جزوه‌ی آموزشیِ برنامه‌نویسی برای آمادگی المپیاد کامپیوتر

تهیه و تنظیم : عرفان عبدی

ویرایش : جواد عابدی

irprogramming.ir

مقدمه

در این جزوه‌ی آموزشی تلاش شده به صورت مختصر دانش‌پژوهان با زبان برنامه نویسی c++ آشنا شوند. مرحله‌ی سوم المپیاد کامپیوتر به صورت برنامه‌نویسی برگزار می‌شود. این آزمون از تعدادی مساله تشکیل شده است که پاسخ نهایی هرکدام یک عدد خواهد بود. حل هر مساله معمولا نیازمند محاسبات پیچیده‌ای است که به کمک برنامه‌نویسی می‌توان پاسخ را بدست آورد.

به عنوان مثال فرض کنید پاسخ مساله برابر با مجموع اعداد اول بین یک تا یک میلیون باشد. بدیهتا شما نمی‌توانید روی کاغذ این عدد را محاسبه کنید، اما با نوشتن چند خط کد ساده می‌توان این مقدار را بدست آورد.

هنگامیکه کد مساله‌ای را نوشتید می‌توان توسط ابزارهایی که به آنها compiler گفته می‌شود کد را به یک برنامه‌ی اجرایی تبدیل و خروجیِ آن را مشاهده کرد. اگر در کدِ نوشته شده تمام قواعد برنامه نویسی رعایت شده باشد compiler با موفقیت آن را به فایل اجرایی تبدیل می‌کند ولی در غیر اینصورت خطاهای کد را نمایش می‌دهد و شما می‌توانید پس از رفع مشکلات دوباره کد خود را compile کنید.

شما می‌توانید کد خود را در هر جایی روی کامپیوتر حتی در notepad بنویسید ولی ابزارهای استانداردی برای برنامه‌نویسی وجود دارد که علاوه بر ویرایش، امکانات اضافه‌ای در اختیار شما قرار می‌دهد.

نرم افزارهای دیگری نیز هستند که این دو قابلیت را همزمان برای شما فراهم می‌کنند، یعنی هم می‌توانید در آن به آسانی کد بزنید و هم کد را در همانجا compile کنید از جمله نرم افزارهای کم‌حجم و ساده برای برنامه‌نویسی برنامه‌ی devcpp می‌باشد که می‌توانید آنرا از آدرس زیر دریافت کنید:

http://www.bloodshed.net/devcpp.html

شما در هر محیطی کد بزنید فرقی در اجرای برنامه‌ی نهایی ندارد ولی برای آنکه از یک رویه پیروی کنیم در این جزوه فرض می‌کنیم در devcpp کد می‌زنیم.

آشنایی با c++

برای آغاز، نرم افزار devcpp را باز کرده و File → New → Source File را انتخاب کنید.

حال در صفحه‌ی روبرو قطعه کد زیر را بنویسید:

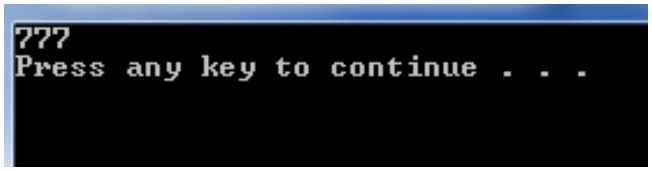
```
#include<iostream>
using namespace std;

int main(){
    cout<<777<<endl;

    system("pause");
}
```

سپس از منوی Execute گزینه ی Compile & Run را انتخاب کنید یا اینکه از کلید F9 را فشار دهید.

سپس شما باید صفحه‌ی زیر را ببینید:



در خط اول خروجی عدد ۷۷۷ چاپ شده است. به جای ۷۷۷ عدد دیگری را بنویسید و نتیجه را مشاهده کنید. دستور cout برای نمایش مقادیر در خروجی است.

