بسم الله الرحمن الرحیم

آموزش برنامه نویسی C++

سال اول دبیرستان

مرکز پرورش استعدادهای درخشان شهید بهشتی نیشابور

محمدرضا بلوکی

فهرست مطالب

فصل هفتم : اشنایی ساختمان

فصل هشتم : توابع بازگشتی

پیش گفتار

امروزهمگام با تحول و پیشرفت سخت افزار کامپیوتر ، نرم افزار نیز با همان مقدار و گاهی فراتر از آن ، دستخوش تغییر و بوده و هست . اگر چه در این حرکت ، هدف اصلی علوم نرم افزاری ایجاد برنامه های کاربردی ، برنامه های سیستمی و سایر برنامه های سودمند بوده است ؛ گاهی هدف به فراموشی سپرده شده و تنها بعنوان مصرف کننده نرم افزار از جنبه تولیدی این علم غافل می شوند.

از میان زبان های برنامه نویسی ، برنامه نویسی C++ ، به دلیل ارتباط نزدیک آن با سخت افزار سیستم ، از جایگاه ویژه ای برخوردار است و همین امر، این زبا ن را به یکی از ابزارهای قدرتمند ایجاد برنامه های سیستمی تبدیل است .

زبان C++ از واسط متنی جهت ارتباط با کاربر استفاده می کند و چون نسبت به زبان های دارای واسط گرافیکی نظیر C# از چشم انداز گرافیکی بهره نمی برد از محبوبیت کمتری برخودار است . اما چون امکانات مود نیاز کاربر باید از طریق کد ایجاد شود ، این زبان ابزاری مناسب و قدرتمند جهت تقویت کد نویسی است .

این کتاب آموزشی به منظور ارتقاء کیفی سطح یادگیری دانش آموزانی که می خواهند برنامه نویسی را بهتر و ساده تر بیاموزند ، تهیه شده است . این مجموعه مبتنی بر توضیحات مقدماتی در رابطه با ساختارهای C++ است ؛ با ذکر مثال های متعدد به شیوه ای خاص .

همه می دانیم که یادگیری برنامه نویسی بر دو اصل خلاقیت و پشتکار استوار است . در همین راستا در این کتاب آموزشی از روش جدیدی برای حل مسائل استفاده شده است . با تقسیم مسائل به بخش های کوچکتر و با حل آنها ، می توان راه حل را گسترش داد ه و نهایتا موفق به حل کل مسئله شد .
یادگیری این شیوه بسیار ساده است و با تمرین و ممارست می توان از عهده حل مسائل پیچیده برامد که در حالت عادی ممکن است حل آن غیر ممکن به نظر رسد.

آشنایی با زبان برنامه نویسی c++

زبان برنامه نویسی c++ از زبان برنامه نویسی c پیروی می کند. زبان c یک زبان ساخت یافته و زبان c++ زبانی شئ گراست.

ویژگی های زبان های ساخت یافته :

1- برنامه به اجزای کوچکتری به نام تابع تقسیم می شود.
2- مشکل اصلی این نوع برنامه نویسی جدا بودن داده ها از رویه ها یا متد هاست.
3- هر جزء از برنامه ی ساخت یافته می تواند به صورت مستقل عمل کند.

ویژگی های برنامه ی شئ گرا :

1- هر برنامه از قطعات نرم افزاری به نام شئی ساخته می شود که منجر به افزایش سرعت اجرا می گردد.
2- خوانا تر شدن برنامه و عیب یابی ساده تر.
3- پیاده سازی سریع تر برنامه.
4- کاهش حجم کد نویسی.

زبان c++ که در گروه زبان های شئ گرا قرار دارد دارای ویژگی های منحصر به فردی است که برخی از آن ها در بخش زیر آورده شده اند:

1- هیچ محدودیتی در برنامه نویسی وجود ندارد.
2- جزء زبان های برنامه نویسی سیستمی است چون ارتباط نزدیکی با اسمبلی و زبان ماشین دارد.
3- زبان برنامه نویسی کوچکی است لذا از سرعت اجرای بالایی برخوردار است .
4- نسبت به حروف حساس است (case sensitive) .

دستورالعمل های c++ داری ویژگی های زیر است:

1- هر دستور به ; ختم می شود.
2- حداکثر طول هر خط 255 نشانه است.
3- هر دستور را می توان در چند سطر نوشت .
4- در هر سطر می توان چند دستور تایپ کرد.

متغیر

داده های مورد نیاز برنامه در فضایی از حافظه RAM به نام متغیر ذخیره می شوند. در حقیقت هر متغیر شناسه یا نامی است که داده ها را در خود نگهداری می کند. در c++ باید قبل از استفاده از متغیر آن ها را تعریف کرد. الگوی زیر برای تعریف یا معرفی یک متغیر به کار می رود :

نوع	نام متغیر(ها)	;

انواع معتبر در زبان c++ عبارتند از: char , int , float , double , void . نوع char برای نگهداری یک نشانه ، نوع int برای نگهداری اعداد صحیح و نوع های float و double برای نگهداری اعداد اعشاری به کار می رود و نوع void معادل پوچ است.

نکته : با افزودن پیشوندهای **unsigned , signed , long , short** می توان نوع های زیر را استفاده کرد:

نوع	اندازه بر حسب بیت	محدوده
char	8	-128 to 127
unsigned	8	0 to 255
signed char	8	-128 to 127
int	16	-32768 to 32767
unsigned int	16	0 to 65535
signed int	16	-32768 to 32767
long int	32	-2G to 2G
unsigned long int	32	0 to 4G
float	32	7 رقم دقت 10^{38} to 10^{-38}
double	64	15 رقم دقت 10^{308} to 10^{-308}
long double	80	19 رقم دقت 10^{4932} to 10^{-4932}

شرایط نام گذاری شناسه ها یا نام متغیرها :

1- از حروف ، ارقام و (_)underscore تشکیل می شود.
2- نباید اولین نشانه ی آن رقم باشد.
3- حداکثر تعداد نشانه ها 31 است.
4- نباید جزء کلمات رزرو شده یا کلیدی باشد.

مثالهایی از تعریف متغیر:

```
int   x,y
unsigned long int   p;
char  ch1
```

نکته: در هنگام تعریف متغیر می توان به آن مقدار اولیه نسبت داد.مثلا :

```
int    s = 0,  count = 100;
char      p = 'a';
```

نکاتی در مورد مقادیرثابت معتبر در c++ :

1- ثابت های نشانه ای درون ' ' قرار می گیرند.
2- ثابت های رشته ای درون " " قرار می گیرند.
3- ثابت های عددی را می توان بصورت صحیح ، اعشاری یا نماد علمی استفاده کرد مثلا :

```
float   y = 3.75;
y = 3e5 ;
```

نکته : نماد علمی معتبر از دو بخش مانتیس و توان تشکیل شده است.که مانتیس عددی در بازه [1,10)و توان عددی صحیح می باشد. مثلا: 3.2e-2 که منظور $3.2*10^{-2}$ است.

$$485.3289 = 4.853289 * 10^{2}$$
$$0.0002487 = 2.487 * 10^{-4}$$

تعریف ثابت

ثابت ها ، شناسه هایی هستند که مقدار آن ها درون برنامه قابل تغییر نیست.برای اعلان ثابت ها از دستور const استفاده می کنیم:

const نوع داده نام ثابت = مقدار ثابت

مثلا :

```
const   int   n=100 ,  float  p = 3.14;
```

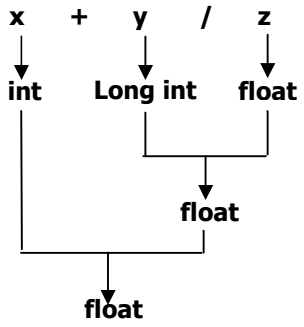
عملگر های ریاضی

نمادهایی هستند که برای اجرای اعمال خاص ایجاد شده اند. لیست عملگر های ریاضی به ترتیب اولویت در جدول زیر آمده است.

عملگر	نام	مثال
++ پیشوندی -- پیشوندی	افزایش یک واحد کاهش یک واحد	++x; --x;
* / %	ضرب تقسیم باقیمانده ی تقسیم صحیح	x*y x/y x % y
+ -	جمع تفریق	x+y x-y
= += -= /= *= %=	واگذاری ترکیب عملگر واگذاری با عملگرهای مقایسه ای	مثلا: s+=x; یعنی: s=s+x;
++ پسوندی -- پسوندی	افزایش یک واحد کاهش یک واحد	x++; x--;

نکته : نوع حاصل از اجرای یک عملگر بستگی به نوع عملوندهای آن دارد که معادل نوع داده ی بزرگتر است مثلا :

```
int x =5;
long int y;
float  z;
```



نکته : اگر داده یا مقدار اعشاری، درون متغیر صحیح قرار گیرد بخش اعشاری آن از بین می رود.

```
long int    x;
float   y = -46.91;
x = y;
```

نتیجه اجرای دستورات مقابل ، قرار گرفتن 46- در متغیر x است .

نکته : برای تبدیل یک نوع داده به نوع دیگر ، قبل از داده ی موردنظر، نوع دلخواه را قرار می دهیم (نوع یا داده را درون پرانتز قرار می گیرد) مثلا :

```
int   x ;
cout<<float (x) /2;
cout<< (float) x /2;    یا
```

در این مثال نوع متغیر x را به float تبدیل کرده و بر 2 تقسیم می کنیم ، تا حاصل اعشاری شود.

نکته : در تبدیل int به char ، تنها از 8 بیت با ارزش کمتر عدد استفاده می شود.

نکته : در تبدیل نوع long به int ، از 16 بیت با ارزش کمتر استفاده می شود.

نکته : در تبدیل double به float ، دقت عدد کم می شود.

ساختار برنامه ی c++

برنامه c++ از تعدادی تابع و کلاس تشکیل می شود. بدنه ی اصلی برنامه ، تابع main() است. به عبارت دیگر هیچ برنامه ای بدون تابع main اجرا نمی شود. الگوی یک برنامه ی ساده در c++ به صورت زیر است :

```
#include< فایل سرتیتری یا سرآیند >
void main ()
{
        اعلان داده ها
        دستورت اجرایی

}
```

نکته: توابع و سایر اطلاعات مورد نیاز برنامه درون فایل هایی با پسوند h. بنام header file قرار دارد ، که برای اتصال این فایلها به برنامه خودمان، آن ها در بخش include به برنامه معرفی می کنیم.

نکته: در صورت استفاده از چند فایل سرآیند باید برای هرکدام از دستور #include جداگانه استفاده کرد.

نکته: دستور العمل یک تابع با { شروع و به } ختم می شود.

نکته: کلمه void کنار main نوع بازگشتی تابع main است ، به این مفهوم که این تابع مقدار بازگشتی ندارد. اگر این کلمه حذف شود ، پیش فرض int است ، در این حالت برای جلوگیری از ظاهر شدن اخطار (warning) باید در انتهای برنامه از دستوری نظیر ; return 0 استفاده کرد.

نکته: توضیحات درون برنامه جهت راهنمایی برنامه نویس نوشته می شوند و جنبه ی اجرایی ندارند.این توضیحات درون /* */ قرار می گیرند. در ضمن برای این که یک خط به عنوان توضیح مشخص شود ، ابتدای آن می توان از // استفاده کرد.

نکته: از نکته قبل می توان برای اجرا نشدن یک یا چند خط برنامه استفاده کرد . یعنی آنها را به عنوان توضیحات درون برنامه مشخص کرد.

دستورات ورودی و خروجی

معمولا داده های مورد نیاز برنامه را از طریق صفحه کلید دریافت می کنیم و نتایج بدست آمده از اجرای برنامه ، توسط صفحه نمایش ، در اختیار کاربر قرار می گیرد .

دستور چاپ cout:

cout شئ است که برای چاپ اطلاعات به کار می رود و در فایل iostream.h قرار دارد. نحوه ی استفاده از این شئی به صورت زیر است :

```
cout<<1 عبارت >>2 عبارت >>...;
```

نکته: برای چاپ متن یا رشته ی ثابت ، باید آن را درون " " قرار داد.
نکته: برای کنترل چاپ عبارت ها در خروجی و هم چنین چاپ نشانه های خاص ، از کاراکتر های کنترلی استفاده می کنیم. این کاراکتر ها با \ شروع می شود. برخی از کاراکتر های کنترلی در جدول زیر آورده شده است:

کاراکتر کنترلی	عملکرد
\n	انتقال مکان نما به خط بعدی
\t	انتقال مکان نما به هشت ستون بعدی
\a	شنیدن صدای Beep
\b	حذف کاراکتر قبل (معادل کلید Back space)
\r	معادل کلید Enter
\\	کاراکتر \ چاپ می شود
\"	کاراکتر " چاپ می شود
\?	کاراکتر ؟ چاپ می شود

مثال) خروجی دستورات مقابل چیست ؟

```
x=10 ;          y=25 ;
cout << x  << y  << "\n" ;
cout << x << "\t" << y <<"\n" ;
cout << "x=" <<x<<"\n\n" ;
cout <<"x+y=\? " << x+y ;
```

```
1025
10      25
x=10

x+y=?35
```

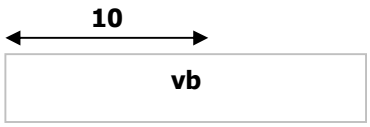
مثال) خروجی دستورات مقابل چیست ؟

```
cout <<"\"turbo\"\n" ;
cout <<"vissaul \n basic 6.0 \b\b\b" ;
```

```
"turbo"
Visual
basic
```

نکته : برای تعیین میدان یامحدوده ی چاپ یک عبارت ، از تابع **setw(n)** استفاده می کنیم . عبارتهای قرار گرفته بعد از **setw** ، از یک فضای **n** تایی ، به عنوان میدان تایی استفاده می کنند و حاصل عبارت در سمت راست میدان چاپ می شود . این متد در **iomanip.h** قرار دارد .

مثال)




```
cout <<setw(10)<<"vb" ;
```

خواندن داده ها ورودی با cin :

این شئ برای خواندن داده ها از صفحه کلید بکار می رود و همانند **cout** در **iostream.h** قرار دارد.

```
cin >>1 متغیر >>2 متغیر>>... ;
```

نکته : برای خواندن چند داده در یک خط ، هنگام وارد کردن داده ها باید حد اقل بین آن ها یک فاصله قرار گیرد .

مثال) با دریافت شعاع دایره مساحت و محیط آن را چاپ کنید .

```
#include <iostream.h >
void main ( )
{
   float    r , s , p ;
   cout << " enter the radius : "
   cin >> r ;
   s = r * r * 3.14 ;
   p = 2 * r * 3.14 ;
   cout << " area = "  << s << " \t " << " prime = " << p ;
}
```

توضیح :

پس از دریافت یک عدد و قرار دادن آن در متغیر **R** ، با روابط شناخته شده در دایره ، مساحت و محیط دایره ای به شعاع **R** را بدست آورده و درون متغیر های **P** و **S** قرار داده ایم . نهایتاً **P** و **S** را بعنوان خروجی چاپ می کنیم .

نکته : برای اجرای برنامه از منوی **Run** گزینه ی **Run** استفاده می کنیم . یا از کلید میانبر **Ctrl + F9** استفاده می کنیم .در ضمن برای ترجمه برنامه از **Alt +F9** استفاده می کنیم .

نکته : پس از اجرای برنامه و وارد کردن داده های ورودی ، به محیط ویراستار بر می گردیم . بدون اینکه نتیجه ی برنامه را دیده باشیم . برای رفتن به پنجره ی کاربر(**User Screen**) از **Alt + F5** استفاده می کنیم .

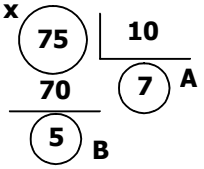
نکته : بهتر است در پایان برنامه از تابع **() getch** استفاده کنیم تا برنامه متوقف شده و پس از مشاهده ی خروجی ؛ با فشردن یک کلید به محیط ویراستار باز گردیم . این تابع در فایل سر تیتری **conio.h** قرار دارد .

مثال) برنامه ای بنویسید که با دریافت یک عدد دو رقمی مجموع ارقام آن را چاپ کند .

توضیح :

در مثال قبل رابطه ی بین ورودی **R** و خروجی های **P** و **S** را می دانستیم و از همان روابط برای حل مسئله استفاده کردیم . در این مثال برای پیدا کردن راه حل مسئله ابتدا یک مثال عددی حل می کنیم .

- برای جداسازی ارقام ، از تقسیم بر **10** استفاده می کنیم .



پس از حل مثال عددی ، به اعداد متغیر موجود در مثال ، نام مناسبی نسبت می دهیم (مقسوم علیه ثابت است پس نیاز به اختصاص متغیر ندارد اما مقسومِ ، خارج قسمت و باقی مانده را نام گذاری کرده ایم) . عملیات اجرا شده بر روی مثال عدد ی را ، در قالب برنامه پیاده سازی می کنیم .

```
# include <conio .h >
#  include < iostream .h>
void main()
{
   int    x , A , B , sum ;
   clrscr ( ) ;
   cout <<  "x =
   cin >> x ;
   A= x / 10;
   B=x % 10; "
   sum = A+B ;
   cout <<"sum = " << sum ;
   get ch ( );
}
```

تمرین :

1- برنامه ای بنویسید که با دریافت سه عدد ، میانگین آن ها را چاپ کند .

2- برنامه ای بنویسید که با دریافت جرم جسم بر حسب پوند ، آن را بر حسب کیلو و گرم چاپ کند . مثلاً با وارد کردن عدد 10 پوند ، دو مقدار 4 کیلو گرم و 530 گرم چاپ شود . (هر پوند 453 گرم است)

3- با دریافت طول جسمی بر حسب متر ، سانتی متر و میلی متر ِ طول جسم را به میلی متر تبدیل کند .

4- 10% درصد حقوق یک کارمند ، مالیات حقوق است . پس از دریافت مقدار حقوق ، میزان مالیات آن را چاپ کنید .

5- با دریافت عدد X رقم یکان آن را چاپ کنید .

6- زمان جاری h ساعت، m دقیقه و s ثانیه است . تعیین کنید چند ثانیه از شبانه روز سپری شده است .

7- با دریافت زمان جاری بر حسب ثانیه ، آن را به ساعت ، دقیقه و ثانیه تبدیل کنید .

8- شرکتی به هر کدام از کارمندان خود مبلغ X ریال پرداخت می کند . اگر 15 درصد به حقوق هر یک اضافه کند ، شرکت باید چقدر هزینه ی بیشتری بابت 10 نفر پرداخت کند .

9- طول جسمی x یارد ، y فوت و z اینچ است . طول جسم را به متر ، سانتی متر و میلی متر تبدیل کنید .

3 فوت = 1 یارد 12 اینچ = 1 فوت 25 میلی متر = 1 اینچ

10- پس از دریافت عدد سه رقمی x جای رقم های یکان و دهگان آن را تغییر داده و جذر عدد حاصله را چاپ کند . (برای پیدا کردن جذر از تابع sqrt(X) استفاده می کنیم و این تابع را در فایل سر آیند math . h است .)

11- پس از دریافت مقدار یک زاویه بر حسب درجه ، sin آن را چاپ کنید . (برای پیدا کردن sin از تابع sin(x) که در math .h است استفاده می کنیم

راهنمایی : ورودی این تابع بر حسب رادیان است که با رابطه ی R = D *3.14 /180 می توان درجه را به رادیان تبدیل کرد .

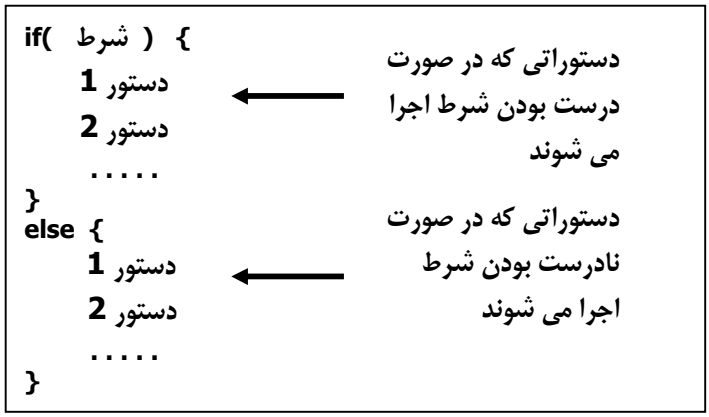
12- با دریافت یک نشانه کد اسکی آن را چاپ کند .

ساختارهای کنترلی

در فصل قبل با دستورات C++ آشنا شدیم . برای حل مسائل برنامه نویسی ، علاوه بر امکانات فصل قبل ، به ساختارهای شرطی و تکراری نیاز داریم . ساختارهای کنترلی زبان C نسبت به سایر زبان ها ، از انعطاف پذیری بیشتری برخوردار است . به طوری که با حداقل تعداد دستورات می توان مسائل را پیاده سازی کرد .

ساختار if :

این ساختار با بررسی شرط مسئله ، از میان دو حالت ممکنه یکی را اجرا می کند . یعنی در صورت درست بودن شرط ، گروهی از دستورات ودر صورت نادرست بودن ، گروه دیگری از دستورات اجرا می شود .



نکته : بخش else اختیاری است و در صورت نیاز ، تنها می توان از بخش if استفاده کرد .
نکته : اگر بخش if یا else فقط یک دستور را اجرا کنند ، می توان { } را حذف کرد .

عملگر های منطقی و مقایسه ای

برای ایجاد عبارت های شرطی که ارزش درست یا نادرست داشته باشند از عملگر های مقایسه ای یا رابطه ای استفاده می کنیم . این عملگر ها عبارتند از:

==	تساوی	>=	بزرگتر مساوی
=!	نا مساوی	<	کوچکتر
>	بزرگتر	=<	کوچکتر مساوی

برای ترکیب عبارت های مقایسه ای از عملگر منطقی && (and) یا !! (or) استفاده می کنیم .
جدول درستی این عملگر ها به صورت زیر است :

A	B	A && B	A || B
F	F	F	F
F	T	F	T
T	F	F	T
T	T	T	T

نکته : عملگر ! (not) برای برعکس کردن نتیجه ی شرط بکار می رود .

مثال) برنامه ای بنویسید که با دریافت عدد x زوج یا فرد بودن آن را گزارش دهد .

