی این است که از تابع **time** استفاده کنیم زیرا این تابع هر بار بر اساس زمان سیستم عدد مختلفی را به عنوان خروجی برمی‌گرداند. 

برای مثال کد زیر را اجرا کنید. 

```
srand(time(0));  
for(int i = 0; i < 5; i++)  
    printf("%d", rand());
```

مشاهده می‌شود که این بار برخلاف دفعه‌ی قبل، بعد از هر بار اجرای برنامه، خروجی متفاوتی را مشاهده می‌کنیم که دلیل آن تغییر مقدار **seed** می‌باشد. 

توجه: به خاطر داشته باشید که برای درست کار کردن تابع رندوم، تنها یک بار باید به آن **seed** ای را اختصاص دهید. در غیر این صورت نمی‌توان تضمین کرد که برای عدد رندوم تولید شده احتمال آمدن عدد x با عدد y برابر باشد. 



اگر دوست دارید در مورد تابع time و خروجی‌اش تحقیق کنید. به نتایج جالبی می‌رسین :)
این تابع مقدار ثانیه‌های سپری شده از تاریخ ۱ ژانویه ۱۹۷۰ رو برمی‌گردونه. پس با هر بار اجرای تابع rand، مقدار متفاوتی رو از دفعه‌ی قبل بدست میاریم که قابل پیش‌بینی نیست.

برای غیرقابل پیش‌بینی‌تر کردن این مقدار می‌تونیم از کارهای جالب و خلاقانه‌ی دیگه‌ای هم استفاده کنیم، مثلاً خروجی تابع تایم رو به جای ثانیه به میلی‌ثانیه تبدیل کنیم.
یه مثال دیگه برنامه‌هایی هستن که بر اساس حرکت موس کاربر یا حرکت دادن موبایل و... می‌تونن seed بسازن و با کمک اون عدد رندوم تولید کنن.

سوال سوم: کمی هندسه به قبلی اضافه کنیم!



فرض کنید مثلثی دارید که دو ضلع و زاویه‌ی بین آن مشخص است، حال می‌خواهید با استفاده از اطلاعات فعلی، تمامی مجهولات دیگر مسئله یعنی ضلع دیگر مثلث و اندازه‌ی ۲ زاویه‌ی مجهول را حساب کنید. برای محاسبه‌ی اندازه‌ی ضلع دیگر احتمالا از دبیرستان به‌خاطر دارید که می‌توان از قانون کسینوس‌ها استفاده نمود.

(اگر یادتون نیست با گوگل احساس راحتی کنید! از اون بپرسین!)



پس از محاسبه‌ی ضلع دیگر، به کمک قانون سینوس‌ها اندازه‌ی زاویه‌ی کوچک‌تر را بدست بیاورید (به نظرتان چرا زاویه‌ی کوچک‌تر؟ چرا هر ۲ زاویه یا حتی زاویه‌ی بزرگ‌تر نه؟! و در نهایت هم اندازه‌ی زاویه‌ی آخر را بدست آورید.



یک شرط دیگر مسئله این است که برنامه قرار نیست اندازه‌ی اضلاع و زاویه‌ی بین را دریافت کند و باید هر ۳ را با کمک عدد رندوم تولید کنید! به این شکل که زاویه عددی بین ۰ تا ۹۰ و اضلاع هم هر کدام بین ۵ تا ۲۵ باشند. یعنی کد شما هم صورت سوال را طراحی می‌کند و هم پاسخ آن را می‌یابد!