خطا در کامپایل

فرض کنید کدی که نوشته‌اید خطای دستوری داشته باشد در این صورت هنگامی که آنرا کامپایل می‌کنید اجرا نخواهد شد. برای مثال در کد قبلی از انتهای خط چهارم یعنی:

```
cout<<777<<endl;
```

; را بردارید و دوباره برنامه را اجرا کنید، خطای زیر را مشاهده خواهید کرد:

```
int main(){
    cout<<777<<endl

    system("pause");
}
```

Compiler | Resources | Compile Log | Debug | Find Results | Close

Line	File	Message
	C:\Users\ee\Documents\Untitled1.cpp	In function `int main()':
7	C:\Users\ee\Documents\Untitled1.cpp	expected `;' before "system"

همانطور که می‌بینید در پایین صفحه خطاهای دستوری کد نوشته شده است.

متغیرها

در خیلی از موارد نیاز به ذخیره مقادیر داریم تا در ادامه از آنها استفاده کنیم، به این منظور می‌توانید از متغیرها استفاده کنید.

در زبان c++ متغیرها انواع مختلفی می‌توانند داشته باشند. برای اینکه متغیری از جنس اعداد صحیح تعریف کنیم به این صورت عمل می‌کنیم:

```
int    a;
```

حال ما یک متغیر از جنس اعداد صحیح به نام a داریم و می‌توانیم از آن استفاده کنیم. به مثال زیر توجه کنید:

```
#include<iostream>
using namespace std;

int main(){
    int a = 5;
    int b;
    b=9;

    cout<<a<<endl;
    cout<<a+b<<endl;
    a=17;
    cout<<a<<endl;

    system("pause");
}
```

در این مثال ابتدا متغیر a با مقدار ۵ و سپس متغیر b با مقدار ۹ تعریف شده است.

در ادامه متغیر a را در خروجی چاپ کرده‌ایم.

پس انتظار داریم در خط اول خروجی

مقدار ۵ چاپ شود و سپس مقدار a+b یعنی ۱۴ را

در خروجی ببینیم.

بعد از آن مقدار متغیر a را به ۱۷ تغییر داده‌ایم و دوباره a را چاپ کرده‌ایم

که در خروجی مقدار ۱۷ را مشاهده خواهیم کرد:

خروجی:

```
5
14
17
Press any key to continue . . .
```

ورودی خواندن

همانطور که در مثال‌های بالا دیدید متغیرهایی تعریف کردیم و به آنها مقدار دادیم.

ما می‌توانیم با استفاده از دستور cin مقدار متغیرهای خود را از ورودی بخوانیم (به همان سادگی که مقدار متغیر را در خروجی چاپ می‌کردیم).

به مثال زیر توجه کنید:

```
#include<iostream>
using namespace std;

int main(){
 int a,b;
 cin>>a;
 cin>>b;

 cout<<a+b<<endl;

 system("pause");

}
```

همانطور که می‌بینید در این مثال

ابتدا دو متغیر a و b را تعریف کرده‌ایم.

سپس در خط بعد از ورودی مقدار a را خوانده‌ایم و در خط بعد آن مقدار b را خوانده‌ایم

و در نهایت مجموع آنها را در خروجی چاپ کرده‌ایم.

خروج به شکل زیر خواهد بود:

```
10
23
33
Press any key to continue . . .
```

همانطور که می‌بینید در زمان اجرای برنامه خط اول به a مقدار ۱۰ و به b مقدار 23 را داده‌ایم.

دستورات شرطی:

اگر بخواهیم دستوراتی در شرایط خاصی اجرا شوند می‌توانیم از if استفاده کنیم، ساختار if به شکل روبرو است:

If (شرط مورد نظر){

دستوراتی که باید انجام شود

}

برای مثال فرض کنید می‌خواهیم عددی را از ورودی بخوانیم، در صورتی که کوچکتر و یا مساوی ۲۰ بود آن عدد را چاپ کنیم و در غیر این صورت (یعنی اگر بزرگتر از ۲۰ بود) عدد ۵۰ را چاپ کنیم:

```
#include<iostream>
using namespace std;

int main(){
    int n;
    cin>>n;
    if(n<=20){
            cout<<n<<endl;
    }
    if(n>20){
            cout<<50<<endl;
    }
}
```

توجه کنید برای نوشتن شرطِ برابریِ دو عبارت باید از == استفاده کنیم. به مثال زیر توجه کنید:

```
if(n==2){
    cout<<1<<endl;
}
```

دستور بالا به این معنی است که اگر n برابر ۲ بود شرط برقرار است.