توضیح :

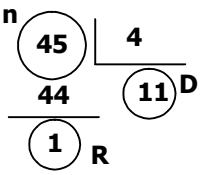
برای تعیین زوج یا فرد بودن عدد ، باقیمانده عدد بر 2 را بدست می آوریم . اگر باقیمانده صفر شود عبارت even (زوج) وگرنه عبارت odd (فرد) را چاپ می کنیم .

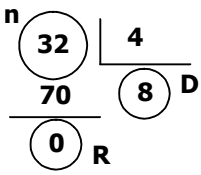
```
# include<conio .h >
# include<iostream .h>
void main ( )
{
     int  x , r ;
     cout<< "x = " ;
     cin>> x ;
     r = x % 2;
     if ( r == 0 )
          cout << "even " ;
     else
        cout << "odd " ;
     getch ( ) ;
}
```

مثال) با دریافت تعداد نشانه های پیام کوتاه ، هزینه مکالمه را بدست آورید . با این شرط که هزینه هر 4 نشانه یا کمتر ، 20 ریال است .

توضیح :

برای بدست آوردن راه حل مسئله ، ابتدا دو مثال عددی فرضی حل می کنیم :





برای پیدا کردن تعداد نشانه های چهار تایی درون عدد ورودی 45 ، آن را بر 4 تقسیم کرده ایم . 11 چهار نشانه وجود دارد که هزینه آن 220 ریال است . چون باقیمانده مخالف صفر است ، باید 20 ریال دیگر به هزینه اضافه شود . چون هزینه ی کمتر از چهار نشانه نیز 20 ریال است .

برای نوشتن برنامه ابتدا به اعداد درون مثال متغیر نسبت می دهیم و بر اساس ورودی n ، باقیمانده و خارج قسمت را بدست می آوریم . سپس توسط ساختار شرطی if ، هزینه را برای دو حالت پیش آمده ، حساب می کنیم .

روش اول :

```
#include<conio.h >
#include<iostream.h>
void main ( )
{
      int   n , D , R , b ;
     cout << "enter a number  =" ;
     cin >> n ;
     D = n / 4 ;
     R = n % 4;
     if  ( R  == 0 )
          s = D * 20 ;
     else
          s = D * 20 +20 ;
     cout << s ;
     getch ( ) ;
}
```

روش دوم :

```
#include<conio.h >
#include<iostream.h>
void main ( )
{
     int  n , D , R , b ;
     cout << "enter a number  =" ;
     cin >> n ;
     D = n / 4 ;
     R = n % 4;
     if  ( R != 0 )
          D ++;
     s = D*20
     cout << s ;
     getch ( ) ;
}
```

تمرین

1- تعیین کنید عدد x بر y بخش پذیراست یاخیر؟
2- بااستفاده ازرقم یکان عدد ، تعیین کنید عدد x بر 5 بخش پذیراست یاخیر؟
3- تعیین کنیدعدد ورودی x بر 2 یا 3 بخش پذیراست یاخیر ؟
4- با دریافت عدد x قدرمطلق آن را چاپ کنید ؟
5- با دریافت عدد x گرد شده آن را چاپ کنید ؟
6- هزینه هر دقیقه مکالمه 20 است . با دریافت تعداد دقایق مکالمه تلفن هزینه را بدست آورید به شرط اینکه ، کمتراز 3 دقیقه باید هزینه 3 دقیقه را پرداخت کند ؟
7- با دریافت عدد x جذر آن را چاپ کنید ؟(اعداد منفی جذر حقیقی ندارند)
8- با دریافت تعدا دقایق مکالمه تلفن هزینه مکالمه را بدست آورید .
   a. کمتر از 10 ، هزینه هر دقیقه 30 ریال است
   b. بیشتر از 10 دقیقه ، 10 دقیقه اول هر دقیقه 30 ریال و به هزینه باقی دقایق 4٪ اضافه می شود.
9- در صورتی که نشانه ورودی جزء ارقام 0 تا 9 است ، پس از کم کردن 48 واحد از کد اسکی نشانه ورودی ، آن را چاپ کنید ؟
10- با دریافت تاریخ یکی از روزهای سال تعداد روزهای سپری شده از سال را چاپ کنید ؟
11- با دریافت تعداد روزهای سپری شده ازسال، تاریخ آن روز را چاپ کنید ؟
12- با دریافت تاریخ تولد یک فرد تعیین کنید چند سال ، چند ماه و چند روز از عمر فرد سپری شده است ؟
13- با دریافت 3 عدد ، اگر این 3 عدد یک مثلث متساوی الاضلاع تشکیل می دهند، مساحت و محیط آن را چاپ کنید ؟
14- با دریافت سه عدد ، کمترین مقدار بین آن ها را چاپ کنید .
15- با دریافت سه عدد ، تعیین کنید عدد بزرگتر چندمین عدد ورودی است .
16- با دریافت سه عدد ، مقادیر درون آن ها را جابجا کرده بطوریکه لیست خروجی صعودی باشد.

ساختار if...else متداخل یا لانه ای

به وسیله **if** می توان از میان دو حالت ، یکی را انتخاب و اجرا کرد. اگر بخواهیم از میان چندین حالت ممکن ، یکی را اجرا کنیم ، بجای بکار بردن **if** های مجزا ، از **if...else** های متداخل استفاده می کنیم تا سرعت اجرا افزایش یابد . این ساختار را می توان به صورت زیر استفاده کرد :

```
if( شرط ) {
   .....
   .....
}
else if ( شرط )  {
   .....
   .....
}
.....
.....
else {
   ....
   ....
}
```

مثال) بادریافت نشانه و دو عدد ورودی **x,y** اعمال زیررا اجرا کنید .

- اگر ch ='+' : x+y
- اگر ch ='-' : x-y
- اگر ch ='*' : x*y
- اگر ch ='/' : x/y
- در غیر این صورت پیغام مناسبی چاپ شود .

```
#include<iostream.h>
#include<conio.h>
void main( )
{
   Clrscr();
   float  x , y;
   char   ch ;
   cout<< "enter two  numbers  =" ;
   cin>>x>>y ;
   cout<<"enter  character =" ;
   cin>>ch;
   if ( ch == '+')
       cout<< x+y ;
   else if ( ch == '-' )
       cout<<x-y;
   else if ( ch == '*' )
       cout<<x*y;
   else if ( ch == '/' )
       cout<<x/y;
   else
       cout<<"invalide data";
}
```

توضیح: پس از دریافت ورودیهای **ch,x,y** ، مقدار **ch** را بررسی می کنیم و متناظر با مقدار **ch** یکی از اعمال

ریاضی را اجرا می کنیم . اگر هیچ کدام از شرط ها درست نباشد دستور else آخری اجرا می شود .

مثال) با دریافت شماره ی یکی از ماههای سال تعداد روزهای آن ماه را به همراه نام فصل آن ماه چاپ کنید .

```
#include<iostream.h>
#include<conio.h>
void main( )
{
   clrscr();
   int  x ;
   cout<< "enter no. of month =" ;
   cin>>x ;
   if ( x>0 && x<=12 ) {
      if( x<=3) {
         cout<<"spring"<<"\t";
         cout<<31 <<"\n";
      }
      else if( x<=6) {
         cout<<"summer"<<"\t";
         cout<<31 <<"\n";
      }
      else if(x<=9) {
         cout<<"autumn"<<"\t";
         cout<<30 <<"\n";
      }
      else if(x<=11) {
         cout<<"winter"<<"\t";
         cout<<30 <<"\n";
      }
      else {
         cout<<"winter"<<"\t";
         cout<< 29<<"\n";
      }
   }
   else
       cout<<"data out of  range";
   getch();
}
```

توضیح :

ابتدا توسط (if (x>=0 && x<=12 معتبر بودن داده ی ورودی را تعیین بررسی کرده ایم اگر شرط درست باشد ورودی x را بررسی کرده و تعداد روزهای آن ماه و نام فصل آن را چاپ می کنیم و گرنه پیغام خارج بودن داده ی ورودی از محدوده ی معتبر را نمایش می دهیم . در شرط داخلی اگر x<=3 باشد عدد 31 و کلمه spring را چاپ می کنیم اگر شرط غلط باشد بخش else اجرا می شود یعنی x<=3 نیست پس x>3 است اما با شرط x<=6 آن را محدود کرده ایم یعنی اگر 3<x<=6 باشد عدد 31و کلمه summerرا چاپ کرده ایم و....

تمرین

1-با دریافت نمره ی یک دانش آموز ، بر اساس نمره ی ورودی و شرایط زیر. حروف **D,CB,A** را چاپ کنید .

a. اگر **18**< نمره باشد حرف **A** چاپ شود .
b. اگر **15**< نمره= < **18** باشد حرف **B** چاپ شود .
c. اگر **10**< نمره= < **15** باشد حرف **C** چاپ شود .
d. اگر نمره < **10** باشد حرف **D** چاپ شود .

2-با دریافت حقوق یکی از کارکنان شرکت و شرایط زیر ، میزان حقوق دریافت شده و مالیات را چاپ کنید .

a. اگر **10000**> حقوق باشد معاف از مالیات
b. اگر **1000000**> حقوق => **100000** باشد **7**٪ مالیات
c. اگر **1000000**=< حقوق باشد **10**٪ مالیات

3-با دریافت شماره ی یکی از روزهای هفته نام آن روز را چاپ کنید .

4-با دریافت سه عدد **x,y,z** اگر این سه عدد تشکیل مثلث را می دهند تعیین کنید مثلث متساوی الاضلاع ، متساوی الساقین یا قائم الزاویه است یا مثلث مشخصی نیست ؟

5-با در یافت تاریخ یکی از روزهای سال تعیین کنید روز مورد نظر چند شنبه است در حالی که می دانیم روز اول سال دوشنبه است ؟

ساختار **switch**

شکل ساده تری از دستور if....else متداخل است که برای اجرای یک حالت ، از میان چندین حالت ممکن به کار می رود :

```
switch( عبارت ) {
   case  مقدار 1 :
     .....
     .....
   break;
   case  مقدار 2 :
     .....
     .....
   break;
   .....
   .....
   default :
     .....
     .....
}
```

اگر عبارت دستور switch با مقدار 1 برابر باشد ، مجموعه دستورات case اول و اگر با مقدار 2 برابر باشد مجموعه دستور case دوم .... اجرا می شود و اگر هیچ حالتی بر قرارنباشد ، دستورات بخش default اجرا می شود .

نکته : عبارت switch می تواند از انواع صحیح یا نشانه ای باشد .

نکته : ساختار switch فقط شرط تساوی را بررسی می کند .

نکته : اگر مقدار دو تا از case ها برابر باشد پیغام خطا ظاهر می شود .

نکته : بخش default اختیاری است .

نکته : وجود break برای هر case لازم است و گرنه پس از اجرای case , حالات بعدی ، بدون درست بودن شرط شان اجرا می شوند . مثلا :

```
n=2;
switch(n) {
   case  1 :  cout<<"A";
   case  2 :  cout<<"B";
   case  3 :  cout<<"C";
   case  4 :  cout<<"D";
}
```

چون n=2 است پس case 2: اجرا می شود یعنی B چاپ می شود اما چون break ندارد سایر case ها نیز اجرا می شوند یعنی CD نیز چاپ می شود . پس خروجی این مثال BCD است .

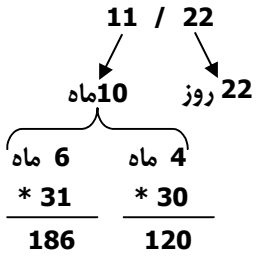
نکته : در حالت های switch می توان عملگر or (یا) را بصورت زیر شبیه سازی کرد :

```
switch  (ch)  {
   case 'a':
   case 'A': cout << 1 ;
   break ;
     ....
     ....
}
```

در این مثال اگر ch='A' یا ch='a' باشد عدد 1 چاپ می شود .

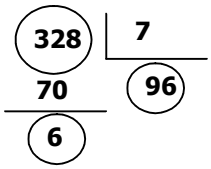
مثال) با دریافت شماره ی یکی از روزهای سال ، تعیین کنید روز مورد نظر چه روزی از ایام هفته است. درصورتی که می دانیم 11/22 دوشنبه است .

توضیح : ابتدا تعداد روزهای سپری شده از سال را بر اساس تاریخ 11/22 بدست می اوریم :

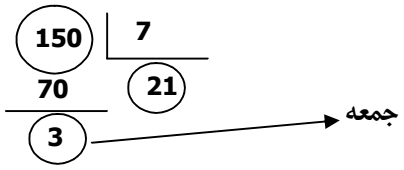


تعدا د روزهای سپری شده =186+120+22=328

با تقسیم این عدد بر 7 تعداد هفته های کامل را جدا کرده ، باقیمانده تقسیم را بدست می آوریم .



همین عمل را بر روی عدد ورودی نیز تکرار می کنیم . یعنی عدد ورودی را بر 7 تقسیم کرده و باقیمانده ی تقسیم را بدست می آوریم اگر باقیمانده 6 باشد دوشنبه و اگر 5 باشد یکشنبه و .... خواهد بود . مثلا اگر ورودی 150 باشد :




```
#include<iostream.h>
#include<conio.h>
void main( )
{
   int x,r;
   clrscr ( ) ;
   cout<<"x ="; cin>>x ;
   r = x % 7;
   switch (r)   {
      case 6 : cout<<"monday";
      break;
      case 0 : cout<<"Tuesday";
      break;
      case 1 : cout<<"Wednesday ";
      break;
      . . . . .
    }
}
```

تمرین : تمرین 3 ، 5 و مثال 2 ، 1 بخش قبل (if.....else متداخل) را با استفاده از switch حل کنید.

ساختار تکرار for(1)

برای تکرار گروهی از دستورات به تعداد دفعات معین ، مناسب ترین ساختار **for** است . این ساختار تا زمانی که شرط حلقه درست باشد ، دستورات حلقه را تکرار می کند.

```
for ( دستوراتی که قبل از شرط حلقه اجرا می شوند ; شرط حلقه ; دستوراتی که بعد از دستورات حلقه اجرا شوند ) {
    ..................
    ..................
}
```

مثال) با دریات شعاع **5** دایره (یکی یکی) مساحت و محیط آن ها را چاپ کنید.

توضیح :

ابتدا بدون استفاده از ساختارهای تکرار مسئله را حل می کنیم و با دریافت شعاع یک دایره مساحت و محیط آن را چاپ می کنیم :

```
cout<<"enter theradivs";
cin>>r;
s = r* r*3.14;
p = 2*r*3.14;
cout<<s<<"\t"<<p<<"\n";
```

پس از چاپ مساحت و محیط اولین دایره ، به مقادیر درون متغیرهای **R , s , p** نیازی نداریم ، پس می توانیم از این متغیرها و همان قطعه برنامه قبل ، برای دریافت شعاع دایره دوم و چاپ مساحت و محیط آن استفاده کرد.

برای دایره های سوم ،چهارم و پنجم نیز همین قطعه برنامه تکرار می شود.

برای مشخص کردن اینکه قطعه برنامه چند بار اجرا شود از یک متغیر به عنوان شمارنده ی حلقه استفاده می کنیم.

این متغیر در مرحله اول **I = 1** ، برای اجرا دومین بار **I = 2** و برای پنجمین بار **I = 5** خواهد بود ، یعنی با هر بار اجرای قطعه برنامه فوق به این متغیر یک واحد اضافه می شود تا **I** به **5** برسد.

مقدار اولیه متغیر حلقه را در بخش اول **for** ، افزایش یک واحد به آن را در بخش سوم و شرط **I<=5** را در بخش وسط ساختار **for** قرار می دهیم . سپس می نویسیم.

```
#include<iostream.h>
#include<conio.h>
void main ( )
{
   int I;
   float R,S,P;
   for (I =1 ; I ,=5; I++) {
      cout <<"Enter the radivs :";
      cin>>R;
      S = R*R*3.14;
      P = 2*R*3.14;
      cout<<S<<"\t"<<P<<\n";
   }
   getch( ) ;
 }
```

مثال) مجموع 10 عدد ورودی را چاپ کنید.
توضیح : در این مثال نیز اعداد را یکی یکی دریافت می کنیم ؛ یعنی به کمک یک حلقه 10 بار عمل دریافت را تکرار می کنیم.

```
for ( i = 1; i <= 10 ; i + +) {
  cout <<"x =";
  cin>>x ;
  . . . . . .
 }
```

همانطور که دیده می شود اعداد ورودی یکی یکی دریافت شده و درون x قرار می گیرند. بنابراین با دریافت یک عدد ، مقدار قبلی x از بین می رود. برای پیدا کردن مجموع x های ورودی ، از یک متغیر کمکی نظیر S با مقدار اولیه صفر به صورت S = S + x; استفاده می کنیم.

```
#include<iostream.h>
#include<conio.h>
void main ( )
{
   int x , I , s = 0;
   for(  i=1; i<=10;i + +) {
      cout<<"x =";
      cin>>x ;
      s = s + x;
   }
   cout<<"sum ="<<s;
   getch( );
}
```

تحلیل :
با دریافت اولین x و جمع آن با S = 0 ، اولین مقدار درون S قرار می گیرد . پس از دریافت عدد x دوم ، آن را با عدد x اول که درون S است جمع کرده و حاصل جمع را درون S قرار می دهیم ، در نتیجه درون S مجموع دو عدد اول قرار می گیرد. مقدار S را با عدد سوم جمع می کنیم و حاصل را باز هم در S می ریزیم . . . نهایتاً مجموع 10 عدد ورودی که یکی یکی درون x قرار گرفته بودند ، در S خواهد بود .

تمرین :

1-از میان **10** عدد ورودی ، اعداد مثبت را به همراه جذر آن ها چاپ کنید.
2-با دریافت مقدار **10** زاویه بر حسب درجه، **sin** و **cos** هر یک را چاپ کنید.
3-از میان **10** عدد ورودی کمترین مقدار را چاپ کنید.
4-با دریافت **10** عدد ورودی ، تعیین کنیدکدام عدد بیشترین مقدار بین **10** عدد بوده است.
5-با دریافت **10** عدد ، تعداد اعداد زوج و فرد را جداگانه چاپ کنید.
6- با دریافت نمره و تعداد واحد (ضریب) **5** درس ، معدل دانش آموز را حساب کنید.
7-تعداد و قیمت **5** کالای خریداری شده توسط یک مشتری را دریافت و هزینه کل خرید را چاپ کنید.
8-با دریافت **x , n** حاصل x^n را چاپ کنید.
9-با دریافت نمره میانی،پایانی و مستمر های یک درس **10** دانش آموز ، نمره هر دانش آموز را در آن درس چاپ کنید.
10- میزان حقوق پایه **20** کارمند شرکتی دریافت می شود . با فرض اینکه **10٪** حقوق پایه ، مالیات هر فرد باشد ، میزان کل مالیات کارکنان و حقوق دریافتی آنها را چاپ کنید.

ساختار for (2)

مثال) اعداد طبیعی کمتر از n را چاپ کنید.

توضیح : می خواهیم در خروجی اعداد مقابل را چاپ کنیم . 1 2 3 4 . . . n

ابتدا برای چاپ تک تک اعداد ، از دستورات جداگانه استفاده می کنیم و اعداد را یکی یکی چاپ می کنیم . برای چاپ عدد 1 دستور ساده زیر را می نویسیم :

```
cout<<1;
```

برای چاپ عدد 2 نیز همین دستور را تکرار می کنیم : cout<<2;

برای چاپ عدد های بعدی نیز همین تک دستور باید تکرار شود. پس با حلقه for این دستور را تکرار می کنیم.

در ضمن عدد چاپ شده ثابت نیست پس یک متغییر به آن نسبت می دهیم که از 1 تا n تغییر کند ، این متغیر همانند متغیر حلقه در مثال های قبل است . پس می نویسیم :

```
#include <iostream.h>
void main ( )
{
   int i , n;
   cout<<"n =" ;
   cin >>n;
   for( i =1; i <=n; i+ +)
       cout << i <<"\t";
}
```

- بدلیل تک دستور بودن دستورات حلقه از { } استفاده نشده است. به عبارت دیگر اگر برای For از {} استفاده نشود ، دستور بعد از آن به عنوان دستور حلقه تکرار می شود .

مثال) اعداد زوج بین A,Bرا چاپ کنید بشرط اینکه A<B باشد.

روش اول:

به کمک یک حلقه for اعداد صحیح بین A,B را تولید می کنیم . سپس هر عدد را بررسی کرده و در صورت زوج بودن ، آن را چاپ می کنیم.

حلقه زیر ، توسط متغیر حلقه i اعداد A تاB را ایجاد می کند.

```
for ( i=A ; i <=B; i++)
```

از میان این i ها ، اعداد زوج را چاپ می کنیم یعنی می نویسیم :

```
if (i % 2 == 0)
   cout<<i;
```

پس برنامه کامل به صورت زیر خواهد شد :

```
#include <iostream.h>
void main ( )
{
   int A,B,i;
   cout <<"A,B=";
   cin>>A>>B;
   for( i = A ; i <=B: i + +)
      if( i % 2 = = 0)
          cout<<i<<"\t";
}
```

روش دوم :

در این روش ابتدا عدد A را از نظر زوج بودن بررسی می کنیم ، در صورتی که فرد باشد ، آن را به عدد زوج بعدی تبدیل می کنیم. در هر صورت می خواهیم مقدار اولیه درون A زوج باشد .

```
if ( A % 2 == 1) A+ +;
```

متغیر حلقه را از A شروع می کنیم و اولین عدد زوج را چاپ می کنیم ، برای چاپ عدد زوج بعدی به متغیر حلقه 2 واحد اضافه می کنیم . در این صورت دیگرنیازی به دستور if داخل حلقه نیست .

```
#include <iostream.h>
void main ( )
{
    int  A,B,I;
    cout <<"A,B =";
    cin<<A<<B;
    if(A%2 ==1) A++;
    for(I= A;I<=B;I= I+2)
        cout<<I<<"\t";
}
```

مثال) مجموع مقسوم علیه ها x را چاپ کنید.

همان طور که می دانیم مقسوم علیه های یک عدد ، می تواند در محدوده ی 1 تا خود عدد باشد ، پس عدد x را بر تک تک این اعداد تقسیم کرده و اگر باقیمانده تقسیم صفر شود ، عدد مورد نظر ، مقسوم علیه x است . سپس مجموع آن ها را بدست می آوریم.

