ویرایش دوم

C++

آموزش برنامه نویسی بدون ترس

ایده پرداز

- ☑ آموزش در کوتاهترین زمان
- ☑ آموزش کاربردی برنامه نویسی

مولف:زهرا بیات

فهرست مطالب

تقدیم به:

استاد مرحومم، جناب آقای

مهندس مسعود شکری پور

او که فکر و ذهنم را با ایده‌های جدیدش پرورش‌داد و راه رسیدن به تعالی را به من آموخت. امیدوارم این کتاب روحش را شاد کند. از شما عزیزانی‌که این‌کتاب را می‌خوانید خواهش می‌کنم برای شادی روحش یک فاتحه بفرستید.

با تشکر از:

همسر مهربانم‌که مرا برای نوشتن این کتاب یاری داد و بعد از خدای بزرگ و مهربان دومین حامی‌من بود. هر چند که تشکر از او کمترین کار است.

مقدمه

با توجه به استقبال بی‌نظیر شما دوستداران برنامه‌نویسی از کتاب سی پلاس پلاس بدون ترس، این کتاب در ویرایشی جدید مطابق با نظرات و پیشنهادات دوستان تغییر یافت.

باز هم ممنونم که این کتاب را خواندید و نظرات و پیشنهادات سازنده‌تان را با من درمیان گذاشتید.

یکی از مهمترین مشکلات و سوالات شما عزیزان این بود که بعد از خواندن کتاب حالا چطور شروع به‌کار کنیم؟

من در این کتاب مسائل را ساده‌تر و روان‌تر مورد بحث قرار دادم و سعی کردم، خودم را به جای خواننده کتاب قرار دهم. اما به این نکته توجه کنید که برای راه رفتن، زمین خوردن ضروریست. اگر زمین نخورید هیچ وقت راه رفتن را یاد نمی‌گیرید. ترس زمین خوردن را از ذهن خود بیرون کنید و شجاعانه بلند شوید و حرکت کنید. از همین امروز با سوالات کوچک در ذهن خود برنامه بنویسید. کم‌کم این سوالات را وسیع کنید.

من مطمئنم شما می‌توانید برنامه‌نویس‌خوبی شوید. به شرطی‌که از خطا گرفتن کامپایلرها خسته نشوید. بجای اینکه شما خسته شوید بگذارید کامپایلر خسته شود.

باید مصمم باشید تا به هدف خود برسید. امیدوارم همیشه با این طرز تفکر به هدف‌های عالی دست پیدا کنید و این کتاب راهی نو برای شما باشد.

به امید خدا و دلگرمی شما دوستداران برنامه نویسی تصمیم بر این گرفتم که مجموعه‌کتاب‌های برنامه‌نویسی‌ای، تحت عنوان برنامه‌نویسی بدون ترس نوشته و انتشاردهم و با این عمل خدا پسندانه سطح علمی شما عزیزان ارتقا یابد.

هدف من از نوشتن این کتاب علاوه بر اشتراک‌گذاری تجربه‌های چند ساله‌ی خودم و دوستان برنامه‌نویسم خواهد بود، ایجاد یک تعامل دوطرفه بین ما و شما دوستان عزیز خواننده می‌باشد. شما می‌توانید تمامی سوالات برنامه‌نویسی خود را از طریق ایمیل با من در میان بگذارید و من سعی می‌کنم تا حد توانم به تک تک سوالات شما پاسخ دهم، اگر شما این‌کتاب را خواندید و مطالب آن برایتان مفید بود، برای کمک به ادامه‌ی این مسیر مبلغی را به دلخواه خود، به شماره کارت زیر واریز نمایید.

شماره حساب: ۶۰۷۸۳۶۰۵۸۸

شماره کارت: ۶۱۰۴۳۳۷۰۲۶۶۹۳۸۸۵

حساب بانک ملت زهرا بیات قلی لاله

پس از مطالعه کامل این کتاب هر سوال یـا ابهـامی داشـتید، بـرای مـن ایمیـل بفرستید.

فصل‌اول

اساس برنامه‌نویسی

هدف‌های رفتاری:

- ✓ مقدمات برنامه‌نویسی را یاد می‌گیرید.
- ✓ تفکر برنامه‌نویسی‌تان‌عوض می‌شود.
- ✓ تحلیل برنامه را می‌آموزید
- ✓ خطایابی برنامه را فرا می‌گیرید.
- ✓ با تعریف متغیر و چارچوب C++ برنامه هم آشنا می‌شوید.

اساس برنامه‌نویسی

برای نوشتن هر برنامه‌ای، باید دو کار اساسی انجام دهیم:

۱. شکستن برنامه و نوشتن کد

۲. خطایابی

هنگام نوشتن برنامه باید به مواردهای زیر دقت کنیم:

۱. اول باید **مطلب را درک کنیم** و بفهمیم.

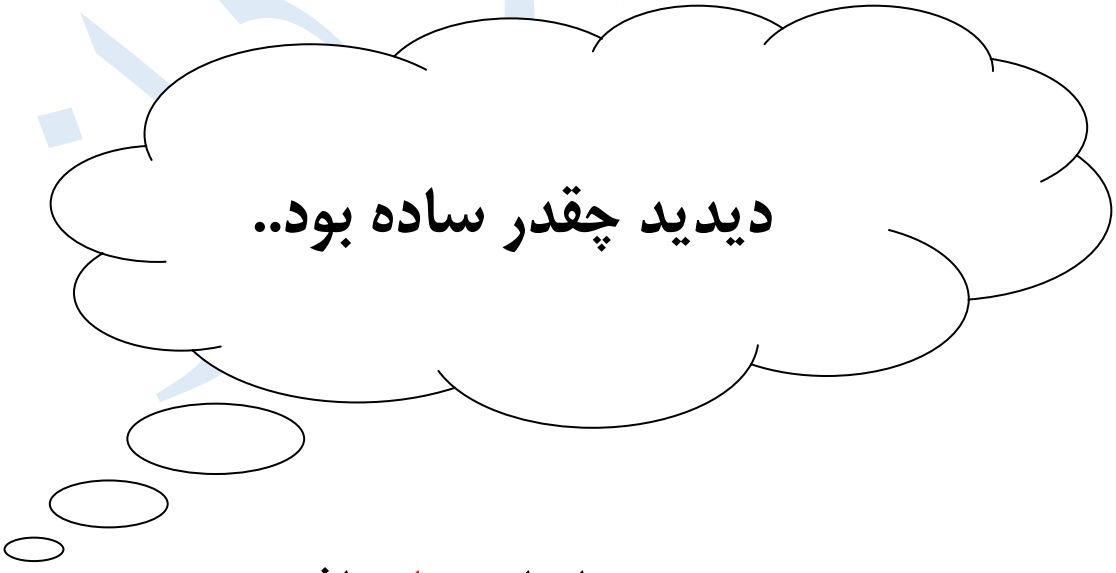
۲. مسئله را **تحلیل** و به **قطعات کوچک بشکنیم.**

۳. مرحله آخرحل مسئله و **نوشتن کدهای برنامه‌نویسی**است.

یک مثال ساده: جمع دو عدد

برای جمع دو عدد

- اول: باید **دو ظرف** حاضر کنیم، تا اعداد داخل آن‌ها قرار بگیرند.
- دوم: باید از کاربر خواهش کنیم **دو عدد** وارد کند.
- سوم: سپس باید دو عدد را **جمع** کنید.
- چهارم: در آخرهم با یک پیغام محترمانه **نتیجه**را به کاربر نشان دهید.



پس با ما همراه باشید...

به دستوراتی که برای کاربر مورد استفاده قرار می‌گیرد ***IO*** می‌گویند.

تمام دستورات برنامه‌نویسی در نرم‌افزارهای مختلف از یک قانون مشابه استفاده می‌کنند، فقط از نظر ظاهر متفاوتند. به مثال‌های زیر دقت کنید تا به اصل موضوع پی ببرید:

برنامه‌ای بنویسید که ***Hello*** را چاپ کند.

Visual basic

```
Print “Hello”
```

Pascal

```
Begin
Write(“Hello”);
End
```

Web

```
<?
Response.write( “Hello”);
%>
```

C

```
Printf(“Hello”)
```

C++

```
Cout<< “Hello”;
```

مطمئنم تا اینجا همه چیز را یاد گرفته‌اید، پس اگر حاضرید به سراغ خطایابـای و روش‌های حل آن رویم:

انواع خطا در برنامه‌نویسی

خطای متنی:

خطای متنی خطایست‌که، هنگام رخ دادن مانع اجرای برنامه می‌شـود و رفـع آن خیلی ساده است. فقط کافیست پیغام را بخوانی.

هر خطایی‌که در متن برنامه رخ می‌دهد یکی از حالات زیر را دارد :

- **خطا در تعریف متغیر**

مثال

```
x = 5;
```

در این مثال خطا می‌گیردچون x باید قبل از مقداردهی تعریف شود:

```
int x;
x = 5;
```

- **خطا درنوع متغیر**

```
int x;
x = 5.1;
```

مقدارx از نوع صحیح تعریف شده در حالی‌که مقدار اعشاری بـه آن نسـبت داده شده پس در اینجا دو نکته مهم وجود دارد:

✓ **تعریف متغیر**.

✓ **جنسیت** دو طرف‌که باید یکسان باشد.

خطا در دستور

- ✓ خطا در **شکل دستور** که مانع اجرای برنامه می‌شود، مثل غلط املایی.
- ✓ خطا در **ساختار دستور**.

خطای املایی

```
Prnt  “h”
```

این دستور از نظر املایی مشکل دارد.(***i*** دستور ***Print*** جا افتاده است.)

خطای ساختاری

```
Cout<<”xx”
```

در***C++*** باید آخرهمه‌ی دستورات سمی‌کالن (;)گذاشته شود. دستور بالا بخاطر سمی‌کولن دچار خطا شده است.

برای حل این مشکلات بهتر است روی کلمه اشتباه قرار بگیرید و کلید ***F1***رابزنید.(البته در پاسکال باید کلید ***Ctrl*** را هم با ***F1*** بگیرید.) می‌بینید که صفحه‌ای باز شده و شما را راهنمایی می‌کند.

خطای ساختاری در VB

```
Print[“hello”]
```

دستور***Print*** در ***VB*** نیازی به [] ندارد و این جزء خطای دستوریست.

خطای ساختاری در پاسکال

```
For x:=1 to 5.5do
End
```

در پاسکال نمی‌توانیم در دستور ***For*** از اعشار استفاده کنیم.

خطای منطقی

این دسته خطاها در اثر اشتباه برنامه‌نویس درطراحی الگوریتم درست برای برنامه و یا گاهی دراثر درنظر نگرفتن بعضی شرایط خاص در برنامه، ایجاد می‌شود. متأسفانه این دسته خطاها در زمان کامپایل اعلام نشده و در زمان اجرای برنامه، خود را نشان می‌دهد. بنابراین، این خود برنامه‌نویس‌است که پس از نوشتن برنامه باید آن را تست کرده و خطاهای منطقی را پیدا و رفع کند. متاسفانه ممکن است برنامه‌نویس خطای منطقی برنامه خود را تشخیص ندهد و این خطا پس از مدت‌ها و تحت یک شرایط خاص توسط کاربر برنامه کشف شود.

فرض کنید برنامه‌ای برای تشخیص شماره ملی نوشته‌اید، که ورودی‌اش از نوع صحیح است. همان‌طورکه می‌دانید در نوع صحیح تعداد صفر قبل عدد حساب

نمی‌شود. پس در این صورت اگر کد ملی با صفر شروع شود، عملیات مورد نظر روی آن انجام نمی‌گیرد. شاید برنامه را با چند کد ملی متفاوت امتحان کنید و برنامه به ظاهر مشکلی نداشته باشد، اما در موردی‌که شماره ملی با صفر شروع شود با دردسر روبرو خواهید شد.

خطا در عبارت متنی

```
"Hello"=5
```

قرار دادن عدد درون عبارت متنی اشتباه است و کامپایلر خطا می‌گیرد.