سوال چهارم: ساعت ۲

مشابه جلسه‌ی قبل، سوالی مربوط به ساعت اما با رویکردی متفاوت را می‌خواهیم بررسی کنیم. این بار از شما می‌خواهیم برنامه‌ای بنویسید که با دریافت تعداد ثانیه‌ها به صورت یک عدد صحیح، محاسبه کند عدد داده شده معادل چند روز، چند ساعت، چند دقیقه و چند ثانیه است. عدد 9876543210 را وارد کنید و ببینید آیا برنامه به درستی کار می‌کند؟



این برنامه را با شبه‌کدی که در جلسه‌ی قبل نوشتید مقایسه کنید. آیا محاسبات، کار را در مقایسه با دقیقه دقیقه پیش‌رفتن در زمان ساده‌تر نکرد؟



❓ و اما سوال آخر: ماشین حساب

سوال آخر مثل همیشه اختصاص دارد به گُدخدا و Botfather.

این دو کدر^۱ بزرگ در یک دوره‌ی آموزشی از شرکت‌کننده‌ها خواستند تا یک ماشین حساب با قابلیت محاسبه‌ی چهار عمل اصلی و برخی توابع مثلثاتی را پیاده‌سازی کنند. آن‌ها نمونه کدهای جالبی دریافت کرده‌اند و از آنجایی که معتقدند مطالعه‌ی کدهای دیگران و خطایابی یکی از تمرین‌های موثر برای یادگیری برنامه‌نویسی است، یک نمونه از کدهای دارای ایراد را برای تمرین در این دستورکار فرستاده‌اند و از شما می‌خواهند خطاهای موجود در کد را رفع کنید تا برنامه به درستی کار کند. آیا می‌توانید نوع هر یک از خطاها را مشخص کنید؟



برای رفع کردن اشکالات من سعی می‌کنم راهنمایی‌هایی بکنم تا راحت‌تر بتوانید آن‌ها را پیدا کنید. از Botfather هم دعوت می‌کنم که در بخش توضیحات هر کدام از ایرادات در کنار من باشد. بیا بید با هم قدم به قدم سعی کنیم تا ایرادات موجود در کد رفع شوند. قبل از هر کاری باید بفهمیم که این ماشین حساب چه قابلیت‌هایی قرار است داشته باشد و بعد شروع کنیم به تست کردن برنامه. ممکن است در ابتدا اصلاً کد کامپایل نشود. آیا این اتفاق برای کد شما هم افتاد؟ پس باید سعی کنیم تا مشکل به وجود آمده را رفع کنیم. به نظر شما چه چیزی این مشکل را به وجود آورده؟

