



# شبه کد ورودی و خروجی

جلسه چهارم

## بسم الله الرحمن الرحيم



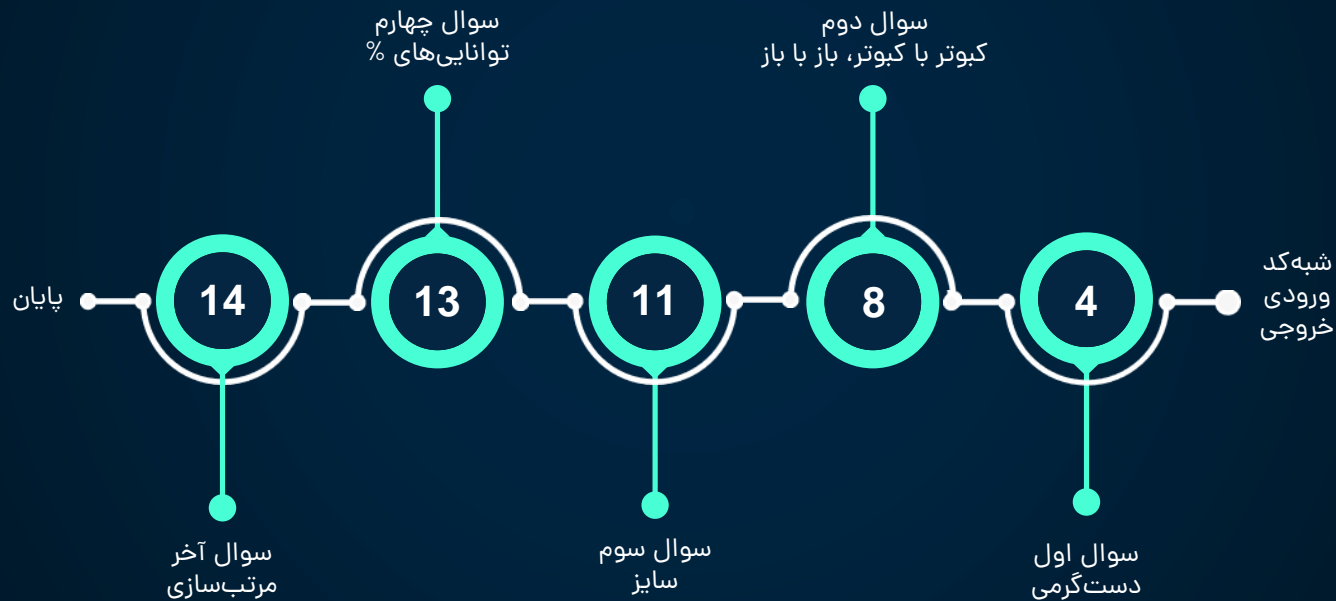
کارگاه مبانی برنامه نویسی - دانشکده مهندسی کامپیوتر دانشگاه هوایی شهید ستاری

برای قدم گذاشتن در دنیای برنامه‌نویسی لازم است تا بتوانیم با کامپیوتر ارتباط برقرار کنیم. در این مسیر باید بدانیم چطور می‌توانیم اطلاعات خود را در کامپیوتر وارد و ذخیره کنیم و همچنین چطور باید از کامپیوتر بخواهیم تا نتیجه‌ی کار خود را به ما نشان دهد.

به همین دلیل، این جلسه از کارگاه را به گرفتن ورودی از کاربر و نشان دادن خروجی توسط برنامه اختصاص داده‌ایم.



# فهرست



# سوال اول: دست‌گرمی

برای اینکه کامپیوتر ورودی‌ای را از کاربر دریافت کند، باید ابزار مناسب این کار را داشته باشید. اولین ابزاری که برای این کار با آن آشنا می‌شوید، تابع `scanf` است.

```
scanf("%type", &variable);
```

ابزار مورد نیاز بعدی ما تابع `printf` است که به برنامه‌ی ما این امکان را می‌دهد تا بتواند خروجی را نشان دهد.

```
printf("%type", variable);
```

هر دوی این ابزارها از جعبه ابزاری با نام `stdio` مخفف `standard input output` آمده‌اند. پس باید حتما در ابتدای برنامه، نام این جعبه ابزار (که در دنیای کامپیوتر و برنامه‌نویسی به جای جعبه ابزار به آن **کتابخانه**<sup>۱</sup> گفته می‌شود) آورده شود.