همچنین برای نوشتن شرطِ نابرابریِ دو عبارت باید از عملگر=! استفاده کنیم.

حلقه‌ها:

اگر بخواهیم دستوراتی را چند بار به صورت مکرر انجام دهیم از حلقه‌ها استفاده می‌کنیم. ساختار کلی یک حلقه به شکل زیر است:

for(تعریف متغیر و مقداردهی اولیه; شرط خاتمه‌ی حلقه; اتفاقی که پس از اجرای دستورات داخل رخ می‌دهد){

اعمالی که باید اتفاق بیفتند

}

مثلا فرض کنید می‌خواهیم ۱۰۰ بار عدد ۷ را چاپ کنیم. برنامه‌ی زیر را برای آن می‌نویسیم:

```
#includ[1]ream>[2]
using n[illegible]ce st[illegible]

int main()[illegible]
 for([3]int i = 0; i < 100; i = i+1){
        cout<<7<<endl;
 }

}
```

در برنامه‌ی روبرو ابتدا یک متغیر به اسم i تعریف کرده‌ایم و مقدار اولیه آن را برابر صفر قرار داده‌ایم. سپس شرط حلقه را چک می‌کنیم یعنی بررسی می‌کنیم مقدار i کمتر از ۱۰۰ است یا خیر اگر این شرط برقرار بود داخل بدنه‌ی حلقه می‌شویم و عدد ۷ را چاپ می‌کنیم، سپس به قسمت سوم حلقه بازمی‌گردیم و دستوری که آنجا هست را اجرا می‌کنیم یعنی i را برابر i+1 قرار می‌دهیم در واقع یک واحد به آن اضافه کرده و شرط حلقه یا همان قسمت ۲ آنرا چک می‌کنیم اگر برقرار بود داخل حلقه می‌شویم این فرایند را آن قدر انجام می‌دهیم تا شرط قسمت دوم حلقه دیگر برقرار نباشد.

چند مثال:

حال که با مباحث اولیه آشنا شدید چند مثال حل می‌کنیم:

مثال اول: می‌خواهیم برنامه‌ای بنویسیم که عدد n را از ورودی بخواند و اعداد 1 تا n را در خروجی چاپ کند.

```
#include<iostream>
using namespace std;

int main(){
    int n;
    cin>>n;
    for(int i =1 ;i<=n;i=i+1){
            cout<<i<<endl;
    }
    system("pause");
}
```

```
7
1
2
3
4
5
6
7
Press any key to continue . . .
```

مثال دوم: می‌خواهیم ابتدا یک عدد از ورودی بخوانیم و به تعداد آن، عدد طبیعی از ورودی دریافت کرده و از بین آنها عدد بیشینه را چاپ کنیم. مثلا فرض کنید در ابتدا عدد ۵ و سپس ۵ عدد وارد کنیم. بعد از آن اعداد ۴، ۱، ۱۰، ۷ و ۳ را وارد کرده باشیم. در نتیجه خروجی باید ۱۰ باشد:

```
#include<iostream>
using namespace std;

int main(){
    int n;
    cin>>n;
    int max=0;
    int input;
    for(int i = 0; i<n; i= i+1){
            cin>>input;
            if(input>max){
                    max=input;
            }
    }
    cout<<max<<endl;
    system("pause");
}
```

همانطور که می‌بینید ابتدا متغیر n را از ورودی خوانده‌ایم سپس دو متغیر

تعریف کرده‌ایم، یکی برای اینکه ماکسیمم را در آن قرار دهیم و متغیر ورودی است.

سپس در یک حلقه که n دفعه اجرا می‌شود هر بار یک ورودی

دریافت می‌کنیم اگر از ماکسیممی که تا به الان خوانده‌ایم بیشتر

بود ماکسیمم را برابر آن قرار می‌دهیم.

