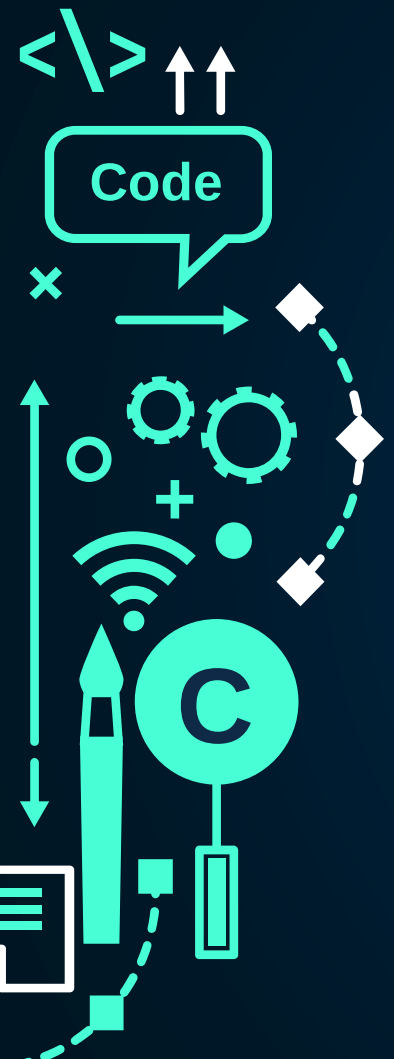


بسم الله الرحمن الرحیم

تابع بازگشتی

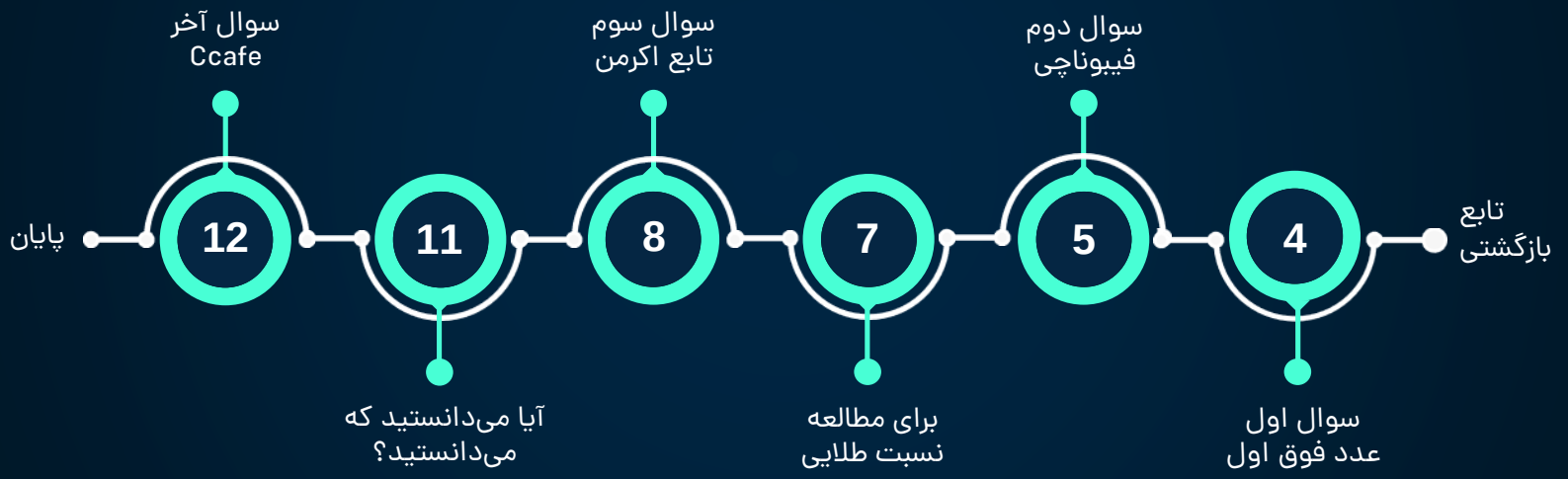
جلسه هفتم

همه‌ی ما با مفهوم تابع از ریاضی دبیرستان آشنایی داریم... تابع‌های برنامه‌نویسی یک گسترش[1] از همان توابع ریاضی هستند. هدف اولیه‌ی ایجاد توابع برنامه‌نویسی، ساخت یک خلاصه[2]، برای محاسبه‌ای بزرگ‌تر است. مثلا فرض کنید برنامه‌ای نوشته‌ایم که با گرفتن یک عدد طبیعی n به عنوان ورودی، مجموع اعداد 1 تا n را حساب می‌کند. حالا می‌خواهیم برنامه‌ای بنویسیم که این‌کار را چندین بار و تحت شرایط مختلفی انجام دهد. به‌جای این‌که کد محاسبه‌ی این مجموع را چندین بار کپی کنیم، می‌توانیم با استفاده از توابع، این کد را یک بار بنویسیم و چندین‌بار استفاده کنیم. (کتاب‌های گاج را چندین‌بار بخوانید. :)))))))

اهمیت استفاده از توابع در برنامه‌نویسی غیر قابل انکار است؛ طوری که استفاده نکردن از آن در برنامه‌های بزرگ و طولانی تقریبا غیر ممکن خواهد بود.