ابتدا عددx را بر 1 تقسیم می کنیم و مقسوم علیه بودن آن را بررسی می کنیم.

```
if(x%1 == 0)
    s+=1;
```

سپس این عمل را بر روی عدد 2 تکرار می کنیم

```
if(x%2 == 0)
    s+=2;
```

توسط این دستور اعداد 3 ،4 ،...x را نیز بررسی می کنیم . به جای این اعداد متغیر I را قرار داده و توسط حلقه for این دستور را تکرار می کنیم :

```
for(i=1;i<=x;i++)
  if(x%i == 0)
    s+=i;
```

در ضمن مجموع i هایی که شرط مورد نظر را داشته باشد یعنی مقسوم علیه x باشند را به کمک دستور s+ = i; با هم جمع کرده ایم . نهایتا s را چاپ می کنیم .

```
#include<iostream.h>
void main( )
{
  int x,i,s=0;
  cin>>x;
  for(i=1;i<=x;i++)
    if(x%i= = 0)
      s+=i;
  cout<<s;
}
```

مثال) با دریافت دو عدد x,n حاصل x^n را چاپ کنید.

توضیح :

برای محاسبه x^n از رابطه ریاضی $x\times x\times x\times ...$ به تعداد n بار استفاده می کنیم. برای تکرار یک عمل ، به تعداد n بار ، از حلقه ای نظیر حلقه زیر استفاده می کنیم:

```
for(i=1;i<=n;i++)
    . . . . . .
```

هر دستور یا دستوراتی که درون این حلقه قرار گیرد n بار تکرار می شود.ما می خواهیم ضرب x ها در این حلقه را تکرار کنیم ، پس از متغیری کمکی نظیر p با مقدار اولیه 1(عضو خنثی در ضرب) بصورت p = p* x ; استفاده می کنیم . یعنی می نویسیم:

```
#include<iostream.h>
void main( )
{
    int x,n,i;
    double p=1;
    cout<<"x,n=";
    cin>>x>>n;
    for(i=1;i<=n;i++)
        p*=x;
    cout<<p;
}
```

نکته : چون مقدار p به سرعت زیاد می شود آن را از نوع double انتخاب کرده ایم.

تمرین:

1-اعداد فرد طبیعی کوچکتر مساوی n را چاپ کنید.

2-مجموع اعداد طبیعی کمتر از n را چاپ کنید.

3-اعداد طبیعی قابل قسمت بر 2 یا 3 کمتر از n را چاپ کند.

4-مجموع اعداد صحیح بین دو ورودی a,b را چاپ کنید. (A<B)

5-حاصل ضرب اعداد طبیعی کمتر مساوی n (!n) را چاپ کند.

6-تعداد مقسوم علیه های x را به همراه خود مقسوم ها چاپ کند.

7-اعداد طبیعی بین a,b را چاپ کنید.(ممکن است a>b یا b>a باشد)

8-پس از به دست آوردن تعداد مقسوم علیه های x تعیین کنیدx اول است یا خیر.

9-پس از به دست آوردن مجموع مقسوم علیه های x (به جز خود x)تعیین کنید عدد x تام است یا خیر(عدد تام عددی است که مجموع مقسوم علیه هایش (به جز خود عدد) با عدد برابر باشد)

10-نشانه های معادل با کدهای اسکی 100 تا 200 را چاپ کنید.

11-اعداد دو رقمی که رقم یکان و دهگان برابر دارند را چاپ کنید.

12-مجموع زیر را به دست آورید.

$$1+\frac{1}{2}+\frac{1}{3}+...+\frac{1}{n}$$

13-فاکتوریل اعداد 1 تا n را چاپ کند.

14-مجموع سری مقابل را به دست آورید.

$$1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}.+..\pm\frac{1}{n}$$

15-مجموع سری مقابل را به دست آورید.

$$1+\frac{1}{2!}+\frac{1}{3!}+\frac{1}{4!}.+..+\frac{1}{n!}$$

16- مجموع زیر را به دست آورید. $1+\frac{x^1}{2!}+\frac{x^2}{3!}+\frac{x^3}{4!}.+..+\frac{x^n}{n!}$

17- از میان ده عدد ورودی، دو عددی که بیشترین مقدار را دارند چاپ کند.

18- با دریافت شماره یکی از ماه های سال **87** تقویم آن ماه را چاپ کند(برای انتقال مکان نما به سطر و ستون دلخواه از تابع **gotoxy(x,y)** (سطر**y** و ستون**x**) استفاده کنید)

19- برنامه ای بنوسید که مجموع زیر چاپ شود.

$$1+\frac{1}{2}+\frac{1}{4}+\frac{1}{7}.+\frac{1}{11}+..+\frac{1}{n}$$

ساختار for متداخل یا لانه ای

در بخش قبل با ساختار **for** برای تکرار گروهی از دستورات آشنا شدیم . ممکن است در یک برنامه ، خود ساختار **for** نیز توسط ساختار **for** دیگری تکرار شود که در این حالت ساختار متداخلی از **for** ایجاد می شود.

مثال: برنامه ای بنویسید که اعداد مقابل چاپ شود.

```
1  2  3  4  5  6  7  8  9  10
1  2  3  4  5  6  7  8  9
1  2  3  4  5  6  7  8
1  2  3  4  5  6  7
1  2  3  4  5  6
1  2  3  4  5
1  2  3  4
1  2  3
1  2
1
```

توضیح :

این برنامه را نیز می توان به برنامه های کوچک تری تقسیم کرد. در این مثال ابتدا خط اول این خروجی را چاپ می کنیم:

```
for(i=1;i<=10;i++)
   cout<<i<<" ";
cout<<"\n";
```

چون حلقه قبل خاتمه یافته است ؛ از همان متغیر حلقه برای چاپ خط دوم استفاده می کنیم :

```
for(i=1;i<=9;i++)
   cout<<i<<" ";
cout<<"\n";
```

برای چاپ اعداد سایر خطوط نیز همین قطعه برنامه را بکار می بریم . فقط کافی است شرط حلقه را بصورت **i<=8 , i<=7 , ... , i<=2** بنویسیم . پس این قطعه برنامه را درون یک حلقه دیگر تکرار کرده و متغیر حلقه آن را طوری در نظر می گیریم که ابتدا **1,2,...10** باشد یعنی می نویسیم:

```
for(j=10,j>=1;j--) {
  for(i=1;i<=10;i++)
     cout<<i<<" ";
  cout<<"\n";
}
```

برنامه کامل مثال فوق به صورت زیر نوشته می شود:

```
#include<iostream.h>
void main()
{
  int i,j;
  for(j=10,j>=1;j--) {
    for(i=1;i<=10;i++)
       cout<<i<<" ";
    cout<<"\n";
  }
}
```

مثال:برنامه ای بنویسید که اعداد مقابل چاپ شود.

```
1
2  3
4  5   6
7  8   9   10
11  12   13  14  15
16  ..... n
```

توضیح:

این برنامه را می توان از رو شهای مختلفی حل کرد .

روش اول : همانطور که دیده می شود خروجی اعداد **1** تا **n** است . بنابراین می توان متغیری مانند **k=1** در نظر گرفته و پس از چاپ ، هر بار یک واحد به آن اضافه کرده تا به **n** برسیم.

در خط اول این برنامه یک عدد، در خط دوم دو عدد،در خط سوم سه عدد و... چاپ شده است. برای چاپ**k** به تعداد دلخواه از یک حلقه استفاده می کنیم. برای چاپ خط اول از قطعه برنامه زیر استفاده می کنیم:

```
k=1;
for(i=1;i<=1;i++){
    cout<<k<<" ";
    k++;
}
cout<<"\n";
```

در دومین خط خروجی، عمل چاپ **k** را دو بار انجام می دهیم:

```
for(i=1;i<=2;i++){
    cout<<k<<" ";
    k++;
}
cout<<"\n";
```

در سایر خطوط همین عملیات به تعداد**3،4** و... تکرار می شود. با قرار دادن این قطعه برنامه درون یک حلقه دیگر و انتخاب متغیری نظیر**j** این عمل را اجرا می کنیم :

```
K=1;
for(j=1;      ;j++){
  for(i=1;i<=j;i++){
    cout<<k<<" ";
    k++;
  }
  cout<<"\n";
}
```

شرط حلقه اول را با توجه به مقدار **k** که باید تا **n** برسد مشخص می کنیم یعنی **(k<=n)** . برای حلقه داخلی نیز باید این شرط را اضافه کنیم . چون تغییرات **k** درون حلقه داخلی انجام می گیرد و با رسیدن متغیر **K** به **n** ، باید برنامه خاتمه یابد .

پس برنامه را به صورت زیر تکمیل می کنیم :

```
#include<iostream.h>
void main( )
{
   int i,j,k=1,n;
   cout<<"n=";cin>>n;
   for(j=1;k<=n;j++) {
     for(i=1;i<=j&&k<=n;i++) {
        cout<<k<<" ";
         k++;
     }
     cout<<"\n";
   }
}
```

روش دوم:

درخط اول عدد **1** چاپ شده است پس می نویسیم:

```
for(i=1;i<=1;i++)
    cout<<i<<" ";
cout<<"\n";
```

در خط دوم اعداد **2** تا **3** چاپ می شود پس می نویسیم:

```
for(i=2;i<=3;i++)
    cout<<i<<" ";
cout<<"\n";
```

در خط بعدی اعداد **4** تا **6** چاپ می شود:

```
for(i=4;i<=6;i++)
    cout<<i<<" ";
cout<<"\n";
```

مقدار اولیه **i** در هر حلقه ، بر اساس مقدار نهایی **i** در حلقه ی قبلی تعیین می شود . پس می توان مقدار نهایی **i** را درون متغیر دیگری نظیر **j** قرار داده و در تکرار مجدد حلقه از آن استفاده کرد .

```
for(i=j;i<=    ;i++)
    cout<<i<<" ";
j=i;
cout<<"\n";
```

مقدار نهایی حلقه از رابطه**i<=j+k** به دست می آید . عدد **k** در حلقه اول صفر، در حلقه دوم **1** ,در حلقه ی بعدی **2** و...است.

```
i=1;
for(k=0 ;     ; k++) {
   for(i=j ; i<= j+k  ; i++)
      cout<<i<<" ";
    j=i;
   cout<<"\n";
}
```

شرط حلقه اول را بر اساس مقدار متغیری که چاپ می شود یعنی **i** مشخص می کنیم پس می نویسیم:

```
#include<iostream.h>
Void main()
{
  Int n, k,j=1,i;
  Cout<<"n=";   cin>>n;
  for(k=0 ;  i<=n  ; k++) {
    for(i=j ; i<= j+k &&  i<=n ; i++)
       cout<<i<<" ";
    j=i;
    cout<<"\n";
  }
}
```

مثال) اعداد سه رقمی که رقم یکان زوج و دهگان فرد داشته باشند را چاپ کنید.
توضیح: برای چاپ هر عدد خروجی ، ارقام مورد نظر را در ارزش هر رقم ضرب می کنیم . در جدول زیر اعداد خروجی را بر همین اساس ایجاد کرده ایم .

```
110  = 1*100 + 1*10 + 0
112  = 1*100 + 1*10 + 2
114  = 1*100 + 1*10 + 4
116  = 1*100 + 1*10 + 6
118  = 1*100 + 1*10 + 8
_____________________
130  = 1*100 + 3*10 + 0
132  = 1*100 + 3*10 + 2
.........
138  = 1*100 + 3*10 + 8
_____________________
.........
190  = 1*100 + 9*10 + 0
.........
198  = 1*100 + 9*10 + 8
=====================
210
212
....
298
=====================
310
312
....
398
=====================
..........
=====================
990
992
.....
998
```

ابتدا گروه اول اعداد را چاپ می کنیم یعنی 110 تا 118:

```
for(i=0;i<=8;i=i+2)
   cout<<1*100+1*10+i;
```

برای چاپ اعدا گروه دوم 130 تا 138 ، حلقه فوق را به صورت زیر تغییر می دهیم:

```
for(i=0;i<=8;i=i+2)
   cout<<1*100+3*10+i;
```

این برنامه را برای چاپ گروهی از اعداد که رقم صدگان 1 دارند ، در حلقه ی دیگری تکرار می کنیم یعنی:

```
for(j=1;j<=9;j=j+2)
  for(i=0;i<=8; i=i+2)
    cout<<1*100+j*10+i;
```

برای چاپ اعدادی که رقم یکان ودهگان آن ها مانند گروه قبل باشند و تنها رقم صدگان 2 داشته باشند می توان برنامه ی قبل را به فرم زیر تغییر داد:

```
for ( j=1; j<=9 ;j=j+ 2 )
  for ( i=0 ; i<=8 ; i=i+2 )
    cout<<2*100+j*10+i ;
```

چاپ اعدادی که رقم صدگان 3 ،4،... ، 9 داشته باشند نیز با تکرار همین قطعه حاصل می شود . پس این حلقه ها را توسط حلقه ی سوم تکرار می کنیم:

```
for ( k=1 ; k<=9 ; k++)
  for ( j=1 ; j<=9 ; j=j+2 )
    for ( i=0 ; i<=8 ; i=i+2 )
      cout<<k*100 + j*10 + i ;
```

برنامه ی کامل را می توان به صورت زیر نوشت:

```
#include <iostream.h>
void main ( )
{
   int  i ,j, k ;
   for ( k=1 ; k<=9 ; k++)
     for ( j=1 ; j<=9 ; j=j+2)
       for ( i=1 ; i<=8 ; i=i+2 )
         cout<< k*100 + j*10 + i ;
}
```

مثال) اعداد اول کوچک تر از n را چاپ کنید.

توضیح:

در این مثال باید اعداد 1و2و3و...وn را از نظر اول بودن بررسی کنیم و در صورت اول بودن ، آن ها را چاپ کنیم.
ابتدا مسئله را برای یک عدد فرضی مانند x حل می کنیم .یعنی می خواهیم در صورت اول بودن x آن را چاپ کنیم.
برای تعیین اول بودن x روش های مختلفی وجود دارد.یکی از سریع ترین راه حل ها این است که عدد x را بر 2 تا جذر x تقسیم کنیم.اگر در این بازه بر عددی بخش پذیر نباشد یعنی x اول است.
برای این که بخش پذیر بودن x بر این اعداد را بررسی کنیم ، یک متغیر کمکی مانند flag=0 را در نظر می گیریم.
اگر عدد x بر هرعددی بخش پذیر باشد ، تغییری در این متغیر ایجاد می کنیم ، مثلاً flag=1 می کنیم.

```
flag = 0 ;
for ( i=2 ; i<=sqrt (x) ; i+ + )
  if ( x%i == 0 )
     flag = 1 ;
```

در صورتی که x بر عددی بخش پذیر باشد ، یعنی اول نیست. پس به ادامه ی حلقه نیاز نیست و توسط دستور break از حلقه خارج می شویم. پس برنامه را به صورت زیر کامل می نماییم :

```
flag = 0 ;
for ( i=2 ; i<= sqrt (x) ; i++ )
  if( x%i= =0 ){
    flag =1 ;
    break ;
  }
```

پس از خروج از حلقه مقدار flag را بررسی می کنیم. اگر flag = 0 مانده باشد یعنی x بر هیچ عددی در بازه ی 2 تا جذر x ، بخش پذیر نبوده است ، یعنی اول است و آن را چاپ می کنیم .

```
if ( flag == 0 )
   cout<<x ;
```

همان طور که گفتیم باید اعداد1و2و3و...وn را بررسی کنیم . یعنی x می تواند 1,2,3,... باشد و توسط قطعه برنامه فوق اول بودن آن ها بررسی شود. پس دستورات فوق را در یک حلقه ی دیگر با متغیر حلقه x ، تکرار می کنیم.

```
#include<math.h>
#include<iostream.h>
void main ( )
{
   int n, x, flag ;
   cout<<"n=" ;
   cin>>n ;
   for ( x = 1 ; x<= n ; x ++ ) {
      flag = 0 ;
      for ( i = 2; i<= sqrt (x) ; i++ )
            if ( x%i== 0 ){
               flag = 0 ;
               break;
            }
       if ( flag == 0 )
          cout<<x<<"\t" ;
    }
 }
```

تمرین:

1-برنامه ای بنویسید که شکل مقابل چاپ شود.

```
*
* * *
* * * * *
. . . . . .
* * * * * . . . *
```

تا n

2-برنامه ای بنویسید که سطر **5** تا سطر **10** جدول ضرب را چاپ کند.

3-برنامه ای بنویسیدکه اعداد تام کمتر از **n** را چاپ کند.

4-اعداد سه رقمی که رقم یکان و صدگان برابرو رقم دهگان فرد دارند چاپ کنید.

5-برنامه ای بنویسید که مبلغ **1000** ریال را به کمک سکه های **50** -**100**-**250** -**500** ریالی خرد کند.

ساختار while

معمولاً حلقه ی **for** را برای تکرار گروهی از دستورات به تعداد دفعات معین به کار می برند. در حالی که به کمک این ساختارمی توان تا زمان درست بودن شرایطی خاص، دستوراتی را تکرار کرد. ساختار **while** نیز به همین منظور یعنی تکرار دستورات تا زما ن درست بودن شرایطی خاص به کار می رود.

نکته : در **do...while** چون شرط در انتهای ساختار قرار دارد ، دستورات حلقه حداقل یک بار انجام می شوند.

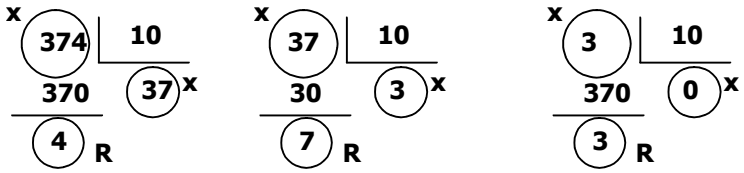
```
while( شرط ) {
    دستور 1 ;
    دستور 2 ;
    .......
}
```

```
do {
    دستور 1 ;
    دستور 2 ;
    .......
} while( شرط ) ;
```

مثال) مجموع ارقام ورودی **x** را چاپ کنید.

توضیح:

برای پیدا کردن مجموع ارقام ، باید ارقام عدد را از هم جدا کرد . برای این منظور از تقسیم های متوالی بر **10** استفاده می کنیم. مثلاً :



تا زمانی که خارج قسمت تقسیم مخالف صفر باشد عمل تقسیم تکرار می شود.چون می خواهیم دستورات تکراری و مشابه را درون حلقه تکرار اجرا کنیم ، متغیر های یکسانی به اجزای تقسیم ها نسبت می دهیم. مقسوم تقسیم اول همان عدد ورودی **x** است ، پس کلیه ی مقسوم ها را با **x** مشخص می کنیم . مقسوم تقسیم دوم همان خارج قسمت تقسیم اول است ، پس اگر مقسوم دومی **x** باشد ،خارج قسمت تقسیم قبلی نیز **x** خواهد بود. به همین دلیل خارج قسمت تقسیم ها را نیز با **x** نام گذاری کرده ایم . در تقسیم اول باید باقی مانده و خارج قسمت را پیدا کنیم :

```
R = x%10;
x = x/10;
```

در تقسیم های بعدی نیز همین دو دستور تکرار می شود(چون متغیر های نسبت داده شده مشابه یکدیگرند).
این دو دستور تا زمانی که x>0 یا x!=0 است تکرار می شود.پس می توان نوشت:

```
while ( x != 0 ) {
   R= x % 10 ;
   x = x / 10 ;
   . . . .
}
```

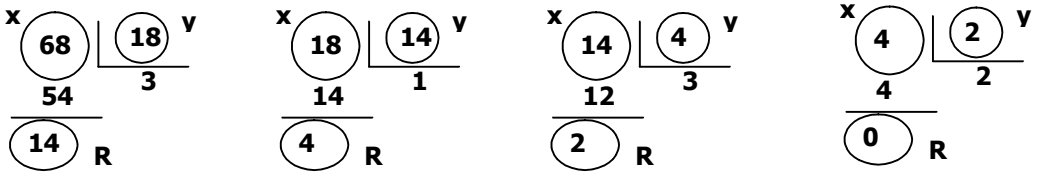
برای پیدا کردن مجموع ارقام باید R ها را با هم جمع کرد. پس می نویسیم:

```
#include <iostream.h >
void main ( )
{
   int  x, R,s ;
   cout<<"enter a number =" ;
   cin>>x ;
   while  ( x != 0 )  {
      R = x % 10 ;
      x =x / 10 ;
      s = s + R ;
   }
   cout<< s ;
}
```

مثال) با دریافت دو عدد x،y بزرگ ترین مقسوم علیه مشترک بین آن دو را به دست آورید.
توضیح:
سریع ترین روش برای حل این مسئله،روش نردبانی است.

	3	1	3	2	
68	18	14	4	2	0
54	14	2	4		

تقسیم های انجام شده در این روش را جداگانه انجام می دهیم تانسبت دادن متغیر ها راحت تر باشد.



