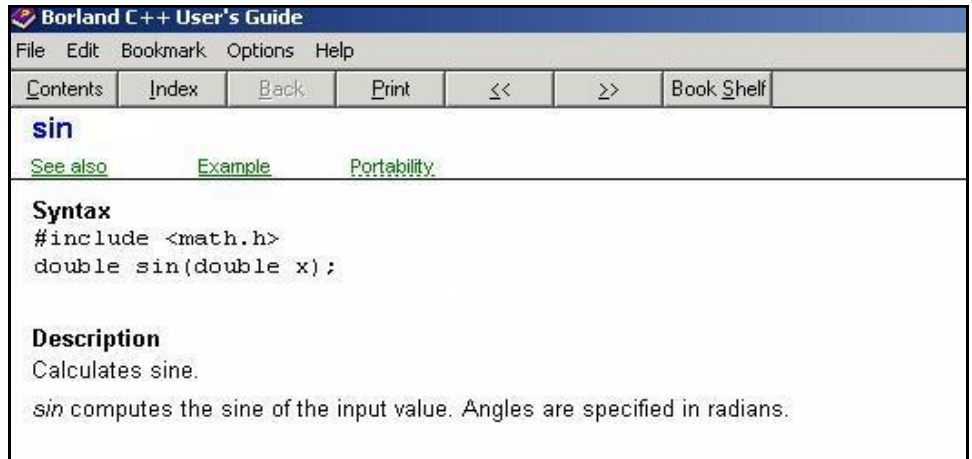
توابع کتابخانه ای

- **انواع توابع کتابخانه ای:**
  - **توابع ورودی–خروجی: printf، scanf، getche، getch و...**
    - **header file: اغلب stdio.h ، conio.h**
  - **توابع ریاضی: abs (برای محاسبه قدر مطلق) ،pow (برای محاسبه توان) ، sqrt (محاسبه رادیکال) sin، cos، tan، sinh، cosh، tanh،asin، acos، atan، exp (محاسبه e^x) ، log (لگاریتم در مبنای e) ، log10 (log در مبنای ۱۰)**
    header file: math.h
  - **توابع برای کار با فایل ها: fopen ،fwrite ،fread و ...(در بخش آشنایی با فایل ها توضیح داده خواهند شد).**
    header file: stdlib.h
  - **کارهای گرافیکی، کاربا پورت ها، نوشتن برنامه اسمبلی و.....**

توابع کتابخانه ای

نکته: در کتاب های برنامه نویسی C و همچنین help محیط های نرم افزاری در مورد ورودی و مقدار برگشتی توابع و نوع آنها و همچنین عملکرد آنها توضیح داده شده است.



در شکل help محیط borland c برای تابع sin نشان داده شده است.
ملاحظه می شود که نوع مقدار برگشتی double است. تابع یک ورودی از نوع double دارد و برای محاسبه sin به کار می رود.
header file این تابع هم math.h است.

ادامه توابع کتابخانه ای

- **مثال:**

```
#include <stdio.h>
#include <math.h>
void main ()
{
printf("%f",sin(3.141));
}
```

نکته: متغیرهای محلی و سراسری

برنامه زیر را در نظر بگیرید.

```
#include <stdio.h>
void func (int)
void main ()
{
int y;
y=5;
func (6);
printf("%d",y);
}

void func (int z)
{
    int y;
    y=z*z;
}
```

سوال: مقدار چاپ شده بر روی مانیتور چقدر است:

عدد ۵ یعنی مقدار y تعریف شده درتابع main چاپ می شود نه ۳۶
که مقدارy تعریف شده در func است.

ادامه

- **مثال ۲) برنامه زیر را در نظر بگیرید:**

```
#include <stdio.h>
void func (int )

void main ()
{func(3);
printf("%d",y);
}

void func (int z)
{    int y;
   y=z*z;
}
```

این برنامه مواجه با خطای کامپایلی زیر خواهد شد:
y: undeclared identifier
یعنی y تعریف نشده است. به عبارت دیگر main متغیری به نام y را نمی شناسد.

