



# بازگشتی

جلسه هفتم

تابع

بسم الله الرحمن الرحيم



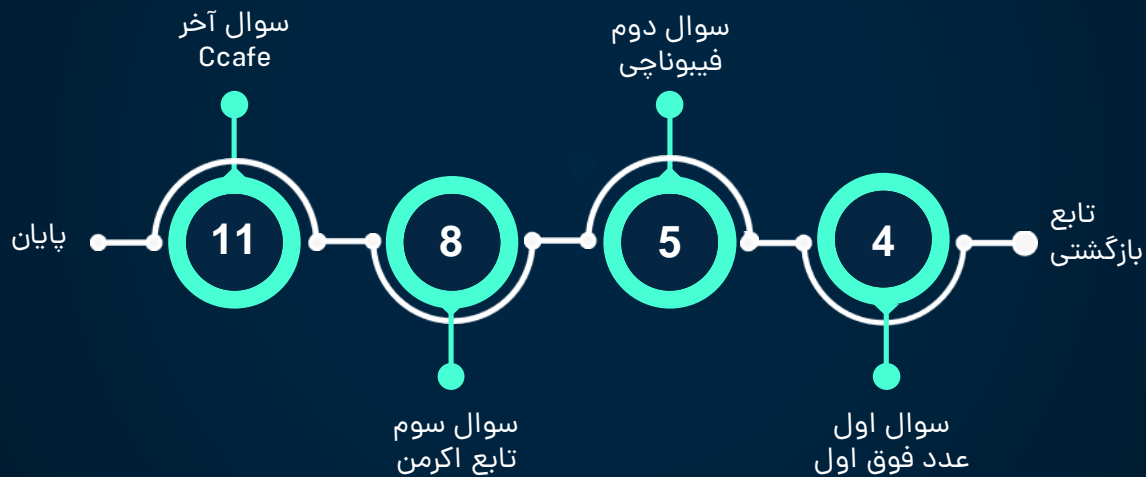
کارگاه مبانی برنامه‌نویسی - دانشکده مهندسی کامپیوتر دانشگاه امیرکبیر



همه‌ی ما با مفهوم تابع از ریاضی دبیرستان آشنایی داریم،  
تابع‌های برنامه‌نویسی یک گسترش<sup>۱</sup> از همان توابع ریاضی  
هستند. هدف اولیه‌ی ایجاد توابع برنامه‌نویسی، ساخت یک  
خلاصه<sup>۲</sup>، برای محاسبه‌ای بزرگ‌تر است. مثلاً فرض کنید برنامه‌ای  
نوشته‌ایم که با گرفتن یک عدد صحیح  $n$  به عنوان ورودی،  
مجموع اعداد 1 تا  $n$  را حساب می‌کند. حالا می‌خواهیم برنامه‌ای  
بنویسیم که این کار را چندین بار و تحت شرایط مختلفی انجام  
دهد. به جای این که کد محاسبه‌ی این مجموع را چندین بار کپی  
کنیم، می‌توانیم با استفاده از توابع، این کد را یک بار بنویسیم و  
چندین بار استفاده کنیم. (کتاب‌های گاج را چندین بار بخوانید.


( ((((((:

# فهرست





# سوال اول: عدد فوق اول

اگر ارقام عددی را جدا کنیم و به صورت یک رقمی، دو رقمی، سه رقمی و ... بنویسیم به طوری که تمام این ترکیبات جدا شده عدد اول باشند، آن عدد را عددی فوق اول می‌دانیم. به عنوان مثال 173 عددی فوق اول است چون 3 و 17 و 173 اول می‌باشند و اگر یکی از این اعداد تفکیک شده عدد اول نبود، در آن صورت بدیهی‌ست که عدد دریافتی نیز فوق اول نمیشد!

با توجه به این تعریف برنامه‌ای بنویسید که عددی را به عنوان ورودی از کاربر دریافت کرده و فوق اول بودن یا نبودن عدد را مشخص کند. 

