

In The Name of Allah



برنامه نویسی پیشرفته



ساختار درس در پایان ترم

برنامه نویسی به زبان C

مبنای امتحان، مطالب ذکر شده در کلاس می باشد.

مرجع کامل دستورات و ساختار برنامه نویسی C در کتاب «برنامه نویسی به زبان C»، مولف: «عین الله جعفر نژاد قمی» نیز قابل دسترس می باشد.

محیط برنامه نویسی

Visual Studio .NET

Dev C++

فهرست مطالب

- مقدمه

- تاریخچه زبان
- ویژگی های زبان
- تعریف متغییر و ثابت
- انواع داده ها
- انواع عملگرها

- برنامه نویسی در زبان C

- ساختار کلی برنامه به زبان C
- دستور ورودی و خروجی
- ساختارهای شرط
- ساختارهای حلقه
- حلقه های تو در تو
- تابع
- آرایه
- ماتریس
- آرایه های کاراکتری یا رشته ها

تاریخچه زبان C

- در سال 1970 در آزمایشگاه شرکت Bell زبان C ارائه گردید
- در اوایل 1980 در شرکت Bell زبان C++ را با اضافه کردن بحث شی گزایی به زبان C ارائه نمود

ویژگی های زبان C

- زبان C یک زبان سطح بالا و همه منظوره است که دارای قابلیت های یک زبان سطح پایین مانند دستیابی مستقیم به حافظه، کار کردن با بیت، بایت و آدرس را داراست.
- برنامه های نوشته شده به زبان مستقل از ماشین بوده و بر روی هر کامپیوتری اجرا می شوند
- زبان C یک زبان ساخت یافته است به این معنی که پرش ندارد و هر برنامه توسط ساختار ترتیب، انتخاب و تکرار قابل نوشتن است.
- زبان C یک زبان حساس به متن یا Case Sensitive است بدین معنی که حروف کوچک و بزرگ در آن متفاوت در نظر گرفته می شوند. مثلا for با For یکسان نیست.

کلمات کلیدی در C

- کلمات کلیدی یا keywords به کلماتی گفته می شود که کامپایلر آن ها را می شناسد و معنای خاصی برای آن زبان دارند

- کلمات کلیدی در زبان C مانند

| | | | | | |
|-------|-------|--------|--------|-------|-----------|
| int | float | double | char | for | while |
| if | else | struct | switch | case | otherwise |
| break | | void | | const | main |

قانون نام گذاری شناسه ها

- قانون نام گذاری شناسه ها (اسم متغیرها، ثابت ها و توابع) در زبان C
 - اسم شناسه ها می تواند ترکیبی از حروف الفبا (A...Z, a...z) یا اعداد (0...9) و یا خط ربط ('_' underline) باشد با این شرط که اسم شناسه با عدد شروع نشود
- نکته: برای اسم گذاری از کلمات کلیدی نباید استفاده شود. و حداکثر طول یک شناسه 31 کاراکتر می باشد

• مثال

- نام گذاری های مجاز

Ali Zab A2 Sum_2 AB_C3

- نام گذاری های غیر مجاز

Ali@B AB.C A*B main SUM+ A-1 C#A
2Avg My Avg "AB"

متغیر

- متغیر به مکانی از حافظه گفته می شود که می تواند مقداری را به خود اختصاص داده و این مقدار می تواند در طول اجرای برنامه تغییر کند
- نام گذاری متغیرها از قواعد نام گذاری شناسه ها پیروی می کند.
- در زبان C هر متغیر باید قبل از استفاده تعریف شود تا مقادیر و اعمال مجاز روی آن متغیر مشخص گردد.

انواع داده ها در زبان C

| نوع داده | حافظه | مقادیر |
|----------|--------|--|
| char | 1 بایت | یک کاراکتر مانند 'a' ، '9' ، 'B' ، '@' |
| int | 2 بایت | 32767 تا -32768 |
| float | 4 بایت | 3.4e38 تا 1.2e-38 |
| double | 8 بایت | 1.8e308 تا 2.2e-308 |
| void | --- | کاراکتر تهی |

- می توان گفت که نوع char یک int یک بایتی می باشد به این معنی که تمامی اعمالی که برای int قابل استفاده است برای char نیز می تواند استفاده شود!

انواع داده ها در زبان C

- می توان با استفاده از کلمات signed/unsigned و همچنین short/long به همراه انواع داده های در زبان C، نوع های جدیدی مانند جدول زیر ایجاد نمود.

| نوع داده | حافظه | مقادیر |
|-------------------|--------|---------------------------|
| unsigned char | 1 بایت | 0 تا 255 |
| unsigned int | 2 بایت | 0 تا 65535 |
| long int | 4 بایت | -2147483648 تا 2147483647 |
| unsigned long int | 4 بایت | 0 تا 4294967295 |

تعریف متغیر

- تعریف متغیر:

<نام متغیر> <نوع متغیر>;

همه دستورات در زبان C به سمیکالون ; ختم می شوند

- مثال:

- تعریف متغیر myCh از نوع کاراکتر

```
char myCh;
```

- تعریف متغیر X از نوع int

```
int X;
```

- تعریف متغیر p و q از نوع float

```
float p, q;
```

مقداردهی اولیه به متغیر

● پس از تعریف با دستور انتساب

```
int x;  
int sum, z;  
char ch1, ch2;  
float p;
```

```
x = 5;  
sum = 4;  
z = -3;  
ch1 = 'A';  
ch2 = 'b';  
p = 3.14;
```

● هنگام تعریف متغیر

```
int x=5;  
int sum=4, z=-3;  
char ch1='A', ch2='b';  
float p=3.14;
```


ثابت ها یا Constant

- ثابت ها در زبان C مقادیری هستند که در برنامه تعریف می شوند و مقدار آن ها در طول اجرای برنامه قابل تغییر نیست. اگر در طول اجرای برنامه بخواهیم مقدار یک ثابت را تغییر دهیم کامپایلر خطا خواهد داد
- نام گذاری ثابت ها از قواعد نام گذاری شناسه ها پیروی می کند
- فایده استفاده از ثابت: (1) جلوگیری از خطای برنامه نویس، (2) خوانایی در برنامه (3) اگر بخواهیم مقدار یک ثابت را عوض کنیم تنها لازم است در هنگام تعریف مقدارش تصحیح گردد و تغییر بقیه دستورات برنامه لازم نیست.