توضیحات چیست؟

عباراتی‌که برای فهم مسئله نوشته‌اید و فقط جنبه نمایشی دارد را توضیحات می‌گویند. توضیحات درکدنویسی هیچ تاثیری ندارد و توضیحات در هر زبان برنامه‌نویسی متفاوت است.

```
C   /*     */

Vb  '

pascal {                    }
```

قبل از حل مسئله باید مسئله را به:

۱. ورودی

۲. خروجی

۳. شرط

۴. حلقه

بشکنیم.

تعریف متغیر:

متغیر ظرفیست که در آن اطلاعات قرار می‌گیرد. و همان‌طورکه می‌دانید چون ما برای هرغذا از ظرف مخصوصش استفاده می‌کنیم، برای انواع متغیرها هم باید ازظرف مخصوص خودش استفاده کنیم. مثلا اگر عدد صحیح استفاده می‌کنیم از نوع ***int*** اگر اعشاری ،***Float*** و...

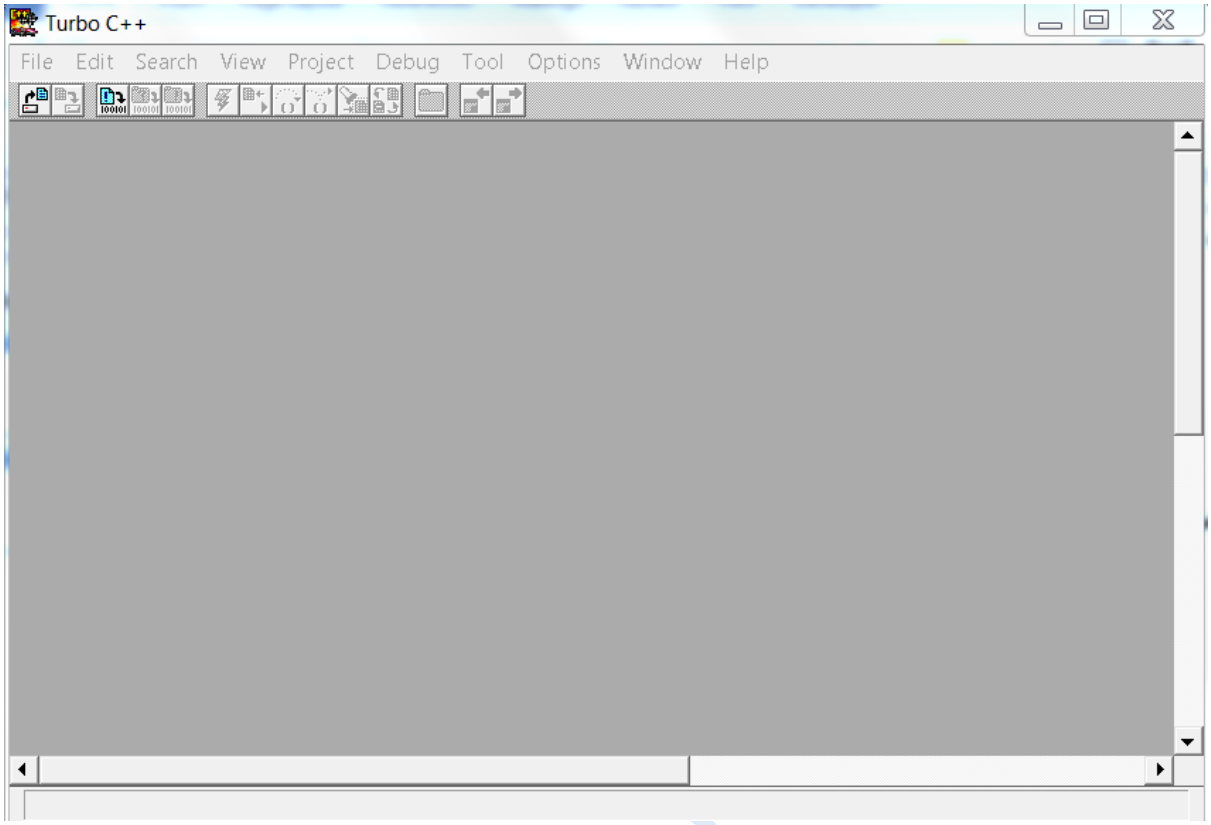
هرگاه پارامتری را نوشتیم و خط زدیم یعنی نیاز به متغیر داریم.

برای نوشتن برنامه در C++ قبل از هرکاری باید از هدرهایی استفاده کنیم. هدرها مجموعه بسته‌هایست‌که درون آن‌ها دستورات برنامه نویسی قرار گرفته است. در حقیقت ما هدرها را می‌نویسیم تا بتوانیم از دستورات مختلف برنامه‌نویسی استفاده کنیم.

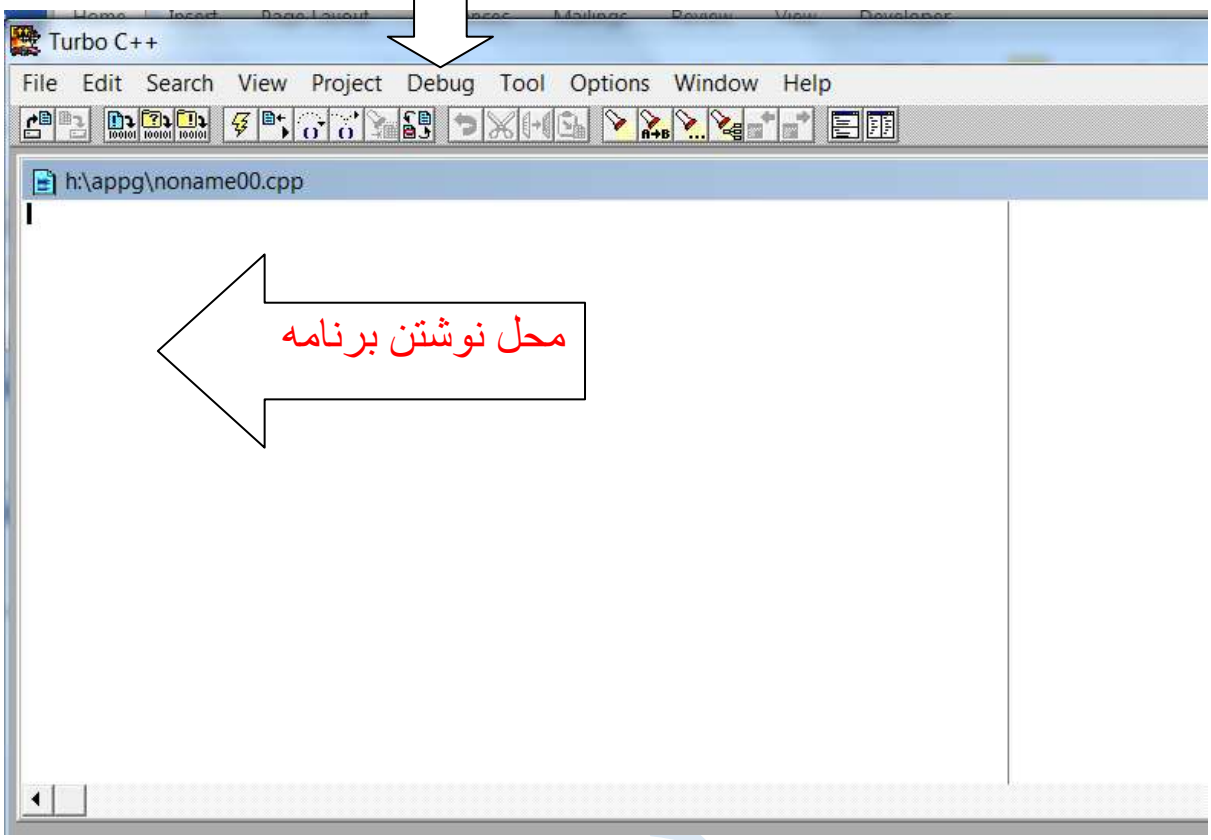
نگران نباشید این هدرها آنقدر تکرار می‌شوند که نیازی به حفظ کردن آن‌ها ندارید.

محیط نوشتن برنامه

من برای نوشتن برنامه‌هایم از محیط Turbo c++ استفاده می‌کنم. که محیط آن به صورت زیر می‌باشد.



برای نوشتن برنامه در این محیط باید از منوی ***File*** گزینه ***New*** را انتخاب کنیم. و بعد از نوشتن برنامه از منوی ***Debug*** گزینه***Run*** را اجرا کرده تا خروجی‌هایتان را ببینید.



چار چوب C++

```
#include <fstream.h>
void main()
{

}
```

هدر

تعامل کاربربا رایانه

شما برای جمع دو عدد در سیستم باید روند زیر را پشت سر بگذارید.

- ابتدا باید **اولین عدد** را **وارد** ماشین حساب کنید، بعد باید کلید **علامت جمع** را بزنید، سپس **دومین عدد** را وارد کنید.

- مرحله دوم ماشین‌حساب **نتیجه** را برای شما **چاپ** می‌کند.

دیدید که تعاملی بین کاربر و سیستم برقرار شد. ما هم در برنامه‌نویسی نیاز به دستوراتی برای برقراری ارتباط بین سیستم و کاربر داریم.

دستور **دریافت ورودی** از کاربر را با ***cin>>***

و دستور **چاپ خروجی** را با ***cout<<***

برنامه

برنامه‌ای بنویسید **شعاع را دریافت و مساحت و محیط** دایره را محاسبه کند.

راه‌حل :

برای این سوال ما نیاز به :

- ظرفی برای نگهداری عدد دریافتی از کاربرداریم ←تعریف متغیر

- دستوری برای دریافت عدد ← دستور cin>>
- چاپ مساحت x*x*3.14; ← دستور cout<<
- چاپ محیط x+x*3.14; ← دستور cout<<

برای تمیزی کار سعی کنید دستورات را نسبت به هم با فاصله مناسب بنویسید. و قبل از هردستور ورودی یا خروجی، برای کاربر پیغامی نمایش دهید. تا روند کار با محتوا شود.

همان‌طور که قبلا گفتم قبل از نوشتن کدها باید هدر مربوط به دستورات را بنویسیم. در اینجا دستور cin و cout از هدر <fstream.h> واکشی می‌شود. پس ما هم باید از این هدر قبل از برنامه استفاده کنیم.

کد:

```
#include <fstream.h>
void main()
{
 int x;          //تعریف متغیر برای دریافت عدد

 cout<<"masahat ra vared kon:" ;//چاپ پیغام ورود مساحت
     cin>> x;  //دریافت ورودی وریختن در متغیر

 cout<<"masahat" ;  //چاپ پیغام مساحت
```

```
    cout<< x*x*3.14;                    //چاپ مساحت

cout<<"mohit" ;                         //چاپ پیغام محیط
    cout<< x+x*3.14;                    // چاپ محیط
}
```

if (شرط)

گاهی در برنامه‌نویسی نیاز به شرط یا مقایسه پیش می‌آید، مثلا مشخص کـردن زوج یا فرد بودن عدد ورودی یا تشخیص بزرگتر بودن بین دو عدد و ....

در اینجور مواقع نیاز به بررسی شرط‌هایی داریم. و باید از دستور ***if*** استفاده کنیم. در زیر درمورد انواع ***if*** ها صحبت شده است.

if یک خطی

اگر بعد از بررسی شرط فقط یک دستور نوشتیم، ***if*** مـا بـه صـورت یـک خطـی نوشته می‌شود.

if(شرط)

د ستـو ر

مثلا

اگر **a** بزرگتر از **b** بود، کلمه تایید را چاپ کن.

```
if(a>b)
  cout<<"taid";
```

if چند خطی

اگر بعد از بررسی شرط بیش‌از یک دستور نوشتیم باید دستورات بعد از **if** را بـین دو اکولاد باز و بسته{} قرار دهیم.

```
if(شرط)
{
 دستور
}
```

مثلا

اگر **a** بزرگتر از **b** بود، کلمه تایید را چاپ کن و سپس **a** را درون متغیر **c** بریز.

```
if(a>b)
{
  cout<<"taid";
  c=a;
}
```

دستور else{}

هرگاه نیاز داشتیم شرط مخالف هم بررسی شود از ***else*** استفاده می‌کنیم. معمولا در پاسخ به **وگرنه** استفاده می‌شود. باید در ***else*** حتما از{} استفاده کنیم.

توجه داشته باشید دستور ***if*** قبل از ***else***، از قانون بالا پیروی می‌کند. یعنی اینکه قرار دادن {}برای***if*** بستگی به تعداد دستورات بعد آن دارد.