# سوال دوم: فیبوناچی

 **الف)** با استفاده از رابطه‌ی بازگشتی دنباله‌ی فیبوناچی، برنامه‌ای بنویسید که  $n$  را گرفته و جمله‌ی  $n$ ام فیبوناچی را با روش بازگشتی به دست آورد.  
اگر  $n$  برابر 100 باشد، برنامه‌ی شما می‌تواند جواب را در چند دقیقه تولید کند؟

 همان‌طور که مشاهده کردید، جواب در چند دقیقه تولید نمی‌گردد، با توجه به این‌که از نظر منطقی برای محاسبه جمله‌ی 100ام حداکثر نیاز به محاسبه‌ی 100 جمله‌ی قبل است. چرا این محاسبه به این میزان طولانی شده است؟



**ب)** تابع فیبوناچی را به صورت غیربازگشتی بنویسید، این بار تابع را با  $n = 100$  فراخوانی کنید، آیا جواب حاصل در زمان چند دقیقه حاصل می‌گردد؟



این‌طور به نظر می‌رسد که راه حل‌های بازگشتی در برخی از موارد کارآیی لازم را ندارند، به نظر شما آیا روشی برای بهبود این موضوع وجود دارد یا واقعا استفاده از روش‌های بازگشتی بی‌ثمر است؟



**پ)** برنامه‌ای بنویسید که عدد  $n$ ام فیبوناچی را با استفاده از فرمول عمومی آن به دست می‌آورد.

$$F(n) = \frac{\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n}{\sqrt{5}}$$



سرعت عمل‌کرد این برنامه را با برنامه‌ی قسمت **ب** برای  $n$ های مختلف حساب کنید. به ازای چه  $n$ هایی تفاوت سرعت این دو مشهود می‌شود؟

✓ بررسی این‌گونه مسائل و میزان سرعت الگوریتم‌های مختلف و ارتباط آن‌ها با مقدار ورودی در حوزه‌ای به نام **Complexity Theory** بررسی می‌شود. شما در درس «ساختمان داده‌ها» به صورت مقدماتی با این بخش آشنا خواهید شد.

✓ هم‌چنین، نسبت طلایی به صورت  $\frac{a+b}{a} = \frac{a}{b} \stackrel{\text{def}}{=} \varphi$  تعریف می‌شود که جواب آن همان

$\varphi = \frac{1+\sqrt{5}}{2} = 1.6180339887 \dots$  است که در قسمت اول فرمول عمومی فیبوناچی دیدیم.

این عدد بسیار در طبیعت ظاهر می‌شود (با عدد اویلر یا  $e$  اشتباه نگیرید!) و بسیاری از سازه‌ها را برای ماکسیمم کردن زیبایی آن‌ها با استفاده از این عدد می‌سازند.

# سوال سوم: تابع اکرمین

تابع اکرمین یکی از ساده‌ترین و در عین حال جالب‌ترین توابع بازگشتی‌ای است که کشف شده‌است. از بعد ریاضی و تئوری، این تابع اهمیت بسیار زیادی دارد و مسائل و اثبات‌های مختلفی برای آن ارائه می‌شود اما در این بخش ما کاری به بخش ریاضی و تئوری این تابع نداریم (هرچند توصیه می‌کنیم حتماً برید و درموردش بخونید:) خیلی جالب و قشنگه) در اینجا می‌خواهیم به کمک هم این تابع را پیاده‌سازی کنیم.

قبل از هر چیز این تابع اصلاً چجوری است؟

تابع اکرمین ۲ متغیر به عنوان ورودی می‌گیرد  $(m, n)$  و با توجه به شرایط دو متغیر به شکل زیر خروجی مطلوب را

می‌دهد:

$$A(m, n) = \begin{cases} n + 1 & \text{if } m = 0 \\ A(m - 1, 1) & \text{if } m > 0 \text{ and } n = 0 \\ A(m - 1, A(m, n - 1)) & \text{if } m > 0 \text{ and } n > 0 \end{cases}$$

Where  $m$  and  $n$  are non-negative integers





دقت کنید همانطور که در انتهای تعریف تابع نوشته شده بود، مقادیر ورودی باید نامنفی باشند.