تعریف ثابت:

Const <مقدار ثابت> = <نوع ثابت> <نام ثابت>

● مثال:

```
const float p = 3.14;  
const int Len = 100, Num = 200;
```


انواع عملگرها در زبان C

- عملگر به نمادهایی گفته می شود که اعمال خاصی را روی عملوندهای خود انجام می دهند. (عملوند به مقادیری گفته می شوند که عملگرها بر روی آن ها عمل می کنند)

● انواع عملگرها

- عملگرهای محاسباتی
- عملگرهای جایگزینی محاسباتی
- عملگرهای رابطه ای (مقایسه ای)
- عملگرهای منطقی
- عملگرهای بیتی (در این درس به آن نمی پردازیم)

عملگر انتساب

- عملگر انتساب یا جایگزینی برای مقداردهی به یک متغیر می باشد. که در زبان C بصورت $=$ نشان داده می شود.
- پیش از این در بخش الگوریتم و فلوچارت، عملگر انتساب را بصورت \leftarrow نشان دادیم. از این به بعد از عملگر $=$ برای این منظور استفاده خواهیم کرد.
- توجه عملگر انتساب را با عملگر تساوی در زبان C که بصورت $==$ نشان داده می شود، به اشتباه به کار نبرید!
- مثال:

$x = 5;$

$sum = x + 2 * y + 8;$

$Z = Y = F = 2;$

انتساب همزمان 2 به سه متغیر Z، Y و F \leftarrow

عملگرهای محاسباتی

| مثال | نماد عملگر | عملگر |
|----------------|------------|---------------------------|
| $-a$ | $-$ | منهای یکانی |
| $a++$ یا $++a$ | $++$ | عملگر افزایشی (پلاس پلاس) |
| $a--$ یا $--a$ | $--$ | عملگر کاهشی (ماینس ماینس) |
| $a * b$ | $*$ | ضرب |
| a / b | $/$ | تقسیم |
| $a \% b$ | $\%$ | باقیمانده |
| $a + b$ | $+$ | جمع |
| $a - b$ | $-$ | تفریق |

• توجه: در زبان C عملگر توان نداریم

عملگرهای افزایشی و کاهشی

- عملگر ++ و -- تک عملوندی هستند.
- عملگر ++ یک واحد به عملوند خود اضافه می کند. به عبارت دیگر سه دستور زیر معادل یک دیگر هستند.

$X++;$

$++X$

$X = X + 1;$

- عملگر -- یک واحد از عملوند خود کم می کند. به عبارت دیگر سه دستور زیر معادل یک دیگر هستند.

$Y--;$

$--Y;$

$Y = Y - 1;$

نکته

- در عملگرهای ++ و --
- اگر عملگر قبل از متغیر ظاهر شود مانند ++X ابتدا مقدار متغیر اضافه می شود و سپس مقدار جدید در محاسبات شرکت داده می شود.
- ولی اگر عملگر بعد از متغیر ظاهر شود مانند X++ ابتدا X در محاسبات شرکت داده می شود و بعد از آن مقدار X یک واحد افزایش (یا کاهش) داده می شود.

```
X = 4;  
Y = ++X * 3;
```

در این جا بعد از اجرای دستورات
X=5 و Y=15 می شود

```
X = 4;  
Y = X++ * 3;
```

در این جا بعد از اجرای دستورات
X=5 و Y=12 می شود

• مثال:

مثال

- بعد از اجرای دستورات زیر هر متغیر چه مقداری خواهد داشت؟

$X = 3;$

$Y = 7;$

$Z = ++X * Y--;$

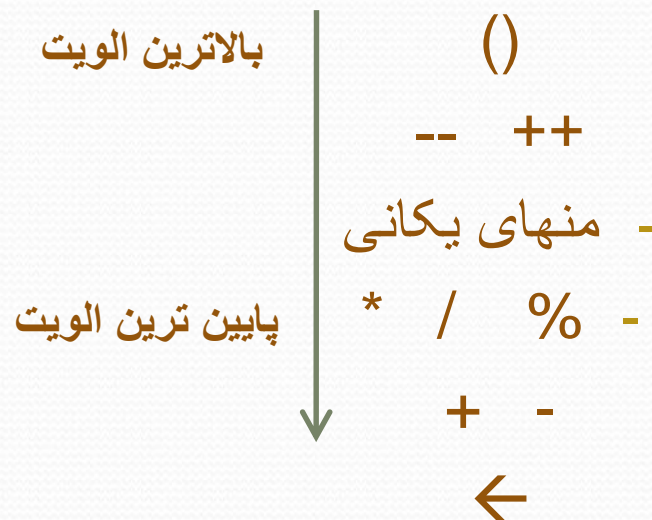
$X=?$

$Y=?$

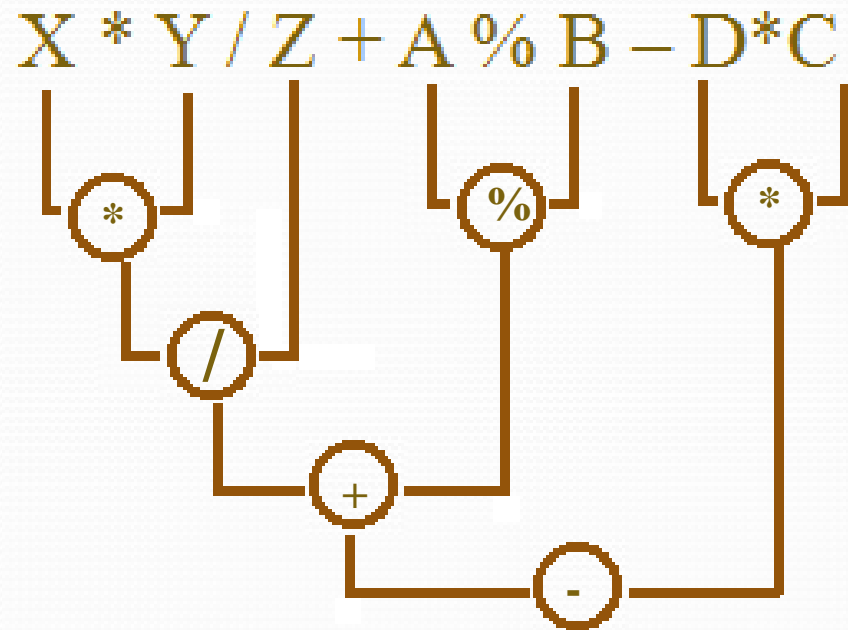
$Z=?$

اولویت عملگرهای محاسباتی

- اولویت عملگرهای محاسباتی بصورت زیر از بالا به پایین کاهش می یابد.
بعبارتی ابتدا باید پرانتزها ارزیابی شوند. سپس حق تقدم با $\%$ $*$ $/$ و بعد از آن حق تقدم با $+$ و $-$ است و در نهایت حق تقدم با عملگر انتساب است
- عملگرهایی که دارای اولویت یکسانی هستند مثل $+$ و $-$ از چپ به راست ارزیابی می شوند



مثال



$$(((X * Y) / Z) + (A \% B)) - (D * C)$$

عملگرهای جایگزینی محاسباتی

| مثال | نماد عملگر | عملگر |
|------------|------------|-----------------|
| $a *= b$ | $*=$ | ضرب مساوی |
| $a /= b$ | $/=$ | تقسیم مساوی |
| $a \% = b$ | $\% =$ | باقیمانده مساوی |
| $a += b$ | $+=$ | جمع مساوی |
| $a -= b$ | $-=$ | منها مساوی |

• عبارت $x += y$ معادل است با $x = x + y$

• عبارت $x *= 10$ معادل است با $x = x * 10$

• اولویت همه عملگرهای جایگزینی محاسباتی یکسان است

عملگرهای رابطه ای

- عملگرهای رابطه ای برای مقایسه دو عبارت یا عدد به کار می روند و حاصل آن ها True (درست) و یا False (نادرست است)

| عملگر | نماد عملگر | مثال |
|--------------|------------|----------|
| بزرگتر | > | $X > Y$ |
| بزرگتر مساوی | >= | $X >= Y$ |
| کوچکتر | < | $X < Y$ |
| کوچکتر مساوی | <= | $X <= Y$ |
| تساوی | == | $X == Y$ |
| نامساوی | != | $X != Y$ |

عملگرهای منطقی

- عملگرهای منطقی برای ترکیب چند عملگر رابطه ای استفاده می شوند
- سه عملگر منطقی بصورت زیر هستند

| مثال | نماد عملگر | عملگر |
|-------------------------------------|-------------|----------|
| $(a > b) \ \&\& \ (c < 5)$ | $\&\&$ | AND |
| $(a \leq 6) \ \parallel \ (c == b)$ | \parallel | OR |
| $!(b < 7)$ | $!$ | NOT نقیض |

عملگرهای منطقی

- حاصل عبارت های منطقی True یا False می باشد. فرض کنید X و Y دو عبارت منطقی باشند در این صورت داریم:

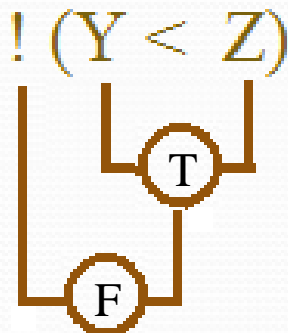
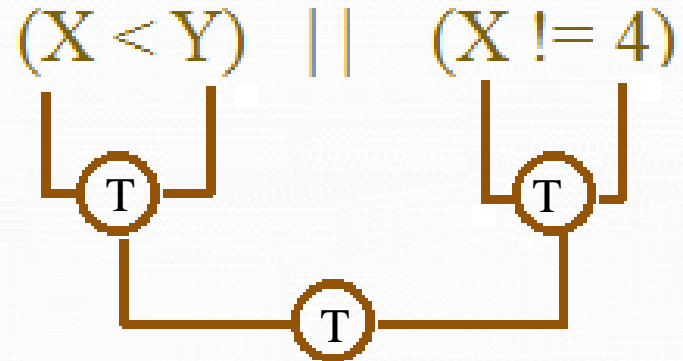
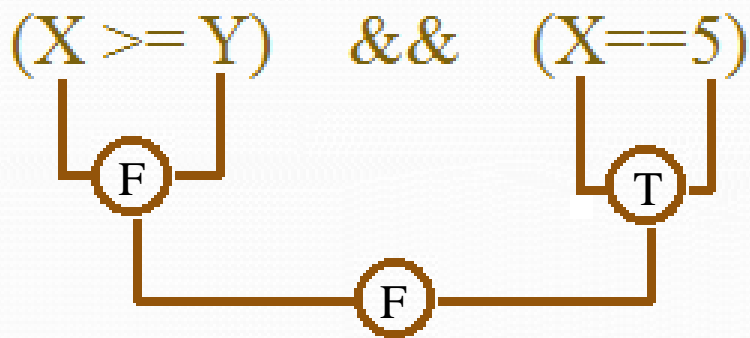
| X | Y | X && Y |
|---|---|--------|
| F | F | F |
| F | T | F |
| T | F | F |
| T | T | T |