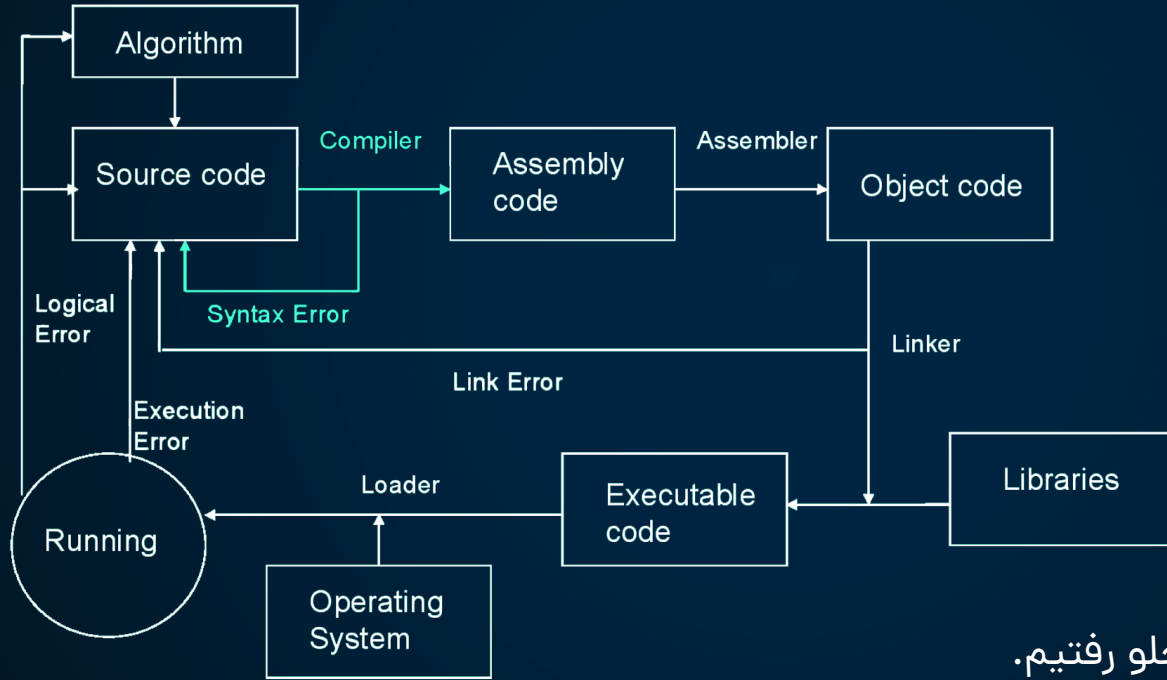


این خطا قبل از اجرا شدن برنامه رخ داده است به همین دلیل به آن **خطای زمان کامپایل**^۱ یا **خطای نوشتاری**^۲ می‌گویند.

هر گونه خطا در نوشتن برنامه مثل نبودن " ; " یا دقت نکردن به کوچک و بزرگ نوشتن حروف در زبان C می‌تواند منجر به این خطا شود. چون کامپایلر برای تبدیل کد نوشته شده به یک برنامه‌ی اجرایی، ابتدا خط به خط کد را چک می‌کند و فایل اجرایی را می‌سازد.



برای رسیدن به برنامه‌ای که هدف نهایی ماست لازم است تا برای رفع کردن تمامی خطاها یک روندی را داشته باشیم تا ایرادی باقی نماند. برای مثال می‌توانیم از چنین مسیری استفاده کنیم.



پس تا اینجا یک مرحله جلو رفتیم.





احتمالا برخی از شما Main را هم درست کردید و به این موضوع که C تفاوت بین حروف بزرگ و کوچک را متوجه می‌شود دقت کردید. اما نوشتن Main به جای main یک خطای کامپایل نیست و ما در این شرایط با Linker error مواجه می‌شویم. علت رخ دادن این ارور را در آینده در مبحثی به نام تابع متوجه خواهید شد.



حال با درست شدن برخی از ایرادها برنامه کامپایل می‌شود اما با این اخطار مواجه می‌شویم:
warning: implicit declaration of function 'printf' [-Wimplicit-function-declaration]