و بعد از خاتمه‌ی حلقه مقدار ماکسیمم را چاپ می‌کنیم.

```
5

4
1
10
7
3
10
Press any key to continue . . .
```

مثال سوم: می‌خواهیم عددی را از ورودی بخوانیم اگر آن عدد اول بود در خروجی یک چاپ کنیم و اگر اول نبود صفر چاپ کنیم.

```
#include<iostream>
using namespace std;

int main(){
    int n;
    cin>>n;

    int cnt=0;

    for(int i=2;i<n;i=i+1){
            if(n%i==0){
                        cnt=cnt+1;
            }
    }
    if(cnt==0){
                cout<<1<<endl;
    }
    if(cnt!=0){
                cout<<0<<endl;
    }
    system("pause");
}
```

ابتدا عدد را می‌خوانیم سپس به ازای تمام

اعداد بین ۲ تا n-1 چک می‌کنیم اگر n

حداقل بر یکی از آنها بخشپذیر باشد یعنی

اول نیست ولی اگر بر هیچکدام بخشپذیر

نبود یعنی اول است. لازم به ذکر

است برای اینکه باقیمانده‌ی عددی را

بر عدد دیگر محاسبه کنیم می‌توانیم

از عملگر ٪ استفاده کنیم.

ابتدا عدد n را از ورودی می‌خوانیم

حال یک متغیر به اسم cnt تعریف می‌کنیم.

که تعداد مقسوم‌علیه‌های n در بین اعداد ۲ تا n-1 را در آن ذخیره کنیم.

سپس یک حلقه از ۲ تا n-1 اجرا می‌کنیم و...

چند متغیر دیگر

تا به اینجا با متغیر از نوع int که برای ذخیره‌سازی اعداد صحیح است آشنا شدیم. لازم به ذکر است اعداد از نوعِ int یک محدوده‌ی خاص دارند و اعداد بیرون این محدوده نمی‌توانند باشند. در جدول زیر چند متغیر جدید به همراه محدوده و کاربردشان را نوشته‌ایم:

نام متغیر	کاربرد	محدوده
Int	اعداد صحیح	-2^{31} تا $2^{31}-1$
long long	اعداد صحیح بزرگ	-2^{63} تا $2^{63}-1$
Float	اعداد اعشاری	3.4E +/- 38
Double	اعداد اعشاری بزرگ	1.7E +/- 30

توابع

تا اینجا هر کدی زده‌ایم شامل کلماتی ثابت در خط اول و دوم بوده‌اند:

```
#include<iostream>
using namespace std;
```

اینکه include چه کاری می‌کند را در ادامه خواهیم دید، اما پس از این دو خط دستورات خود را همیشه در داخل

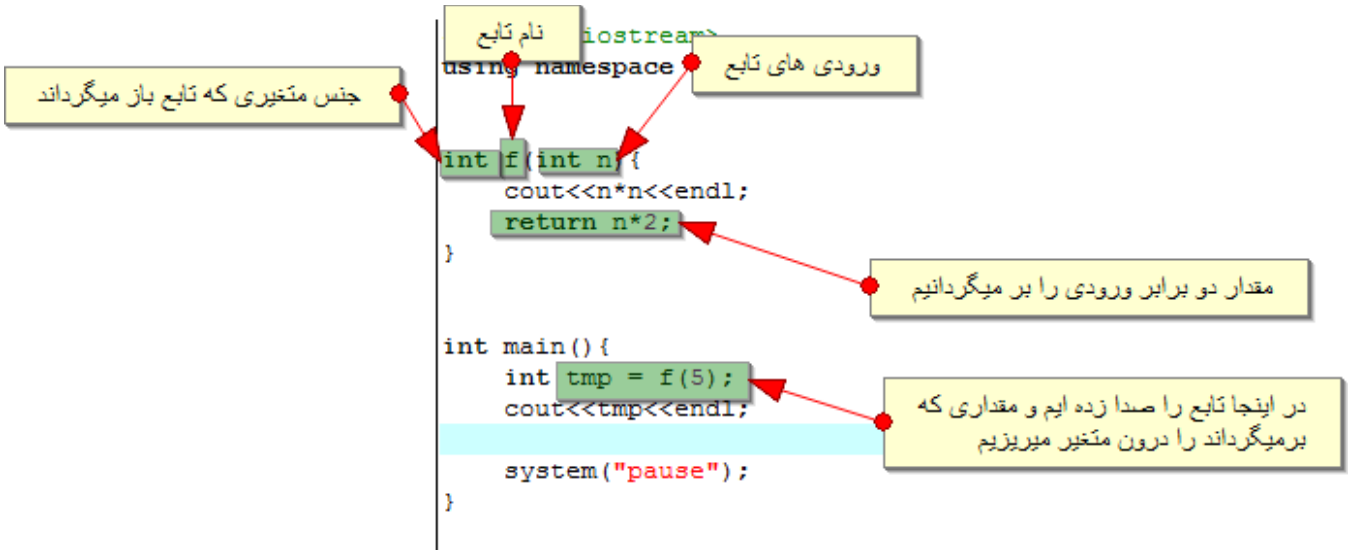
```
int main(){

        دستورات مورد نظر

}
```