| X | Y | X Y |
|---|---|--------|
| F | F | F |
| F | T | T |
| T | F | T |
| T | T | T |

| X | ! X |
|---|-----|
| F | T |
| T | F |

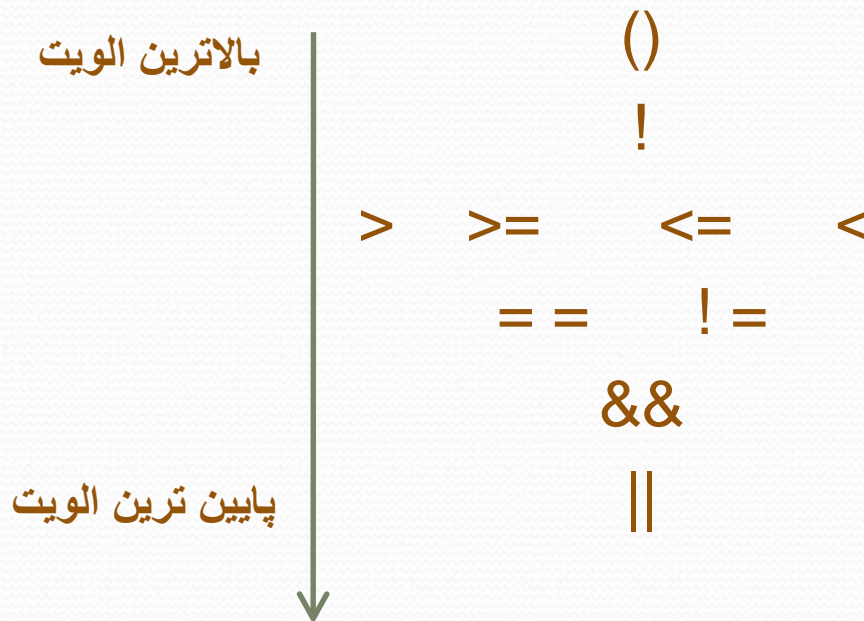
مثال

- فرض کنید که $X=5$ و $Y=7$ و $Z=10$ در این صورت حاصل عبارت های زیر بصورت زیر محاسبه می شود



اولویت عملگرهای رابطه ای و منطقی

- اولویت عملگرهای رابطه ای و منطقی بصورت زیر از بالا به پایین کاهش می یابد.
- عملگرهایی که دارای اولویت یکسانی هستند (در یک سطر نوشته شده اند) از چپ به راست ارزیابی می شوند



ساختار کلی برنامه زبان C

```
#include <نام فایل سرآیند>
int main( )
{
    ;تعریف متغیرها و ثابت ها
    ;دستورات برنامه
    return 0 ;
}
```

هر برنامه در زبان C یک و تنها یک تابع با نام main دارد

شروع اجرای یک برنامه از تابع main میباشد

هر متغیر و ثابتی که در برنامه استفاده میشود می بایست در ابتدای برنامه تعریف شود

کتابخانه ها

- توابع مورد نیاز برای نوشتن برنامه ها در زبان C و C++، مانند توابع ورودی و خروجی، در قالب فایل های کتابخانه ای دسته بندی شده اند. این فایل ها را فایل سرآمد یا Header می گویند و معمولاً با پسوند h ذخیره شده اند.
- برای استفاده از یک تابع که در کتابخانه خاصی وجود دارد، آن کتابخانه می بایست در ابتدای برنامه بصورت زیر include گردد.

#include <iostream>

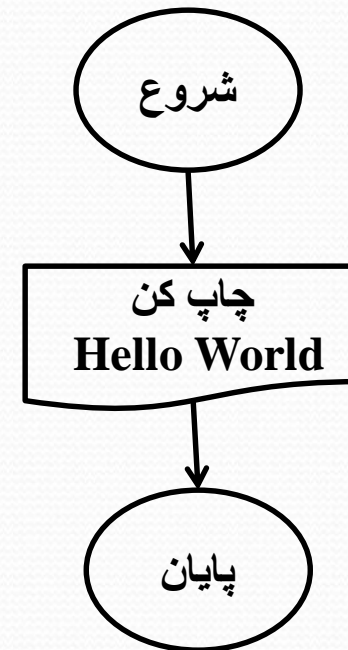
#include <math.h>

- مثلاً توابع ورودی و خروجی داخل کتابخانه ای با نام `iostream` می باشند.
- توابع ریاضی مانند تابع رادیکال `sqrt(x)` و یا تابع سینوس `sin(x)` داخل کتابخانه `math.h` می باشند.

مثال

- برنامه ای بنویسید که پیام "Hello World" را روی صفحه نمایش چاپ نماید.

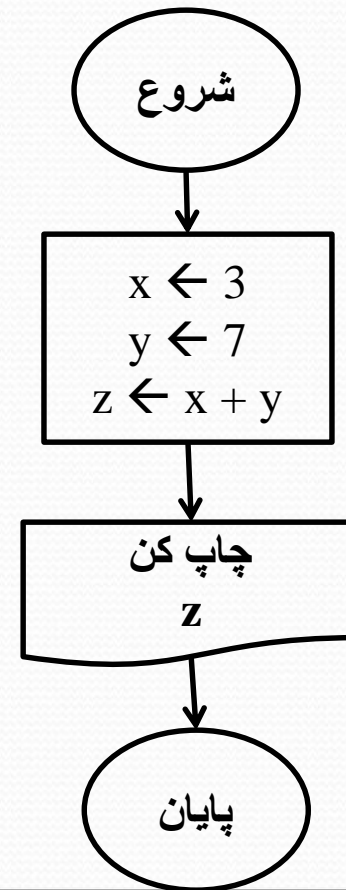
```
#include <iostream>
using namespace std;
int main( )
{
    cout << "Hello World!" ;
    return 0 ;
}
```



مثال

- برنامه ای بنویسید که دو مقدار 3 و 7 داخل دو متغیر از نوع صحیح قرار داده و حاصل جمع آن ها را محاسبه و چاپ کند.

```
#include <iostream>
using namespace std;
int main( )
{
    int x=3, y=7, z;
    z = x+y;
    cout << z ;
    return 0 ;
}
```



چاپ روی صفحه نمایش

- برای چاپ یک جمله یا مقدار بر روی صفحه نمایش در زبان C از دستور `printf` استفاده می شود که در کتابخانه `stdio.h` قرار دارد.
- تابع چاپ روی صفحه نمایش در زبان C++ تابع `cout` می باشد که در کتابخانه `iostream` و در `namespace` با نام `std` قرار دارد.
- در این درس برای سادگی بیشتر از تابع `cout` برای چاپ روی صفحه نمایش استفاده می کنیم.