دلیل این اتفاق چیست؟



به عنوان راهنمایی می‌پرسم. آیا می‌دانید stdio مخفف چه چیزی است؟



Header files “stdio.h” and “stdlib.h” in C

<https://b2n.ir/248303>

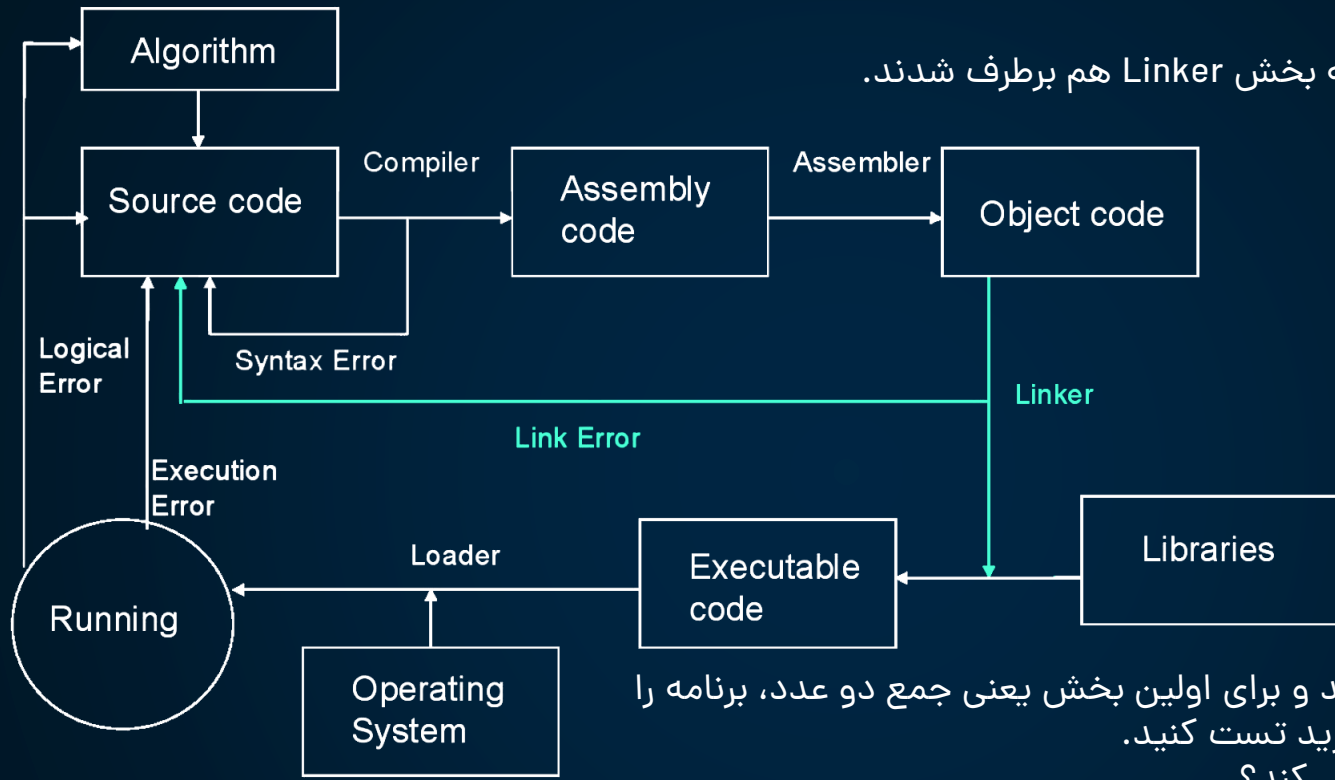


در صورت اضافه نکردن کتابخانه‌های مورد نیاز با ارور یا اخطار مواجه می‌شویم تا زمانی که آن را برطرف کنیم. اگه کامپایلر C بتواند تابع را به صورت **ضمنی** یا **implicit** تعریف کند در این صورت با ارور مواجه نمی‌شویم. برای عمیق‌تر شدن در این موضوع می‌توانید بیشتر تحقیق کنید. به عنوان مثال می‌توانید به لینک زیر مراجعه کنید و اگر دوست داشتید درباره‌ی این موضوع در کلاس تدریس‌یاری بیشتر بحث کنید.





خب خطاهای مربوط به بخش Linker هم برطرف شدند.