تا زمانی که R!= 0 باشد تقسیم ها را اجرا می کنیم. در این مثال نیز چون عملیات شبیه است متغیر های یکسانی را به اعداد نسبت داده ایم.در تقسیم اول باید R را پیدا کنیم تا در تقسیم دوم به عنوان y جدید استفاده شود. در ضمن y تقسیم ، به عنوان x در تقسیم دوم استفاده شده است.پس درتقسیم اول دستورات زیر را می نویسیم:

```
R = x % y ;
x = y ;
y = R ;
```

دو دستور واگذاری ، مقادیر xوy تقسیم بعدی را آماده می کنند. در تقسیم دوم نیز همین دستورات تکرار می شود.
به همین دلیل توسط یک حلقه
این دستورات را تکرارمی کنیم .

```
while (          ){
   R = x % y ;
   x = y ;
   y = R ;
}
```

شرط **while** می تواند **R!= 0** باشد. چون مقدار اولیه **R** مشخص نیست و **R** را در **y** قرار داده ایم ، پس شرط را می توان با **y** تأمین کرد یعنی **y!= 0** . پس می نویسیم :

```
while ( y!=0 ){
   R = x % y ;
   x = y ;
   y = R ;
}
```

در مثال جواب **BMM** مقسوم علیه تقسیم آخر است که در **y** قرار دارد اما چون **y** را در **x** واگذار کرده ایم ، پس **BMM** همان مقدار نهایی **x** است .

```
# incude < iostream.h >
void main ( )
{
   int  x , y , R ;
  cout<<"x,y =" ;
  cin>>x>>y ;
  while ( y!= 0 ) {
    R = x % y ;
    x = y ;
    y = R ;
  }
  cout<<"BMM ="<<x ;
}
```

مثال) تا زمانی که هر جمله ی سری زیر بزرگ تر از **10** است، مجموع سری را به دست آورید.

$$1-\frac{1}{2!}+\frac{1}{4!}-\frac{1}{6!}+\dots$$

توضیح:

مخرج کسرهای این مثال فاکتوریل اعداد زوج است. برای تولید آن ها ابتدا فاکتوریل های اعداد متوالی را ایجاد کرده ، سپس فاکتوریل اعداد زوج را به کار می بریم . در ریاضی عبارت های زیر معتبر است :

$$f(0!) = (1)$$
$$f(1!) = (0!) \overset{f}{*} (1)\ i$$
$$f(2!) = (1!) \overset{f}{*} (2)\ i$$
$$f(3!) = (2!) \overset{f}{*} (3)\ i$$
$$f(4!) = (3!) \overset{f}{*} (4)\ i$$
$$\dots\dots$$

با توجه به متغیرهای نسبت داده شده به اعداد دستور **f = i * f** باید در یک حلقه به ازای **i = 1** تا **i=n** تکرار شود . در این صورت فاکتوریل اعداد **1!** ، **2!** ، **3!** ، . . . ، **n!** ایجاد شود یعنی می توان نوشت :

```
f = 1 ;
for ( i = 1 ; i<= n ; i + + ){
    f = f * i ;
     .....
  }
```

این قطعه برنامه را می توان بصورت مقابل با **while** نوشت .

```
f=1; i= 1;
while (          ) {
   f=f*i;
   i=i*1;
   .........
}
```

(فعلاً با شرط آن کاری نداریم)

از میان فاکتوریل های تولید شده ، تنها فاکتوریل اعداد زوج !2,!4,... در مخرج کسرها به کار رفته است.
برای تبدیل شدن جملات سری به مجموع ، عبارت سری را به بصورت زیر در نظر می گیریم .

$$1+(-1)\times\frac{1}{2!}+(1)\times\frac{1}{4!}+(-1)\times\frac{1}{6!}+\dots$$

با این تغییر ، سری به مجموع تبدیل می شود و متغیرها را می توان بصورت زیر به آنها نسبت داد و سری را به صورت زیر نوشت :

$$1+a\times\frac{1}{f}+a\times\frac{1}{f}+a\times\frac{1}{f}+\dots$$

پس ما باید مجموع $a\times\frac{1}{f}$ ها را بدست آوریم. یعنی می توان نوشت:

```
f=1; i=1; a=-1;
while (          ) {
    f=f*i;
    if (i%2==0) {
       s=s+a*1/f;
       a=-a;
    }
    i++;
}
```

در صورت که مقدار I زوج باشد مجموع $a\times\frac{1}{f}$ s=s+ را بدست می آوریم . در ضمن a را هر دفعه در 1- ضرب می کنیم تا یکی در میان 1- ، 1 ، 1- ، ... شود.
شرط حلقه به حاصل 1/F بستگی دارد. تا زمانی که $1/F > 10^{-7}$ باشد مجموع را بدست می آوریم . بخاطر اینکه جمله اضافی در if شرکت نکند، دستور f = f * I را به بعد از if منتقل می کنیم تا در هنگام تولید یک f جدید ابتدا شرط while بررسی شده و اگر $1/F > 10^{-7}$ باشد وارد شرط و حلقه می شویم.

```
#include<iostream.h>
void main()
{
    double   s=0 , f=1;
    int   i , a=-1;
    while (1/f > 1e-7) {
        if (i%2==0) {
          s=s+1/f*a;
          a=-a;
        }
        i++;
        f=f*i;
    }
    cout<<"sum of serie=" << s;
}
```

تمرین:

1- تعداد ارقام عدد ورودی **x** را چاپ کنید.

2- مجموع ارقام زوج عدد ورودی **x** را چاپ کنید.

3- مقلوب عدد ورودی **x** را چاپ کنید. **637 → 736**

4- تا زمانی که عدد ورودی **x** مثبت است، جذر عدد ورودی را چاپ کنید.

5- عددی اعشاری دریافت کنید. برنامه ای بنویسید که تعداد رقم های اعشار آن را چاپ کند.

6- برنامه ای بنویسید که با دریافت یک عدد اعشاری، بخش های اعشار و صحیح آن را به عنوان دو عدد صحیح چاپ کند.

7- برنامه ای بنویسید که با دریافت یک عدد اعشاری آن را وارونه کند **643.21 → 12.346**

8- با دریافت عدد **x** ، تعیین کنید این عدد فاکتوریل چه عددی است. در صورتی که عدد **x** فاکتوریل عددی نیست پیغام مناسبی چاپ شود.

9- جملات سری فیبوبوناچی را چاپ کنید **1 1 2 3 5 8 13 ... n**

(هرجمله از مجموع دو جمله قبل بدست می آید)

10- **n** امین جمله ی سری فیبوناتچی را چاپ کنید مثلاً **n = 8 → چاپ 21**

11- برنامه ای بنویسید که دریافت **n** را تا زمانی تکرار کند که **n** خارج از محدوده **1** تا **12** باشد . سپس این عدد را بعنوان شماره یکی از ماه های سال فرض کرده تعداد روز های آن ماه را چاپ کنید .

12- با دریافت عدد **n**، تعیین کنید این عدد جزء سری فیبوناتچی هست یا خیر

13- فرض کنید عدد ورودی **x** در مبنای **8** باشد آن را به عددی در مبنای **10** تبدیل کنید.

14- با فرض اینکه عدد ورودی **x** در مبنای **10** است آن را به عددی در مبنای **2** تبدیل کنید.

آرایه

در فصل های قبل داده ها ی مورد نیاز برنامه را در متغیرهای ساده یا ثابت ها ذخیره می کردیم. اگر تعداد داده ها زیاد باشد، استفاده از متغیرهای ساده برنامه نویسی را مشکل و کدها را طولانی می کند. در برخی مواقع نیز برنامه نویسی غیر ممکن می شود. به همین دلیل از آرایه استفاده می کنیم.
آرایه: به مجموعه ای از داده ها با نوع و نام یکسان گفته می شود که دارای اندیس های متمایز باشند. به هر عنصر یک آرایه متغیر اندیس دار می گوییم.

آرایه های یک بعدی

سیستمی از داده ها که با یک اندیس شناسایی می شوند آرایه یک بعدی می گوییم. در C++ آرایه یک بعدی را به فرم مقابل معرفی می کنیم :

```
نوع عناصر آرایه  نام آرایه  [تعداد عناصر];
```

نکته : طول آرایه یا تعداد عناصر ، یک عدد مثبت است که ذکر آن الزامی است. مثلاً : `int x[10];`
این تعریف آرایه ی x را با 10 عنصر از نوع int معرفی می کند.
نکته : اندیس آرایه در C++ از صفر شروع می شود. پس عناصر آرایه عبارتند از:

X[0]	X[1]	X[2]	...	X[9]

نکته: در هنگام معرفی آرایه می توان به هر عنصر مقدار اولیه نسبت داد. مثلاً:

```
int y[5] = {75,0,46,-10,6};
```

نکته: اگر تعداد مقادیر نسبت داده شده کمتر از تعداد عناصر باشد ، سایر عناصر آرایه صفر می شوند.

```
int x[5] = {100,-15};
```

x[0] x[1]

x[2] ، x[3] ، x[4] نیز صفر هستند.

نکته: زبان C نسبت به حدود اندیس ها حساس نیست. لذا برنامه نویس باید رعایت حدود اندیس ها را بنماید.
مثلاً : در دستورات مقابل باوجود خارج بودن اندیس های آرایه از حدود آنها ،
این دستورات با خطا مواجه نمی شوند.

```
int x[10];
cout<< x[12];
x[-1] = 100;
```

نکته: در صورت نسبت دادن مقدار اولیه می توان تعداد عناصر آرایه (در تعریف آرایه) را حذف کرد. مثلاً:

```
int x[ ] = {100,55,-40,75};
```

– در این تعریف آرایه x متناظر با تعداد مقادیر اولیه ، فضا اشغال می کند یعنی این تعریف معادل دستور زیر است :

```
int x[4] = {100,55,-40,75};
```

مثال) پس از دریافت نمرات ده دانش آموز و قرار دادن آنها در یک آرایه، نمرات بالای 12 را به همراه شماره هر نمره بصورت دو ستون مجاور هم چاپ کنید.

توضیح:

ابتدا آرایه ای بصورت مقابل تعریف می کنیم.

```
float   mark [10];
```

تک تک عناصر آرایه را دریافت می کنیم:

```
cin>> mark [0];
cin>> mark [1];
. . . . . .
Cin>> mark [9];
```

همانطور که دیده می شود دستور cin تکرار می شود. لذا از یک حلقه for برای تکرار دریافت استفاده می کنیم و متعیر حلقه را نیز متناظر با اندیس های عناصر تغییر می دهیم.

```
for (i=0 ; i<10 ; i++)
    cin>>mark [i];
```

در مرحله بعدی تک تک عناصر را با نمره 12 مقایسه کرده و اگر بیش از 12 باشند آنها را چاپ می کنیم. پس حلقه ی دیگری به صورت زیر می نویسیم:

```
for (i=0 ; i<10 ; i++)
    if (mark [i] > 12)
        cout<<mark [i] <<"\t"<<i+1<<"\n";
```

در این حلقه نیز از متغیر i استفاده کرده ایم. چون متغیر i که در حلقه قبل به کار رفته است، بدون استفاده است. نهایتا برنامه را به صورت زیر کامل می کنیم:

```
#include <iostream.h>
void main ( )
{
    float mark [10];
    for (int i=0 ; i<10 ; i++) {
        cout<< "enter score=";
        cin>> mark [i];
    }
    for (i=0 ; i<10 ; i++)
        if (mark [i] > 12)
            cout<< mark [i] << "\t" << i+1<< "\n";
}
```

مثال) در یک آرایه 10 عنصر منحصر به فرد قرار دهید. سپس عناصر را چاپ کنید.

توضیح:

در صورتی عنصری در آرایه درج می شود که قبلاً در آرایه وجود نداشته باشد. فرض کنید 5 عنصر در آرایه وجود دارد و حالا می خواهیم عنصر ششم را در آرایه درج کنیم. پس از دریافت عنصر جدید ، آن را با تک تک عناصر موجود در آرایه مقایسه می کنیم. در صورتی که با عناصر موجود در آرایه یکی نباشد، آن را به آرایه اضافه می کنیم.

```
flag = 0;
cin>> a;
for (i=0 ; i<5 ; i++)
    if (a== x[i])
        flag = 1;
```

در این قطعه برنامه از متغیر کمکی **flag** بعنوان یک نشانه استفاده کرده ایم. اگر عنصر جدید **(a)** با عناصر موجود در آرایه برابر باشد، این متغیر را تغییر می دهیم. نهایتاً اگر **flag = 0** بماند، یعنی **a** با هیچ عنصر آرایه مساوی نیست. اما اگر **flag = 1** شود، یعنی عنصر تکراری است. برای درج عنصر جدید می نویسیم:

```
if  (flag == 0)
   x [i]= a;
```

چون حلقه خاتمه یافته است پس مقدار **i=5** است. بنابراین **a** را بعنوان عنصر ششم در آرایه قرار می دهیم. قطعه برنامه فوق را می توان برای درج تک تک عناصر تکرار کرد. یعنی در یک حلقه دیگر قرار می دهیم:

```
for (j=0; j<10; j++) {
    flag= 0;
    cout<<"new item=";
    cin>> a;
    for (i=0 ;  i<j ;  i++)
        if (a== x[i]) {
            flag= 1;
            break;
        }
    if (flag == 0)
        x [i]= a;
}
```

حال می توان آرایه دریافت شده را چاپ کرد.

```
for (i=0 ;  i<10 ;  i++)
    cout<< x[i] <<"\t";
```

برنامه کامل را می توان بصورت زیر نوشت:

```
#include <iostream.h>
void main ( )
{
 Int I,j,flag,a,x[10];
 for (j=0; j<10; j++) {
    flag= 0;
    cout<<"new item=";
    cin>> a;
    for (i=0 ;  i<j ;  i++)
        if (a== x[i]) {
            flag= 1;
            break;
        }
    if (flag == 0)
        x [i]= a;
  }
  for (i=0 ;  i<10 ;  i++)
     cout<< x[i] <<"\t";
}
```

تمرین :

1-در یک آرایه 10 عضوی اعداد صحیح قرار دهید ، سپس اعداد زوج آرایه را چاپ کنید .
2-پس از دریافت عناصر یک آرایه 10 عضوی ، تعداد اعداد مثبت را چاپ کنید .
3-پس از دریافت آرایه 10 عضوی صحیح ، میانگین اعداد فرد آرایه را چاپ کنید .
4-10 عدد اعشاری در آرایه قرار دهید . کمترین مقدار آرایه را بدست آورده و عناصری که با کمترین مقدار برابرند را چاپ کنید .
5-پس از دریافت 10 عدد اعشاری در یک آرایه ، دو عضوی که حاوی بیشترین مقدار هستند را چاپ کنید .
6-در یک آرایه 10 عدد صحیح قرار دهید سپس اعداد اول موجود در آرایه را چاپ کنید .
7-در یک آرایه 10 عدد صحیح قرار دهید سپس با جابجایی عناصر ، لیست را وارونه کنید .
8-در یک آرایه نمره 10 دانش آموز را درج می کنیم پس از گرد کردن نمرات و قرار دادن آنها در آریه ، تعداد هر نمره را چاپ کنید .
9-تمرین قبل را بدون گرد کردن عناصر انجام دهید .
10-میانگین 10 عدد ورودی را بدست آورده سپس اعدادی را چاپ کنید که از میانگین بزرگتر باشند .

مرتب سازی حبابی

یکی از روش های مرتب سازی آرایه ، مرتب سازی حبابی است . این روش را بر روی یک لیست 5 عضوی فرضی اجرا می کنیم سپس آن را پیاده سازی می کنیم .

فرض کنید اعداد مقابل در یک لیست 5 عضوی وجود داشته باشد . 7 , 55 , 30 , 40 , 3

عناصر آرایه را دو به دو مقایسه می کنیم یعنی عنصر اول با دوم ، دوم با سوم و ... . در صورتی که عنصر اولی از دومی بزرگتر باشد جای آنها را عوض می کنیم .

مقادیر نهایی لیست پس از گام اول	چهارم با پنجم	سوم با چهارم	دوم با سوم	اول با دوم
7	7	7	7	7
30	30	30	55	55
55	55	55	30	30
5	400	400	400	400
400	5	3	3	3

در پایان این گام ، عنصری که بیش از همه باشد به انتهای لیست منتقل می شود . حال عملیات را بر روی لیست تکرار می کنیم .

مقادیر نهایی لیست پس از گام اول	چهارم با پنجم	سوم با چهارم	دوم با سوم	اول با دوم
7	7	7	7	7
30	30	30	30	30
5	5	55	55	55
55	55	5	5	5
400	400	400	400	400

در پایان این گام ، 55 به انتهای لیست (قبل از 400) منتقل می شود . با اجرای عملیات در گام های سوم و چهارم لیست مرتب می شود.

پیاده سازی مرتب سازی حبابی

با توجه به توضیحات قبل ، یک آرایه 10 عنصری فرضی را بصورت صعودی مرتب می کنیم.

توضیح :

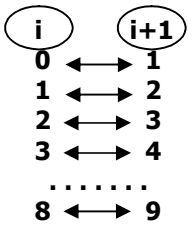
پس از دریافت عناصر لیست ، ابتدا گام اول را اجرا می کنیم یعنی عنصر اول با دوم و دوم با سوم و ... نهم ودهم را مقایسه می کنیم و در صورت لزوم جای دو عنصر را عوض می کنیم . برای سادگی ابتدا اولین مرحله از این گام را اجرا می کنیم :

```
if (x[0]>x [1] ) {
    t = x[0];
    x[0]=x[1] ;
    x[1]= t
}
```

در مرحله دوم عناصر دوم و سوم آرایه را مقایسه می کنیم.

```
if (x[1]>x [2] ) {
    t = x[1];
    x[1]=x[2] ;
    x[2]= t
}
```

در مراحل بعدی نیز همین عملیات تکرار می شود اندیس های مقایسه شده عبارتند از :



اگر ستون اول را با متغییر i شناسایی کنیم ستون دوم را می توان i+1 نامید .

پس توسط یک حلقه گام **1** را بفرم زیر می نویسیم :

```
for (i = 0; i < = 8; i+ + )
      if (x[i] > x [i+1]) {
         t = x[i];
         x[i] = x [ i+1];
         x[i+1] = t ;
      }
```

* در مرتب سازی حبابی n عنصر ، باید عملیات مرتب سازی n-1 گام اجرا شود . پس اگر قطعه برنامه فوق را n-1 بار یعنی **9** بار تکرار کنیم آرایه مرتب می شود .

```
for( j=1; j< = 9 ; j + + )
   for (i = 0; i < = 8; i+ + )
      if (x[i] > x [i+1]) {
         t = x[i];
         x[i] = x [ i+1];
         x[i+1] = t ;
      }
```

برنامه کامل مثال مرتب سازی **10** عنصر عددی را می توان بصورت زیر نوشت

```
#include<iostream.h>
void main ()
{
    int  x[10], t ,i ,j ;
    for ( i= 0; i < 10; i + + )
          cin >> x[i];
    for( j = 1 ; j < = 9; j + +)
       for( i =0 ; i < = 8 ; i + +)
          if ( x[i] > x [i+1]) {
             t = x[i];
             x[i] = x[i+1];
             x[i+1] = t ;
          }
    for ( i= 0 ; i< = 9 ; i+ + )
        cout << x[i] <<' "\t";
}
```

جستجوی باینری

در سوالات قبلی برای یافتن یک عنصر در آرایه تک تک عناصر ارایه را با عنصر مورد جستجو مقایسه کردیم . این روش جستجو را خطی می نامند . بدلیل سرعت پایین این روش ، در لیست های طولانی از روش جستجوی باینری استفاده می کنیم . شرط لازم برای اجرای جستجوی باینری مرتب بودن لیست است .
فرض کنید اعداد زیر در یک آرایه قرار داشته باشند .

5	75	110	200	275	300	304	700	1000	1200
L=0				Mid=4					R=9

بر اساس اندیس های ابتد ا و انتهای لیست مورد جستجو میانه را بدست می آوریم .

$$Mid = \frac{0+9}{2} = 4$$

یعنی 275 عنصر میانه است . حال عنصر میانه را با عنصر مورد جستجو A مقایسه می کنیم اگر [A < x[Mid باشد یعنی عنصر مورد جستجو در نیمه سمت چپ لیست است پس لیست جدید اندیس های [0 to 3] را خواهد داشت . پس L = 0 خواهد ماند اما R = 3 می شود .

```
if ( A< x[ Mid] ) R = Mid -1 ;
```

اگر عنصر مورد جستجو بزرگتر از عنصر میانه باشد عنصر در سمت راست لیست است .

```
if (A> x [Mid] ) L= Mid +1;
```

اگر عنصر مورد جستجو با x[mid] برا بر با شد عنصر پیدا شد ه است ومی توان جستجو را خا تمه داد.
پس از تغییر اندیس شروع یا پایان ، لیست جدیدی ایجاد می شود و همان عملیا ت مرحله قبل را بر روی لیست جدید تکرار کنیم . این عمل را تا زما نی ادامه می دهیم که عنصر پیدا نشده باشد یا اندیس L< = R با شد .

تمرین :

1-برنا مه جستجوی را بر روی یک لیست مرتب شد ه فرضی صعو د ی اجرا کنید .

تمرین تکمیلی :

1-در یک آرایه 10 عنصری , 5 عدد به صورت مرتب شده درج کنید . برنا مه ای بنویسید که با دریا فت یک عنصر جدید آن را طوری در آرایه قرار دهد که لیست باز هم مرتب با شد.

2-در یک آرایه 10 عدد منحصر به فرد قرار دهید.سپس برنامه ای بنویسید که با دریافت یک عدد آن را در آرایه جستجو کرده و از لیست حذف کند.اگر عنصر در لیست وجود ندارد ، پیغام مناسب چاپ شود.

3-دو آرایه نزولی حداکثر 10 عنصری در اختیار داریم.این دو آرایه را در یک آرایه سوم طوری درج کنید که آرایه حاصل باز هم نزولی باشد.

رشته ها

در زبان C++ نوع استانداردی بنام رشته وجود ندارد ، بلکه برای تعریف یک رشته باید آرایه ای از نشانه ها را تعریف کرد . مثلاً

```
char s[20] ;
```

این دستور رشته ای بنام S را معرفی می کند که حد اکثر می تواند حاوی 19 نشانه باشد .
نکته : انتهای رشته در زبان C به نشانه '\0' یا عدد صفر یا مقدار NULL ختم می شود به همین دلیل رشته S می تواند حداکثر 19 نشانه را در خود ذخیره کند.
نکته : در هنگام معرفی رشته می توان مقدار اولیه به آن نسبت داد . مثلاً :

```
char s[20] = "Turbo c+ + " ;
```

وضعیت قرار گرفتن نشانه ها در این آرایه بصورت زیر است :

S[0]	S[1]	S[2]	S[3]	S[4]	S[5]				S[19]
'T'	'u'	'r'	'b'	'o'	NULL	. . . .			