دستور خروجی cout

- حالت کلی استفاده از دستور cout

؛ عبارت خروجی << cout

یا خروجی 3 << خروجی 2 << خروجی 1 << cout

به ترتیب خروجی ها را چاپ خواهد کرد

- عبارت خروجی در دستور cout سه حالت می تواند داشته باشد:
 - اگر عبارت خروجی یک متغیر باشد، مقدار متغیر در خروجی چاپ می شود
 - اگر عبارت خروجی یک عبارت محاسباتی باشد، حاصل عبارت محاسباتی در خروجی چاپ می شود.
 - اگر عبارت خروجی یک رشته یا string باشد، آن رشته عیناً در خروجی چاپ می شود.
(به کلمات یا جملاتی گفته که بین ” ” نوشته می شوند، مانند ”This is a string” یا ”one=1” رشته یا string گفته می شود)
- توجه: کاراکترها داخل ‘ ’ نوشته می شوند مانند ‘A’ یا ‘@’ و رشته ها داخل ” ” نوشته می شوند مانند ”Test” یا ”Hello world!”

دستور خروجی cout

```
int x=10, y=4;
```

```
cout << x; _____>
```

```
cout << x+y; _____>
```

```
cout << "Test x:" << x; _____>
```

```
cout << "x+y = " << x+y; _____>
```

```
cout << x << "+y = " << x+y; _____>
```

```
cout << x << " + " << y << " = " << x+y; _____>
```

خروجی

10

14

Test x:10

x+y= 14

10 + y = 14

10 + 4 = 14

خواندن از صفحه کلید

- برای خواندن یک مقدار از ورودی صفحه کلید در زبان C از تابع `scanf` استفاده می شود که در کتابخانه `stdio.h` قرار دارد.
- برای خواندن یک مقدار از ورودی صفحه کلید در زبان C++ از تابع `cin` استفاده می شود که در کتابخانه `iostream` و `namespace std` با نام `std` قرار دارد.
- در این درس برای سادگی بیشتر از تابع `cin` برای خواندن از ورودی صفحه کلید استفاده خواهیم کرد.

دستور ورودی cin

- حالت کلی استفاده از دستور cin

```
cin >> اسم متغیر ;
```

```
cin >> اسم متغیر 1 >> اسم متغیر 2 ;
```

مقادیر را به ترتیب از صفحه کلید خوانده و در متغیر مربوطه قرار می دهد

- توجه: مقدار متغیر وارد شده می بایست با نوع متغیر سازگاری داشته باشد مثلاً اگر نوع متغیر عدد صحیح (int) است ورودی نیز باید عدد صحیح وارد شود وگرنه مقدار مربوطه به درستی خوانده نخواهد شد.

```
int x, y;  
float testp;  
cin >> x >> y;  
cin >> testp;
```

- مثال:

مثال

- برنامه ای بنویسید که سه عدد صحیح را از ورودی دریافت کرده و مجموع آن ها را محاسبه و چاپ کند

مثال

- برنامه ای بنویسید که طول و عرض یک مستطیل را به عنوان ورودی دریافت کرده و محیط و مساحت آن را محاسبه و چاپ کند

توضیحات یا Comment

- در یک برنامه توضیحات یا comment به مطالبی گفته می شود که صرفاً جهت توضیح برنامه بوده و کامپایلر کاملاً از آن ها صرف نظر می کند.
- توضیحات برای افزایش خوانایی در برنامه نوشته می شوند.
- نحوه اضافه کردن توضیح به برنامه

روش اول: توضیح یک سطر

```
// comment
```

روش دوم: توضیح چند سطر

```
/*  
  
...  
comments  
  
....  
*/
```

مثال

- برنامه ای بنویسید که دو عدد اعشاری A و B را از کاربر دریافت کرده و محتویات آن دو را با یکدیگر جابجا نموده و سپس آن ها را چاپ نماید.

مثال

- برنامه ای بنویسید سه عدد صحیح را از ورودی دریافت کرده و میانگین آنها را محاسبه و چاپ کند.
- نکته: در صورتی که در یک عبارت محاسباتی همه عملوندها صحیح باشند، حاصل عبارت نیز صحیح خواهد بود. به عنوان مثال:

تقسیم صحیح

$$5/2 = 2$$

تقسیم اعشاری

$$5.0/2 = 2.5$$

$$5*1.0 / 2 = 2.5$$

$$5/2.0 = 2.5$$

ساختارهای شرط

دستور اگر (if)

دستور اگر...آنگاه (if...else...)