مثلا

اگر ***a*** بزرگتر از ***b*** بود، ***a*** را چاپ کند وگرنه ***b*** را چاپ کند.

```
if(a>b)
  cout<<"a";
else
{
  cout<<"b"
}
```

دستور else if(){}

هرگاه شرط ما بیشتر از ۲ مورد باشد باید از این دستور استفاده کنیم.

مثلا

کاربر سه عدد را وارد کند، سپس مشخص شود کدام عدد بزرگتر است.

```
if(a>b &&  a>b)
  cout<<”a”;
elseif(b>a and  b>c)
{
  cout<<”b”
}
else{
  cout<<”c”
}
```

برنامه

برنامه‌ای بنویسید که **دو عدد** را بخواند بعد **جابه‌جا** کند.

راه حل اول:

ابتدا نیاز به سه ظرف داریم، ظرف اول و دوم برای نگهداری عدد و ظرف سوم به عنوان واسط برای جابه‌جایی دو عدد . ← سه متغیر ***A*** , ***B*** , ***C*** (ظرف ***A, B*** حاوی عدد و ظرف ***C*** به عنوان واسط است.)

به عنوان مثال:

A=10 **B=5**

اول باید ظرف ***A*** را در ***C*** بریزیم.

1) C=A ➔ C=10

بعد ظرف ***B*** را در ***A***

2) A=B ➔ A=5

و در آخرظرف ***C*** را در ***B***

3) B=C ➔ B=10

حالا اعداد جابه‌جا شد.

B=10 **A=5**

راه حل دوم

ابتدا به دو ظرف نیاز داریم. باید در ظرف اول، ظرف دوم را هم اضافه کردیعنی:

x=x+y

سپس در ظرف دوم، ظرف اول را از ظرف دوم کم کرد.

y=x-y

و در آخر در ظرف اول، ظرف اول را از ظرف دوم کم کرد.

x=x-y

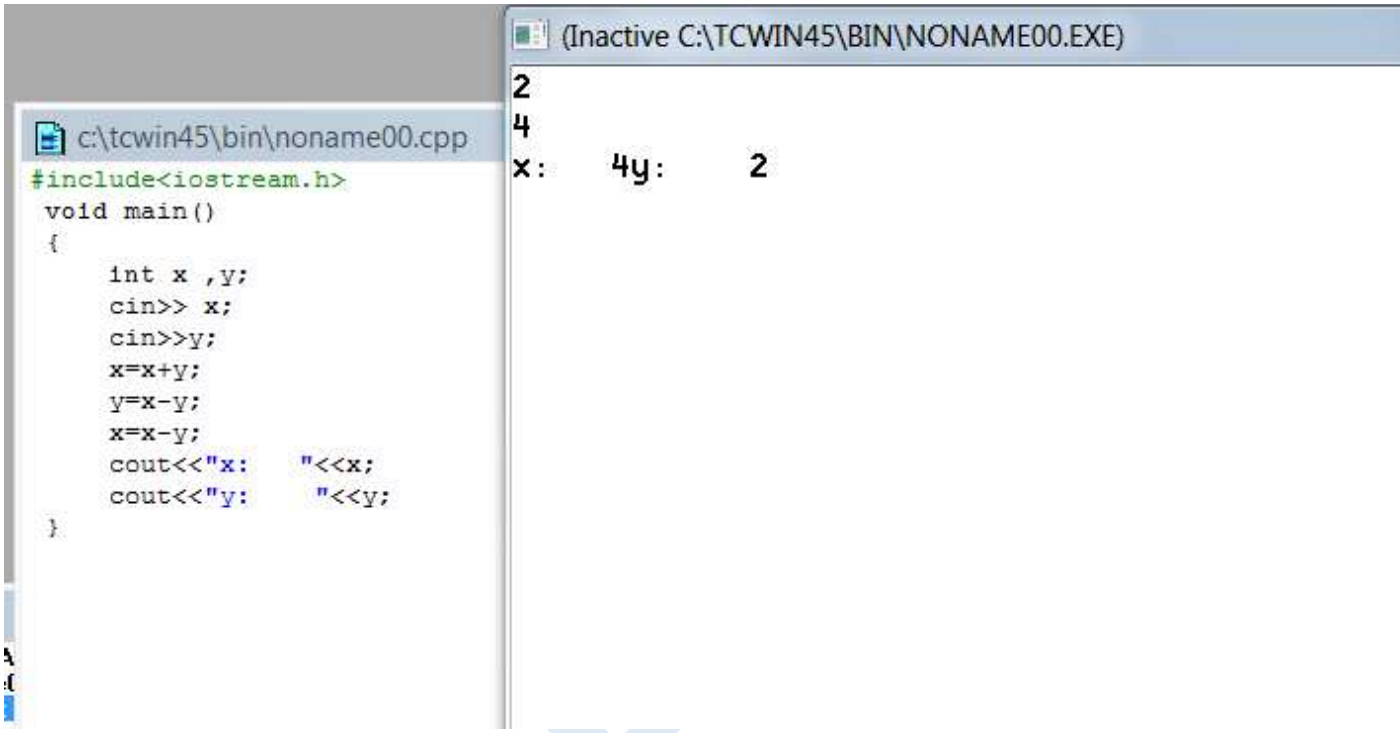
x=2 y=4

x=x+y x=2+4= 6 ➔ x=6 y=4

y=x-y y=6-4=2 ➔ x=6 y=2

x=x-y x=6-2=4 ➔ x=4 y=2

x=4 y=2



برنامه

برنامه‌ای بنویسید دو عدد دریافت هر کدام **بزرگ‌تر** بود آن را چاپ کند.

راه‌حل:

ابتدا دریافت دو عدد ← پس نیاز به دستور ***cin>>*** داریم.

سپس مقایسه دو عدد ← باید از شرط استفاده کنیم.

در آخر چاپ بزرگترین عدد ← نیاز به دستور *cout>>* داریم.

کد:

```
#include <fstream.h>
void main()
{
 int x;  //تعریف متغیر برای دریافت عدد اول
 int y;//تعریف متغیر برای دریافت عدد دوم

 cout<<"adad aval ra vared kon:    "//چاپ ورودااولین عدد
     cin>>x;//دریافت اولین عدد

 cout<<"adad aval ra vared kon:    "//چاپ ورود دومین عدد
     cin>>y;// دریافت دومین عدد

     if(x>y)//شرط بزرگ بودن
     cout<<"x:  "<<x;//اگر عدد اول بزرگ بود آن را چاپ کند
     else// وگرنه
     {
          cout<<"y:    "<<y;//عدد دوم راچاپ کند
     }

}
```

فصل دوم

حلقه‌ها

هدف‌های رفتاری:

- ✓ تعریف حلقه را فرا می‌گیرید.
- ✓ مفهوم متغیرجمع، ضرب و حافظه را یاد می‌گیرید.
- ✓ در پایان هم با حلقه‌های وابسته و تودرتو آشنا می‌شوید.

حلقه‌ها

هرگاه یک فرآیند تکراری با روند مشخص در مسئله باشد، از حلقه استفاده می‌کنیم.

حلقه‌ها برای کارهای تکراری مورد استفاده قرار می‌گیرند. فرض‌کنید نیاز باشد ۱۰۰ عدد دریافت کنید و درون متغیری بریزید اگر ۱۰۰ بار دستور

```
cin>>a
```

```
b=a
```

را تایپ کنیم مسلما خسته می‌شویم. پس یک **راه‌حل منطقی** این است که از یک حلقه استفاده کنیم تا این دستور دائم در حلقه تکرار شود.

برای نوشتن حلقه باید به سه سوال جواب داد:

۱- **از کجا** ← شروع حلقه

۲- **تا کجا** ← انتهای حلقه

۳- **چند تا چند تا** ← گام حرکت

```
For(چند تا چند تا; تاکجا ;از کجا)
For(int i=0 ;i<=100;i++)
```

این حلقه از ۰ شروع تا ۱۰۰ ادامه دارد و گام حرکت آن یکی‌یکی است.

گام حرکت ما می‌تواند معکوس هم باشد. در این صورت باید شروع و انتها با هم جابه‌جا شوند و گام حرکت یکی‌یکی کم شود. البته جای علامت => وجهتش هم باید تغییر کند.

```
For(چند تا چند تا; تاکجا ;از کجا)
For(int i<=100 ;i>=1;i--)
```

در ویرایشی قبلی‌کتاب،ب یشترسوالات خوانندگان این بود که علامت ++ یـا -- چیست؟

اگر دوتا ++ باشد، یعنی پلاس پلاس و باید یک گام به جلو حرکت کنیم. اما اگر دوتا -- باشد یعنی ماینـاس ماینـاس و باید یک گام به عقب حرکت کنیم.

برنامه

برنامه‌ای بنویسید که **اعداد زوج** بین ۱ تا ۱۰۰ را چاپ کند.

راه‌حل:

- پیدا کردن **اعداد زوج** بین ۱ تا ۱۰۰ کاریست تکراری ← پس به یک حلقه نیازمندیم.

- بررسی **زوج بودن** عدد ← نیاز به شرط داریم.

- **چاپ کردن** اعداد زوج بین ۱ تا ۱۰۰ ← دستور ***cout<<***

وقتی شما می‌خواهید بفهمید عدد زوج است یا فرد چه کاری انجام می‌دهید؟؟

مطمئنا عدد را بر۲ تقسیم می‌کنید، اگر باقیمانده صفر شود عدد زوج است وگرنه فرد.

در ***C++*** برای فهمیدن باقیمانده از علامت %(مد) استفاده می کنیم.

```
#include <fstream.h>

void main()
{
     int i;      //تعریف متغیر

     for(i=۱;i<=100;i++)       //حلقه ۱ تا ۱۰۰
     {
          if(i%2==0)      //بررسی زوج بودن
```

```
            cout<<i<<"zoj";//در صورت برقراری، چاپ اندیس و کلمه زوج
        else//وگرنه
        {
            cout<<i<<"fard";//چاپ اندیس و کلمه فرد
        }
    }
}
```

نکته: اگر بعد از متغیر = بگذاریم به معنای انتساب است. یعنی آن آیتم را(مثلا عدد) به آن متغیر انتساب می‌دهیم و محتوای متغیر آن عدد می‌شود. اما برای مقایسه باید از == استفاده کنیم.

حافظه

مکانیست که اعداد، رشته و... درون آن قرار می‌گیرد.

متغیرجمع

متغیریست‌که در یک مکان حافظه قرار می‌گیرد، قبل از حلقه صفر، داخل حلقه اضافه و خارج حلقه استفاده می‌شود.

متغیر ضرب

متغیریست که در یک مکان حافظه قرار می‌گیرد، قبل از حلقه یک، داخل حلقه اضافه و خارج حلقه استفاده می‌شود.

هر وقت در صورت مسئله **جمع** داشتیم، به **متغیرجمع** و هر وقت **ضرب** داشتیم به **متغیرضرب** نیازمندیم.

برنامه

برنامه‌ای بنویسید که **۱۰ عدد از کاربر دریافت** کند اگر **عدد زوج** بود آن را **جمع** کند و در پایان **جمع کل** را نمایش دهد.

راه حل:

- کار **تکراریست**(دریافت ۱۰ عدد) ←**نیاز به حلقه داریم.**
- **زوج بودن** عدد ←**شرط**
- **جمع** اعداد زوج ←متغیر جمع

چون در این حلقه هم شرط و هم دریافت ورودی وجود دارد باید دستورات داخل حلقه بین **دو آکولاد** قرارگیرد.

```
#include <fstream.h>
void main()
{
     int i,a,x;      //تعریف متغیر

     x=0;  //متغیر جمع
for(i=۱;i<=10;i++) //حلقه ۱ تا ۱۰۰
     {
    cout<< "daryaft adad"; //پیغام برای دریافت عدد
          cin>>a//دستور دریافت عدد
          if(a%2==0)      //بررسی زوج بودن
           x=x+a; //جمع اعداد زوج
     }
cout<<"jam kol"<<a;  //پیغام جمع کل و چاپ جمع کل
}
```

حالا شما این چند نمونه سوال را حل کنید...

برنامه

۱-برنامه‌ای بنویسید که **معادله** $ax+bx=c$ را چاپ کند.