نکته : به تک تک نشانه های رشته می توان همانند یک آرایه دسترسی پیدا کرد . در شکل فوق برخی از نشانه های رشته به صورت متغیر اندیس دار مشخص شده اند .
نکته : مثال قبل را می توان بصورت زیر نیز تعریف کرد :

```
char s[20] = {' T ',' u' , ' r ', ' b' , ' o ' , '  ' , ' c ', ' + ', ' + ' , ' \o ' };
```

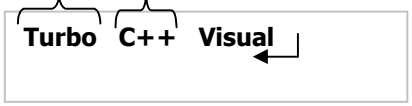
نکته : در ضمن در صورت نسبت دادن مقدار اولیه می توان حد آرایه را مشخص نکرد :

```
char s[ ] = "C+ + ";
```

دریافت متغیر رشته ای

همانند متغیر های عددی می توان توسط cin عمل دریافت رشته را انجام داد . اما چون cin ورودی های خود را با فاصله جدا می کند ، اگر بین رشته ورودی فاصله باشد ، دو مقدار محسوب شده و فقط قسمت اول درون متغیر رشته ای قرار می گیرد . مثلاً :

```
char s1[10], s2[10];
cin>> s1 >> s2 ;
```



با توجه به توضیح قبل دستورcin برای دریافت رشته مناسب نیست(مگر رشته هایی که حاوی فضای خالی نباشند)به همین دلیل متد cin.get را مورد استفاده قرار می دهیم .

```
Cin.get(متغیر رشته ای , تعداد نشانه ['نشانه'] );
```

دو پارامتر اول اجباری است ، در صورت استفاده از این دو پارامتر دریافت زمانی خاتمه می یابد که کاربر کلید Enter را فشار دهد و یا تعداد نشانه های ورودی n-1 شود .

مثال :

```
char  s[10];
cin.get(s,3);
```

Visual↵

```
char  s[10];
cin.get(s,10);
```

vb↵

در مثال اول تعداد نشانه های رشته ورودی 2 است و چون حداکثر تعداد نشانه ها 1-10 تعیین شده پس دریافتS کامل صورت می گیرد ، یعنی vb درون S قرار می گیرد . اما در مثال بعدی حداکثر تعداد نشانه های ورودی 1-3 یعنی2 مشخص شده پس تنها Vi وارد رشته S می شوند.
پارامتر سوم اختیاری است و بیانگر نشانه ای است که می خواهیم پایان دریافت را تعیین کند مثلا:

```
char  s[10];
cin.get(s,10,'.');
```

در این مثال تنها رشته "731" وارد S می شود.

731.46↵

نکته : برای چاپ متغیر رشته ای همانند داده های عددی از cout استفاده می کنیم.
مثال : با دریافت یک رشته ،تک تک نشانه های رشته را به همراه کداسکی آنها چاپ کنید.

```
#include<iostream.h>
void main( )
{
   char s[20];
   cout<<"Enter string";
   cin.get(s,20);
   for(int i=0;s[i]!=NULL;i++) {
      int x=s[i];
      cout<<s[i]<<"\t"<<x<<"\n";
  }
}
```

توضیح :
تک تک نشانه های رشته را همانند یک آرایه از رشته جدا می کنیم . برای دسترسی به کد اسکی نشانه های مورد نظر ، آن را درون متغیر عددی x قرار می دهیم تا معادل عددی نشانه (کد نشانه)بدست آید . این عمل را تا پایان رشته یعنی تا زمانی که s[i] مخالف نشانه پایانی رشته (Null) باشد تکرار می کنیم.

مثال: تعداد حروف کوچک و بزرگ موجود در رشته ورودی S را چاپ کنید.
توضیح :
در این مثال نیز باید تک تک نشانه های رشته را بررسی کنیم . اگر اولین نشانه رشته درمحدوده حروف بزرگ باشد، از متغیر k برای بدست آوردن تعداد واگر درمحدوده حروف کوچک باشد از متغیر p برای تعیین تعداد نشانه استفاده می کنیم .

```
if(s[0]>='A' && s[0]<='Z')
   k++;
else  if(s[0]>='a' && s[0]<='z')
   p++;
```

همین دستور را باید بر روی[1]s,[2]s,... (تا پایان رشته) تکرار کنیم . پس می نویسیم:

```
#include<iostream.h>
void main( )
{
   int k=0,p=0;
   char s[20];
   cout<<"S=";cin.get(s,20);
   for(int i=0;s[i]!=0;i++)
      if(s[i]>='A' && s[i]<='Z')
         k++;
      else if(s[i]>='a' && s[i]<='z')
         p++;
   cout<<" count of uppercase="<<k<<"\n";
   cout<<" count of lowercase="<<p<<"\n";
}
```

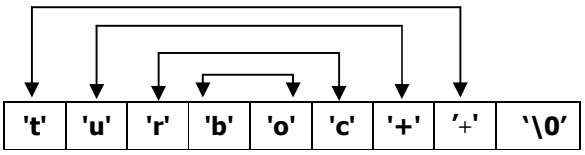
نکته: دراین مثال از نشانه های 'Z' تا 'A'برای تعیین حدود مقایسه استفاده کردیم در حالی که می توانستیم از کداسکی آنها نیز استفاده کنیم. در این صورت شرط درون حلقه را با دستور زیر تغییر می دهیم :

```
if(s[i] > = 65&&  s[i]<=90)
   K++;
```

مثال) رشته ورودی s را وارون یا مقلوب نمایید .

توضیح :

فرض کنید رشته ورودی "turbo c++" باشد برای مقلوب کرد ن رشته با ید نشا نه ها را بصورت زیر جا بجا کنیم:



در مرحله ی اول جا ی s[0] را با s[7] عوض می کنیم :

```
ch=s[0];
s[0]=s[7];
s[7]=ch;
```

همین عملیات را باید بین نشانه های مقابل تکرار کنیم .

I p
S[0] ⟷ s[7]
S[1] ⟷ s[6]
S[2] ⟷ s[5]
S[3] ⟷ s[4]

اندیس های ستون اول را با i=0 نام گذاری می کنیم و هر باریک واحد به آن اضافه می کنیم ، برای ستون دوم می توان متغیر دیگری نظیر p را انتخاب کرد که از 7 شروع شده و هر بار یک واحد کم می شود .

```
for(i=0 , p=  7 ;      ;  i+ + , p -- ) {
   ch = s[i];
   s[i]= s[p];
   s[p]= ch;
}
```

شرط پایان حلقه توسط دو متغیر i , p تعیین می شود . حلقه تا زما نی ادا مه دارد که i< p با شد .

نکته : در این مثال بخش اول و سوم حلقه **for** شا مل دو دستور است که توسط ویرگول از هم جدا شده اند .
نکته : توجه داشته با شید که حلقه با تغییرات **i** ، چهار بار تکرار می شود پس برای تغییرات **p** نباید حلقه دیگری (بصورت متدا خل) فرض کنیم .

```
#include<iostream.h>
#include<string.h>
void main()
{
   char s[20];
   int  i,p;
   cout<<"s=";
   cin.get(s,20);
   for(i=0 , p=  7 ; i<p  ;  i++ , p -- ) {
      ch = s[i];
      s[i]= s[p];
      s[p]= ch;
   }
   cout<<s;
}
```

نکته : برای تعیین مقدار اولیه **p** از طول رشته استفاده شده است که تا بع **sterlen** این عمل را انجام می دهد . این تابع در فایل **String.h** است .

تمرین :

1. تعداد نشا نه های عددی و غیر عددی رشته **s** را چاپ کنید .
2. رشته **s** را به تعداد نشانه هایش در خروجی چاپ کنید .
3. به کد هر نشانه رشته **s** , **3** واحد اضافه کرده و رشته حاصله را چاپ کنید .
4. حروف کوچک رشته را به بزرگ و حروف بزرگ را به کوچک تبدیل کنید .
5. پس از دریافت دو رشته s_1 و s_2 , رشته s_2 را به انتهای رشته s_1 اضافه کنید .
6. به ابتدای رشته **s** ورودی ،رشته **"Mr"** را اضافه کنید.
7. پس از دریافت رشته **s** و دو عدد**n,m** ، از نشانه **n** ام رشته **s** ، **m** نشانه را جدا کرده و درون رشته دیگری مانند s_2 قرار دهید، سپس رشته s_2 را چاپ کنید.
8. رشته ورودی عددی در مبنای **10** است ، پس از تبدیل این رشته به عدد، جذر عدد حاصله را چاپ کنید.
9. عدد **n** را از مبنای **10** به رشته ای در مبنای **2** تبدیل کنید.
10. برنامه ای بنویسید که تنها امکان دریافت رشته ای در مبنای **8** امکان پذیر باشد یعنی تنها نشانه های **'0'** تا **'7'** به عنوان نشانه های معتبر در رشته ورودی قرار گیرند.

آرایه دوبعدی:

فرض کنید در یک آرایه تک بعدی نام و شماره تلفن مشترکین قرار گرفته باشد. در این لیست ، پیدا کردن یک فرد براحتی امکان پذیر نیست ، چون نمی توان لیست را بر اساس پارامتری نظیر نام مرتب کرد . اما اگر در این لیست ،به جای آرایه ی یک بعدی ، از دو بعدی استفاده شود به راحتی می توان لیست را بر اساس نام یا شماره تلفن مرتب کرد.

در آرایه ی دوبعدی برای مشخص کردن هر عنصر از سایر عناصر، دو عدد به عنوان اندیس استفاده می شود. اندیس اول سطر ، و اندیس دوم ستون عنصر مورد نظر را تعیین می کند . آرایه ی دو بعدی در قالب زیر تعریف می شود:

نوع آرا یه نا م آرایه [بعدا] [بعد2] ;

مثلا : int x[5][10];

این آرایه شا مل 5 سطر و 10 ستون است یعنی 50 عنصر از نوع صحیح را معرفی می کندکه عناصر آن عبارتند از:

x[0][0]	x[0][1]	x[0][2]	....	x[0][9]
x[1][0]	x[1][1]		....	x[1][9]
....	....	....	....	....
x[a][0]	x[9][1]		....	x[9][9]

نکته : همانند آرایه ی تک بعدی می توان به عناصر آرایه ی دوبعدی ، مقدار اولیه نسبت داد:

int x[2] [3] = { {7,3,0}و{4,15,-1};

این دستوررا می توان به فرم زیر نیز نوشت :

int x [2] [3] ={7,3,0,15,-1};

- در حالت اول سطر و ستون ها بهتر دیده می شوند .

نکته : در هنگام نسبت دادن مقدار اولیه می توان تعداد عناصر بعد اول راحذف کرد . مثلا :

int x [ ] [4] = {15,100,50,-1,7};

این تعریف یک ارایه ی2 4 راتعریف می کند. که سطر اول آن کامل ، و از سطر دوم تنها یک مقدار مشخص است. سایر مقادیر سطر دوم نامعین است .

مثال) پس از دریافت یک آرایه ی 5 10 عناصر آن را مانند عناصر یک ماتریس چاپ کنید .

توضیح :

هدف این مثال آشنایی با دریافت وچاپ عناصرماتریس یا آرایه ی دو بعدی است . فرض کنید که آرایه به صورت مقابل معرفی شده باشد. int x [5] [10] ;

عناصر این ماتریس در بخش قبل مشخص شد .از همان عناصر برای توضیح دریافت استفاده می کنیم .

ابتدا می خواهیم ردیف اول آرایه را دریافت کنیم ، که شامل عناصر زیراست :

x[0][0]	x[0][1]	x[0][2]	x[0][3]	....	x[0][9]

برای دریافت این ردیف ، تک تک عناصر آرایه را دریافت می کنیم .

```
cin>>x[0][0];
cin>>x[0][1];
cin>>x[0][2];
. . . . . .
cin>>x[0][9]
```

می توان توسط یک حلقه این دریافت ها را به صورت زیر خلاصه کرد :

```
for (i=0; i<10; i++)
   cin>> x [0] [i];
```

در ردیف دوم عناصرزیر وجود دارند :

x[1][0]	x[1][1]	x[1][2]	x[1][3]	. . . .	x[1][9]

دریافت این ردیف نیز همانند ردیف اول است ، تنها اندیس اول آرایه 1 است :

```
for (i=0; i<10; i++)
   cin>> x [1][i];
```

در ردیف های سوم و چهارم وپنجم نیز همین قطعه برنامه تکرار می شود. لذا توسط حلقه دیگری این قطعه کد را تکرار می کنیم و از متغیر حلقه جدید برای اندیس اول آرایه استفاده می کنیم :

```
for(j=0;j<5;j++)
  for (i=0; i<10; i++)
    cin>> x [j] [i];
```

در مرحله ی بعدی می خواهیم عناصر آرایه را چاپ کنیم . در ابتدا فقط عناصر ردیف اول را چاپ می کنیم:

```
for (i=0; i<10; i++)
   cout<< x [0] [i]<<"   ";
```

پس از چاپ این ردیف ، مکان نما رابه خط بعدی منتقل می کنیم . تا ردیف دوم در خط دیگری چاپ شود .

```
cout<<"\n";
```

ردیف های بعدی را نیز مانند ردیف اول چاپ می کنیم. مثلا در ردیف دوم داریم :

```
for (i=0; i<10; i++)
   cout<< x [1] [i]<<"   ";
cout<<"\n";
```

پس می توان با تکرار این دو دستور ، کلیه ی عناصر ماتریس را چاپ کرد:

```
for(j=0;j<5;j++) {
  for (i=0; i<10; i++)
    cout<< x [j] [ i ]<<"   ";
  cout<<"\n";
}
```

نهایتا می توان برنامه را به صورت زیر کامل کرد :

```
#include<iostream.h>
#include<iomanip.h>
void main()
{
  int x [5][10],i,j;
  for (j=0;j<5;j++)
     for (i=0;i<10;i++)
        cin>>x [j][i];
  for(j=0;j<5;j++){
     for(i=0;i<10;i++)
        cout<<stew(4)<<x [j][i];
     cout<<endl;
  }
}
```

مثال) پس از دریافت یک ماتریس عناصر بالای قطر اصلی را چاپ کنید.

توضیح (روش اول) :

دریافت آرایه در مرحله ی قبل توضیح داده شد لذا فقط نحوه چاپ عناصر را توضیح می دهیم .

چاپ عناصر مورد نظر رابا توجه به شکل زیر اجرا می کنیم :

	j=0	j=1	j=2	j=3	j=4
i=0	x00	x01	x02	x03	x04
i=1	x10	x11	x12	x13	x14
i=2	x20	x21	x22	x23	x24
i=3	x30	x31	x32	x33	x34
i=4	x40	x41	x42	x43	x44

-در قطراصلی اندیس سطرها (i) با اندیس ستون ها (j) برابراست یعنی i = j ؛ در عناصر بالای قطر اصلی i < j و در عناصر زیر قطر اصلی i > j است . لذا می توان قطعه برنامه ی زیر را برای چاپ عناصر بالای قطر اصلی نوشت :

```
for(i=0;i<5;i++) {
   for(j=0;j<5;j++)
      if( i < j)
         cout<< x [i][j]<<"  ";
    cout<<"\n";
}
```

روش دوم : در روش اول از میان کلیه عناصر آرایه ، عناصر بالای قطر اصلی را انتخاب و چاپ کرده ایم. برای اجرای سریع تر این برنامه ، تنها عناصر بالای قطر را چاپ کنیم .

برای چاپ اولین ردیف بالای قطر x00 x01 x02 x03 x04 می نویسیم :

```
for(j=1;j<5;j++)
   cout<< x [0][j];
cout<<"\n";
```

برای چاپ ردیف بعدی می توان نوشت:

```
for(j=2;j<5;j++)
   cout<< x [1][j];
cout<<"\n";
```

در سایر ردیف ها نیز این برنامه تکرار می شود . پس می توان نوشت :

```
for(i=0;i<=4;i++){
   for(j=i+1;j<5;j++)
      cout<< x [i][j];
   cout<<"\n";
}
```

نهایتا برنامه را به صورت زیر کامل می کنیم :

```
#include<iostream.h>
void main ()
{
   int  x[5][5];
   for(int i=0;i<5;i++)
      for(int j=0;j<5;j++)
         cin>> x[i] [j];
   for(i=0;i<4;i++) {
      for(j=i+1;j<5;j++)
         cout<< x [i] [j]<<" ";
      cout<<"\n";
   }
}
```

تمرین :

1-پس از دریافت یک ماتریس **5 5** مجموع عناصر قطر اصلی آن را چاپ کنید.

2-عناصر غیر صفر دو ماتریس **5 5** پایین مثلثی را دریافت کنید و مجموع این دو ماتریس راچاپ کنید.

3-پس از دریافت یک ماتریس **5 5** عناصر قطر فرعی را چاپ کنید.

4-کد شناسایی و معدل **10** دانش آموز را دریافت کنید. پس از مرتب سازی ارایه بر اساس معدل ، کد و معدل هر یک راچاپ کنید؟

5-با دریافت دو ماتریس**3 4** و**4 5** حاصل ضرب آن ها را چاپ کنید؟

گرافیک

زبان c++ می تواند خروجی برنامه ها رادردو حالت یاmode متنی وگرافیکی ایجاد کند . صفحه ی نمایش مد متنی دارای 80 ستون و 25 خط است و کاراکترها در این سطرها یا ستو ن ها قرار می گیرند . در مد گرافیکی صفحه ی نمایش از نقاطی بنام pixel تشکیل می شودو باتوجه به کارت گرافیکی و تنظیمات صورت گرفته ، تعداد نقاط یا ابعاد صفحه نمایش تعیین می شود . به طور پیش فرض ، خروجی در mode متنی قراردارد . برای استفاده از محیط گرافیکی باید از توابع موجود در C++ استفاده کرد.

1) تابع initgraph :

برای ورود به محیط گرافیکی باید براساس کارت گرافیکی و به کمک درایو آن ، مقادیر اولیه مورد نیاز کارت گرافیکی را فراهم کنیم . نحوه ی استفاده ازاین تابع به صورت زیر است:

```
initgraph (درایورگرافیک , مد گرافیک , مسیر فایل راه انداز یا درایور);
```

پارامتر اول درایو کارت گرافیکی است که توسط اعداد صحیح یامقادیرثابت تعیین شده در++c مشخص می شود . این مقادیر درجدول آورده شده است :

Constant	Value
DETECT	0
CGA	1
EGA	2
MCGA	3
EGA64	4
EGAMONO	5
IBM8514	6
HERCMONO	7
ATT400	8
VGA	9
PC3270	10

نکته : مقدار صفر ، شناسایی کارت گرافیک را بطور خودکار انجام می دهد . به همین دلیل ماباز این مقدار برای شناسایی کارت گرافیک استفاده می کنیم.

پارامتر دوم ، مد یا حالت گرافیکی را تعیین می کند . مقادیر این پارامتر دردرایو های گرافیکی مختلف ، متفاوت است وتعیین کننده ی ابعاد وتعداد رنگ های قابل پشتیبانی است . در درایور VGA که کارت های گرافیکی فعلی از آن استفاد ه می کنند ، مقادیر مد به صورت زیر است :

مد گرافیکی	مقدار	ابعاد صفحه	تعداد رنگ
VGALO	0	300 * 640	16
VGAMED	1	350 * 640	16
VGAHL	2	480 * 640	16

پارامتر سوم مسیر فایل درایور است ، که در زبان c بطور پیش فرض TC\BGI\ است.

مثال) دستو رات لازم برای ورود به محیط گرافیکی را بنویسید.

```
int  gdriver =DETECT, gmode;
initgraph(&gdriver,&gnode,"E:\\tc\\BGI");
```

توضیح :
چون شناسایی خودکار انتخاب شده است ، بطور اتومات مقادیر gdriver,gmode مشخص می شود .
نکته : اگر فایل راه انداز در مسیر جاری باشد ، رشته پوچ " " را به عنوان پارامتر سوم قرار می دهیم .

مثال)قطعه کدی بنویسید که محیط گرافیکی را با راه انداز کارت VGA در ابعاد 350*640 قرار دهد.

```
int  gdriver =9, gmode=1;
initgraph(& gdriver,&gnode,"E:\\tc\\BGI");
```

نکته : راه انداز VGA فایلی بنام EGIVGA.BGI واقع درشاخه BGI است.

2) تابع graphresult( ) :

در صورتی که آخرین عمل گرافیکی اجرا شده با خطا مواجه شود ، این تابع کد خطا را بر می گرداند . مقادیر بازگشتی این تابع در محدوده 0 تا 15- است .
اگرکارت گرافیکی به طور صحیح مقدار دهی شود ، خروجی این تابع صفر یا ثابت grOk است . ولی اگر گرافیک درست نصب نشود ، مقدار بازگشتی 1- است.

- سایر کدها ی خطا در help زبان c++ آورد ه شده است .

با توجه به توضیحات بالا دستورات لازم بری ورود به محیط گرافیکی را به صورت زیر کامل می کنیم :

```
int  gdriver,  gmode;
gdriver  =DETECT;
init graph(&gdriver,  &  gmode,  "E:\\tc\\BGI");
if(graphresult()!=grOk){
    cout<<"Graphic  error";
    exit(1);
}
```

توضیح :
در این برنامه اگر خطایی در تشخیص کارت گرافیکی رخ دهد از برنامه خارج می شویم .

3) تابع closegraph() : با اجرای این تابع سیستم از مد گرافیکی خارج می شود.

4) تابع setcolor(): تعیین کننده رنگ قلم در محیط گرافیکی است . مقدا ر پارامتر این تابع می تواند در محدوده صفر تا getmaxcolor باشد .

5) تابع getmaxx() : ماکزیمم مقدار ستون ها را بر می گرداند .

6) تابع getmaxy() : ماکزیمم مقدار سطر ها را بر می گرداند .

- در مد 9 کارت VGA مقدار getmaxcolor عدد 15 .مقدار getmaxx عدد 639 و getmaxy عدد 479 است.

8) تابع line() : برای رسم یک خط با مشخص کر دن مختصات دو راس آن بکار می رود .

```
line(x1 , y1  , x2 , y2) ;
```

(x1 ,y1) : مختصات نقطه ی اول (x2,y2) مختصات نقطه ی دوم است .

9) تابع lineto() : از نقطه ی جاری تا نقطه ای به مختصات x,y خطی رسم می کند .