دستور انتخاب چند گزینه ای (switch...case)

ساختار شرط

• ساختار کلی اگر ... آنگاه ... (if)

if (شرط)

دستور 1

if (شرط)

{

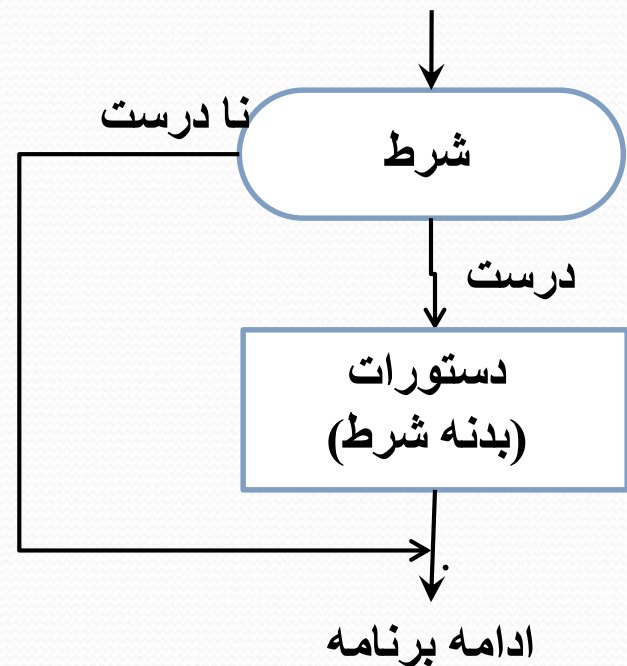
دستور 1 ;

دستور 2 ;

.....

}

اگر بدنه شرط بیش از یک دستور باشد،
حتماً باید دستورات را داخل {...} گذاشت



ساختار شرط

- ساختار شرط اگر... آنگاه ... وگرنه
(if..else)

if (شرط)

{

دستور 1

دستور 2

....

}

else

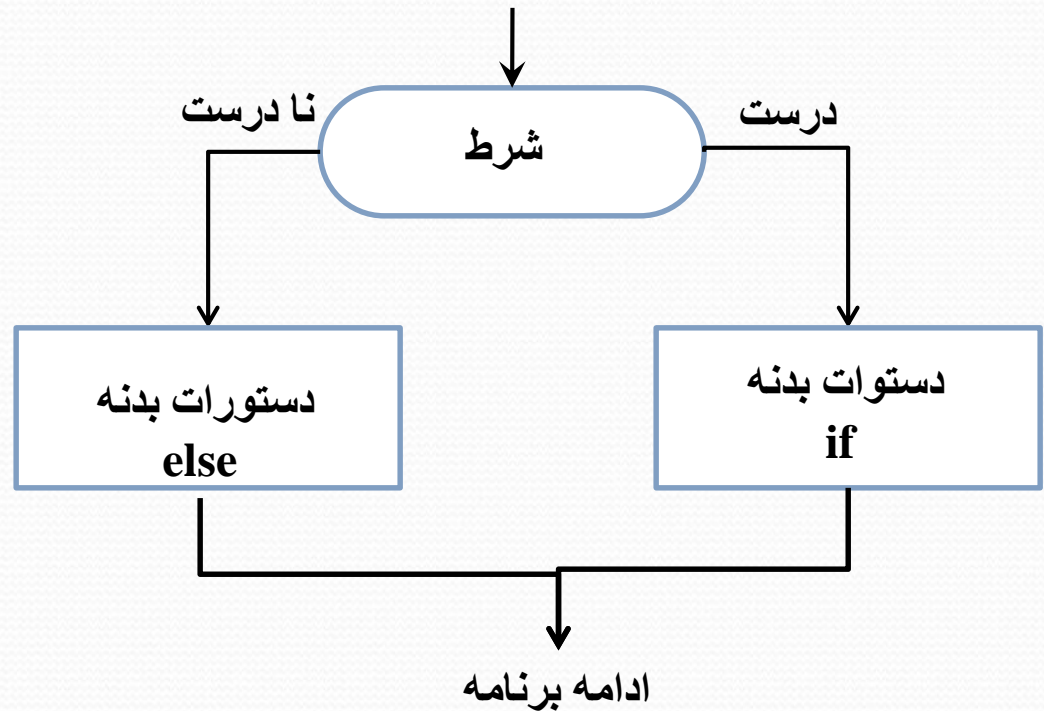
{

دستور 1

دستور 2

}

ادامه برنامه



مثال

- فلوجارتی رسم کنید که یک عدد را به عنوان ورودی دریافت کرده و مشخص کند آن عدد فرد است یا زوج. سپس برنامه آن را بنویسید.

مثال

- فلوجارتی رسم کنید که سه عدد را از ورودی دریافت کرده و ماکزیمم بین سه عدد را محاسبه و چاپ کند. سپس برنامه مربوط به آن را بنویسید.

مثال

- برنامه ای بنویسید که ضرایب یک معادله درجه دو را به عنوان ورودی دریافت کرده و ریشه های آن را محاسبه و چاپ کند.

switch(عبارت)

```
{  
    case <مقدار 1> :  
        دستور 1  
        دستور 2  
        break;  
    case <مقدار 2> :  
        دستورات  
        break;  
    case <مقدار 3> :  
        دستورات  
        break;  
    default:  
        دستورات  
}
```

ادامه برنامه

ساختار شرط switch ... case

- دستور switch برای چک کردن حالت تساوی بصورت چند انتخابی می باشد. استفاده می شود.

- ساختار کلی دستور switch

- توجه: دستور break باعث خروج از بدنه switch می شود و کنترل اجرای برنامه را به اولین دستور بعد از switch (ادامه برنامه) منتقل می کند

- اگر break گذاشته نشود چه اتفاقی می افتد؟

مثال

- برنامه ای بنویسید که یک عدد تک رقمی را از کاربر گرفته و معال حروفی آن را چاپ نماید. مثلا برای 0 عبارت “zero” چاپ شود.

تمرین

- برنامه ای بنویسید که یک کاراکتر و دو عدد صحیح A و B را از کاربر دریافت کرده و ماشین حساب ساده ای بصورت زیر پیاده سازی نماید.