تابعی به نام main می‌نوشتیم. در واقع هنگامی که برنامه‌ی ما شروع به اجرا می‌کند از اولین خط درون main شروع به عمل کردن می‌کند تا به انتهای آن برسد.

حال می‌خواهیم در برنامه توابع دیگری تعریف کنیم. با یک مثال شروع می‌کنیم:



برای هر تابعی که تعریف می‌کنیم ابتدا باید جنس متغیری که به عنوان خروجی برمی‌گرداند را مشخص کنیم. در مثال بالا خروجی از نوع int است. می‌توانیم از هر نوع دیگری از متغیرها نیز استفاده کنیم و درصورتیکه بخواهیم خروجی نداشته باشیم، باید به جای آنها void بنویسیم. سپس نام تابع را می‌نویسیم که در مثال f نوشته‌ایم. سپس اگر بخواهیم ورودی داشته باشیم به تعداد دلخواه ورودی قرار می‌دهیم (ابتدا نوع متغیر و سپس نام آن). در نهایت داخل {} دستورات مورد نظر را می‌نویسیم.

در برنامه هر جا که بخواهیم می‌توانیم تابع را صدا بزنیم و تمام دستوراتی که درون آن است اجرا می‌شود. در این مثال در خط اولِ تابع main، متغیر tmp را تعریف کرده‌ایم و آنرا برابر خروجیِ تابع f با ورودی ۵ قرار داده‌ایم. یعنی ابتدا تابع f با ورودی ۵ صدا زده می‌شود، پس هنگامی که تابع می‌خواهد اجرا شود مقدار ورودی خود یعنی همان n را برابر ۵

قرار می‌دهد و آن تابع را اجرا می‌کند. پس ابتدا ۵*۵ یا ۲۵ را چاپ می‌کند و پس از آن مقدار ۵*۲ را برمی‌گرداند که مقدار متغیر tmp را برابر ۱۰ قرار می‌دهد و در نهایت ۱۰ را در main چاپ می‌کنیم.

حال برای اینکه بیشتر با تابع آشنا شویم چند مثال می‌زنیم.

مثال اول: می‌خواهیم یک تابع بنویسیم که به آن دو عدد بدهیم و آن تابع جمع آنها را حساب کند.

ما در این برنامه x را برابر حاصل add(5,10) قرار می‌دهیم تا پس از اجرای آن مقدار x برابر ۱۵ شود.

سپس همینکار را به ازای a=20 و

b=13 انجام می‌دهیم.

```
#include<iostream>
using namespace std;

int add(int a,int b){
   int c = a+b;
   return c;
}

int main(){
    int x;
    x=add(5,10);
    cout<<x<<endl;
    x=add(20,13);
    cout<<x<<endl;

    system("pause");
}
```

مثال دوم: برنامه‌ای بنویسید که عدد n را از ورودی بگیرد و یک مثلث با ستاره‌ها چاپ کند که طول ضلع آن n باشد.

برای مثال اگر n برابر ۳ باشد باید شکل زیر چاپ شود:

*

**

***

برای اینکه این سوال را حل کنیم و همچنین از تابع استفاده کرده باشیم یک تابع می‌نویسیم که یک ورودی بگیرد و به تعداد آن ورودی ستاره چاپ کند سپس در main این تابع را با ورودی‌های مختلف در یک حلقه n بار صدا می‌زنیم.

```
#include<iostream>
using namespace std;


void f(int n){
    for(int i = 0 ; i < n ; i++){
            cout<<"*";
    }
    cout<<endl;
}


int main(){
    int x;
    cin>>x;
    for(int i=1;i<=x;i++){
            f(i);
    }

    system("pause");
}
```

مثال سوم: فرض کنید ابتدا از ورودی عدد n را می‌خوانیم و سپس n زوج عدد دریافت کند. می‌خواهیم به ازای هر زوج عدد داده شده ماکسیمم آن دو را چاپ کنیم.