راهنمایی:

- در حقیقت معادله بالا بعد از حل شدن به $x=b/a$ تبدیل می‌شود.
- در اینجا به **سه متغیر** نیاز داریم که b,a را کاربر وارد می‌کند.
- مجموع آن‌ها در حافظه جمع x قرار می‌گیرد.
- و در آخر این مجموع چاپ می‌شود.

برنامه

۲-برنامه‌ای بنویسیدکه **دو عدد** را خوانده **بزرگترین** آن را چاپ کند.

راهنمایی:

- نیاز به تعریف **سه متغیر** x,y,z داریم .
- سپس باید سه عدد ازکاربر بگیریم و درون این متغیرها بریزیم.

بعد با سه شرط مشخص‌کنیم کدام متغیر بزرگتر است.(یک بـار متغیـر اول را بـا متغیر دوم و سوم مقایسه می‌کنیم، یک بار متغیر دوم را با متغیر اول و دوم مقایسـه می‌کنیم، یک بار هم متغیر سوم را با متغیر اول و دوم مقایسه می‌کنیم).

برنـامه

۳-برنامه‌ای بنویسیدکه **اعداد فرد** ۱ تـا ۱۰۰ را جمـع کنـد و **میـانگین** آن را نمایش دهد.

راهنمایی:

- پیدا کردن **اعداد فرد** بین ۱ تا ۱۰۰ کاری تکراریست ← پس بـه یـک حلقه نیازمندیم.
- بررسی **فرد بودن** عدد ← نیاز به شرط داریم.
- وقتی شما می خواهید بفهمید عدد فرد است چه کار انجام می‌دهید؟؟

مطمئنا عدد را بر۲ تقسیم می‌کنید، اگر باقیمانده یک شود عدد زوج است وگرنـه فرد.

```
if(i%2==1)
```

برای جمع‌کردن اعداد فرد باید متغیر جمع تعریف کنیم و درون حلقه بعد از بررسی شرط اعداد شامل شرط را جمع کنیم.

```
x=x+i
```

چون ما فقط یک میانگین می‌خواهیم پس باید خارج حلقه، متغیر جمع را تقسیم بر تعداد و چاپ کنیم.(اگر فرمول میانگین را درون حلقه بگذاریم هر دفعه که اعداد فرد جمع می‌شود یک میانگین به تعداد کل اعداد فرد بین ۱ تا ۱۰۰ چاپ می‌شود.) البته باید بدانیم بین عدد ۱ تا ۱۰۰ فقط ۵۰ عدد فرد وجود دارد. پس متغیرجمع باید تقسیم بر ۵۰ شود.

```
y=x/50
```

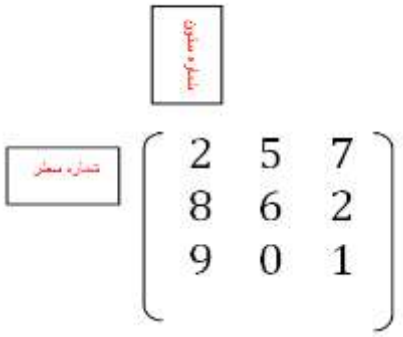
حلقه‌های تودرتو

هرگاه حرکت به صورت ماتریسی بود یعنی طول و عرض را نیاز داشتیم از حلقه تودرتو استفاده می‌کنیم.

به مثال زیر دقت کنید:

ما می‌خواهیم برنامه‌ای بنویسیم‌که درون خانه‌های ماتریس ۳*۳ ای مقدارهـای مورد نظرمان را بریزد. پس باید ابتدا آدرس هر خانه‌ی این ماتریس را پیـدا کـرده و مقدار مورد نظر را درون آن قرار دهیم.

فرض کنید که ماتریس زیر را قرار است از کاربر دریافت کنیم.



آدرس خانه‌های این ماتریس به شرح زیر است:

(1,1)=2 (2,1)=8 (3,1)=9

(1,2)=5 (2,2)=6 (3,2)=0

(1,3)=7 (2,3)=2 (3,3)=1

همانطورکه می‌بینید، به ازای هر سطر سه ستون نوشته می‌شود. پس نیاز به حلقه تودرتو داریم.

در حلقه تودرتو، دو حلقه وجود دارد. یکی خارجی و دیگری داخلی. به ازای هربار اجرای حلقه خارجی، یک دور کامل حلقه داخلی اجرا می‌شود. در زیر دستور حلقه تودرتو را می‌بینید.

در حقیقت تعداد سطردر تعداد ستون ضرب می‌شود. پس اگر ۳ سطر و ۱۰ ستون هم داشته باشیم به ازای یک سطر ۱۰ تا ستون نوشته می‌شود.

```
For(inti=0 ;i<=10; i++)
    {
    دستور
    For(int j=0 ;j<=10; j++)
        {
    دستور
        }
    دستور
    }
    دستور
```

در هر کدام از مکان‌ها می‌توانیم دستور بنویسیم.

به مثال‌های زیر دقت‌کنید تا موضوع برایتان بهتر روشن شود.

برنامه‌ای بنویسید که **شکل** ماتریسی زیر را چاپ کند.

```
****
****
****
****
```

همانطورکه می‌بینید شکل به صورت ماتریسی است پس به **دو حلقه** نیاز داریم. که باید درون حلقه داخلی **ستاره‌ها**را چاپ کنیم. اگر ستاره‌ها دورن حلقه خارجی چاپ شود، در خروجی فقط سه ستاره خواهیم داشت. چون همانطور که در بالا دیدیم یک ماتریس۴*۴ فقط ۴ سطر دارد پس اگر دستور در حلقه خارجی باشد ۴ ستاره چاپ می‌شود.

راه حل:

```
#include <fstream.h>
void main()
{
     for(int i=0 ;i<=3;i++)//حلقه۴ تایی خارجی برای چاپ ستاره
     {
          for(int j=0 ;j<=3;j++)//حلقه ۴ تایی داخلی برای چاپ ستاره
          {
               cout<<"*";//چاپ ستاره
          }
          cout<<endl   ;//بعداز یک دور چاپ باید اینتر زده شود.
     }
}
```

حل حلقه:

خروجی	j	i
****	۰و۱و۲و۳	۰
****	۰و۱و۲و۳	۱
****	۰و۱و۲و۳	۲
****	۰و۱و۲و۳	۳

برنامه

برنامه‌ای بنویسید که شکل زیر را چاپ کند.

۱۱۱
۲۲۲
۳۳۳

حل حلقه:

خروجی	j	i
۱۱۱	۱و۲و۳	۱
۲۲۲	۱و۲و۳	۲
۳۳۳	۱و۲و۳	۳

از حل این حلقه به این نتیجه می‌رسیم که درخروجی فقط *i* چاپ شده. پس داخل حلقه داخلی مقدار *i* را چاپ می‌کنیم.

کد:

```
#include<fstream.h>
void main()
{
     for(int i=1 ;i<=3;i++)   //حلقه ۴ تایی
     {
          for(int j=1;j<=3;j++)//حلقه ۴ تایی داخلی
          {
               cout<<i   ;      //چاپ اندیس
          }
          cout<<endl   ; //زدن اینتر
     }
}
```

برنامه‌ای بنویسید که شکل زیر را چاپ کند.

۱۲۳

۲۴۶

۳۶۹

حل حلقه:

i	j	خروجی
۱	۱و۲و۳	۱۲۳

i*j

۲	۱و۲و۳	۲۴۶
۳	۱و۲و۳	۳۶۹

از حل این حلقه به این نتیجه می‌رسیم که در خروجی فقط ***j*i*** چاپ شده. پس داخل حلقه داخلی مقدار ***j*i*** را چاپ می‌کنیم.

کد:

```
#include <fstream.h>
void main()
{
     for(int i=1 ;i<=3;i++)   //حلقه ۴ تایی
     {
          for(int j=1 ;j<=3;j++)  //حلقه ۴ تایی داخلی
          {
               cout<<i*j   ;//ضرب دواندیس حلقه‌ها
          }
          cout<<endl   ;//زدن اینتر
     }
}
```

حالا برنامه‌های زیر را شما بنویسید.

خروجی زیر را چاپ کنید.

۱)

۱۰۳

۰۰۰

۳۰۹

راهنمایی:

حل حلقه:

خروجی	j	i
۱۰۳	۱و۲و۳	۱
۰۰۰	۱و۲و۳	۲
۳۰۹	۱و۲و۳	۳

در نتیجه اگر $i=2$ یا $j=2$ بود باید عدد صفر چاپ شود وگرنه باید .

```
if(i==2 || j==2)
```

۲)

100

120

123

حل حلقه:

خروجی	j	i
100	۱و۲و۳	۱
120	۱و۲و۳	۲
123	۱و۲و۳	۳

در نتیجه اگر $i<j$ باشد، عدد صفر وگرنه j را چاپ می‌کند.

حلقه تودرتوی وابسته

هرگاه شکل ماتریسی متقارن نبود،(مثلا ماتریس ما به صورت مثلث، لوزی و..بود.) و حلقه‌ها هم وابسته بودن. حلقه ما از نوع تودرتو می‌شود.

دراین نوع ماتریس‌ها حلقه داخلی به حلقه خارجی وابسته است. حلقه خارجی هر عددی شود، حلقه داخلی به همان مقدار تکرار می‌شود.

مثلا فرض کنید شکل زیر را باید چاپ کنیم:

*

**

***

اگر به صورت ماتریس آن را حل کنیم:

(1,1)=*

(2,1)=* (2,2)=*

(3,1)=* (3,2)=* (3,3)=*

همانطورکه می‌بینید عدد سطرهرچند باشد، عدد ستون به همان اندازه تکرار می‌شود. مثلا وقتی حلقه خارجی ۱ باشد حلقه داخلی یک بار اجرا می‌شود. وقتی ۲ باشد ۲ بار و....

حل حلقه:

خروجی	j	i
*	۱	۱
**	۱و۲	۲
***	۱و۲و۳	۳

کد:

```
#include<fstream.h>
void main()
{
     for(inti=1 ;i<=3;i++)    //حلقه ۴ تایی خارجی
     {
```

```
            for(int j=1 ;j>=i;j++)  //حلقه ۴ تایی داخلی
            {
                cout<<"*"   ;              //چاپ ستاره
            }
        cout<<endl   ;        //زدن اینتر
        }
}
```

برنامه‌های زیر را شما بنویسید.

۱)

۱

۱۲

۱۲۳

حل حلقه:

i	j	خروجی
۱	۱	1
۲	۱۲	۱۲
۳	۱۲۳	۱۲۳

حلقه وابسته است. همان‌طور که می‌بینید درخروجی عدد ***j*** نوشته شده.پس ما هم عدد ***j*** را چاپ می‌کنیم.

۲)

۱

۲۲

۳۳۳

حل حلقه:

خروجی	j	i
۱	۱	۱
۲۲	۱۲	۲
۳۳۳	۱۲۳	۳

حلقه وابسته است.همان‌طورکه می‌بینید درخروجی عدد*i* نوشته شده.پس ما هم عدد *i* را چاپ می‌کنیم.

۳)

۱

۲۴

۳۶۹

حل حلقه:

خروجی	j	i
۱	۱	۱
24	۱۲	۲
369	۱۲۳	۳

حلقه وابسته است. همان‌طورکه می‌بینید در خروجی عدد***i*j*** نوشته می شود.پس ما هم عدد ***i*j*** را چاپ می کنیم.

۴)

۱

**

۳۳۳

*****

حل حلقه

خروجی	j	i
۱	۱	۱
**	۱و۲	۲
333	۱و۲و۳	۳
****	۱و۲و۳و۴	۴

حلقه وابسته مثل بالا که باید در آن شرط زوج بودنi بررسی شود و درصورت زوج بودن*چاپ شود وگرنه عدد *i* چاپ شود

فصل سوم

آرایه‌ها

هدف‌های رفتاری:

- ✓ تعریف آرایه و نحوه استفاده آن را یاد می‌گیرید.
- ✓ با انواع آرایه‌های دو بعدی و سه بعدی آشنا می‌شوید.
- ✓ مفهوم ماتریس خلوت را هم می‌آموزید.