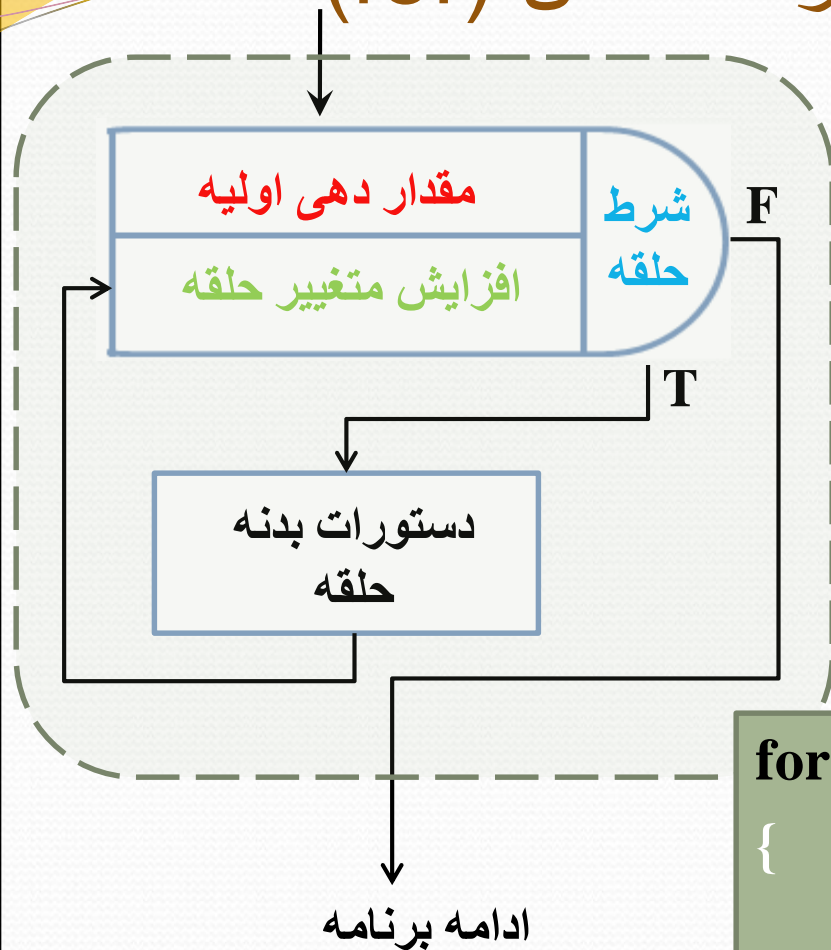
| عمل | کاراکتر |
|--------------|-----------------|
| $A+B$ | + |
| $A-B$ | - |
| $A*B$ | * |
| A/B | / یا \ |
| $A\%B$ | % |
| A به توان B | ^ |
| چاپ پیام خطا | در غیر این صورت |

حلقه ها

حلقه با تعداد تکرار مشخص (for)

حلقه با تعداد تکرار نامشخص (while)

ساختار حلقه با تعداد تکرار مشخص (for)



```
for( افزایش متغیر حلقه ; شرط حلقه ; مقدار دهی اولیه )  
{  
    دستورات بدنه حلقه  
    ....  
}  
ادامه برنامه
```


مثال

- مثال 1) فلوچارتی رسم کنید که عدد N را به عنوان ورودی دریافت کرده و سپس N عدد از کاربر دریافت کرده و مجموع آنها را محاسبه و چاپ کند. سپس برنامه مربوط به آن را بنویسید.
- مثال 2) سوال قبل را طوری تغییر دهید که مجموع اعداد زوج و فرد را بصورت مجزا محاسبه و چاپ کند.

مثال

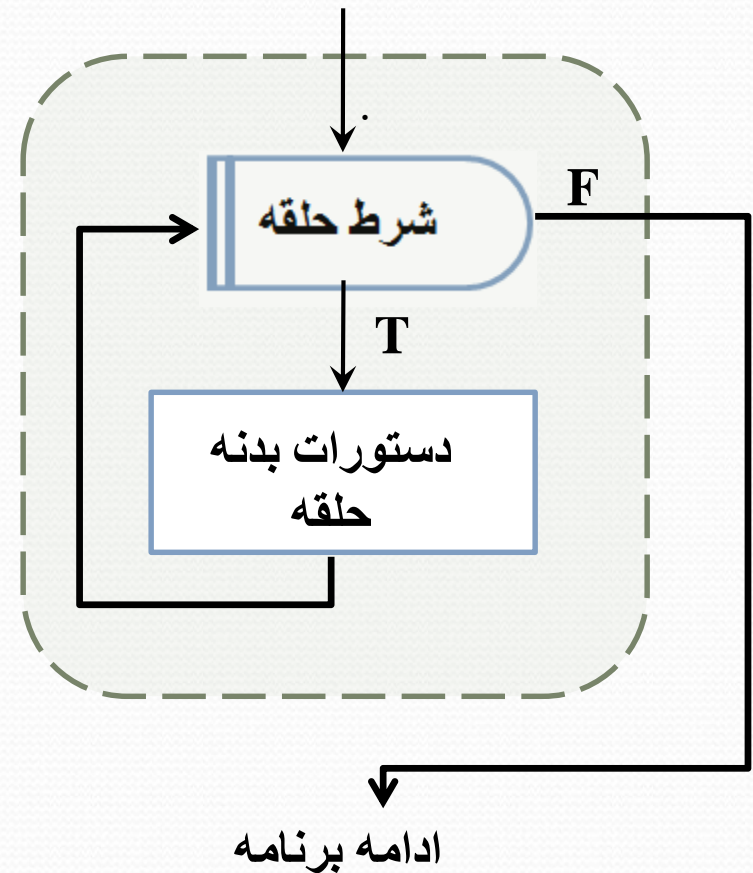
- فلوجارتی رسم کنید که عدد N را از کاربر دریافت کرده و فاکتوریل آن را محاسبه و چاپ کند. سپس برنامه مربوط به آن را بنویسید.