برای حل این سوال یک تابع می‌نویسیم که دو ورودی بگیرد و ماکسیمم آنها را بازگرداند سپس آنرا چاپ می‌کنیم.

ابتدا یک تابع تعریف کرده‌ایم که به

آن دو عدد می‌دهیم و در آن شرطی

گذاشته‌ایم که اگر a بزرگتر از b

بود a را برگرداند. هنگامی که در

تابع به return می‌رسیم تابع همان

جا تمام می‌شود و خروجی‌اش را

همانجا که آنرا صدا زده‌ایم

قرار می‌دهد.

پس اگر داخل شرط a>b نشویم یعنی

```
#include<iostream>
using namespace std;
int max(int a,int b){
    if(a>b){
            return a;
    }
    return b;
}

int main(){
    int n;
    cin>>n;
    int c,d;
    for(int i = 0 ; i < n ; i++){
          cin>>c>>d;
          cout<<max(c,d)<<endl;
    }

    system("pause");
}
```

b بزرگتر مساوی a بوده است

پس b را برمی‌گردانیم.

آرایه ها

در بسیاری از موارد میخواهیم تعداد زیادی داده نگه داریم ، به نظر منطقی نمی آید که ۱۰۰ متغیر عدد صحیح به شکل زیر تعریف کنیم:

int a0,a1,a2,a3,a4, ….,a99

زبان سی برای این منظور آرایه را معرفی میکند ، دستور نوشتن آرایه به شکل زیر است

int a[100];

در واقع این دستور ۱۰۰ متغیر با نام های a[0] ,a[1],a[2] ,…a[99] تعریف میکند و ما برای مثال برای دسترسی به آنها میتوانیم به فرمت زیر استفاده کنیم:

a[0] = 54;

a[10] = 59*4 *a[0]

مثال :برنامه ای بنویسید که ۱۰ عدد را از ورودی بخواند و آنها را به ترتیب عکس درخروجی چاپ کند.

برای حل این سوال یک آرایه تعریف میکنیم و ۱۰ ورودی خود را به ترتیب در آن میریزیم سپس آنهارا به ترتیب معکوس چاپ میکنیم

```
#include<iostream>
using namespace std;

int main(){
    int arr[20];

    for(int i=0;i<10;i++){
            cin>>arr[i];
    }
    for(int i=9;i>=0;i--){
            cout<<arr[i]<<" ";
    }
    cout<<endl;
    system("pause");
}
```

مثال:میخواهیم ابتدا از ورودی یک عدد را بخوانیم و در متغیر n بریزیم سپس n عدد از ورودی دریافت کرده و در نهایت آنهارا به صورت مرتب شده در خروجی چاپ کنیم مثلا اگر ورودی ها به صورت زیر باشند

۵

۴ ۴ ۲ ۱۰ ۱

خروجی ما میشود ۱ ۲ ۴ ۴ ۱۰

برای اینکار ما یک آرایه تعریف میکنیم و اعداد را درون آن میریزیم

از عدد دوم شروع میکنیم اگر از عدد قبل خود کوچکتر بود آن دورا باهم جا به جا میکنیم سپس این کار را برای عدد سوم انجام میدهیم تا به عدد آخر برسیم پس از این کار عدد بیشینه در جایگاه آخر قرار گرفته است اگر این کار را n بار انجام دهیم کل اعداد به صورت مرتب شده قرار خواهند گرفت.

```
#include<iostream>
using namespace std;

int main(){
    int arr[1000];
    int n;
    cin>>n;
    for(int i = 0 ; i < n ; i++)
            cin>>arr[i];

    for(int i = 0 ; i < n ; i++){
            for(int j = 1 ; j < n ;j++){
                    if(arr[j]<arr[j-1]){
                            int tmp = arr[j];
                            arr[j]=arr[j-1];
                            arr[j-1]=tmp;
                    }
            }
    }
    for(int i = 0 ; i < n ; i++){
            cout<<arr[i]<<" ";
    }
    cout<<endl;

    system("pause");
}
```

```
5
1 10 2 4 4
1 2 4 4 10
Press any key to continue . . .
```