آرایه

هرگاه به تعداد زیادی متغیر از یک جنس نیاز داشته باشیم از آرایه‌ها استفاده می‌کنیم. تصورکنید قرار است برنامه‌ای بنویسید که نمرات ۵ دانش‌آموز را بگیرد سپس:

۱. بیشترین نمره

۲. کمترین نمره

۳. نمرات کمتر از ۱۰

۴. میانگین نمرات

۵. و..

را محاسبه و چاپ کند.

در این صورت مجبورید ۵ متغیر تعریف کنید و عملیات بالا را روی تک‌تک این متغیرها به صورت جداگانه انجام دهید. مطمئنم شما هم با من هم نظرید و این کار را دشوار و طاقت فرسا تلقی می‌کنید. حالا اگر این ۵ دانش آموز به ۵۰ دانش آموز

تغییر کند، کار دشوارتر هم خواهد شد. چون مجبورید این بار ۵۰ متغیر تعریف کنید. پس بهتر است به سراغ یک راه حل منطقی برویم.

آرایه‌ها مانند بسته‌هایی هستند که هر چقد نیاز دارید می‌توانید به خانه‌های آن اضافه کنید.

پس:

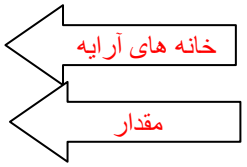
آرایه فضای ذخیره‌سازیی است که در آن باید سه عمل انجام شود:

۱- ورود اطلاعات

۲- پیمایش

۳- چاپ

آرایه‌ها از صفر شروع می‌شوند و دسترسی به اطلاعات آن با ایندکس**(شمارنده حلقه)** می‌باشد.



x(0)	x(1)	x(2)	x(3)	x(4)
10	20	30	40	50

تعریف آرایه در C++

مثلا:

```
int x[5]
```

در c++ آرایه از صفر شروع می‌شود. آرایه بالا ۵ خانه را شامل می‌شود.

برای نوشتن در آرایه ها باید به صورت زیر عمل کنیم:

x[0]=10;

x[1]=2;

x[2]=0;

x[3]=4;

,...

دیدید که کار تکرایست پس نیاز به حلقه داریم. در آرایه حلقه ما از صفر شروع می‌شود و پایان آن تعداد خانه‌های آرایه است. مثلاآرایه ۵ تایی به صورت زیر است.

```
x[4];
```

به طور مثال می‌خواهیم یک آرایه ۵ تایی را مقداردهی کنیم:

- نیاز به تعریف آرایه ۵ تایی داریم.
- نیاز به یک حلقه.
- و نیاز به دستور دریافت عدد.

اینجا شماره خانه آرایه همان شماره اندیس حلقه است.

```
int x[4];
     for(int i=0 ;i<=4;i++)
     {
          cin>>x[i];
     }
```

برای خواندن از آرایه هم باید یک حلقه دیگر تعریف کنیم و این بار دستور چاپ را بنویسیم.

```
for(int j=0 ;j<=4;j++)
     {
          Cout<<x[j];
     }
```

در اینجا با شماره اندیس به شماره خانه آرایه دست پیدا می‌کنیم و عدد آن خانـه را چاپ می‌کنیم.

برنامه:

برنامه‌ای بنویسید **۵ عدد را بخواند** سپس چاپ کند.

کد:

```
#include <fstream.h>
void main()
{
     int x[4];//تعریف آرایه ۵ تایی
     for(int i=0 ;i<=4;i++)//تعریف حلقه ۵ تایی
     {
          cin>>x[i];//پر کردن آرایه
     }
     for(i=0 ;i<=4;i++)//تعریف حلقه ۵ تایی
     {
     cout<< x[i];//چاپ کردن آرایه
     }
}
```

برنامه

برنامه‌ای بنویسید که در یک آرایه ۵ تایی **خانه‌های زوج آرایه** را مقداردهی کند.

راه‌حل:

- در این سوال فقط به خانه های زوج نیازمندیم پس باید شرط گذاشته شود.

کد:

```
#include <fstream.h>
void main()
{
     int x[4]; //تعریف آرایه ۵ تایی
     for(int i=0 ;i<=4;i++)   //حلقه ۵ تایی
     {
          if(i%2==0)          //بررسی زوج بودن شمارنده حلقه
            cin>>x[i];        //در صورت زوج بودن خانه آرایه ای که با اندیس زوج برابر است را مقداردهی می کنیم
     }
     for(i=0 ;i<=4;i++) //حلقه ۵ تایی
     {
          if(i%2==0)     //بررسی زوج بودن شمارنده حلقه
          cout<< x[i];//چاپ خانه های زوج مقداردهی شده
     }
}
```

برنامه

برنامه‌ای بنویسید که ۵ عدد را **خوانده** و **جمع و میانگین** آن را چاپ کند.

کد:

```
#include <fstream.h>
void main()
{
     int x[4]; //تعریف آرایه ۵ تایی
   int y=0;          //تعریف متغیر برای میانگین
     for(int i=0 ;i<=4;i++)  //حلقه ۵ تایی
     {
          cin>>x[i];      //پر کردن آرایه
          y=y+x[i]; //جمع کل خانه های آرایه
     }
     cout<<y/5;          //چاپ میانگین
}
```

حالا یک مقدار سوالات سخت‌تر می‌شود...

برنامه

نمرات ۵ دانش آموز را دریافت و

۱- کمترین نمره

۲- بیشترین نمره

۳- نمرات کمتر از ۱۰

۴- میانگین نمرات

راه‌حل:

در این سوال باید سه کار انجام شود:

۱-مقداردهی اولیه

۲-مقایسه

۳- استفاده

در سوال ۱و۲ نیاز به متغیری داریم که اولین نمره در آن قرار گیرد و آن متغیر با خانه‌های بعدی مقایسه، اگرخانه‌ی مورد نظرکمتر یا بیشتر بود باید آن عدد در متغییر ریخته شود.

مثال: شما اعداد روبرو را وارد می‌کنید.

X[0]	X[1]	X[2]	X[3]	X[4]
4	2	5	9	1

راه‌حل:

به طور پیش فرض $c=0$ است. بعد از پر شدن حلقه c اولین خانه آرایه می‌شود.

c با تک تک خانه‌ها **مقایسه** می‌شود. اگر c بزرگ‌تر باشد c برابر آن خانه آرایه می‌شود.

i	x[i]	c	out low
0	4	4	4
1	2	4	2
3	5	2	2
4	9	2	2
5	1	1	1

نتیجه ۱ می‌شود. برای بزرگتر هم همین‌کار را می‌کنیم.

کد:

```
#include  <fstream.h>
void main()
{
 int x[4];     //تعریف آرایه ۵ تایی
 int min;//تعریف متغیر مقایسه
     min=0;
     for(inti=0 ;i<=4;i++)    //حلقه ۵ تایی
     {
          cin>>x[i];          //پر کردن آرایه
     }
                    min=x[0]; //ریختن اولین خانه آرایه در یک متغیر به منظور مقایسه
     for(i=0 ;i<=4;i++)//حلقه ۵ تایی
     {
          if(min>x[i])  // بررسی بزرگتر بودن متغیر
            min=x[i];       //اگر متغیر بزرگ باشد یعنی خانه آرایه کوچک است پس باید
                            //متغیر تغییر کند.
     }
     cout<< min;     //چاپ متغیر که الان کوچک ترین عدد است
}
```

نکته: *x[i]* بیرون حلقه مفهومی ندارد. چون ***i*** در حلقه استفاده می‌شود.

قسمت۲:

کد:

```
#include  <fstream.h>
void main()
{
     int x[4]; //تعریف آرایه ۵ تایی
```

```
	int max;		تعریف متغیر//
	for(int i=0 ;i<=4;i++)
	{
		cin>>x[i];		پر کردن آرایه//
	}
			max=x[0]; ریختن اولین خانه آرایه در یک متغیر به منظور مقایسه//
	for(i=0 ;i<=4;i++)		حلقه ۵ تایی//
	{
		if(max<x[i])		اگر متغیر کوچک باشد یعنی خانه آرایه بزرگ است پس
باید متغیر تغیر کند.//
			max=x[i];		ریختن اولین خانه آرایه در یک متغیر به منظور
مقایسه//
	}
	cout<< max;		چاپ متغیر که الان بزرگ ترین عدد//
}
```

قسمت ۳:

کد:

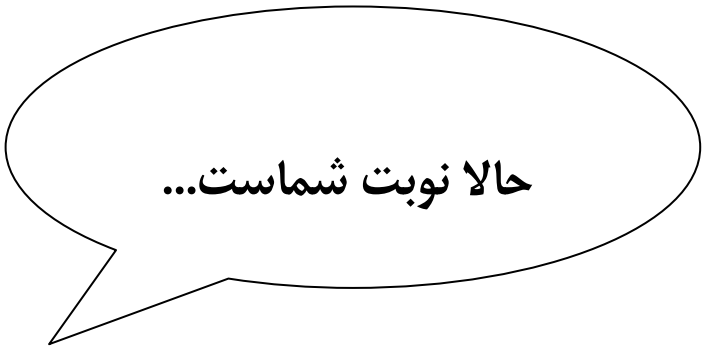
```
#include  <fstream.h>
void main()
{
	int x[4]; تعریف آرایه ۵ تایی//
	for(int i=0 ;i<=4;i++)  حلقه ۵ تایی//
	{
		cin>>x[i];		پر کردن آرایه//
	}
	for(i=0 ;i<=4;i++) حلقه ۵ تایی//
	{
		if(x[i]<10)	// بررسی کوچکتر از ۱۰ بودن خانه آرایه
			cout<<x[i];	اگر خانه آرایه کوچکتر از ۱۰ بودچاپ می شود.//
```

```
    }
}
```

قسمت۴:

کد:

```
#include <fstream.h>
void main()
{
    int x[4]; //تعریف آرایه ۵ تایی
    int c;    //تعریف متغیر
    for(int i=0 ;i<=4;i++)  //حلقه ۵ تایی
    {
        cin>>x[i];      //پر کردن آرایه
    }
    c=0;        //متغیر را صفر می کنیم(متغیر جمع)
    for(i=0 ;i<=4;i++)        //حلقه ۵ تایی
    {
        c=c+x[i]; //جمع خانه های آرایه
    }
    cout<< c/5;           //میانگین خانه های آرایه
}
```



برنامه:

۱) آرایه‌ای ۵ تایی را **مقداردهی** کنید، سپس **یک عدد را بخوانید**، و برنامه اعلام کند عدد درکدام خانه آرایه قرار گرفته.

راهنمایی:

در این سوال :

- ابتدا در یک حلقه ۵ تایی یک آرایه ۵ تایی را مقداردهی می‌کنید.
- سپس خارج حلقه عددی را از کاربر می‌گیرید.
- دوباره در یک حلقه دیگر آن متغیر دارای عدد را با تک‌تک خانه‌ها بررسی می‌کنید. اگر شرط برقرار بود، اندیس حلقه را چاپ کند.

شرط:

```
if(c=x[i])
cout<<i;
```

برنامه:

۲) برنامه‌ای بنویسید که آرایه ۵ تایی را مقداردهی و سپس اعداد زوج لیست را برابر صفر کند.

۳) راهنمایی:

- دراین سوال باید در حلقه دوم شرط زوج بودن را بررسی کرد و در صورت برقراری شرط مقدار آن خانه را صفر قرار دهیم.

```
if(x[i]%2=0)
x[i]=0;
```

برنامه:

۴) برنامه‌ای بنویسیدکه در یک آرایه ۵ تایی اگر محتوا با اندیس برابر بود، محتوا را چاپ کند.

راهنمایی:

- در این سوال باید با یک شرط بررسی‌کنیم مقدار خانه آرایه با اندیس آن برابر است یا خیر؟ مثلا خانه شماره ۱ برابر ۱ است یا نه و...

شرط

```
if(x[i]=i)
cout<<x[i];
```

برنامه:

۵) در همان آرایه ۵ تایی عددی از کاربر بگیرد در صورت تکراری بودن، تعداد آن را چاپ کند.

راهنمایی:

- بعد از پر کردن آرایه در خارج حلقه عددی از کاربر دریافت می‌کنید.
- سپس در حلقه بعد با شرطی آن را با خانه‌های آرایه مقایسه می‌کنید، اگر برابر یک بود شمارنده اضافه می‌شود.

```
if(c==x[i])
m=m+1;
```

برنامه:

۶) برنامه‌ای بنویسیدکه تعداد خانه‌های زوج و خانه‌های فرد در یک آرایه ۵ تایی را چاپ‌کند.