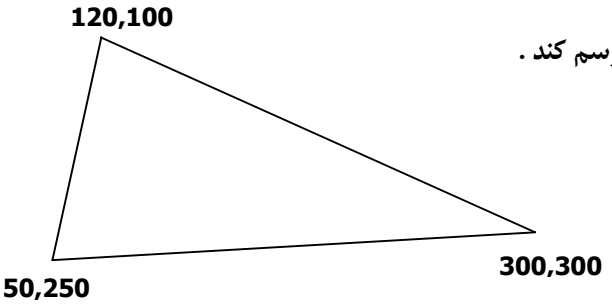
```
lineto(x ,  y) ;
```

10) تابع moveto() : تنظیم نقطه ی جاری با نقطه ای به مختصات x,y .

```
moveto(x,y) ;
```

11) تابع linerel() : از نقطه ی جاری تا نقطه ی x,y خطی رسم می کند تفاوت آن با lineto ، نسبی بودن مختصات x,y نسبت به نقطه ی جاری است .

مثال) برنامه ای بنویسید که شکل مقابل را رسم کند .



توضیح :

پس از ورود به محیط گرافیکی در مد VGA ، توسط روش های زیر می توان مثلث بالا را رسم کرد.

روش اول : استفاده از line :

```
line (120,100,50,250);
line(50,250,300,300);
line(300,300,120,100);
```

روش دوم : استفاده از lineto :

```
move to(120,1000);
lineto(50,250);
lineto(300,300);
lineto(120,100)
```

پس از انتقال نقطه ی جاری به نقطه ی (120,100) ، توسط lineto می توان تا نقطه(50,250)یک خط رسم کرد. پس از اجرای این دستور نقطه جاری نیز به (50,250)منتقل می شود. سپس تا(300,300)خطی رسم می کنیم و ...

روش سوم : در این روش از linerel استفاده کرده و مختصات نقاط را نسبت به نقطه ی جاری تعیین می نماییم . مثلا اگر نقطه ی جاری (120,100) باشد ، برای رفتن به (50,250) باید 70 واحد از x کم کرده و 150 واحد به y اضافه کنیم. ...

```
move to(120,100);
linerel(-70,150);
linerel(250,50);
linerel(-180,-200)
```

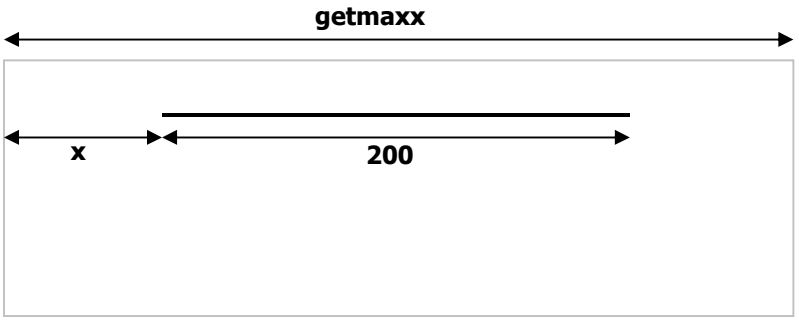
برنامه را با اجرای روش سوم کامل می نویسیم :

```
#include<conio .h>
#include<graphics.h>
#include<stdlib.h>
void main()
{
   int  gd,gm;
   gd=DETECT;
   intgraph(&gd,& gm,"e\\tc\\bgi");
   if(graphresult() != grOk)  exit(1);
   moveto(120,100);
   linerel(-70,150);
   linerel(250,50);
   linerel(-180,-200)
   getch();
   closegraph();
}
```

مثال: خطوطی موازی با طول فرضی 200 در وسط عرض صفحه نمایش به فاصله ی 15 نقطه ازیکدیگر ، به صورت افقی رسم کنید . رنگ های خطوط از 1 تا getmaxcolor باشد.

توضیح :

خط اول را با فاصله 50 نقطه از بالای صفحه رسم می کنیم . برای رسم این خط در وسط عرض صفحه نمایش ، فاصله x را مطابق شکل زیر بر اساس مقادیر getmaxx و طول فرضی 200 بدست می آوریم.




```
x=(getmaxx - 200)/2;
```

پس از پیدا کردن مقدار x ، خط اول با دستورات زیر رسم می کنیم :

```
setcolcor(1);
line(x,50,x+200,50)
```

خط دوم را با فاصله ی 15 نقطه از خط اول به صورت زیر رسم می کنیم :

```
setcolcor(2);
line(x,50+15,x+200,50+15);
```

خط سوم نسبت به خط اول به اندازه ی 2*15 نقطه پایین تر رسم می شود :

```
setcolcor(3);
line(x,50+2*15,x+200,50+2*15);
```

سایر خطوط نیز بر اساس همین ساختار رسم می شوند.

دستورات فوق را به کمک یک حلقه تکرار می کنیم :

```
for (i=1 ; i<=getmaxcolor ; i++) {
   setcolor(i);
   line(x,50+   *15,x+200,   *15);
}
```

ضریب 15 را می توان با توجه به مقدار i ، یا یک متغیر دیگر ، مشخص کرد . اگر متغیری مانند k را انتخاب کنیم ، برنامه بصورت زیر نوشته می شود :

```
for (i=1 , k=0 ; i<=getmaxcolor ; i++,k++) {
   setcolor(i);
   line(x,50+ k*15,x+200, k*15);
}
```

واگر از i استفاده کنیم ، می نویسیم :

```
for (i=1  ; i<=getmaxcolor ; i++) {
   setcolor(i);
   line(x,50+ (i-1)*15,x+200, (i-1)*15);
}
```

برنامه را می توان به صورت زیر کامل کرد :

```
#include<graphics.h>
#include<conio.h>
#include<stdlib.h>
 void main( )
{
    int   gd=DETECT  , gm;
    initgraph(&gd,  &gm ,  "e:\\te\\bgi" )
    if (graphresult( ) == grOk)  {
       int  x =  (maxx-200)/2;
       for(int i =1  ;  i<= getmaxcolor() ; i++ ) {
          setcolor ( i);
          line(x, (i-1)*15,x+200,(i-1)*15);
       }
       getch();
       closegraph ();
    }
}
```

11) تابع circle () : از این تابع برای رسم دایره استفاده می شود.

```
circle(x , y , radius);
```

x , y : مختصات مرکز و Radius شعاع دایره است .

12) تابع arc () : برای رسم کمانی از یک دایره به شعاع Radius و مرکز x , y به کار می رود :

```
arc (x , y , Stangle , Endangle , Radius);
```

Stangle = زاویه ی شروع کمان Endangle = زاویه ی پایان کمان

نکته : مقادیر Stsangle و Endangle بر حسب درجه است .

13) تابع ellipse ( ) : رسم بیضی یا کمانی از بیضی.

```
ellipse ( x , y , stangle , endangle , xradius , yradius );
```

xradius = شعاع افقی yradius = شعاع عمودی

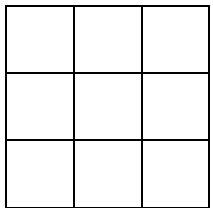
مثال) در مرکز صفحه ی نمایش دایره ای به شعاع عدد ورودی رسم کنید.

```
void  main  ( )
{
   int  gr=0  ,  gm , r  ;
   intitgraph  (&gr , &gm , " e:\\tc\\bgi" ) ;
   if ( graphresult ()  != 0 )  exit( 1 ) ;
   cout <<" enter the radius  =  "  ;
   cin>> r;
   circle ( germaxx/2 , getmaxy/2 ,r );
   getch ( ) ;
   closegraph ( ) ;
}
```

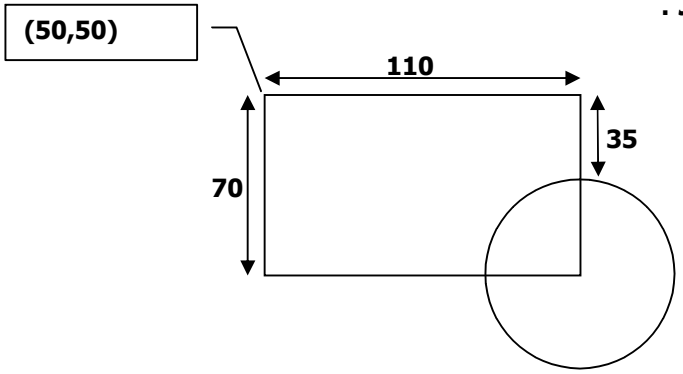
تمرین :

1-پس از ترسیم یک مثلث متساوی الاضلاع فرضی میانه های مثلث را رسم کنید.

2-با استفاده از دستور **line** جدول زیر را رسم کنید .



3 - شکل مقابل را رسم کنید .



4-ابتدا بر روی صفحه دایره ای فرضی رسم کنید ، برنامه ای بنویسید که این دایره از سمت چپ صفحه ی نمایش به سمت راست منتقل شود. راهنمایی : (برای انتقال دایره ، ابتدا بر روی دایره جاری ، دایره ای با رنگ زمینه رسم می کنیم و دایره جدید را با تغییر **x** مرکز دایره رسم می کنیم . سرعت اجرا برنامه ، مانع از دیدن حرکت دایره می شود ، لذا از تابع **delay** برای ایجاد تاخیر استفاده می کنیم. پارامتر این تابع مقداری بر حسب میلی ثانیه است و در فایل سر تیتری **Dos.h** است)

مثال) برنامه ای بنویسید که در وسط صفحه ی نمایش ، دوایری هم مرکز ، با شعاع های تصادفی در محدوده ی **10** تا **200** رسم کند . در ضمن رنگ دوایر نیز تصادفی تولید شود. این برنامه با فشردن یک کلید خاتمه می یابد .

توضیح :

برای تولید یک عدد تصادفی از تابع **random** استفاده می کنیم . این تابع عددی تصادفی در بازه **0** تا **num – 1** تولید می کند .

random(num) ⟵ عدد تصادفی

- این تابع در فایل سر تیتری **stdlib.h** است .

نکته : برای پیدا کردن عدد تصادفی در محدوده ی **A** تا **B** **(A<B)** از رابطه ی زیر استفاده می کنیم .

random (B-A+1) +A

در این مثال به اعداد تصادفی در محدوده ی [200 , 10] احتیاج داریم . رابطه ی **random(191) +10** این عدد را تولید می کند .

با توجه به توضیحات فوق ابتدا یک دایره ی تصادفی رسم می کنیم :

```
c=  random( getmaxcolor+1);
r= random(191)+10;
x = getmaxx/2;
y= getmaxy/2;
setcolor(c) ;
circle(x,y,r);
```

با قرار دادن این برنامه در یک حلقه می توان دوایری با شعاع های تصادفی رسم کرد .

```
do {
      c=  random( getmaxcolor+1);
      r = random( 191)+10;
      x= getmax x/2  ;
      y=  getmax  y/2 ;
      setcolor ( c) ;
      circle  ( x , y , r );
    }while  ( ! kbhit () );
```

برای شرط این حلقه از تایع ( )kbhit استفاده می کنیم که در فایل سر تیتری conio.h است. تا زما نی که کلیدی فشرده نشود این تابع مقدار صفر را بر می گرداند ، وبا فشردن یک کلید مقدار باز گشتی مخالف صفر می شود. پس می توان از شرط (0=! ()kbhit)یا (()kbhit!) در while استفاده کرد.
برنامه ی کامل به صورت زیر نوشته می شود :

```
#include<graphics. h>
#include<stdlib.h>
#include<conio.h>
void main ( )
{
    int  gr= DETECT , gm ;
    randomize( ) ;
    initgraph( &gr,&gm," e:\\tc\\bgi") ;
    do{
          setcolor(random( getmaxcolor+1)
          circle(getmaxx/2 ,  getmaxy/2 , random(191)+10 );
    }while  ( !kbhit() );
    getch( ) ;
    closegraph ( );
}
```

توابع :

یکی از ویژگی های مهم برنامه نویسی ساخت یافته ، استفاده از روالها یا توابع است . در برنامه های بزرگ می توان برنامه را به اجزاء کوچکتری به نام تابع تقسیم کرد. مزیت های استفاده از این روش ، عبارت است از :

1-کاهش حجم کد نویسی
2-خوانا تر شدن برنامه و تجزیه و تحلیل ساده تر
3-خطا یابی ساده تر

درزبان c++ می توان تابع را در دوقالب یا الگوی زیر معرفی کرد :

1) تابعی که مقدار بازگشتی ندارد : هدف استفاده از این نوع تابع ، انجام یک عمل خاص است . مثلاً تابعی که عناصر یک آرایه را مرتب کند ، تابعی که مقسوم علیه های x را چاپ کند ، تابعی که صفحه ی نمایش را پاک می کند و یا تابعی که خطی را رسم کند و ... .

الگوی نوشتن این نوع تابع به صورت زیر است :

```
void  اسم تابع (لیست پارامتر ها و نوع آن ها)
{
----------
----------      } بدنه تابع
----------
}
```

نکته : هنگام استفاده از این نوع تابع , اسم تابع و پارامترهای آن را به عنوان یک دستور مستقل می نویسیم ,

مثلا :

```
clrscr() ;
line(100,50,420,50);
```

2) تا بعی که یک مقدار بازگشتی دارد : در این نوع تابع پس از اجرای وظیفه ی تابع , یک مقدار به برنامه ی فراخوان بازگردانده می شود .

```
نوع بازگشتی  اسم تابع (لیست پارامتر ها و نوع آن ها)
{
----------
----------      } بدنه تابع
----------
   return مقدار بازگشتی ;
}
```

در برنامه ی فراخوان می توان از مقدار بازگشتی استفاده کرد .مثلا :

```
x= random (191)+10;
y= sqrt(x-5);
s= s+1/fac(i);
```

چگونگی انتخاب نوع تا بع و وظیفه ی آن :

یکی از مهمترین مراحل برنامه نویسی ، تجزیه و تحلیل برنامه است . در این مرحله وظیفه ی هر تابع و نوع آن ها مشخص می شود . انتخا ب وظیفه ی هر تابع برا ساس قواعد زیر صورت می گیرد :

1-وظایف برنامه ، بین برنامه ی اصلی و توابع بدرستی تقسیم شود .
2-تابع به گونه ای نوشته شود که کاربرد بیشتری داشته باشد .
3-تنها یک وظیفه بر عهده ی تابع باشد

مثال) برنامه ای بنویسید که اعداد اول کمتر از n را چاپ کند .
توضیح : در این برنامه می توان تابع را به صورت های مختلفی تعریف کرد ، که بستگی به نظر برنامه نویس دارد . مثلا :

1-تابع f(x) : عدد x را بعنوان پارامتر دریافت کرده و در صورت اول بودن عدد ، آن را چا پ کند .
2-تابع f(x) : عدد xرا بعنوان پارامتر دریافت کرده و تعداد مقسوم علیه های آن را برمی گرداند
3-تابع f(x) : عدد x را بعنوان پارامتر دریافت کرده و در صورت اول بودن عدد 1 وگرنه عدد صفر را بر می گرداند .
4-تابع f(x) : عدد x را بعنوان پارامتر دریافت و اعداد اول کمتر از آن را چاپ می کند .

نکته : در حا لت 1و4 تابع مقدار بازگشتی ندارد اما در حا لت 2و3 مقدار بازگشتی دارد .
نکته : در حا لت 4 کل وظیفه ی برنامه بر دوش تابع است ، لذا انتخاب مناسبی نیست .
نکته : در حالت 2و3 تابع می تواند کاربرد بیشتری داشته باشد.

همان طور که دیده می شود انتخاب وظیفه یا عملکرد تابع مهم است و برنامه اصلی مطابق با وظیفه تهیین شده برای تابع ، نوشته می شود .

این مثال را را بر اساس تعریف 2 حل می کنیم. ابتدا با توجه به وظیفه ی مشخص شده برای تابع ، برنامه آن را می نویسیم.

```
int f  (int x)  {
   int k=0;
   for(int i=1;i,=x; i++)
      if(x%i == 0) k++;
    return k;
}
```

x پارامتر ورودی تابع است که از برنامه ی فراخوان (برنامه استفاده کننده از تابع) به تابع آورده شده است و k تعداد مقسوم علیه های x است ، که برگردانده می شود .
از این تابع در برنامه ی اصلی استفاده می کنیم و اعداد 1و2و3و... تا n را به تابع می فرستیم . اگر تعداد مقسوم علیه ها (مقدار بازگشتی) دو باشد ، یعنی عدد اول است .
برای سادگی ابتدا عدد 1 را به تابع ارسال می کنیم و بازگشتی عدد را بررسی می کنیم .

```
if( f(1)==2)
   cout<<1<<"\t";
```

همین دستور را بر روی اعداد 2 تا n تکرار می کنیم . پس توسط یک حلقه ، قطعه برنامه ی فوق را تکرار می کنیم :

```
for(i=1;i,=n;i++)
    if( f(i)==2)
      cout<<i<<"\t";
```

نهایتا می توان برنامه را به صورت زیر کامل کرد :

```
#include<iostream.h>
void main( )
{
    int f(int x);
    int n;
    cout<<"n="; cin>>n;
    for(int  i=1;i<=n;i++)
        if(f(i)==2)
           cout<<i<<"\t";
}
int f(int x)
{
    int k=0;
    for(int i=1; i<=x; i++)
        if(x%i==0)   k++;
     return k;
}
```

نکته: در این مثال برنامه ی تابع **f** ، بعد از تابع فراخوان **(main)** آورده شده است ؛ به همین دلیل ، در بخش تعاریف تابع **main** ، معرفی تابع **f** را نیز انجام داده ایم . اما اگر تابع قبل از برنامه ی فراخوان قرار گیرد ، نیازی به این معرفی نیست.

نکته : در هر دو تابع متغیری به نام **i** استفاده شده است . چون این متغیر در هر دو تابع محلی است ، هیچ ارتباطی بین آن دو وجود ندارد. (متغیر محلی تنها درون همان بلاکی که تعریف می شود قابل دسترسی است و با خروج از بلاک فضای اشغال شده را به سیستم بر می گرداند .)

مثال):برنامه ای بنویسید که حاصل عبارت زیر را چاپ کند.

$$1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\ldots\pm\frac{x^n}{n!}$$

توضیح:

برای حل این سوال ، از دو تابع استفاده می کنیم.

1- **fac(x)** : تابعی که با دریافت **x** ، فاکتوریل **x** را برگرداند .

2- تابعی که با دریافت دو عدد **n** و **x** ، حاصل x^n را به دست آورد. البته چنین تابعی به نام **Pow** در زبان **c** تعریف شده است و نیازی به نوشتن برنامه ی این تابع نیست .

تابع فاکتوریل را به صورت زیر می نویسیم :

```
long double fac(int x)
{
   long double f=1;
   for(int i=1;i<=x;i++)
       f*=i;
   return f;
}
```

حال به جای جملات سری از توابع مورد نظر استفاده می کنیم . در ضمن تفریق ها را نیز به جمع تبدیل می کنیم.

$$1+\overset{a}{\boxed{(-1)}}\times\frac{pow(x,2)}{fac(2)}+\overset{a}{\boxed{(1)}}\times\frac{pow(x,4)}{fac(4)}+\overset{a}{\boxed{(-1)}}\times\frac{pow(x,6)}{fac(6)}+\ldots+(\pm1)\times\frac{pow(x,n)}{fac(n)}$$

در این عبارت می خواهیم حاصل جمع $a \times \frac{pow(x,i)}{fac(i)}$ رابدست آوریم . متغیر i ابتدا 2 سپس 4،6و... خواهد بود لذا برنامه را به کمک یک حلقه به صورت زیر می نویسیم :

```
a=-1; s=0;
for(int i=2; i<=n;i=i+2){
     s=s+pow(x,i)/fac(i)*a;
     a=-a;
}
cout<<s+1;
```

چون مقدار اولیه ی s=0 فرض شده است ، در پایان ، s+1 را چاپ کرده ایم ، تا اولین 1 سری نیز در جمع شرکت کند. می توانستیم مقدار اولیه ی s را 1 در نظر بگیریم تا نیازی به جمع عدد 1 در هنگام چاپ نباشد.
برنامه ی کامل شده را به صورت زیر می نویسم:

```
long double fac(int x)
{
   long double f=1;
   for(int i=1; i<=x; i++)
      f*=i;
   return  f;
}
void main()
{
   double  s=1;
   int x ,n,a=-1;
   cout<<"enter two numbers x,n";
   cin>>x>>n;
   for(int i=2; i<=n; i+=2) {
      s+=a*pow(x,i)/fac(i);
      a=-a;
   }
   cout<<s;
}
```

مثال):پس از دریافت یک آرایه ی ده عضوی به عنوان نمرات ده دانش آموز، فراوانی هر نمره را چاپ کند.
توضیح:
این مسئله را قبلا در حالتی خاص ، یعنی گرد بودن نمرات به عنوان تمرین بررسی کردیم . دراین مثال می خواهیم بدون گرد کردن اعداد ، فراوانی را بدست آوریم . مناسب ترین شرط برای پیدا کردن سریع فراوانی هر عضو ، مرتب بودن آرایه است. پس ابتدا آرایه را به کمک تابع sort مرتب می کنیم، سپس آرایه مرتب شده را به تابع دیگری به نام count منتقل می کنیم تا فراوانی ، هر عضو را چاپ کند.
تابع sort(x) : آرایه ی x را به عنوان پارامتر دریافت کرده و آن را به صورت صعودی مرتب می کند.

```
void sort (float x[10])
{
   float  t;
   for(int  j=1;j<=9;j++)
      for(int i=0;i<9;i++)
         if(x[i]>x[i+1]){
            t=x[i];
            x[i]=x[i+1];
            x[i+1]=t;
         }
}
```

تابع count(x) : آرایه ی x را به عنوان پارامتر دریافت کرده و فراوانی هر عنصر را چاپ می کند.