هر چیزی که بخواد از طریق صفحه کلید وارد کامپیوتر بشه، باید از دروازه‌ای که توی شکل توابع به رنگ سبز کشیده شده عبور کنه. اگر قرار باشه در زمان اجرای برنامه، ورودی‌ای از دنیای خارج به برنامه داده بشه، باید مشخصاتش رو به این دروازه‌ها بدیم تا اجازه‌ی ورودش داده بشه.

`scanf(`  `,`   `x);`

بعد هم یک راهنما براش تعیین کنیم تا بتونه با کمک اون بره خونه‌ی خودش توی حافظه. اگر این راهنما نباشه، ممکنه ورودی ما توی کامپیوتر سرگردون بمونه و جاش رو پیدا نکنه!

هرچند برای چاپ کردنش باید دوباره از همون دروازه‌ی سبزرنگ عبور کنه، اما دیگه به راهنمایی احتیاج نداره چون خودش راه رو یاد گرفته. (((

`printf(`  `, x);`



همان‌طور که گفته شد، باید زمانی که متغیری به این توابع داده می‌شود، مشخصاتش هم ذکر شود. متغیرها انواع مختلفی دارند که برخی از آن‌ها در جدول زیر نوشته شده و در ستون روبه‌روی آن‌ها، علامت معادل آن در کتابخانه‌ی `stdio` نوشته شده است.

Format or Type	Format Specifier
int	%d or %i
char	%c
float	%f
double	%lf



بنابراین اگر قرار باشد یک عدد صحیح مانند `x` وارد کامپیوتر شود، باید گفته شود که نوع این متغیر `integer` یا همان **صحیح** است و به این صورت نوشته می‌شود:


```
scanf("%d", &x);
```



حال قطعه کد زیر را اجرا کنید تا توضیحات قبلی بیش‌تر برای‌تان جا بیفتد. نام کتاب‌خانه‌ی مورد نیاز هم همان‌طور که می‌بینید در ابتدای کد آمده است.

```
#include <stdio.h>
```

```
int main() {  
    int x, y;  
    scanf("%d", &x);  
    scanf("%d", &y);  
    printf("The result is: %d\n", (x + y));  
}
```


به نظر شما اگر هر کدام از علامت‌های d را تغییر دهیم یا هرکدام از &ها را برداریم، چه تاثیری در نتیجه خواهد داشت؟ 


## سوال دوم: کبوتر با کبوتر، باز با باز

کامپیوترها همه چیز را به صورت اعداد 0 و 1 ای ذخیره می کنند، اما این رشته های 0 و 1 را به انواع مختلفی تفسیر می کنند و دقیقا به همین دلیل است که ما باید در هنگام استفاده از متغیرها، نوع تفسیر (به صورت دقیق تر، **type** یا **نوع**) آن متغیر را نیز بدانیم. در غیر این صورت، کامپیوتر تفسیر نادرستی از متغیر 0 و 1 ای ما خواهد داشت و نمی تواند محاسبات درستی انجام دهد.

مثلا اگر به کامپیوتر بگوییم که  $x$  یک عدد صحیح است، اما هنگام ورودی دادن به برنامه عدد 12.7 را وارد کنیم چه اتفاقی خواهد افتاد؟ برعکس آن چه طور؟ یعنی اگر از کامپیوتر بخواهیم عدد 56 را به صورت float یعنی با `%f` چاپ کند، مشکلی به وجود خواهد آمد؟




برای بررسی بیشتر، کد زیر را ببینید. خروجی این کد چیست؟ می‌توانید قبل از اجرای برنامه و با کمک کد آن را تفسیر کنید؟ 

این خروجی در اصل چه چیزی که مربوط به ورودی شما باشد را نشان می‌دهد؟   
خروجی‌ها به ازای کاراکترهای مختلفی که وارد می‌کنید چه تغییری می‌کنند؟


```
#include <stdio.h>
```

```
int main() {  
    char x;  
    printf("Enter a character:\n");  
    scanf("%c", &x);  
    printf("%d\n", x);  
}
```

می‌توانید شباهت‌های ورودی و خروجی خود را با   
محتوای جدول موجود در لینک زیر مقایسه کنید؟



[ASCII lookup table](#)

این بار قطعه کد زیر را اجرا کنید. سپس به عنوان ورودی یکبار عدد 5 و یکبار عدد 1904892031 را وارد کنید. 

```
#include <stdio.h>
```

```
int main() {  
    float x;  
    printf("Enter a decimal number:\n");  
    scanf("%d", &x);  
    printf("%f\n", x);  
}
```

عدد مشاهده شده در خروجی چیست؟ به نظر شما چرا چنین اتفاقی برای خروجی افتاد؟ 

# سوال سوم: سائز



هر کدام از متغیرها در کامپیوتر سائز خاصی دارند، یعنی قرار است تعداد بایت<sup>۱</sup> مشخصی از حافظه‌ی کامپیوتر را اشغال کنند.