دقیقا مانند مثال قبلیست.

جمع آرایه متناظر

اگر دو آرایه را پرکنیم و نیاز به جمع خانه‌های آن باشد، باید چه ترفندی را به کار ببریم؟ به مثال زیر دقت کنید.

شماره خانه	1	2	3	4	5
آرایه x	10	20	50	20	10
آرایه y	7	10	20	30	4
جمع آرایه‌ها	17	30	70	50	14

برای حل این سوال باید توسط یک حلقه دو آرایه را پرکنیم و توسط حلقه دیگـر و یک متغیر آن‌ها را جمع و نمایش دهیم.

کد:

```
#include <fstream.h>
void main()
{
  int x[4];     تعریف آرایه ۵ تایی//
 int y[4];تعریف آرایه ۵ تایی//
 int c;          //تعریف متغیر برای جمع دو آرایه

for(int i=0 ;i<=4;i++)   //حلقه ۵ تایی
     {
          cin>>x[i];              //پر کردن آرایه اول
          cin>>y[i];              //پر کردن آرایه دوم
     }
     c=0;              //متغیر جمع که قبل از حلقه باید صفر باشد.

for(int i=0 ;i<=4;i++)          //حلقه ۵ تایی
     {
          c=y[i]+x[i];   //جمع دو آرایه در متغیر جمع
          cout<<c;         //جمع دو آرایه
     }
}
```

نکته: اگر جمع آرایه را خارج حلقه بگذاریم، جمع تمام خانه‌های دو آرایه را می‌بینیم. درحالی‌که ما قصد دیدن جمع خانه‌های متناظر را داریم.

معکوس کردن آرایه‌ای در یک آرایه‌ی:

در این برنامه باید خانه‌های یک آرایه‌ای را دریافت کنیم و سپس آن را در آرایه‌ی دیگر معکوس کنیم.

X[] دریافت	3	2	1
Y[]	1	2	3

حل حلقه:

i	x[i]	y[i]
0	3	1
1	2	2
2	1	3

در حقیقت:

Y[0]=x[2]

Y[1]=x[1]

Y[2]=x[0]

i			
0	2-0=2	y[0]	x[2]
1	2-1=1	y[1]	x[1]
2	2-2=0	y[2]	x[0]

پس برای معکوس‌کردن یک آرایه در آرایه‌ی دیگر، باید خانه‌ی آرایه‌ی دارای مقدار را ازعدد کل آرایه کم کنیم و سپس آن را در آرایه دوم بریزیم.

مثلا اگر آرایه دارای مقدار ما به صورت روبرو باشد: `int x[4];`

باید در آرایه دوم این چنین بنویسیم: `y[i]=x[4-i];`

در اینجا اگر i=0 (i همان شمارنده حلقه می‌باشد.)باشد :y[0]=x[4-0] یعنی خانه اول آرایه خالی با خانه‌ی آخر آرایه‌ی دارای مقدار پر می‌شود.(در نظر داشته باشیدکه خانه‌های آرایه در c++ از صفر شروع می‌شود.)

برنامه:

برنامه‌ای بنویسید آرایه ۵ تایی را مقداردهی و در آرایه دیگر معکوس کند.

کد:

```
#include <fstream.h>
void main()
{
     int x[4];          //تعریف آرایه ۵ تایی
     int y[4];          //تعریف آرایه ۵ تایی
     for(int i=0 ;i<=4;i++)          //حلقه ۵ تایی
     {
          cin>>x[i];      //پر کردن آرایه اول
     }
     for(int i=0 ;i<=4;i++)   //حلقه ۵ تایی
     {
          y[i]=x[4-i];          //معکوس کردن آرایه اول در دوم
          cout<<y[i];           //چاپ آرایه دوم
     }
}
```

آرایه ۲بعدی و ۳بعدی:

آرایه دوبعدی آرایه‌ایست که مانند حلقه‌های تودرتو ماتریسی عمل می‌کند. و آرایه سه بعدی علاوه بر طول و عرض دارای عمق نیز می‌باشد.

تعریف:

```
int x[3][3]
```
دوبعدی

```
int x[2][3][5]
```
سه بعدی

برای پرکردن آرایه باید به تناسب آرایه ازحلقه استفاده کنیم. مثال زیر برای پر کردن و چاپ کردن یک آرایه ۲*۲ می‌باشد.

کد:

```
#include <fstream.h>
void main()
 {
     int j;     //تعریف متغیر برای حلقه
     int i;     //تعریف متغیر برای حلقه
     int x[1][1];          //تعریف آرایه دو بعدی ۲*۲
     for(i=0;i<=1;i++)  //حلقه ۲ تایی خارجی
     {
          for(j=0;j<=1;j++)  //حلقه ۲ تایی داخلی
```

```
        {
            cin>>x[i][j];          //پر کردن آرایه ۲*۲
        }
    }
    for(i=0;i<=1;i++)  //حلقه ۲ تایی خارجی
    {
        for(j=0;j<=1;j++)  //حلقه ۲ تایی داخلی
            {
                cout<<x[i][j];//چاپ آرایه ۲*۲
            }
    }
}
```

برنامه:

برنامه‌ای بنویسید که جمع هرسطر آرایه دو بعدی ۳*۳ را چاپ کند.

جمع	۲	۱	۰	شماره خانه ها
۱۰	۳	۵	۲	۰
۲۱	۵	۷	۹	۱
۴	۳	۱	۰	۲

شماره سطر	out	j	i
(0,0)	۲	0	0
(0,1)	۵	۱	0
(0,2)	۳	۲	0

(1,0)	۹	0	۱
(1,1)	۷	۱	۱
(1,2)	۵	۲	۱
(2,0)	0	0	۲
(2,1)	۱	۱	۲
(2,2)	۳	۲	۲

اگر توجه کنید در هر دوره ***i*** ثابت است. پس برای جمع‌کردن به صورت زیر عمل می‌کنیم: `c=c+x[i][j];`

کد:

```
#include <fstream.h>
void main()
 {
     int j;     //تعریف متغیر برای حلقه
     int i;     //تعریف متغیر برای حلقه
     int x[2][2];//تعریف آرایه ی ۳*۳
     int  c=0; //تعریف متغیر جمع
     for(i=0;i<=2;i++)  //حلقه ۳ تایی خارجی
     {
          for(j=0;j<=2;j++)  //حلقه۳ تایی داخلی
          {
               cin>>x[i][j]; //پر کردن آرایه ۳*۳
          }
     }
     for(i=0;i<=2;i++)  //حلقه ۳ تایی خارجی
```

```
    {
        for(j=0;j<=2;j++)   //حلقه ۳ تایی داخلی
        {
            c=c+x[i][j];          //جمع آرایه۳*۳
        }
    }
}
```

جمع ستون‌ها

۲	۱	۰	شماره خانه ها
۳	۵	۲	۰
۵	۷	۹	۱
۳	۱	۰	۲
۱۱	۱۳	۱۱	جمع

شماره سطر	out	j	i
(0,0)	۲	0	0
(1,0)	۵	0	۱
(2,0)	۳	0	۲
(0,1)	۹	۱	0
(1,1)	۷	۱	۱
(2,1)	۵	۱	۲
(0,2)	0	۲	0

(1,2)	۱	۲	۱
(2,2)	۳	۲	۲

اگر توجه کنید در هر دوره ***j*** ثابت است. پس برای جمع کردن به صورت زیر عمل می‌کنیم: `c=c+x[j][i];`

ماتریس خلوت:

به ماتریسی که تعداد خانه‌های خالی(صفر) آن بیشتر از خانه‌های پر باشد ماتریس خلوت می‌گویند. در برنامه ماتریس خلوت بعد از پرکردن ماتریس از دو شمارنده باید استفاده می‌کنیم. یکی برای شمارش اعداد بالای صفر و دیگری برای شمارش اعداد صفر و درآخر آن‌ها را با هم مقایسه می‌کنیم. اگر خانه‌های صفر بیشتر بود ماتریس خلوت است وگرنه خیر.

کد:

```
#include  <fstream.h>
void main()
{
     int j;          //تعریف متغیر برای حلقه
     int i;          //تعریف متغیر برای حلقه
     int c=0;        //تعریف متغیر جمع
     int m=0;        /تعریف متغیر جمع
     intx[3][3];     //تعریف آرایه ۴*۴
     for (i=0;i<=3;i++) //حلقه ۴ تایی خارجی
```

```
    {
        for (j=0;j<=3;j++) //حلقه ۴ تایی داخلی
        {
            cin>>x[i][j]; //پر کردن آرایه ۴*۴
        }
    }
    for (i=0;i<=3;i++) //حلقه ۴ تایی خارجی
    {
        for (j=0;j<=3;j++) //حلقه ۴ تایی داخلی
        {
            if (x[i][j]=0)//شرط صفر بودن خانه آرایه
            c=c+1;//اگر خانه آرایه صفر باشد،یکی به شمارنده صفر اضافه می شود
            else        //وگرنه
            {
                m=m+1;//یکی به شمارنده صفر نبودن اضافه می شود
            }
        }
    }
    if(c>m)   //اگر شمارنده صفر از شمارنده عدد بزرگ بود
cout<<"khalvat";//یعنی ماتریس خلوت است

}
```

سوالات مربوط به شما :

برنامه:

برنامه‌ای بنویسیدکه اگر در ماتریس ۳*۳ قطر اصلی صفر بود پیغام ok چاپ کند.

راهنمایی:

- در این سوال شماره خانه‌های قطر اصلی را پیدا و برای آن شرط صفر بودن را بررسی کنید. نمونه این سوال در بالا حل شده.

فصل چهارم

کلاس

هدف‌های رفتاری:

- ✓ تعریفی‌ازکلاس وتوابع را می‌آموزید.
- ✓ با حلقه شرطی آشنا می‌شوید.
- ✓ مفاهیم اشاره‌گرها و کاربرد آن را فرا می‌گیرید.

حلقه شرطی(while):

اگر انتهای حلقه ما معلوم نبود احتمالا حلقه ما شرطیست.

در حلقه شرطی کار تکرایست و باید به سه سوال جواب داده شود:

- **از کجا:** شروع حلقه باید عددی مخالف صفر باشد.
- **تا کجا:**معلوم نیست(مبهم) باید شرط توقف داشته باشیم.
- **چند تا چند تا :** یکی‌یکی.

در ***for*** چون انتها را می‌دانستیم پس برای تعداد مشخص استفاده می‌شد. اما در ***while*** به علت ندانستن شروع و پایان حلقه برای تعداد مبهم استفاده می‌شود.

برنامه:

برنامه‌ای بنویسید تا زمانی‌که صفر وارد نکردیم از کاربر عدد دریافت کند.

راه حل:

دراین برنامه به علت ندانستن پایان حلقه نیاز به یک شرط داریم.(شرط مخالف صفر)

```
#include <fstream.h>
void main()
{
     int c=1;          //مقدار اولیه
     while(c!=0)       //شرط
     {
          cin>>c;  //دریافت عدد به منظور ادامه حلقه
     }
}
```

کلاس چیست؟

روند برنامه‌نویسی درگذشته به صورتی بود که همه‌ی برنامه‌ها را یکجا می‌نوشتند. این نوع برنامه‌نویسی دو ایراد بزرگ داشت. یکی اینکه اگر بخواهیم قسمتی از برنامه را تغییر دهیم بایدکل برنامه دستکاری شود. مثلا تغییر یک فرمول در قسمتی از برنامه، کل برنامه را به هم می‌ریزد. دوم اینکه گاهی، کارهای تکراری در کل برنامه دائم تکرار می‌شود.

فرض‌کنید قرار است یک برنامه ماشین حساب بنویسید. اگر همه‌ی دستورات را پشت سر هم بنویسید مطمئنا برنامه‌ی شما شلوغ و گیج‌کننده می‌شود. و دائم باید

دستورات تکراری مثل جمع و تفریق و ضرب و تقسیم را تایپ کنید. مطمئنم فکرکردن هم، به این مسئله‌عذاب آور است. اما برای هر مشکلی راه‌حلی وجود دارد.