توضیح :

چون آرایه مرتب است عناصر مشابه مجاورهم قرار دارند. لذا برای پیدا کردن فراوانی ، هر عنصر را با عنصر بعد از آن مقایسه می کنیم .تا زمانی که عنصری با عنصر بعدی از خود مساوی باشد ، تعداد آن ها را با افزودن یک واحد به یک متغیر کمکی مانند k ، بدست می آوریم. اگر عنصرجاری با عنصر بعدی مساوی نباشد ، دیگر عنصری مشابه عنصر جاری وجود ندارد و تعداد عنصر جاری را به همراه خود عنصر چاپ می کنیم ومقدار k را مساوی 1 قرار می دهیم تا برای پیدا کردن فراوانی عنصر بعدی آماده شده باشد . فقط برای آخرین عنصر ، خارج از حلقه ی for برنامه نوشته ایم، چون i+1 به خارج از آرایه انتقال می یابد.

```
void count(float x[10])
{
   int k=1;
   for(int i=0;i<q; i++)
       if(x[i] == x[i+1]
           k++;
       else {
           cout<<x[i]<<"\t"<<k<<"\t;
           k=1;
        }
    if(x[9]==x[8])
       k++;
    else
        k=1;
    cout<<x[q]<<"\t"<<k<<"\n";
}
```

به کمک دو تابع برنامه ی اصلی را می نوسیم:

```
void main( )
{
   float  x[10];
   for(int i=0;i<=q;i++){
      cout<<"enter mark"<<i+1<<"=";
      cin>>x[i]
   }
   sort(x);
   count(x);
}
```

نکته:چون آرایه ی x محلی تعریف شده است پس فقط درون main اعتبار دارد ، لذا برای استفاده از آرایه در توابع sort و Count آن را به عنوان پارامتر به دورن توابع ارسال کرده ایم.

نکته : اگر آرایه را خارج از بدنه توابع (بالاتر از تمامی توابع) تعریف می کردیم ، آرایه ی x سراسری در نظر گرفته می شد و توسط کلیه توابع بعد از تعریف ، قابل دسترسی بود، و نیازی به ارسال آرایه به تابع نبود.

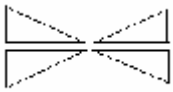
نکته: مقدار اولیه ی متغیر سراسری صفر و مقدار متغیر محلی نامعین است.

نکته: اگر نام متغیر محلی و سراسری یکی باشد اولویت با متغیر محلی است.

تمرین:

1-برنامه ای بنویسید که توسط یک تابع ، **BMM** بین دو عدد ورودی را به دست آورد پس توسط آن **kmm** بین دو عدد را چاپ کند.

2-تابعی بنویسید که با دریافت رئوس یک مثلث آن را رسم کند . به کمک این تابع شکل زیررا رسم کنید.



3-توسط تابعی که مجموع مقسوم علیه های **x** رابرمی گرداند، برنامه ای بنویسید که اعداد تام کمتراز**n** راچاپ کند.

4-تابعی بنویسیدکه میانگین ارقام یک عدد صحیح راچاپ کند.به کمک این تابع برنامه ای بنویسید که بادریافت یک آرایه ی **5** عنصری، هرعنصررا به همراه میانگین ارقام آن چاپ کند.

5-تابعی بنویسید که آرایه ی **x** وعدد **A** رابه عنوان پارامتر دریافت کرده و وجود **A** را در آرایه ی **x** بررسی کند . به کمک این تابع برنامه ای بنویسید که **10**عدد منحصربه فرد دریک آرایه قراردهد. سپس آرایه ی دریافت شده را چاپ کنید.

6-تابعی بنویسیدکه یک مثلث با مختصات تصادفی رسم کند.به کمک این تابع **10** مثلث تصادفی درصفحه چاپ کنید.

7-با دریافت عدد**n** به عنوان توان چند جمله ای ، ضرایب آن را چاپ کنید . مثلا : اگر **n=4** باشد چند جمله ای به فرم زیر نوشته می شود :

$$\binom{4}{0}x^4 + \binom{4}{1}x^3y + \binom{4}{2}x^2y^2 + \binom{4}{3}x^3y + \binom{4}{4}y^4$$

هر یک از ضرایب از رابطه زیر به دست می آید. مثلا:

$$\binom{4}{2} = \frac{4!}{2!\times(4-2)!} = 6$$

فایل

کلیه ی داده های موردنیاز برنامه هایی که تاکنون نوشته ایم ، درقالب متغیر(ساده یا آرایه) ذخیره می شدند. چون متغیرها درحافظه **RAM** تشکیل می شوند ، باخاتمه ی برنامه یا قطع برق ازبین می روند.لذا نمی توان مقدار آن هارا دراجرای مجدد برنامه حفظ کرد. به همین دلیل داده هارا در حافظه های جانبی ذخیره می کنیم ودرحافظه ی جانبی داده ها در قالب فایل ذخیره می شوند.

انواع فایل

1-فایل متنی:

a. داده های این نوع فایل در قالب کد های اسکی ذخیره می شوند .
b. دراین نوع فایل پایان هرخط با کدهای **10و13** (یعنی **n**) مشخص می شود .
c. پایان فایل باکاراکتر انتهایی فایل (کد اسکی **26**)مشخص می گردد.
d. اعداد نیز به صورت رشته ای از نشانه ها ذخیره می شوند ، لذا فضای اشغال شده توسط اعداد ، معادل تعداد نشانه های عدد است.

2-فایل باینری :

a. داده ها به صورت کدهای باینری ذخیره می شوند.
b. دراین فایل **n** معادل کاراکتری با کد اسکی**10** است وکاراکتر کد اسکی **13** در فایل ذخیره نمی شود.
c. اعداد بصورت باینری ذخیره می شوند. لذا فضایی را اشغال می کنند که درون متغیر ها اشغال می کردند .
d. در این فایل کاراکتری بعنوان انتهای فایل وجود ندارد.

روش های دسترسی به فایل

1-ترتیبی : رکورد ها یا عناصر این فایل ، به ترتیب ذخیره شدن ، در فایل قرار می گیرد و با همان ترتیب نیز خوانده می شوند . یعنی اگر بخواهیم رکورد**100** فایل را بخوانیم باید **99**رکورد قبل از آن را نیز بخوانیم.

2-تصادفی : دراین روش به هررکورد یک عدد نسبت داده می شود که توسط آن می توان مستقیما به آن رکورد دسترسی پیدا کرد.

بازکردن فایل

برای دسترسی به محتویات فایل باید ابتدا آن را بازکرد. برای این منظورازتابع **fopen** استفاده می کنیم :

```
fopen(نام فایل ونام آدرس , حالت یا مد);
```

پارامتراول این تابع آدرس و نام فایل برروی دیسک است و پارامتردوم نوع فایل و هدف ازباز کردن فایل را مشخص می کند . این پارامتررشته ای با مقادیر زیراست :

Mode	توصیف یا عملکرد
w یا wt	بازکردن فایل جدید متنی(اگر فایل موجود باشد کلیه داده های آن ازبین می رود.)
r یا rt	بازکردن فایل متنی موجود به منظور خواندن اطلاعات آن (بعنوان ورودی)
a یا at	بازکردن فایل متنی موجود به منظورافزودن اطلاعت به آن
wb	بازکردن فایل باینری جدید
rb	بازکردن فایل باینری موجود به عنوان ورودی
ab	بازکردن فایل باینری موجود جهت افزودن اطلاعات به آن
w+ یا w+t	بازکردن فایل متنی جدید برای خواندن ونوشتن
r+ یا r+t	بازکردن فایل متنی موجود برای خواندن ونوشتن
a+یا a+t	بازکردن فایل متنی موجود برای خواندن ونوشتن
w+b	بازکردن فایل باینری جدید برای خواندن ونوشتن
r+b	بازکردن فایل باینری موجود برای خواندن ونوشتن
a+b	بازکردن فایل باینری موجود برای خواندن ونوشتن

نکته : تفاوت a+t با r+t تنها در موقعیت قرار گرفتن اشاره گردرون فایل است. در r+t موقعیت سنج به اولین رکورد اشاره می کند و دراt+a موقعیت سنج درانتهای فایل قرار می گیرد . به عبارت دیگرr+t ، فایل برای خواندن وa+t ، فایل را به منظور افزودن داده ها باز می کند.

نکته : مقداربازگشتی تابع fopen اشاره گری ازنوع FILE است. این اشاره گر پس از بازشدن فایل به منظور دسترسی به فایل مورد استفاده قرارمی گیرد.مثال:

```
FILE * fp;
fp = fopen ("c:\\st.dat","w+t");
```

نکته: اگرتابعfopen موفق به بازشدن فایل نشود ، مقدار NULL رابرمی گرداند. لذا ازاین مقدار می توان جهت اطمینان ازباز شدن فایل استفاده کرد.مثلاً :

```
FILE *f1;
f1 = fopen ("c:\\boot.ini","rt");
if (f1==NULL) {
    cout«"file not found";
    exit(1);
}
```

با توجه به دستورات نوشته شده اگرتابع fopen نتواند فایل Boot.ini رابازکند ، بانمایش یک پیغام برنامه را خاتمه داده ایم .

تابع **fclose( )** : جهت بسته شدن فایل بازشده ازاین تابع استفاده می شود. مثلاً :

```
fclose(f1);
```

- این دستورفایلf1 که درمثال قبل بازشده است رامی بندد.

روش های ذخیره وبازیابی اطلاعات

دسترسی به اطلاعات فایل به چهارروش امکان پذیراست . که دراین بخش دوروش پرکاربردتر آن بررسی می شود.

1-ورودی وخروجی کاراکتر : یکی ازروشهای خواندن ونوشتن داده ها ، بصورت کاراکتر به کاراکتراست .که درفایل های متنی کاربرد زیادی دارد. برای خواندن یک کاراکترازماکروgetc وبرای نوشتن یک کاراکتر درفایل ازماکروputc استفاده می کنیم.

```
getc ( اشاره گرفایل );  ◄——— نشانه خوانده شده ازفایل
```

```
Putc( نشانه , اشاره گر فایل );
```

مثال) درفایل Boot.ini تعدادنشانه های عددی راچاپ کنید.

توضیح :

برای این منظورتک تک نشانه هارا ازفایل خوانده و درصورت عددی بودن نشانه ی خوانده شده، تعداد آن هارابدست می آوریم. این عمل راتا پایان فایل ادامه می دهیم.

```
#include<stdlib.h>
#include<stdio.h>
void main( )
{
    FILE * f;
    char ch;
    int k=0;
    f=fopen ("c;\\boot.ini","rt");
    if(f==NULL) exit(1)
    ch=getc(f);
    while( ch !=eof ) {
       if(ch>='0'&& ch<='9')
          k++;
       ch=getc(f);
    }
    cout«k;
    fclose(f);
}
```

پس ازبازکردن فایل ابتدا یک نشانه ازفایل رامی خوانیم و درون متغیر ch قرار می دهیم. اگرنشانه ی خوانده شده با کاراکتر انتهایی فایل(ثابتEOF)برابرنباشد، آن را تست کرده و در صورت عددی بودن ، بهk یک واحد اضافه می کنیم . بر روی تک تک نشانه های خوانده شده از فایل همین عمل را تکرار می کنیم، تا به آخرفایل برسیم.

2-ورودی وخروجی رکورد: یکی ازپرکاربردترین شیوه های ذخیره و بازیابی است که اطلاعات را درقالب رکورد ذخیره می کند.رکورد به مجموعه ای ازعناصراحتمالاً غیر هم نوع که دارای وجه اشتراک خاصی باشندگفته می شود .مانند مشخصات یک کتاب، مشخضات یک دانش آموزو...

نکته : این شیوه فقط مختص رکورد نیست و می توان داده های عددی،رشته ای و...را نیزدرفایل ذخیره کرد.

در این روش برای خواندن ازتابع**fread** و برای نوشتن ازتابع **fwrite** استفاده می کنیم.

fread(آدرس بافر , حجم اطلاعات خوانده شده , تعداد , اشاره گر فایل) ;

fwrite(آدرس بافر , حجم اطلاعات خوانده شده , تعداد , اشاره گر فایل) ;

- پارامتراول ، آدرس بافراست. هنگام خواندن یا نوشتن اطلاعات درفایل ازحافظه ای موقت به عنوان بافراستفاده می کنیم ، که درحقیقت یک متغیریا یک آرایه است .
- پارامتردوم ، اندازه یا ظرفیت بافری است که می خواهیم اطلاعات آن را درفایل ذخیره کنیم یا بخوانیم.
- پارامتر سوم ، تعداد داده هایی است که می خواهیم درفایل بنویسیم یا بخوانیم . معمولاً این پارامتر 1 است ؛ مگراینکه بخواهیم یک آرایه را درفایل ذخیره کنیم یا بخوانیم .
- پارامتر چهارم همان اشاره گرفایل است که تابع**fopen** دراختیارما قرارداده است.

مثال) درون فایل**NUM.DAT** درریشه **D:** ، **10** عدد صحیح ذخیره کنید. (این فایل وجود ندارد.)
توضیح : ابتدا دستورات لازم باز کردن فایل را می نویسیم :

```
FILE * f;
int x;
f=fopen ("d:\\ num.dat","wb");
if(f==NULL) exit(1);
```

نکته : بدلیل عددی بودن داده ها ، فایل را از نوع باینری انتخاب کرده ایم . و چون فایل وجود ندارد آن را بصورت **w** بازکرده ایم .
تک تک اعداد را از ورودی دریافت کرده و درون فایل قرار می دهیم . برای ذخیره اولین عدد ، قطعه برنامه ای به صورت زیر می نویسیم :

```
cout<<"x=";
cin>>x;
fwrite(&x,2,1,f);
```

همین قطعه برنامه را برای سایر ورودی ها نیز تکرار می کنیم :

```
for(i=1;i<=10;i++) {
    cout<<"x=";
    cin>>x;
    fwrite(&x,2,1,f);
}
```

برنامه را بصورت زیر کامل می کنیم :

```
#include<iotream.h>
#include<stdlib.h>
#include<stdio.h>
void main()
{
   FILE * f;
   int x;
   f=fopen ("d:\\ num.dat","wb");
   if(f==NULL) exit(1);
   for(i=1;i<=10;i++) {
       cout<<"x=";
       cin>>x;
       fwrite(&x,2,1,f);
   }
   fclose(f);
}
```

مثال) 10 عدد اعشاری درون فایل جدیدی به نام NUM2.DAT قرار دهید.

توضیح :

این فایل را نیز به صورت WB باز می کنیم.

```
FILE *f;
f = fopen ("d:\\num2.dat","wb");
```

در این مثال از یک آرایه به عنوان بافر استفاده می کنیم ، یعنی ابتدا 10 عدد دریافت کرده ودرون یک آرایه قرار می دهیم پس آرایه را در فایل می نویسیم:

```
float x[10];
for ( i = 0;i <10;i++)
   cin>>x[i];
fwrite (x,4,10,f)
```

در دستور fwrite ، پارامتر اول اسم ارایه است که معادل آدرس آرایه x است (نباید از نشانه & استفاده کرد). ظرفیت هر عنصر نوع float ، 4 بایت می باشد که پارامتر دوم تابع را مشخص می کند. تعداد عنصرهای نوشته شده در فایل ، پارامتر سوم را تعیین می کند.

```
#incude<iostream.h>
#include <stdio.h>
void main( )
{
   float x [10];
   file *f;
   f = fopen("d:\\num2.dat","wb");
   for (int i =0;i<10; i++)
      cin>>x[i];
  fwrite (x,4,10,f);
  fclose(f);
}
```

مثال) پس از در یافت یک آرایه ی 5 10 ، عناصر آرایه را در یک فایل به نام num3.dat در مسیر جاری ذخیره کنید.

روش 1 : ابتدا آرایه دو بعدی را بطور کامل دریافت کرده و درون متغیری از نوع آرایه قرار می دهیم . سپس آن را در فایل می نویسیم .(همانند مثال قبل)

```
#include<iostream.h>
#include<stdio.h>
void main( )
{
   int x[5][10];
   file * f;
   f = fopen ("d:\\num3.dat","wb");
   for (int i = 0;i<5;i++)
     for (int j =0;j<10;j++)
        cin>>x[i][j]
   fwrite(x,2,5*10,f);
   fclose (f);
}
```

روش دوم : در روش قبل حجم حافظه توسط x بیهوده اشغال شده است . در حالی که می توانستیم فقط از یک عنصر ساده استفاده کنیم و آن را یکی یکی دریافت و درون فایل ذخیره کنیم. لذا برنامه ی فوق به صورت زیر بازنویسی می شود :

```
#include<iostream.h>
#include<stdio.h>
void main( )
{
    FILE  *f;
    int x;
    f = fopen  ("d:\\num3.dat","wb");
    for  (int i = 0;i<5;i++)
       for (int j =0;j<10;j++){
          cin>>x;
          fwrite(&x,2,5*10,f);
       }
    fclose (f);
}
```

نکته : در این روش می توانستیم بجای دو حلقه for متداخل ، از یک حلقه for با 50 بار تکرار استفاده کنیم.

مثال) از میان محتویات فایل num.dat اعداد زوج را چاپ کنید.

توضیح : در فایل num.dat تعدادی عدد صحیح ذخیره کرده بودیم . حال تک تک اعداد موجود در فایل را خوانده و اعداد زوج را چاپ می کنیم . برای خواندن یک عدد از تابع fread استفاده می کنیم.

```
int x;
fread (&x,2,1,f );
```

عدد خوانده شده باز هم درون بافر (x)قرار گرفته است. اگر x زوج باشد ، آن را چاپ می کنیم.

```
if (x %2 == 0)   cout<<x;
```

پس از اتمام کار بر روی عدد اول ، عدد دوم را می خوانیم و آن را بررسی می کنیم . این عمل را تا زمانی که به پایان فایل f نرسیده باشیم تکرار می کنیم . برای دانستن اینکه به پایان رسیده ایم یا نه ، از تابع ( )feof استفاده می کنیم . در صورتی که به پایان برسیم مقدار باز گشتی این تابع مخالف صفر می شود .

- ***نکته مهم : هنگام خواندن تک تک ا طلاعات، یک داده را خارج از حلقه ، و سایر داده های فایل را درون حلقه می خوانیم. یعنی:***

```
fread (&x,2,1,f);
while ( feof(f) == 0){
    . . . . . .
    . . . . . .
    . . . . . .
    fread (&x,2,1,f);
}
```

با توجه به توضیحات فوق برنامه را به صورت زیرکامل می کنیم :

```
#include<iostream.h>
#include<stdio.h>
void main ( )
{
  FILE  *f;
  int x;
  f = fopen("d:\\num.dat","rb");
  fread (&x,2,1,f);
  while ( feof (f) !=0){
     if (x%2== 0)  cout<<x<<"\t";
     fread (&x,2,1,f);
   }
   fclose(f);
}
```

خواندن اولین داده خارج از

خواندن سایر داده های فایل در انتهای حلقه تکرار

مثال) پس از خواندن محتویات فایل num2.dat آن ها را مرتب کرده و دوباره در فایل قرار دهید.

توضیح: در این فایل 10 عدد float ذخیره شده است ، چون می خواهیم آن ها را مرتب کنیم پس اطلاعات خوانده شده را در یک آرایه 10 عضوی قرار می دهیم .

```
float x[10];
fread (x,4,10,f);
```

ارایه x را مرتب می کنیم :

```
for (j = 1;j<=9;j++)
  for (i = 0;i<9;i++)
    if (x[i]>x[i+1]){
      t = x[i];
      x[i]=x[i+1];
      x[i+1] = t;
}
```

برای جایگزین شدن مجدد آرایه در فایل ، ابتدا فایل را بسته ، و دوباره فایل را به عنوان فایل جدید باز می کنیم و آرایه را در آن ذخیره می کنیم .

```
fclose (f);
f = fopen ("d:\\num2.dat","wb");
fwrite(x,4,10,f);
fclose (f);
```

برنامه ی کامل به صورت زیر است.

```
#include<iostream.h>
#include<stdio.h>
void main( )
{
   FILE  *f;
   float x[10];
   f = fopen ("d:\\num2.dat","rb");
   fread(x,4,10,f);
   for (int j = 1;j<=9;j++)
       for (int i = 0;i<9;i++)
           if (x[i]>x[i+1]){
             float t = x[i];
             x[i] = x[i+1];
             x [i+1] = t;
           }
   fclose (f);
   f = fopen ("d:\\num2.dat","wb");
   fwrite (x,4,10,f);
   fclose (f);
}
```

تمرین:

1-نام خانوادگی 10 دانش آموز را در فایلی به نام name.dat ذخیره کنید.

2-در یک ارایه 100 عدد تصادفی در بازه 100 تا 1000 قرار دهید . پس این آرایه را درون فایلی به نام rnd.dat واقع در ریشه ی d: ذخیره کنید.

3-نمرات 10درس(مستمر 1،میانی،مستمر 2،پایانی) یک دانش آموز را درون یک آرایه 10*4 ذخیره کنید. این نمرات را درون فایلی به نام mark.dat ذخیره کنید.

4-در فایل rnd.dat ،50 عدد تصادفی دیگر در همان بازه اضافه کنید.

5-مجموع اعداد فرد موجود در فایل rnd.dat را چاپ کنید.

6-معدل نمرات دانش آموزی که نمرات آن در فایل mark . dat قرار دارد را به دست آورید.

ساختمان structure

در زبان c++ می توان متغیر ها را به دو نوع ساده و مرکب تقسیم کرد . متغیر ساده ، تنها دارای یک عضو و متغیر مرکب دارای چند عضو است . در بحث ارایه با نمونه ای از متغیر های مرکب آشنا شدیم . در آرایه کلیه عناصر هم نام وهم نوع هستند . درصورتی که عناصر یک متغیر غیر همنوع باشند اما در یک وجه خاص مشترک باشند، از ساختمان استفاده می کنیم . مثلا مشخصات یک دانش آموز شامل نام ، نام خانوادگی ، کد شناسایی ، تاریخ تولد و .... ساختمان دانش آموز را تشکیل می دهد و به هر عنصر این ساختمان یک فیلد گفته می شود.