متغیرهای محلی و سراسری

- در زبان C دو دسته متغیر وجود دارد.
  - محلی (local) .
  - عمومی–سراسری (global).

متغیرهای local

- متغیرهای local متغیرهایی هستند که در داخل توابع تعریف شده اند.
- این متغیرها دارای ویژگی های زیر هستند.
  - تنها در داخل تابعی که تعریف شده اند معتبرند.
  - هنگام اجرای تابع ایجاد می شوند و هنگام خاتمه اجرا از بین می روند.
- همه متغیرهایی که تاکنون در مثال ها دیده ایم متغیرهای local هستند.

مثال

در تابع func ، y محلی است. در نتیجه تنها زمانی که func در حال اجراست وجود دارد. و وقتی که اجرای func تمام می شود از بین می رود.
به همین علت دستور printf("%d",y) اشتباه است چون y ای وجود ندارد که printf آن را چاپ کند.

```
#include <stdio.h>
void func (int )
void main ()
{
func(3);
printf("%d",y);
}


void func (int z)
{
    int y;
     y=z*z;
}
```

ادامه

در این مثال مقداری که چاپ می شود مقدار متغیر Y تعریف شده در main است و نه y تعریف شده در تابع func. چون y تابع func فقط در محدوده خود func شناخته شده است و برای main شناخته شده نیست.

```
#include <stdio.h>
void func (int )

void main ()
{
int y;
y=5;
func (6);
printf("%d",y);
}

void func (int z)
{
    int y;
    y=z*z;
}
```

متغیرهای سراسری (global)

تعریف: به متغیرهایی که خارج از توابع تعریف می شوند متغیرهای سراسری یا global می گویند.

```
#include <stdio.h>
int  func1  (int)
int  func2  (int)

int num=4;
void main ( )
{
   printf("%d", func1(3));
   printf("\n%d", func2(3));
}

int  func1  (int x)
{
    num+=x;
    return num;
}
int  func2  (int x)
{
    num-=x;
     return num;
}
```

دو ویژگی متغیرهای global:

- برخلاف متغیرهای local، از نقطه ای که تعریف می شوند تا انتهای برنامه معتبرند.
- اگر هنگام تعریف مقداری به آنها داده نشود مقدارشان صفر در نظر گرفته می شوند.

در برنامه روبرو، متغیر num global است. چون در خارج توابع تعریف شده. این متغیر در func1، func2 و main اعتبار دارد.

عملکرد برنامه:

- ابتدا num=4 می شود.
- func1(3) اجرا می شود. مقدار num برابر 7 شده و چاپ می شود.
- سپس func2(3) اجرا می شود. مقدار num برابر 4 شده و چاپ می شود.
- اگر به جای int num=4، تنها int num نوشته می شد، مقدار num صفر می شد.

تابع بازگشتی (Recursive function)

- تابعی است که خودش را احضار کند.
- در همه مثال هایی که تاکنون داشتیم احضار یک تابع توسط تابعی دیگر صورت می گیرد. در زبان C این امکان وجود دارد که یک تابع توسط خودش احضار شود(مثال بعد).

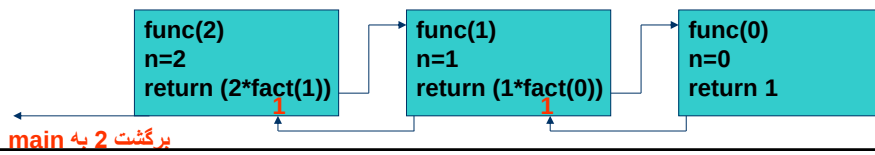
مثال ۱

تابع func، یک تابع بازگشتی است.
چون خودش را احضار کرده است.

```
#include <stdio.h>
long fact(int )

void main()
{
     printf("%d",fact(2));
}

long fact(int n)
{
  if (n==0)
      return 1;
  else
       return (n*fact(n-1));
}
```