روند برنامه‌نویسی همین‌طور ادامه نیافت. برنامه‌نویسی به سمت پیمانه شدن پیش رفت. یعنی‌کد و داده کنار هم قرارگرفتند و مبحث کلاس‌ها را به وجود آوردند. حالا دیگر می‌توانیم در هرکلاس چندین توابع تعریف کنیم و هر تابع را در هر جایی که خواستیم مورد استفاده قرار دهیم. در مثال بالا کافیست یک بار هر عمل را تعریف کرده و جاهای مختلف فقط فراخوانی کنیم.

یک مثال در دنیای واقعی:

مثلا زمانی‌که شما می‌خواهید میوه‌ای بخورید، به کارد، بشقاب و میوه نیازمندید. که هرکدام از این‌ها جایی در آشپزخانه قرار گرفته. دیگر شما نیاز ندارید اول کارد، بعد بشقاب، و بعد میوه را درست کنید. چون یک بار این‌ها درست شدن و شما در کارهای مختلف از آن‌ها استفاده می‌کنید

کلاس‌ها هم بخاطر این مسئله به‌وجود آمدن. در کلاس توابعی تعریف می‌شود، تا کارهای مختلفی را انجام دهد. بیرون کلاس برای هرتابع دستورات نوشته می‌شود و ما در بدنه اصلی، فقط توابع را فراخوانی می‌کنیم و دیگر نیازی به نوشتن دوباره کدها در هرجایی نداریم.

شکل کلی کلاس:

```
Class       نام کلاس//
{
Public:
int x ;تعریف متغیر//
void tavan (void)
};
```

کلمه ***Public*** به مفهوم عمومی بودن متغیر و توابع می‌باشد. و هر جایی قابل استفاده است. کلمه ***void*** به مفهوم نداشتن است. یعنی این تابع ورودی یا خروجی ندارد. و فقط برای کاری ساده مورد استفاده قرار می‌گیرد. البته توابع براساس کارکردشان می‌توانند ورودی و خروجی داشته باشند. که در مباحث آینده به آن می‌پردازیم.

نحوه نوشتن دستورات داخل توابع:

```
voidنام کلاس: : نام تابع(void)
{
دستورات
}
```

نحوه استفاده در بدنه:

```
void main()
{
    classname  x          با این متغیر به توابع آن کلاس دسترسی داریم//
```

```
    x.نام تابع ();

}
```

هر تابع می‌تواند شرایط زیر را داشته باشد.

- **نه ورودی نه خروجی**

در این‌حالت تابع نه ورودی دارد نه خروجی. البته نباید فکر کنید که نمی‌توانیم از دستور دریافت و چاپ استفاده کنیم.

```
void نام تابع (void)
```

این نوع تابع‌ها که نه ورودی دارند نه خروجی، باید در قسمت ورودی و خروجی از کلمه کلیدی ***void*** استفاده کنیم.

مثلا تابعی برای چاپ پیغام *hello* :

این تابع نیازی به دریافت یا برگرداندن ندارد. بلکه فقط برای چاپ پیغامیست.

```
voidclassname::payam (void)
{

    cout<<"hello"
}
```

- **فقط ورودی**(مثلا نوع صحیح)

```
void   نام تابع (int x, int y)
```

اگر تابعی ورودی داشته باید در بدنه تابع فقط فرمول را بنویسیم و در بدنه اصلی برنامه ورودی را بگیریم.

مثلا تابعی برای جمع دوعدد:

```
void classname::jam (int x, int y)
{

     cout<<x+y;
}

void main()
{
     z classname;
     int x,y;
     cin>>x;
     cin>>y;
     z.jam(x,y);
}
```

در بدنه اصلی بعد از گرفتن دو ورودی، آن‌ها را در تابع جایگزین می‌کنیم.

- **فقط خروجی**

```
int نام تابع (void)
```

اگر تابعی فقط خروجی داشته باید دربدنه تابع، بعد از نوشتن فرمول با کلمه کلیدی ***return*** نتیجه را بازگردانی کنیم.

مثلا تابعی برای جمع دو عدد:

```
int classname::jam ()
{
     int x,y;
     cin>>x;
     cin>>y;
return (x+y);
}

void main()
{
     z classname;
     cout<<z.jam();
}
```

- **هم ورودی هم خروجی**

```
int نام تابع (int x)
```

در اینجا باید درون تابع عملیاتی را انجام دهیم و در خروجی آن‌ها را بازگردانی کنیم. درون بدنه برنامه هم، ورودی‌ها را دریافت و چاپ کنیم.

برنامه کنترل دما

دمای هوای محیط را **دریافت** و با توجه به دمای دریافتی **خروجی مناسب** دهد.

دمای بین ۰ تا ۱۰ = سرد

دمای بین ۱۱ تا ۲۰ = خوب

دمای بین ۲۱ تا ۳۰ = گرم

راه حل:

- در این سوال نیاز به ورودی داریم. چون باید یک بار ورودی را بگیریم و در جاهای مختلف استفاده کنیم. پس موقع تعریف تابع برای آن ورودی تعیین می‌کنیم و در بدنه اصلی این ورودی را با ***cin*** دریافت کرده و در تابع جایگزین می‌کنیم.

کد:

```
#include <fstream.h>
     Class dama      //کلاس دما
```

```
{
    Public:
    void temp(int a);   //تابع دما
};
void dama::temp(int a)    //استفاده از تابع .این تابع دارای ورودیست
{
    if (a>0 && a<=10)    //اگر ورودی کمتر از ۱۰و بیشتر از صفر بود
    {
        cout<<"cold";//چاپ کن سرد
    }
    elseif(a>=11   && a<=20) وگرنه اگر ورودی بیشتر از ۱۱ و کمتر از ۲۰
//بود
    {
        Cout<<"good";//چاپ کن خوب
    }
    elseif(a>=21   && a<=30) وگرنه اگر ورودی بیشتر از ۲۱ و کمتر از ۳۰
//بود
    {
        cout<<"hold";//چاپ کن گرم
    }
}
void main()
{
    int t=1;   //تعریف متغیر برای حلقه شرطی
    int y;
    Dama x;//تعریف متغیری از نوع کلاس برای دسترسی به توابع آن
    While(t!=0)//استفاد از متغیر بالا برای شرط توقف
    {
        Cin>>y;//دریافت عدد
        x.dama(y);//استفاده از عدد در تابع دما
    }

}
```

- در بدنه یک متغیر تعریف کردیم تا زمانی که کاربر عددی غیـر صـفر وارد کرد برنامه ادامه داشته باشد، همین‌که صفر وارد کرد برنامه خاتمه یابد.

- سپس متغیری از نوع کلاس برای دسترسی به توابع آن قرار دادیم.

- و درآخر متغیری دیگر برای ورودی تابع نوشتیم.

- حالا باید در حلقه شرطی، توقف را با زدن صفر بررسی کنیم و بـا دسـتور ***Cin>>*** متغیر عددی را دریافت و در تابع موردنظر استفاده کنیم.

برنامه:

کلاسی طراحی‌کنید دو عدد **دریافـت** و**توابـع جمـع، تفریـق، ضـرب** را پیاده‌سازی کند.

کد:

```
#include <fstream.h>
     class calc          // کلاس ماشین حساب
     {
          Public:
          int x;     //تعریف متغیر برای دریافت عدد
          int y;     //تعریف متغیر برای دریافت عدد
```

```
        void jam(void);          //تابع جمع
        void zarb (void);  //تابع ضرب
        void tafrigh(void);      //تابع تفریق
        void daryaft (void);          //تابع دریافت
        void menu (void);  //تابع چاپ منو
};
void calc:: menu(void)
{
        cout<<"1-jam"<<"2-zarb"<<"3-tafrigh";//چاپ منو
}
void calc:: daryaft(void)
{
        cout<<"adad aval ra vared konid";
        cin>> x;        // دریافت عدد


        cout<<"adad dovam ra vared konid";
        cin>>y;         //دریافت عدد
}
void calc:: zarb(void)
{
        cout<<x*y;      //چاپ ضرب دو  عدد دریافتی بالا
}
void calc:: jam(void)
{
        cout<<x+y;      //چاپ جمع دو  عدد دریافتی بالا
}
void calc:: tafrigh(void)
{
        cout<<x-y;      //چاپ تفریق دو  عدد دریافتی بالا
}
void main()
{
        calc a;    //تعریف متغیری از نوع کلاس
        char y=0; //برای انتخاب عددمنو
        a.menu(); //فراخوانی تابع منو
```

```
        a.daryaft();          //قبل ازانجام عملیات دو عدد دریافت می شود
        a=getch();      //ابتداعدد منورامیزنیم
        do
        {
              if(a=='1')//اگر عدد ۱ زده شود عملیات جمع
                    y.jam ();
              elseif(y=='2')//وگرنه اگر عدد ۲ زده شود عملیات ضرب
              {
                    y.zarb ();
              }
              elseif(y=='3')//وگرنه اگر عدد ۳ زده شود عملیات تقسیم
              {
                    y.tafrigh();
              }
              getch();//این دستوربرای دریافت عدد بعدیست
              }(while=='5');//شرطپایان
}
```

نکته ! حلقه *do{ }while()* هم یک حلقه شرطیست که شرط در انتها بررسی می‌شود.

برنامه:

کلاسی طراحی کنید:

۱-تابعی‌که دو عدد را **دریافت** و جمع آن را **برگرداند**.

۲-تابعی که دو عدد را **دریافت** و عدد بزرگتر را **برگرداند**.

```
#include <fstream.h>
     class omome    //کلاس عمومی
     {
          public:
          int  jam(int x, int y);//تابع جمع دارای ورودی و خروجی
          int  max(int x, int y);//تابع عدد بزرگتر دارای ورودی و خروجی
     };
     int omome::jam(int x, int y)
     {
          return(x+y);          //باز گردانی جمع دو عدد
     }
     int omome::max(int x, int y)
     {
          return((x>y)?x:y); //بازگردانی بزرگترین عدد
     }
     void main()
     {
          int a;     //تعریف متغیر  برای دریافت عدد
          int b;     //تعریف متغیر برای دریافت عدد
          omome c    //تعریف متغیراز نوع کلاس
          cin>>a>>b;//دریافتدوعدد
          cout<<c.jam(a,b);  //چاپبازگردانیدوعدد
          cout<<c.max(a,b);//چاپ باز گردانی بزرگترین عدد
}
```

اشاره‌گرچیست؟

برای دسترسی مستقیم به حافظه از اشاره‌گرها استفاده می‌کنیم.

تعریف اشاره‌گر:

```
int *c;
char *c;,...
cin>> c;          یک رشته می‌خواند
cin>>*c;          یک حرف می‌خواند
```

برای دریافت رشته از اشاره‌گر استفاده می‌کنیم و تعریف آن به صورت زیر است.

```
char *c;
c=new (char);
```

- در خط اول اشاره‌گری تعریف می‌کنیم
- و درخط بعد حافظه‌ای برای این اشاره‌گر تخصیص می‌دهیم.

برنامه:

تابعی بنویسید که رشته ای را **دریافت** کند.

کد:

```
void daryaft::reshteh(void)
```

```
{
	char *c;		تعریف اشاره گر//
	c=new(char);		آزاد سازی فضای حافظه//
	cin>>c;		دریافت رشته//
}
```

برنامه:

تابعی بنویسید رشته‌ای دریافت و حروف آن را آن بشمارد.

کد:

```
void daryaft::reshteh(void)
{
	char *c;		تعریف اشاره گر//
	c=new(char);		آزاد سازی فضای حافظه//
	cin>>c;		دریافت رشته//
	int i=0;		تعریف متغیر برای حلقه شرطی//
	while(*(c+i)!='\0')تازمانیکه متن بهپایاننرسد
	{
		i++;// شمارش
	}
	cout<<i;//چاپ تعداد
}
```

تابع سازنده چیست؟

تابعی هم نام با نام کلاس، که مقداردهی اولیه‌ی هر چیز را داخل آن قرار می‌دهند و قبل از اجرا برنامه به صورت خودکار اجرا می‌شود.