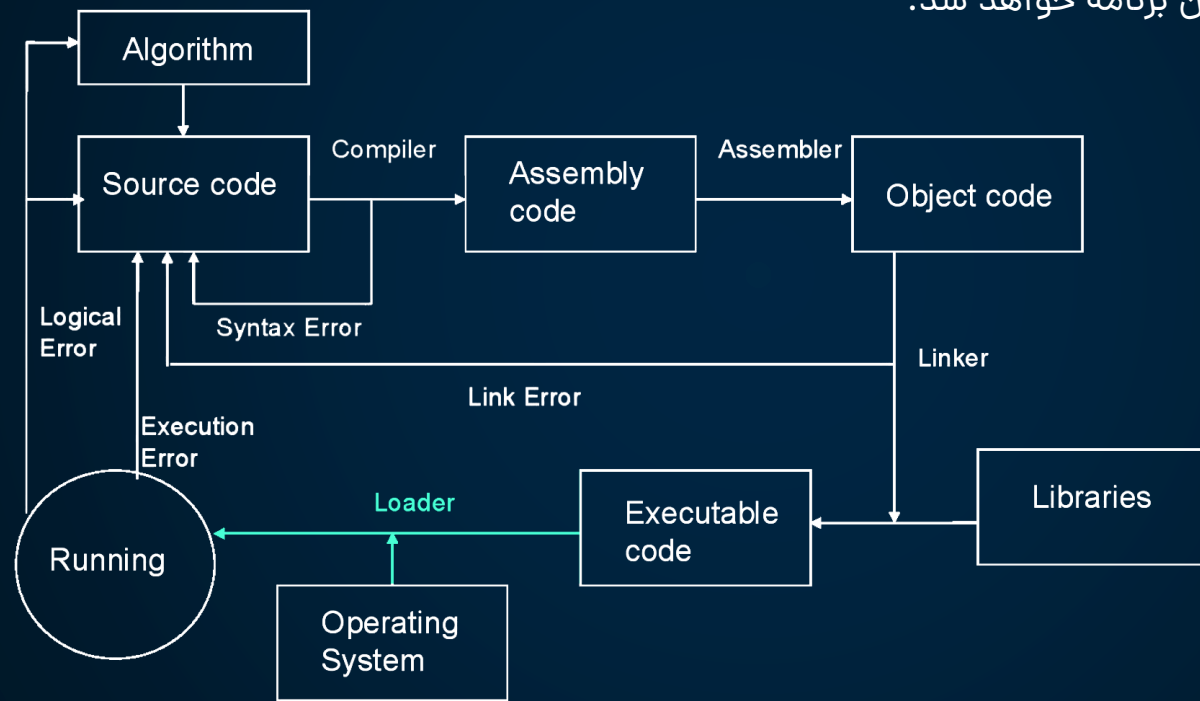
حال برنامه را اجرا کنید و برای اولین بخش یعنی جمع دو عدد، برنامه را هر جور که دوست دارید تست کنید. آیا برنامه درست کار می‌کند؟



این خطا که مانع ادامه پیدا کردن برنامه شده است، **execution error** یا **runtime error** نام دارد چون در زمان اجرا شدن رخ داده است. این بار علت این خطا از کجاست؟



به یاد داشته باشید که قرار ندادن & در scanf یکی از معروف ترین خطاها در زمان اجرای برنامه است. در این زمان برنامه‌ی شما به خانه‌ای از حافظه دسترسی پیدا کرده که سیستم عامل به آن اجازه نمی‌دهد و به همین دلیل مانع ادامه پیدا کردن برنامه خواهد شد.





بالاخره می‌توانیم خود برنامه را تست کنیم که آیا ماشین حساب درست کار می‌کند یا خیر؟
برای شروع خیلی راحت از اولین عملگر یعنی + آغاز می‌کنیم.
آیا حاصل $2 + 3$ درست است؟
حاصل $2/5 + 6/3$ چطور؟



این خطای آخر، یک **خطای منطقی**^۱ است؛ یعنی خطایی است که تنها به دست برنامه‌نویس قابل تشخیص است و به کاربردهایی که او در نظر دارد وابسته است. تنها راه تشخیص چنین خطاهایی هم تست کردن مکرر برنامه به ازای ورودی‌های مختلف است.



می‌توانید برای تسلط بیشتر روی خطاها به هر کدام از لینک‌های زیر مراجعه کنید و اگر دوست دارید در کلاس تدریس‌یاری راجع به آن‌ها با تدریس‌یارها صحبت کنید و یا از طریق ایمیل با من و کدخدا در ارتباط باشید تا بتوانیم شما را در رسیدن به پاسخ سوال‌هایتان راهنمایی کنیم.



Programming Errors in C

<https://b2n.ir/831809>



Compiler, Linker and Run-Time Errors

<https://b2n.ir/948236>



این لینک و لینک‌های مشابه آن نیز با نشان دادن تفاوت‌های بین ارور ها می‌توانند در یادگیری و عمیق‌تر شدن کمک کنند.



Compile errors VS Runtime errors

<https://b2n.ir/999477>

;