در حقیقت ساختمان یک نوع ساخت یافته است که متناظر با خواسته های کاربر قابل تعریف است . الگوی تعریف این نوع ساخت یافته بصورت زیر است :

```
struct  نام ساختمان  {
      نوع فیلد 1      نام فیلد 1 ;
      نوع فیلد 2      نام فیلد 2 ;
      .......
}
```

مثال : ساختمانی به منظور تعریف مشخصات یک دانش آموز معرفی کنید .

```
struct student {
   char  LastName[30];
   char  FirstName[20];
   long int id;
   char   TelNo[10];
}
```

این تعریف نوع جدیدی به نام struct student را به نوع های استاندارد c++ اضافه می کند و کاربر می تواند از این نوع نیز همانند نوع های استاندارد c++ استفاده کند . مثلا می توان متغیری از این نوع تعریف کرد :

```
struct  student  x ;
```

این تعریف متغیر x را از نوع struct student معرفی می کند . پس باید x حاوی فیلدهای موجود در ساختمان struct student باشد . برای دسترسی به این عناصر از الگوی زیر استفاده می کنیم :

```
نام فیلد.نام متغیر ;
```

مثلا می توان اعمال فرضی زیر را بر روی فیلدها انجام داد :

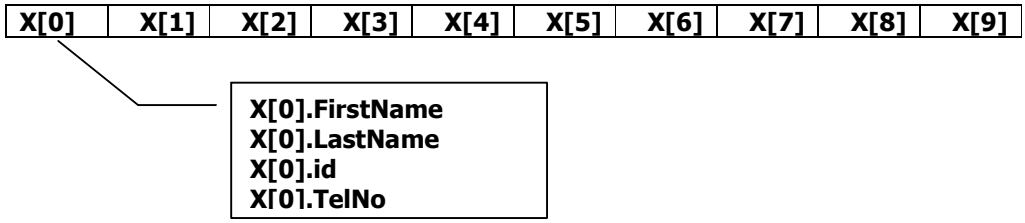
```
cin.get(x.LastName,30);
cout<<x.FirstName;
x.id=1254000;
```

آرایه ای از ساختمان

در صورتی که به تعدادی عنصر از نوع ساختمان معرفی شده نیاز داشته باشیم ، بهتر است بجای تعریف چندین متغیر از نوع ساختمان ، آرایه ای از این نوع جدید معرفی کرد :

```
struct  student  x[10] ;
```

این آرایه دارای 10 عنصر است که هر یک از آن ها از نوع struct stdent می باشند .فیلدهای این عناصر را می توان به صورت زیر تعیین کرد :



سایر عناصر ارایه نیز همانند x[0] دارای فیلدهای تعریف شده هستند.

نکته : امکان انتصاب یک متغیر از نوع ساختمان به متغیر دیگر هم نوع با آن وجود دارد :

```
struct student x,y;
x=y;
```

مثال) مشخصات 10 دانش آموز (نام خانوادگی ، کدشناسایی و معدل) را دریافت کرده سپس نام خانوادگی و معدل افرادی را چاپ کنید که معدل آن ها بالای 18 باشد .

توضیح :

ابتدا متناظر با مشخصات دانش آموزان، ساختمانی به صورت زیر تعریف می کنیم :

```
struct student {
   char  LastName[30];
   long int id;
   float  av;
} ;
```

برای نگهداری مشخصات هر دانش آموز ، باید متغیری از نوع ساختمان فوق داشته باشیم . به همین دلیل آرایه ای 10 عنصری از این نوع تعریف می کنیم :

```
struct   student   x[10] ;
```

با توجه به صورت مسئله ، ابتدا باید مشخصات دانش اموزان را دریافت کنیم . مشخصات دانش آموزان درون فیلدهای هر یک از متغیرها اندیس دار آرایه قرار می گیرد .

برای اجرای ساده تر این عمل ، ابتدا دریافت فیلدهای اولین دانش آموز را اجرا می کنیم . فیلدهای دانش آموز اول عبارتند از :

```
X[0].id      X[0].LastName      X[0].av

cout<<"enter family = ";
cin.get(x[0].LastName);
cin.get();
cout<<"enter id student=";
cin>>x[0].id;
cout<<"enter average=";
cin>>x[0].av;
```

دریافت فیلدهای دانش آموز ان دوم (x[1]) ، سوم (x[2]) و ... نیز باتکرار همین قطعه کد و تنها با تغییر اندیس ها، اجرا می شود .

```
for(i=0;i<10;i++) {
   cout<<"enter family = ";
   cin.get(x[i].LastName,30);
   cin.get();
   cout<<"enter id student=";
   cin>>x[i].id;
   cout<<"enter average=";
   cin>>x[i].av;
}
```

برای چاپ نیز معدل تک تک دانس آموزان را با نمره 18 مقایسه می کنیم ، و مشخصات افرادی را چاپ می کنیم که معدل بالای 18 داشته باشند . برای نفر اول می نویسیم :

```
if(x[0].av>18)
  cout<<x[0].LastName<<"\t"<<x[0].av<<endl;
```

تکرار عملیات توسط حلقه ی زیر خروجی کامل را ایجاد می کند :

```
for(i=0;i<10;i++)
   if(x[i].av>18)
     cout<<x[i].LastName<<"\t"<<x[i].av<<endl;
```

برنامه ی فوق را به صورت زیر کامل می کنیم :

```
#include<iostream.h>
struct student {
   char  LastName[30];
   long int id;
   float  av;
};

void main()
{
    int i;
    struct student x[10];
    for(i=0;i<10;i++)  {
       cout<<"enter family = ";
       cin.get(x[i].LastName,30);
       cin.get();
       cout<<"enter id student=";
       cin>>x[i].id;
       cout<<"enter average=";
       cin>>x[i].av;
    }
    for(i=0;i<10;i++)
    if(x[i].av>18)
       cout<<x[i].LastName<<"\t"<<x[i].av<<endl;
}
```

نکته : در cin.get ، دریافت زمانی خاتمه می یابد که enter (\n) وارد شود . لذا در دریافت رشته بعدی دچار مشکل می شویم ، چون درون بافر \n قرار دارد واین موضوع عدم دریافت های بعدی را درپی خواهد داشت . به همین دلیل دستور cin.get() بدون پارامتر را بعد از دستور دریافت رشته استفاده کرده ایم تا دریافت \n را انجام دهد .

نکته : تعریف ساختمان را می توان درون تابع انجام داد . دراین صورت تعریف محلی است و فقط درون همان تابع قابل استفاده است . معمولا این نوع تعاریف را سراسری می نویسیم چون فقط تعریف است و حافظه اشغال نمی کند .

نکته : می توان همزمان با تعریف نوع ساختمان ، متغیر یا متغیر های مورد نیاز را معرفی کرد . مثلا :

```
struct student {
   char  LastName[30];
   long int id;
   float  av;
} x[10];
```

ساختمان و فایل

در بحث فایل ها با ذخیره و بازیابی داده ها در قالب رکورد آشنا شدیم . برای نوشتن داده ها در فایل از **fwrite** و برای خواندن داده های از **fread** استفاده می کنیم .

مثال) مشخصات تعدادی مشترک تلفن (نام خانوادگی ، شماره تلفن) را در فایلی به نام **Tel.dat** واقع در ریشه **d:** ذخیره کنید .

```
#include<iostream.h>
struct phone {
   char  LastName[30];
   char  TelNo[11];
};
void main()
{
    int i;
    struct phone x;
    FILE  *f;
    f=fopen("c:\\tel.dat","wt");
    do  {
       cout<<"enter family = ";
       cin.get(x.LastName,30);
       if(x.LastName[0]==NULL)  break;
       cin.get();
       cout<<"enter Tel No. =";
       cin.get(x.TelNo,11);
       cin.get();
       fwrite(&x,sizeof(x),1,f);
    } while(1);
    fclose(f)
}
```

توضیح : پس از باز کردن فایل ، تک تک رکوردها را خوانده و درون بافری به نام **x** قرار می هیم . سپس رکوردهای درون **x** را در فایل می نویسیم . این عمل را زمانی خاتمه می دهیم که کاربر هنگام دریافت رشته ی نام خانوادگی ، رشته پوچ را وارد کند (یعنی کاربر فقط **enter** بزند) .

نکته : در این مثال حلقه **do ... while** یک حلقه بینهایت در نظر گرفته شده است . چون شرط حلقه عدد **1** است واین عدد مخالف صفر است ، ارزش **true** دارد . لذا برای خروج از حلقه ، دستور **break** را بکار می بریم .

ساختمان بعنوان پارامتر تابع

همانند داده های ساده می توان عناصری از نوع ساختمان را به تابع ارسال کرد . همچنین می توان مانند ارسال آرایه به تابع ، ارایه ای از ساختمان را به تابع ارسال کرد .

مثال) در یک ارایه مشخصات **5** مشترک تلفن را ثبت کنید . سپس با دریافت نام یا شماره تلفن یک مشترک ، فیلد دیگر را چاپ کنید .

توضیح : پس از تعریف نوع ساختمان ، تابعی برای جستجو عنصر مورد نظر در آرایه می نویسیم . تابعی که با دریافت آرایه بعنوان پارامتر ، و دریافت نام یا شماره تلفن یک مشترک ، شماره تلفن یا نام فرد مورد نظر را چاپ کند .

```
struct phone {
   char  LastName[30];
   char  TelNo[11];
};
void Search( struct phone x[5])
{
    char  s[30];
    cout<<"enter family or tel NO = ";
    cin.get(s,30);

    int found=0;
    for(int i=0;i<5;i++)
      if(strcmp(s,x[i].LastName)==0) {
         cout<<x[i].TelNo;
         found=1;
       }
      else if(strcmp(s,x[i].TelNo)==0) {
         cout<<x[i].LastName;
         found=1;
       }
    if(found==0)  cout<<"Not found";
}
```

در برنامه ی اصلی پس از تعریف آرایه و نسبت دادن مقادیر مورد نظر به آن ، آرایه را به عنوان پارمتر ، به تابع ارسال می کنیم .

```
void main()
{
   struct phone x[5] = {{"aminy","5426254"},
                        {"behrozy","6632548"},
                        {"zamany","225488"},
                        {"akbary","66325555"},
                        {"tahery","332665"} };
   Sreach(x);
}
```

تمرین

1- مشخصات 10 کتاب (نام کتاب ، نویسنده کتاب و ناشر) را درون یک آرایه ذخیره کنید . سپس لیست کتاب های یک نویسنده ی دلخواه را چاپ کنید .

راهنمایی : مقایسه ی رشته ها توسط تابع strcmp(s1,s2) واقع در فایل string.h صورت می گیرد . خروجی این تابع عددی صحیح است . اگر s1<s2 باشد خروجی عددی مثبت و اگر s1>s2 باشد خروجی منفی و در صورت برابر بودن خروجی صفر است .

2- پس از دریافت مشخصات 10 دانش آموز (نام خانوادگی ، معدل) آن ها را بر اساس معدل مرتب کرده ولیست مرتب شده را چاپ کنید.

3- مشخصات 10 مشترک تلفن (نام خانوادگی و شماره تلفن) را دریافت کنید . سپس مشخصات افرادی را چاپ کنید که شماره تلفن آن ها با 33 شروع شود .

4- پس از دریافت مشخصات 10 دانش آموز (نام خانوادگی، معدل) نموداری ستونی از معدل آن ها چاپ کنید.

5- در یک فروشگاه مشخصات کالاهای فروخته شده (کد کالا ، تعداد فروش و قیمت هر واحد) درون یک فایل ذخیره می شود . برنامه ای بنویسید که تک تک داده های ورودی را دریافت کرده و درون فایل ذخیره کند . نها یتا با وارد کردن کد 0 دریافت خاتمه یافته و میزان کل فروش را چاپ کنید .

توابع بازگشتی

در فصل های قبل ، تابع را توسط تابع main یا سایر توابع فراخوانی می کردیم .*اگر درون بدنه یک تابع، مجددا عمل فراخوانی خود تابع صورت گیرد ، به آن تابع بازگشتی (Recursive)می گوییم .*

نکته : نوشتن تابع بازگشتی ، حل بسیاری از مسائل را ساده تر می کند در حالی که نوشتن همان برنامه به صورت ساده بسیار مشکل و گاها غیر ممکن است .

مثال) خروجی برنامه مقابل چیست .

```
double f(int n)
{
    if(n==1)
        return 1;
    else
        return  n*f(n-1);
}
void main()
{
     int n;
     cout<<"n=";
     cin>>n;
     cout<<"factorial  "<< n<<"= "<<f(n);
}
```

توضیح : تابع f(n) بصورت بازگشتی نوشته شده است . چون فراخوانی تابع از درون خود تابع نیز صورت گرفته است . فرض کنید در برنامه ی اصلی عدد ورودی n=5 باشد . در خط آخر ، تابع با عبارت f(n) فراخوانی شده است پس اجرای برنامه را به تابع انتقال داده و 5 را نیز با خود می بریم . در تابع چون شرط نادرست است بخش else اجرا شده و مقدار 5*f(4) برگردانده می شود که معادل همان f(5) است .یعنی :

$$f(5) = 5 * f(4) \qquad (1)$$

همان طور که دیده می شود در این دستور ، دوباره تابع با مقادیر پارامتر n=4 صدا زده شده است . پس دوباره تابع را با این مقدار جدید تکرار می کنیم .

$$f(4) = 4 * f(3) \qquad (2)$$

با تکرار عملیات خواهیم داشت :

$$f(3) = 3 * f(2) \qquad (3)$$
$$f(2) = 2 * f(1) \qquad (4)$$
$$f(1) = 1$$

نهایتا f(1) مقدار معلوم 1 را برمی گرداند . با قراردادن مقدار f(1)=1 درون رابطه (4) ، مقدار f(2) نیز بدست می آید f(2)=2*1=2 . حاصل f(2) را درون رابطه (3) قرار می دهیم و f(3)=3*2=6 را بدست می آوریم . با تکرار همین عملیات f(4)=4*6=24 و f(5)=5*24=120 بدست می اید . نهایتا مقدار f(5) ، چاپ می شود .

نوشتن تابع بصورت بازگشتی

شاید الگوریتم توابع بازگشتی ، از توابع ساده مقداری پیچیده تر به نظر رسد . اما با تحلیل و رسیدن به یک راه حل بازگشتی ، پیاده سازی آن راحت است .

مثال) تابعی بنویسید که حاصل x^n را بصورت بازگشتی برگرداند .

ابتدا برای رسیدن به یک راه حل بازگشتی ، از دانسته های ریاضی استفاده می کنیم :

$$x^0 = 1$$
$$x^1 = x^0 * x$$
$$x^2 = x^1 * x$$
$$x^3 = x^2 * x$$
$$\ldots\ldots$$
$$x^n = x^{n-1} * x$$

همانطور که دیده می شود برای پیدا کردن هر توان از توان قبلی استفاده کرده ایم . *یعنی برای پیدا کردن توان ، دوباره از توان استفاده می کنیم که همان مفهوم بازگشتی است* . برای خارج شدن از این حالت بازگشت ، x^0 را استفاده کرده ایم .در این حالت ، بازگشت صورت نمی گیرد و اجرای تابع خاتمه می یابد. بر اساس تعاریف فوق می توان به این رابطه دو ضابطه ای زیر رسید :

$$x^n = \begin{cases} x^{n-1} * x & n>0 \\ 1 & n=0 \end{cases}$$

قرض کنید اسم تابع را **power** ، و پارامتر های آن را **x,n** در نظر گرفته باشیم یعنی $power(x,n)=x^n$. در این صورت اگر **n>0** باشد مقدار بازگشتی تابعی **power(x,n)** برابر **x*power(x,n-1)** و گرنه مقدار بازگشتی **1** خواهد بود . همین توصیف را به برنامه تبدیل می کنیم :

```
double power( int x, int n)
{
   if(n>0)
       return x*power(x,n-1);
   else
       return 1;
}
```

نکته : در این مثال چون خروجی تابع توان یک مقدار اکیداصعودی است . نوع بازگشتی را **double** انتخاب کرده ایم .

تمرین

1- تابعی بنویسید که حاصل **x*y** را بصورت بازگشتی برگرداند .
2- تابعی بنویسید که باقیمانده تقسیم **x** بر **y** را بصورت بازگشتی برگرداند .
3- تمرین قبل را برای بازگرداندن خارج قسمت حل کنید .
4- مجموع یک ارایه ی **10** عنصری را به کمک تابع بازگشتی برگردانید .
5- مجموع ارقام عدد **x** را به کمک تابع بازگشتی برگردانید .
6- جمله **n** ام سری فیبوناتچی را به کمک تابع بازگشتی برگردانید .
7- باتو جه به تعریف مقابل حاصل ترکیب **m** از **n** را برگردانید .

$$\binom{n}{m} = \binom{n-1}{m} + \binom{n-1}{m-1}$$

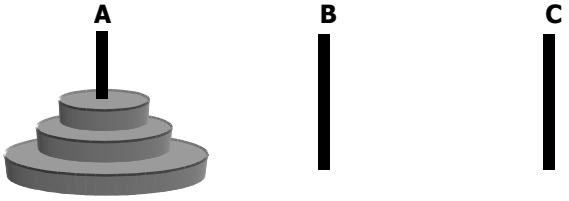
مثال) برنامه ای بنویسید که حرکت دیسک ها در برج هانوی Hanoi را مشخص کند .

توضیح :

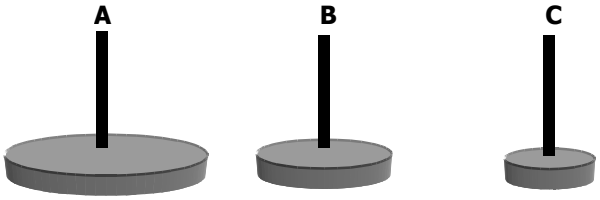
مثال برجهای هانوی ، مثال جالبی در الگوریتم های بازگشتی است . در این مسئله سه پایه وجود دارد ، که آن ها رابا A,B,C نام گذاری می کنیم . هدف نهایی انتقال n دیسک از پایه A به پایه C است ، به طوریکه اولا : هیچگاه دیسک بزرگ تر بر روی دیسک کوچک تر قرار نگیرد ثانیا : در هر بار تنها یک دیسک از پایه ای به پایه دیگر انتقال یابد .

ابتدا وضعیت حرکت دیسک ها را برای n=3 بررسی می کنیم .

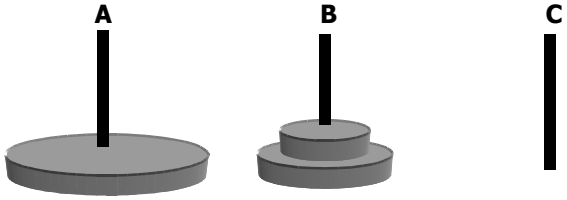
گام اول :



ابتدا مسئله را بر روی سه دیسک توضیح می دهیم . دیسک کوچکتر را بر روی پایه C ودیسک بعدی را بر روی پایه B قرار می دهیم:



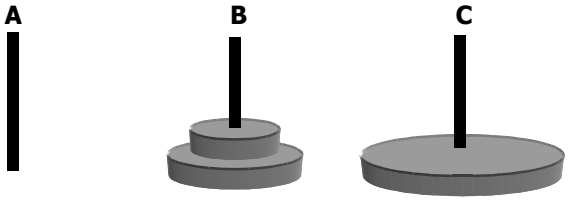
حال می توان دیسک پایه C را به پایه B منتقل کرد :



نتیجه کلی این عملیات انتقال n-1 دیسک از پایه A به پایه B به کمک پایه C است .

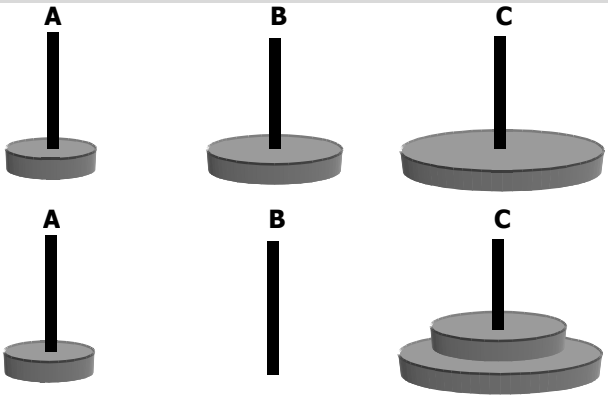
گام دوم :

آخرین دیسک پایه A را به پایه C منتقل می کنیم .

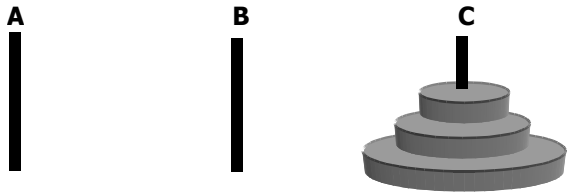


گام سوم :

n-1 دیسک را به کمک پایه A از پایه B به پایه C می بریم .



نکته :*اگر 1 دیسک در پایه A باشد به پایه C می رود .*



الگوریتم این تابع را می توان به صورت زیر خلاصه کرد :

اگر n=1 باشد

1-دیسک از پایه A به پایه C منتقل می شود.

وگرنه

1- n-1 دیسک از پایه A به کمک پایه C به پایه B منتقل می شود

2-آخرین دیسک پایه A به پایه C منتقل می شود

3-n-1 دیسک از پایه B به کمک پایه A به پایه C منتقل می شود

برنامه تابع را بصورت زیر می نویسیم :

```
void tower(int n , char source, char spare , char  target)
{
   if(n==1)
       cout<<"move disk from" << source<<"to"<<target<<endl;
   else
   {
       tower(n-1,source,target,spare);
       tower(1,source,spare,target);
       tower(n-1,spare,source,target);
    }
}
```

در برنامه اصلی پس از دریافت تعداد دیسک ها تابع را صدا می زنیم :

```
Void main()
{
   Int n;
   Cout<<"n=";
   Cin<<n;
   Tower(n,'a','b','c');
}
```

با اجرای برنامه ، حرکت دیسک ها در قالب نوشته های متنی چاپ می شوند . نمونه ای از خروجی به ازای n=3 در زیر آورده شده است . که متناظر با حرکت دیسک ها در توضیح مثال است .

```
move disk from  A  to  C
move disk from  A  to  B
move disk from  C  to  B
move disk from  A  to  C
move disk from  B  to  A
move disk from  B  to  C
move disk from  A  to  C
```

تمرین

1-برنامه ای بنویسید که ترکیبات حاصل از جابجایی عناصر یک آرایه 4 عنصری را نمایش دهد .

2-در یک آرایه مرتب شده فرضی 10 عنصری ،وجود یک عنصر را با استفاده از روش باینری با کمک یک تابع بازگشتی بنویسید .

3-دو آرایه مرتب شده فرضی در اختیار داریم که هر دو بصورت نزولی مرتب شده است . به کمک یک تابع بازگشتی این دو آرایه را در یک آرایه سوم ترکیب کنید ، بطوری که ترکیب آنها نیز مرتب باشد .

منابع :

- برنامه نویسی به زبان C++ : مولف جعفر نژاد قمی
- Help زبان برنامه نویسی ،Turbo C++