نکته‌ی بی‌ربط: هر بایت برابر هشت بیت است و هر بیت نماینده‌ی یک 0 یا 1 در کامپیوتر.

با اجرای کد زیر، سائز هر کدام از متغیرها را به دست آورید.

```
#include <stdio.h>
```

```
int main() {  
    printf("int: %ld \n", sizeof(int));  
    printf("float: %ld \n", sizeof(float));  
    printf("double: %ld \n", sizeof(double));  
    printf("char: %ld \n", sizeof(char));  
}
```

اگر به جای int از long استفاده کنیم چه فرقی می‌کند؟

long long int چگونه؟



قطعه کد زیر را با ورودی 1399 امتحان کنید. چه اتفاقی می افتد؟ `#include <stdio.h>`

```
int main() {  
    char x;  
    printf("Enter a number:\n");  
    scanf("%d", &x);  
    printf("%c\n", x);  
}
```

آیا برنامه با خطایی مواجه شد؟



می توانید دلیل این خطا را با توجه به نکته ای

که از کد قبلی به دست آورده اید توجیه کنید؟

در صورتی که برنامه ی شما با خطایی مواجه نشده است، علت آن عدم هوشمندی کامپایلری است که از آن استفاده می کنید؛ زیرا این کامپایلر بدون آنکه دقت کند متغیر x به چهار بایت نیاز دارند، تنها یک بایت از آن را می خواند و همان بخش را به صورت یک کد اسکی دیده و کاراکتر مربوط به آن را چاپ می کند. برای مثال کد اسکی عدد 2 را در نظر بگیرید... این کد برابر 50 است. یک بار برنامه را با عدد 50 اجرا کنید. همانطور که انتظار می رفت "کاراکتر ۲" چاپ شد. حال  $256+50$  که برابر 306 است را وارد نمایید. مشاهده می کنید که این بار هم کاراکتر 2 را نشان می دهد. در صورتی که عدد 306 ک اسکی یک کاراکتر دیگرست!



## سوال چهارم: توانایی‌های %

این مشخص‌کننده‌های قالبی<sup>۱</sup> که تا الان دیدید، علاوه بر فهماندن نوع متغیر به برنامه، ابزارهای دیگری هم دارند که به صورت کلی به این شکل نمایش داده می‌شود.

`%[conversion character][precision].[width][-]`

برای تمرین هر کدام از این بخش‌ها، یک برنامه بنویسید که ۶ عدد را به عنوان ورودی دریافت کند و آن‌ها را طوری چاپ کند که خروجی شبیه به یک جدول باشد؛ تمامی ستون‌ها به اندازه‌ی ۶ کاراکتر فضا داشته باشند و اعضای ستون دوم همگی تا ۳ رقم اعشار را نشان دهند. برای مثال با دریافت ۶ ورودی زیر جدول سمت چپ را خواهیم داشت.

33\*\*\*\* \*3.000 779892

\*67831 12.000 \*\*\*\*\*0

(علامت \* نشان‌دهنده‌ی فاصله است).

0.00 و 12 و 67831 و 779892.2422 و 3 و 33.0



# و اما سوال آخر: مرتب سازی



سلام بچه ها :) ورودتون یا شایدم ورود دوباره تون رو به برنامه نویسی تبریک می گم. فکر کنم دیگه تا الان حسابی توی سوالات دستورکار با ورودی و خروجی کار کردین و با نکاتش تا حد خوبی آشنا شدین. برای همین به نظرم بهتره که الان برگردیم سراغ شبه کد تا با هم دوباره برخی از نکاتی که لازمه توی الگوریتمها رعایت شه رو بررسی کنیم.