شاید در نگاه اول و با قرار دادن چند عدد ساده در آن به نظر تابع ساده‌ای بیاید و محاسبتش خیلی پیچیده نباشد. جالب است اگر چند عدد کوچک هم در آن بگذارید به همین نتیجه می‌رسید و مشاهده می‌کنید که انگار روند رشد این تابع بسیار کند و آهسته است.



مثلا برای مقادیر  $m=4$  و  $n=3$  بیش‌ترین مقدار آن برابر 29 است که همان مقدار  $A(3, 2)$  است. اما به محض اینکه کمی مقدار ورودی را بیش‌تر کنیم تابع به شکل حیرت‌انگیزی به سرعت رشد می‌کند به شکلی که مقدار  $A(4, 0)$  برابر 13 است اما با زیاد کردن  $n$  فقط به اندازه‌ی 1 واحد مقدار  $A(4, 1)$  برابر 65533 می‌شود! با زیاد کردن مجدد  $n$  این رشد شدیدتر و حیرت‌آورتر هم می‌شود! طوری که مقدار  $A(4, 2)$  یک عدد 19729 رقمی است! یعنی:  $3 - 2^{2^{65536}}$

در جدول زیر رشد این تابع کمی ملموس‌تر است.

$m \backslash n$	۰	۱	۲	۳	۴	$n$
۰	1	2	3	4	5	$n + 1$
۱	2	3	4	5	6	$n + 2 = 2 + (n + 3) - 3$
۲	3	5	7	9	11	$2n + 3 = 2 \cdot (n + 3) - 3$
۳	5	13	29	61	125	$2^{(n+3)} - 3$
۴	۱۳	۶۵۵۳۳	$2^{65536} - 3$	$2^{2^{65536}} - 3$	$2^{2^{2^{65536}}} - 3$	$\underbrace{2^{2^{\dots^2}}}_{n+3} - 3$

حال از شما می‌خواهیم برنامه‌ای بنویسید که با دریافت دو عدد  $m, n$  به عنوان ورودی، مقدار تابع اکرمین را برای آن دو مقدار در خروجی نشان دهد.

# و اما سوال آخر: Ccafe



سلام به همگی... امروز می‌خوایم براتون یکم خاطره تعریف کنیم.

احتمالا شما هنوز با کافی‌شاپ‌های خیابون ولیعصر با دوستانتون خاطره نساختین. از سر کلاس خسته و کوفته برگردین در حالی که یه ربع دیگه باید برین سر کلاس بعدی. این موقع‌ها فقط یه کافی‌شاپ می‌تونه آدمو نجات بده.

اما متاسفانه کرونا در همه‌شونو تخته کرده و حالا حالاها نمی‌تونیم دوباره دسته جمعی بریم با هم یه عصرونه‌ی حسابی بزنیم :



ولی یکی از کافی‌شاپ‌ها به اسم Ccafe هست که هنوز سفارش می‌گیره. می‌شه بریم دم درش، سفارشومونو بدیم و بعد هم دریافتش کنیم. البته چون هر روز فقط یه نفر میاد تا کافی‌شاپ رو بگردونه، برای همین منوشون فقط کیک و کلوچه یا یه لیوان قهوه‌ی حسابی داره.



کلوچه‌هاش واقعا خوش‌مزه‌است :"



حالا ما میخوایم براشون یه برنامه بنویسیم که راحت تر بتونن خریدهای مشتری ها رو مدیریت کنن و برای رعایت بیش تر بهداشت، کمتر لازم باشه مستقیم باهاشون صحبت کنن.