توضیح عملکرد برنامه

```
#include <stdio.h>
long fact(int )

void main()
{
     printf("%d",fact(2));
}

{
  if (n==0)
      return 1;
  else
       return (n*fact(n-1));
}
```



نکته

- **نکته :ایده ای که ما را به برنامه بازگشتی رساند فرمول زیر بود:**

$$n! = n*(n-1)!$$

یا:

fact(n)=n*fact(n-1)

فرمول بالا به روشنی مفهوم بازگشت را درخود دارد. (برای محاسبه fact **به ازای** n **باید همان تابع** fact **به ازای** n-1 **محاسبه شود).**

نکته

- **در هر تابع بازگشتی دو رکن اصلی زیر وجود دارد:**
  - **قانون (فرمول) بازگشت: چگونگی بازگشت را مشخص می کند. در مثال قبل قانون بازگشت** n!=n*(n-1)! **بود.**
  - **شرط خاتمه: مشخص می کند بازگشت چه زمانی متوقف می شود. در مثال قبل** n=0 **شرط خاتمه بود.**

$$fact(n) = \begin{cases} n * fact(n-1) & n > 0 \\ 1 & n = 0 \end{cases}$$

- **برای بتوانیم یک برنامه بازگشتی بنویسیم باید ابتدا این دو رکن را به دست آوریم. بعد از انجام این کار نوشتن برنامه های بازگشتی، به سادگی امکان پذیر است.**

مثال ۲

- **برنامه ای بنویسید که x^n را به صورت بازگشتی محاسبه کند.**
n عدد صحیح غیر منفی و x اعشاری است.

فرض کنید می خواهیم تابعی به نام pow بدین منظور بنویسیم :
ابتدا دو رکن بازگشت را مشخص می کنیم:

- **قانون بازگشت:** $x^n = x * x^{n-1}$
- **شرط خاتمه :** n=0

$$pow(x,n) = \begin{cases} x * pow(x,n-1) & n > 0 \\ 1 & n = 0 \end{cases}$$

برنامه مثال ۲

```
#include <stdio.h>
float pow(float , int )

void main()
{
    printf("%f",pow(2.5,2));
}

float pow(float x, int n)
{
    if (n==0)
        return 1;
    else
        return (x*pow(x,n-1));
}
```

مثال ۳

- **برنامه بازگشتی بنویسید که حاصل عبارت زیر را محاسبه کند:**

$\sqrt{a+\sqrt{a+\sqrt{a+.....}}}$ **(تعداد** n **تا** a **داریم)**

فرض کنید می خواهیم تابعی بازگشتی به نام rad(a,n) **بنویسیم که این کاررا انجام دهد.**

- **قانون بازگشت و شرط خاتمه :**

$$rad(a,n)=\begin{cases}\sqrt{a+rad(a,n-1)} & n>1\\ \sqrt{a} & n=1\end{cases}$$

برنامه مثال ۳

نکته: تابع sqrt از توابع کتابخانه ای است و header file آن math.h است. این تابع برای محاسبه رادیکال استفاده می شود.

```
#include <stdio.h>
#include <math.h>
float rad(float a , int n)
{
    if (n==1)
        return sqrt(a);
    else
        return sqrt(a+rad(a,n-1));
}

void main()
{
    printf("%f",rad(2,3));
}
```

مزایا و معایب برنامه های بازگشتی

- **مزایا:**
  - **خواناتر بودن**
  - **اگر قانون بازگشت به دست آورده شود نوشتن یک برنامه به صورت بازگشتی بسیار ساده تر از معادل غیر بازگشتی آن است.مثلا معادل غیربازگشتی برنامه برج های هانوی برنامه ای طولانی می باشد در حالی که برنامه بازگشتی آن با دو–سه خط نوشته می شود.**
- **معایب:**
  - **مصرف حافظه بیشتری دارند و سرعت اجرای آنها هم معمولا کمتر است. (بررسی به عنوان تحقیق)**