اول از همه، من هم سلام کنم و دوم از همه (!) در تکمیل صحبت های کدخدا بگم که اهمیت الگوریتم پیدا کردن و این که چه طور باید حرفمون رو به کامپیوتر بفهمونیم، خیلی از این که چه طور کدش رو بنویسیم مهم تره. برای تمرین بیشتر این توانایی، پیشنهاد می کنم تو پدیده های اطرافتون دنبال روندهای ریاضی و الگوریتمی بگردین و سعی کنین اون روندها رو به زبان ریاضی در بیارین. اینطوری بعدش خیلی راحت تر می تونین رابطه ی پیدا شده رو برای کامپیوتر قابل درک کنین.



مثلا یکی از این پدیده‌ها چی می‌تونه باشه؟

فرض کنین برای حل مسئله‌ای نیاز به مرتب کردن یه سری عدد داریم. اول از همه باید ببینیم ما خودمون چطور این کار رو انجام می‌دیم تا بتونیم راه‌حلمون رو مرحله به مرحله برای این دوست غیر هم‌زبانمون توضیح بدیم.



خب پس اول بیاین یه مثال بزنیم برای خودمون... ذهن ما برحسب عادت چه‌طور این رشته از اعداد رو مرتب می‌کنه؟ هر راه حل و روندی که تو ذهن‌تونه رو بگین.

$\{3, 11, 5, 2, 17, 4, 1\}$

راه‌های مختلفی هست. حالا ما باید یه طوری عمل کنیم که این روند رو -با کمک الگوریتم‌ها- برای کامپیوتر هم قابل درک کنیم.



یه راه می‌تونه چی باشه؟ اینکه اول بگردیم کوچک‌ترین عدد توی این رشته از اعداد کدومه و اون رو بذاریم اول رشته. بعد بریم توی بقیه ببینیم کوچک‌ترین کدومه و بره جای نفر دوم و ... همین‌طوری تا آخر ادامه‌بدیم تا جای که همه‌ی عدد هامون مرتب شده باشن.

به این الگوریتم می‌گن **selection sort**.



هر کدوم از لینک‌های زیر برای توضیح همین الگوریتم هستن. در صفحه‌ی بعد هم می‌تونین روند کلی الگوریتم رو ببینین.



[Selection Sort Algorithm](#)



[Selection Sort Algorithm Visualizer](#)



[Select-sort with Gypsy folk dance](#)





سعی کنیم با هم گروهی تون شبه کد این الگوریتم رو طوری بنویسین که برای کامپیوتر قابل درک باشه.

- 1- Find the smallest card. Swap it with the first card.
- 2- Find the second-smallest card. Swap it with the second card.
- 3- Find the third-smallest card. Swap it with the third card.
- 4- Repeat finding the next-smallest card, and swapping it into the correct position until the array is sorted.



کسی می تونه بگه منظور از قابل درک بودن چی هست؟  
مثلا اگه به یک آدمی که زبان شما رو متوجه نمی شه بگین کوچک ترین عدد بین این اعداد رو پیدا کن، متوجه حرفتون می شه؟  
پس چطور باید منظورمون رو بهش بفهمونیم؟



الگوریتم دیگه‌ای که می‌خوایم باهاش آشنا بشیم، مرتب‌سازی حبابی یا **bubble sort** هست. اهمیت الگوریتم به اینه که ببینید چندین روش مختلف برای مرتب‌سازی یه رشته از اعداد وجود داره.



این یکی رو سعی کنین از طریق لینک‌های زیر متوجه روندش بشین و بعد با کمک هم‌گروهی‌تون شبه‌کدش رو بنویسین.



[Bubble Sort Algorithm Visualizer](#)



[Bubble-sort with folk dance](#)



الگوریتم‌های مرتب‌سازی خیلی زیادن و محدود به همین دو مورد نیستن. اگر دوست داشتن الگوریتم‌های بیش‌تری رو ببینین با یه سرچ کوچولو به راحتی بقیه‌شون هم پیدا می‌شن. البته توی همون لینک‌هایی که برای selection sort و bubble sort بود هم می‌تونین دنبال بقیه‌شون بگردین؛)



امیدوارم که از این کارگاه لذت برده باشین و هم‌صحبتی با یه موجود زبون‌نفهم (البته خیلی باهوش که به موقعش خیلی خوب می‌فهمه و درس پس می‌ده) براتون دل‌نشین بوده باشه. تا جلسه‌ی بعد خدا نگه‌دار...

;