برنامه، اول از همه باید بدونه که کافه هر روز یه مقدار ثابتی از کیک و کلوچه و قهوه ها رو آماده می کنه. پس می تونیم مقدار اولیه ی اون ها رو define کنیم. یعنی مثلا برای کیک ها داریم:

```
#define init_cake_num 40
```



در ادامه به دو تا تابع نیاز داریم که بتونیم هر زمان منو رو به مشتری نشون بدیم و همچنین خریدها رو مدیریت کنیم. main برنامه مون ولی قراره خیلی خلوت باشه. یعنی اینطوری:

```
int main() {  
    menu(init_cake_num, init_cookie_num, init_coffee_num);  
    while(choice != 5)  
        buy();  
    printf("C U soon\n");  
    return 0;  
}
```



حالا دو تا تابعی که لازم داریم رو باید طوری تعریف کنیم که تابع main خطا نداشته باشه (غیر مستقیم یعنی حواستون به ورودیایی که تابع ها میگیرن باشه).



menu که کارش معلومه. توی کدی هم که براتون آماده کردیم این بخشش تکمیل شده هست. ولی با تابع buy حسابی کار داریم.

این تابع باید اول ورودی کاربر رو بگیره. یعنی همون choice ای که تو main آورده شده. تا زمانی که کاربر 5 رو وارد نکرده، یعنی هنوز میخواد خرید کنه.



خب حالا به نظرتون choice چطور باید تعریف بشه که تو هر دو تابع main و buy قابل استفاده باشه؟



در ادامه سه تا متغیر رو لازم داریم که بتونن موجودی‌ها رو نگهداری کنن. اینجا به نظرتون باید چی کار کنیم؟



اگه سه تا متغیر به صورت عادی و تو خود تابع buy تعریف کنیم (یعنی به صورت **local**)، با هر بار اجرا شدن تابع buy، مقدارهای قبلی پاک می‌شه و متغیر دوباره تعریف می‌شه. یه راه حل می‌شه کاری که برای choice انجام دادین. یعنی استفاده از متغیر **global**. اما ما می‌خوایم یه کار دیگه بکنیم. چه راهی به ذهنتون می‌رسه؟



اگه یادتون باشه متغیرهای **static**، متغیرهایی هستن که با تموم شدن تابع و خارج شدن ازش محتواشون رو از دست نمیدن. پس این بخش رو هم کامل کنین تا بریم سراغ switch case.



در این بخش قراره با توجه به هر خرید، مقدار متغیر مربوط به خرید انجام شده، یکی کم بشه و منو هم دوباره نشون داده بشه.



راستی! جالبترین و خاطره‌انگیزترین بخش Ccafe رو براتون نگفتم. تو Ccafe هر بار که دیگه خریده‌ها تموم میشه و می‌خوای بری، می‌تونی اگه دوست داشته باشی یه عکس یادگاری بگیری. برای همین تو case 5 یعنی وقتی که دیگه مشتری خریدی نداره، ازش پرسیده می‌خوای عکس هم بگیری یا نه؟



متغیر pic ای که تعریف شده در واقع داره تعداد عکس‌ها رو نشون می‌ده. به نظرتون چرا این متغیر دیگه **static** در نظر گرفته نشده؟



فکر کنم الان دیگه فرق متغیرهای **global** و **static** و **local** رو متوجه شده باشین. یه سوال دیگه هم بپرسم و دیگه خسته نباشید: دی

ما شرط خاتمه‌ی خرید رو به صورت دلخواه عدد 5 در نظر گرفتیم. به نظر شما با توجه به کدی که نوشته شده، می‌تونیم این عدد رو 7 در نظر بگیریم؟ 0 چطور؟ این دومی نکته توشه دقت کنین...



خب دیگه همگی خسته نباشید. امیدواریم با کد کافه‌تون حسابی کیف کنین و اگر دوست داشتیم بهش فیچرهای بیش‌تری اضافه کنین. مثل بخش حسابداری یا تابعی که با کمک اون بشه مقدار موجودی رو زیاد کرد و در نهایت یه کافه‌ی حسابی داشت.  
خدا حافظ همگی تا جلسه‌ی بعد 😊



;