برنامه:

سیستم کنترل سرعت

دریافت **نوع** و **سرعت** وسیله

سرعت **بالای ۶۰** کیلومتر تخلف

تعداد وسایل نقلیه براساس نوع

کد:

```
#include <fstream.h>
     class soarat          //کلاس سرعت
     {
          public:
          soarat(); //تابع سازنده
          void menu(void);    //تابع منو
```

```
        void daryaft(void);              //تابع دریافت
        void takhalof(void);             //تابع تخلف
};
soarat::soarat()تابع سازنده برای مقداردهی  اولیه
{
        m=0;
        a=0;
        d=0;
}
void soarat::daryaft(void)
{
        int s;      //تعریف متغیر برای دریافت عدد سرعت
        char v;     //تعریف متغیر برای دریافت وسیله
        cout<<" vaziat";            //چاپ وضعیت
        cin>>s;     //دریافت سرعت
        if(v=='m'&& s>60)//اگر وسیله موتور و سرعتش بیشتر ۶۰بود
                m=m+1;//شمارنده موتور
        elseif(v=='a'&& s>60)//وگرنه وسیله پیکان و سرعتشبیشتر ۶۰بود
        {
                a=a+1;//شمارنده پیکان
        }
        elseif(v=='b && s>60)//وگرنه وسیله سنگین و سرعتش بیشتر ۶۰بود
        {
                b=b+1;//شمارنده ماشین سنگین
        }
voidsoarat::takhalof(void)
{
        Cout<<"motor"<<m;   //چاپ موتور و تعداد آن
        Cout<<"paykan"<<a;  //چاپ پیکان و تعداد آن
        Cout<<"sangin"<<b;  //چاپ سنگین و تعداد آن
}
voidsoarat::manu(void)  //تابع چاپ منو
{
        Cout<<"1-daryaft";
        Cout<<"2-takhalof";
```

```
}
voidmain()
{
	Soarat x; //تعریف متغیری از نوع کلاس
	Char z;	//برای انتخاب عدد منو
	do{
		x.menu(); //فراخوانی تابع منو
		z=getch();	//ابتداعدد منورامیزنیم
		if (z=='1')	//اگر عدد ۱ زده شود عملیات دریافت
			x.daryaft();
		elseif(z=='2')//وگرنه اگر عدد ۲ زده شود عملیات تخلف
		{
			x.takhalof();
		}
		Getch();
	}while(z!='3');
}
```

فصل پنجم

خواندن و نوشتن در فایل

هدف‌های رفتاری:

- ✓ با فایل‌ها آشنا می‌شوید.
- ✓ قادر به نوشتن و خواندن در فایل خواهید بود.

گام‌های نوشتن و خواندن از فایل

- برای نوشتن و خواندن‌از فایل باید ابتدا اشاره‌گری از نوع فایل تعریف کنیم. این اشاره‌گر از هدر ***#include<stdio.h>*** استفاده می‌کند.

- برای نوشتن اشاره‌گری از نوع فایل باید بعد از نوشتن هدر مربوطه:

  - از کلمه کلیدی ***FILE*** استفاده کنیم. (توجه کنید که حتما این کلمه کلیدی را با حروف بزرگ تایپ کنید.)

  - سپس یک * وارد و نام مورد نظر را تایپ و در آخر ; بگذاریم.

```
FILE *f;
```

- حالا نوبت ایجاد فایل مورد نظر در مسیر دلخواه با دستور ***fopen*** است.

در دستور fopen:

```
f=fopen("d:\\x.txt","a");
```

f=اشاره‌گر فایل

مسیر فایل "d:\\x.txt" :

""پارامتر تعیین‌کننده خواندن و نوشتن که

- ***w*** یعنی نوشتن
- ***r*** یعنی خواندن
- ***a*** اضافه کردن به انتهای فایل(اگر از این پارامتر استفاده کنیم هر بار که حروفی وارد می‌کنیم به انتهای حرف‌های قبلی اضافه می‌شود.)

اگر در فایل(اگر از***w*** استفاده کنیم با هر بار اجرا فایل از اول نوشته می‌شود) از ***r***+یا ***w***+ استفاده شود هم می‌توان نوشت هم می‌توان خواند.

- بعد از اینکه دستورات را برای نوشتن یا خواندن تایپ کردید باید فایل مورد نظر را(به منظور خراب نشدنش) ببندید. دستور بستن فایل در زیر قرار گرفته.

```
fclose(f)
```

گام‌های نوشتن

بعد از انجام مراحل اولیه، حالا نوبت تعریف متغیری از نوع کاراکتر است که بتوانیم ورودی را از کاربر بگیریم.

چون نیاز به گرفتن کاراکتر به کاراکتر را داریم باید از دستور ***getche();*** استفاده کنیم.(این دستور در هدر ***#include <conio.h>*** است.) پس نیاز است متغیری از نوع ***char*** تعریف کنیم.

```
char c;

   c=getche()
```

برای نوشتن در فایل از دستور ***putc(c,p);*** استفاده می‌کنیم که ***c*** همان حرف، و ***f*** فایلیست که در آن می‌نویسیم.

برنامه نوشتن در فایل:

```
#include <fstream.h>
#include <stdio.h>
#include <conio.h>
     void main()
     {
          FILE *p;          //تعریف اشاره گری ازنوع فایل
          char c;           //تعریف متغیری از نوع رشته
          p=fopen("c:\\f.txt","w");//باز کردن فایل برای نوشتن در آن
          c=getche();//دریافت اولین حرف
          while(c!='\r')//شرط توقف: زدن اینتر
          {
               putc(c,p);// نوشتن حرف در فایل
               c=getche();//دریافت حرف بعدی
          }
          fclose(p);//بستن فایل
          getch();// زدن یک حرف برای پایان

}
```

- در این برنامه چون نیاز به گرفتن تک‌تک حروف داریم و از طرفی نمی‌دانیم که کاربر چه تعداد حروف وارد می‌کند پس باید از یک حلقه شرطی استفاده کنیم.

- من قبل از حلقه ابتدا کاراکتری را دریافت کردم.

- شرط توقف این حلقه زدن اینتر است.

- بعد از اجرای حلقه اولین بار اولـین کـاراکتر را درون فایـل مـی‌نویسـد و سپس کاراکتر بعدی را دریافت می‌کند.

- و این روال همین طور ادامه می یابد.

- در آخر هم فایل بسته می‌شود.

برنامه خواندن از فایل :

برای خواندن از فایل هم بعد از دستورات اولیه که بالاتر توضیح داده شد، باید بـا استفاده از دستور ***c=getc(k);*** تک تک حرف‌ها را واکشی کنیم ***c***.متغیریسـت کـه اطلاعات فایل ***k*** در آن ریخته می‌شود. برای خواندن از فایـل هـم نیـاز بـه حلقـه شرطی داریم که انتهای فایل را بررسی کند.

```
#include <fstream.h>
#include <stdio.h>
#include <conio.h>
void main()
{
	FILE *k;//تعریف اشاره گری ازنوعفایل
	char c;//تعریف متغیری از نوع رشته
	k=fopen("c:\\f.txt","r");//باز کردن فایل برای نوشتن در آن
	c=getc(k);//دریافت اولین حرف
	while(c!=EOF)دستوربررسی انتهای فایل
	{
		cout<<c;چاپ کاراکتر
```

```
            c=getc(k);دریافت کاراکتربعدی
        }
    getch();// زدن یک حرف برای پایان
    fclose(k);//بستن فایل
}
```

برنامه:

برنامه‌ای بنویسیدکه فایلی را بخواند اگر حرف ***a*** درون آن بود در فایلی دیگر این حرف را با حرف ***b*** جایگزین کند.

راه‌حل:

- در برنامه ما نیاز به دو اشاره‌گر داریم، یکی برای خواندن فایل اول، دومی برای نوشتن در فایل دوم .
- اولین حرف فایل اول را می‌خوانیم و در حلقه شرطی پایان فایل را بررسی می‌کنیم
- و سپس درون حلقه بررسی می‌کنیم اولین حرف دریافتی ***a*** هست یا خیر. اگر ***a*** بود حرف ***b*** را چاپ و ***b*** را در فایل دوم بنویس وگرنه همان حرف ***a*** را چاپ و در فایل دوم می‌نویسد.

کد:

```
#include <fstream.h>
#include <stdio.h>
#include <conio.h>
void main()
{
	FILE *k;//تعریف متغیری از نوع اشاره گر
	FILE *b;//تعریف متغیری از نوع اشاره گر
	char c;//تعریف متغیری از نوع رشته
	k=fopen("c:\\f.txt","r");//باز کردن فایل برای خواندن از آن
	b=fopen("c:\\f1.txt","w");//باز کردن فایل برای نوشتن در آن
	c=getc(k);//دریافت اولین حرف
	while(c!=EOF)دستوربررسی انتهای فایل
	{
		if(c=='a')// اگر حرف گرفته شده برابر حرف مورد نظر بود
		{
			cout<<'b';//حرف روبرو را چاپ
			putc('b',b);//حرف مورد نظر را در فایلمورد نظر بنویس
		}

	else//وگرنه
	 {
		putc(c,b);//حرف خوانده شده را بنویس
		cout<<c;//و چاپ کن
	 }
	 c=getc(k);دریافت کاراکتربعدی
	 }
	getch();// زدن یک حرف برای پایان
	fclose(k);// بستن فایل
	fclose(b);// بستن فایل

}
```

برنامه:

برنامه‌ای بنویسید تعداد حروف فایلی را بخواند و چاپ کند.

راه‌حل:

- در برنامه کافیست که در حلقه، یک شمارنده بگذاریم تا بعد از خواندن هر حرف آن را بشمارد.

کد:

```
#include <fstream.h>
#include <stdio.h>
#include <conio.h>
void main()
{
	FILE *k;//تعریف اشاره گری ازنوع فایل
	char c;//تعریف متغیری از نوع رشته
	int x=0;// تعریف متغیری از نوع عدد برای شمارش
	k=fopen("c:\\f.txt","r");//باز کردن فایل برای خواندن
	c=getc(k);//دریافت اولین حرف
	while(c!=EOF)دستوربررسی انتهای فایل
	{
		c=getc(k);دریافت کاراکتربعدی
		x=x+1;//شمارش کارکتر
	 }
	cout<< x;چاپ کاراکتر
```

```
    getch();// زدن یک حرف برای پایان
    fclose(k);// بستن فایل
    fclose(b);// بستن فایل

}
```

آموزش کامپیوتر مخصوص خانم ها

با تشکر ازشما دوست عزیز که هم اکنون این کتاب را برای خواندن انتخاب کردید. من زهرا بیات موسس آموزشگاه ایده پرداز (در گلستان رباط‌کریم)هستم.

هر کس در زندگی هدفی دارد، اما بهترین هدف ها ، متعالی ترین آنهاست. هدف من از تاسیس ایده پرداز ۲ دلیل داشت:

۱-کمک به **خانم‌ها و کودکانی** که با توجه به سن یا محدودیت مالی **نمی‌توانند** در کلاس‌های آموزشگاه‌ها شرکت کنند .

۲-آموزش نرم افزارهای به روزی که باز هم به دلیل کمبود امکانات بعضی آموزشگاه‌ها تدریس نمی‌شود. یا با هزینه گزاف تدریس می‌شود. و عده ای که طالب این آموزش‌ها هستند از علم رایانه‌ای عقب می مانند. من در این آموزشگاه با به روز ترین رایانه ها و تجهیزات ،به روزترین نرم افزار ها را با قیمت مناسب آموزش می‌دهم.

تخصص من در انواع زمینه های کامپیوتری از جمله:

نرم افزار های اداری(Word, Excel, Access, Powerpoint, Onenote ,...) گرافیکی(فتوشاپ، کورل، فری هند، میکس و مونتاژ، فلش وتری دی مکس..) برنامه نویسی (سی شارپ، Asp.net,Vb,c++,mvc,...)و...

دوستان عزیز ،فقط برای ثبت نام می‌توانید با شماره‌های زیر ارتباط برقرار کنید . و سوالات برنامه‌نویسی خود را به ایمیل من بفرستید تا در اسرع وقت پاسخگو باشم.

آدرس: **جاده ساوه،گلستان ، فلکه اول ،۲۴ متری ارغوان غربی، خیابان ولایت ،پلاک ۳۲**

Email: Bytmohandes@yahoo.com

Tel:۰۹۳۷۳۷۱۹۴۰۵ -۰۹۳۹۳۹۳۹۹۲۴

بهار ۹۲