1- Extension
2- Shorthand

فهرست


تابع بازگشتی
4
سوال اول
عدد فوق اول
5
سوال دوم
فیبوناچی
7
برای مطالعه
نسبت طلایی
8
سوال سوم
تابع اکرمن
11
آیا می‌دانستید که
می‌دانستید؟
12
سوال آخر
Ccafe
پایان

سوال اول: عدد فوق اول

- اگر ارقام عددی را از سمت راست 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 73 و 173 اول هستند.
- اگر یکی از این اعداد تفکیک شده عدد اول نباشد، در آن صورت بدیهی‌ست که عدد دریافتی نیز فوق اول نمی‌باشد!

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 nام 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 مدت کوتاهی 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 کم‌تر از چند دقیقه حاصل می‌گردد؟

این‌طور به نظر می‌رسد که راه حل های 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 nام فیبوناچی را با استفاده از فرمول عمومی آن به دست آورد.

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

سرعت عمل‌کرد این برنامه را با برنامه‌ی قسمت ب برای nهای مختلف حساب کنید. به ازای چه nهایی 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 درس‌های «ساختمان داده‌ها» و «طراحی الگوریتم‌ها» به صورت مقدماتی با این حوزه آشنا خواهید شد.

نسبت طلایی

برای مطالعه

نکته‌ی بسیار جالبی در مورد اعداد سری فیبوناچی وجود دارد که شاید برای شما هم جذاب باشد. آیا تا به حال چیزی در مورد نسبت طلایی شنیده‌اید؟

نسبت طلایی برابر است با $\varphi = \frac{1+\sqrt{5}}{2} = 1.6180339887\ldots$ که به صورت زیر تعریف می‌شود:

هر دو عددی مانند a و b که در $\frac{a+b}{a} = \frac{a}{b} \stackrel{\text{def}}{=} \varphi$ صدق کنند، نسبت آن‌ها طلایی‌است.

ارتباط این موضوع با سری فیبوناچی در این‌جاست که هر دو جمله‌ی متوالی در این سری می‌توانند به جای a و b قرار بگیرند و عددی نزدیک به نسبت طلایی را بسازند.

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

سوال سوم: تابع اکرمن

تابع اکرمن یکی از ساده‌ترین و در عین حال جالب‌ترین توابع بازگشتی‌ای است که کشف شده‌است. از بُعد 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 حتما 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 اصلا چه جور است؟

تابع اکرمن ۲ متغیر به عنوان ورودی می‌گیرد (m, n) و با توجه به شرایط دو متغیر به شکل زیر خروجی مطلوب را تولید می‌کند:

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

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 این تابع در نگاه اول و با قرار دادن چند عدد ساده در آن، به نظر تابع ساده‌ای بیاید و محاسبه‌اش 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: $2^{2^{2^{65536}}} - 3$

در جدول زیر رشد این تابع کمی ملموس‌تر است.

m/n	0	1	2	3	4	n
0	1	2	3	4	5	$n+1$
1	2	3	4	5	6	$n+2=2+(n+3)-3$
2	3	5	7	9	11	$2n+3=2(n+3)-3$
3	5	13	29	61	125	$2^{(n+3)}-3$
4	13	65533	$2^{65536}-3$	$2^{2^{65536}}-3$	$2^{2^{2^{65536}}}-3$	$\underbrace{2^{2^{\cdot^{\cdot^{2}}}}}_{n+3}-3$

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

آیا می‌دانستید که می‌دانستید؟

تا حالا دقت کرده بودین می‌شه برای یه تابع تعداد ورودی متغیر تعریف کرد؟ تابع dynamic رو ببینین. در هر فراخوانی تعداد متغیر متفاوتی داره. حالا به تابع‌های printf داخل کد دقت کنین. تابع printf داخل main با تابع printf داخل تابع dynamic از نظر تعداد ورودی‌هایی که گرفتن با هم چه فرقی دارن؟ دیدین می‌دونستین؟ :)

```c
void dynamic(int first, int second, ...) {
    printf("first = %d, second = %d\n", first, second);
}

int main() {
    printf("Variable numbers of arguments\n");
    dynamic(1, 2, 3);
    dynamic(4, 5, 6, 7);
    return 0;
}
```

و اما سوال آخر: Ccafe

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

احتمالا شما هنوز با کافی‌شاپ‌های خیابون ولیعصر با دوستاتون 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 راحت‌تر بتونن خریدهای مشتری‌ها رو مدیریت کنن و برای رعایت بیش‌تر 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 قهوه‌ها رو آماده می‌کنه. پس می‌تونیم مقدار اولیه‌ی اون‌ها رو define کنیم. یعنی مثلا برای کیک‌ها داریم:

```
#define init_cake_num 40
```

در ادامه به دو تا تابع نیاز داریم که اولی بیاد و منو رو به مشتری نشون بده. دومی هم خریدهای مشتری‌ها رو مدیریت کنه. main برنامه‌مون ولی قراره خییلی 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 تابع‌ها می‌گیرن باشه).

menu که کارش معلومه. توی کدی هم که براتون آماده کردیم این بخشش تکمیل‌شده‌ست. ولی با تابع buy حسابی کار داریم.
این تابع باید اول ورودی کاربر رو بگیره. یعنی همون choiceای 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 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 راه‌حل 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.

راستیی! جالب‌ترین و خاطره‌انگیزترین بخش Ccafe رو براتون نگفتیم. تو Ccafe هر بار که خریدها 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 picای 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 داشتین 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.
خداحافظ همگی تا جلسه‌ی بعد ☺

;