توابع بازگشتی

همانطور که قبل این با توابع آشنا شدیم میتوان آنهارا تعریف کرد و هر جاخواستیم از آنها استفاده کنیم ، حال میخواهیم یک تابع را دورن خودش صدا بزنیم ، به این روش برنامه نویسی که تابعی را درون خودش صدا بزنیم تابع بازگشتی میگویند، برای مثال میخواهیم تابع فیبو ناچی را به صورت بازگشتی بنویسیم:

```
#include<iostream>
using namespace std;

int f(int n){
    if(n<=1)
            return 1;
    return f(n-1)+f(n-2);
}


int main(){
    int n;
    cin>>n;
    cout<<"f(n) => "<<f(n)<<endl;

    system("pause");
}
```

همانطور که میبینید ما یک تابع به نام f داریم که از آن انتظار داریم مقدار عدد فیبوناچی n ام را بدهد هنگامی که به آن ورودی n را میدهیم ، برای اینکه آنرا درست بنویسیم مانند استقرا عمل میکنیم فرض میکنیم توابع با n های کوچکتر درست کار میکنند تابع را با استفاده از n های کوچکتر مینویسیم همچنین برای اینکه تابع یک جا خاتمه یابد مانند استقرا باید یک پایه داشته باشیم که آن هم شرطی است که در اول تابع قرار داده ایم.

```
5
f(n) => 8
Press any key to continue . . .
```

معرفی توابع پیش نوشته شده ی کاربردی

تعدادی تابع وجود دارد که قبلا نوشته شده اند و شما میتوانید از آنها استفاده کنید چند تابع کاربردی را در زیر تعریف میکنیم همچنین برای اینکه از این توابع استفاده کنیم باید کتابخانه ای که در نوشته شده است را به برنامه اضافی کنیم که این کار با نوشتن دستور

#include<نام کتاب خانه ی مورد نظر>

انجام میشود

اگر توجه کرده باشید در ابتدای تمام برنامه ها ما کتابخانه ی iostream را اضافی کرده ایم به برنامه ها که این کتاب خانه برای استفاده از دستورات cin و cout میباشد .

در زیر دستورات کاربردی کتابخانه ی cmath را آورده ایم پس برای استفاده از آنها در ابتدای برنامه باید

#include<cmath> را بزنیم

نام تابع	کاربرد
cos(a)	مقدار کسینوس a را بر میگرداند
sin(a)	مقدار سینوس a را برمیگرداند
tan(a)	مقدار تانژانت a را برمیگرداند
log10(a)	مقدار لگاریتم عدد a در پایه ده را میدهد
pow(a,b)	مقدار a به توان b را برمیگرداند
sqrt(a)	مقدار جذرaرا برمیگرداند
ceil(a)	سقف عددa را برمیگرداند
floor(a)	کف عدد aرا بر میگرداند
fabs(a)	قدر مطلق a را برمیگرداند

حال دستوارت کتابخانه ی algorithmرا مورد بررسی قرار میدهیم قبل از اینکه این دستورات را مورد بررسی قرار دهیم فرض کنید آرایه ی زیر را تعریف کرده ایم

```
int arr[1000];
```

نام تابع	کاربرد
swap(a,b)	این تابع دو متغیر را میگیرد و مقدار آنها را باهم عوض میکند
fill(arr,arr+n,c)	این تابع مقدار arr[0] تا arr[n-1] را برابر c قرار میدهد
sort(arr,arr+n)	این تابع مقادیر داخل خانه های ۰ تا n-1 آرایه arr را مرتب میکند
min(a,b)	این تابع مقدار مینیموم دو عدد a و b را برمیگرداند
max(a,b)	این تابع مقدار ماکسیموم دو عدد a و b را برمیگرداند

در صورت داشتن هرگونه سوال میتوانید سوالات خود

را در داخل سایت irprogramming.ir بپرسید