ساختار حلقه با تعداد تکرار نامشخص (while)

while (شرط حلقه)

```
{  
    ....  
    دستورات بدنه حلقه  
    ....  
}
```

ادامه برنامه



مثال

- فلوجارتی رسم کنید که عدد N را به عنوان ورودی دریافت کرده و تعداد ارقام آن را محاسبه و چاپ کند. سپس برنامه آن را بنویسید.

مثال

- فلوجارتی رسم کنید که یک عدد را از کاربر دریافت کرده و ارقام صفر آن را حذف کند به عنوان مثال عدد 3408 به عدد 348 تبدیل شود. سپس برنامه آن را بنویسید.

مثال

- برنامه ای بنویسید که دو عدد صحیح را از کاربر دریافت کرده و بزرگترین مقسوم علیه (ب.م.م) آن ها را محاسبه و چاپ کند. ب.م.م در انگلیسی gcd می گویند.
به عنوان مثال:

$$\text{gcd}(16,24) = 8$$

$$\text{gcd}(8,9) = 1$$

$$\text{gcd}(84,18) = 6$$

چند دستور خاص

- دو دستور خاص که در حلقه ها (for یا while) از آن ها استفاده می شود:
- **break**: در هر جای حلقه که اجرا شود باعث می شود که اجرای حلقه پایان یافته و اجرای برنامه به اولین دستور بعد از حلقه منتقل شود.
- **continue**: در هر جای حلقه که اجرا شود باعث می شود که اجرای برنامه به تکرار بعدی حلقه منتقل شود.

while (شرط حلقه)

{

....

if(شرط)

break;

....

}

ادامه برنامه

for(i=0; i<100; i++)

{

....

if(شرط)

continue;

....

}

ادامه برنامه

مثال

- دو تکه کد زیر چه چیزی را چاپ می کنند؟

```
for (i=1; i<100; i++)  
{  
    if (i%6 ==0)  
        continue;  
    cout << i <<endl;  
}
```

```
for (i=1; i<100; i++)  
{  
    if (i%6 ==0)  
        break;  
    cout << i <<endl;  
}
```

مثال

- برنامه ای بنویسید که تعدادی عدد را به عنوان ورودی دریافت کرده و میانگین آن ها را محاسبه و چاپ کند. فرض کنید اعداد مثبت هستند و پایان اعداد را یک عدد منفی مشخص می کند.

حلقه های تو در تو

تمرین

- برنامه ای بنویسید که شکل زیر را چاپ نماید.

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

مثال

- برنامه ای بنویسید که عدد N را از کاربر گرفته و عبارت زیر را در خروجی چاپ نماید. به عنوان مثال در صورتی که عدد وارد شده 5 بود، خروجی بصورت زیر باشد:

1

1 2 1

1 2 3 2 1

1 2 3 4 3 2 1

1 2 3 4 5 4 3 2 1

.

تمرین های تکمیلی

ساختارهای شرط و حلقه ها

تمرین

- برنامه ای بنویسید که نمره یک دانشجو را از کاربر دریافت کرده و سپس معال حرفی نمره را بصورت زیر در خروجی چاپ نماید.

| خروجی | نمره |
|-------|-------------------------|
| A | $17 \leq grade \leq 20$ |
| B | $15 \leq grade < 17$ |
| C | $12 \leq grade < 15$ |
| D | $0 \leq grade < 12$ |

تمرین

- برنامه ای بنویسید که عدد π را با استفاده از 100 جمله اول دنباله زیر محاسبه کرده و مقدار به دست آمده را چاپ نماید

$$\pi = 4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \dots$$

تمرین

- برنامه ای بنویسید که عدد N در مبنای 2 را گرفته و معادل آن در مبنای 10 را محاسبه و چاپ کند.

تمرین

- برنامه ای بنویسید که عدد N را از کاربر گرفته سپس N عدد را از کاربر گرفته و ماکزیمم آن ها را محاسبه و چاپ کند.

تمرین

- برنامه ای بنویسید که عدد N را از کاربر گرفته سپس N عدد را از کاربر دریافت کرده و تعداد اعداد زوج و تعداد اعداد فرد را مشخص کرده و چاپ نماید.

تمرین

- برنامه ای بنویسید که مجموع اعداد فرد سه رقمی را محاسبه و چاپ کند.

تمرین

• برنامه ای بنویسید که عدد صحیح N را از کاربر گرفته و مشخص کند کامل است یا خیر.

عدد کامل به عددی گفته می شود که مجموع مقسوم علیه های کوچکتر آن عدد برابر خود عدد شود.

مانند عدد 6 که کامل است زیرا $1+2+3=6$

یا عدد 28 که $1+2+4+7+14=28$

تمرین

- برنامه ای بنویسید که یک عدد را از کاربر دریافت کرده و مقلوب آن را محاسبه و چاپ کند. به عنوان مثال مقلوب عدد 593 برابر است با 395.

تمرین

- برنامه ای بنویسید که عدد N را از کاربر گرفته و مجموع سری زیر را محاسبه و چاپ کند بطوریکه:

اگر N فرد

$$S=1 + (1+2+3) + (1+2+3+4+5) + \dots + (1+2+3+\dots+N)$$

و اگر N زوج باشد

$$S=1 + (1+2+3) + (1+2+3+4+5) + \dots + (1+2+3+\dots+(N-1))$$

تمرین

- برنامه ای بنویسید که عدد N را از کاربر دریافت کند سپس N عدد صحیح و مثبت از کاربر دریافت کرده و مشخص کند هر عدد اول است یا خیر.

تمرین

- برنامه ای بنویسید که عدد N را از کاربر دریافت کند سپس N عدد صحیح و مثبت از کاربر دریافت کرده و تعداد ارقام هر عدد را محاسب و چاپ کند.

تابع

تابع

- یک تابع مجموعه ای از دستورات مستقل و مجزا که وظیفه خاصی را انجام می دهند



- زیاد شدن خطوط برنامه باعث پیچیده شدن، ناخوانایی در برنامه می شود و خطایابی آن را بسیار مشکل می کند. تابع این امکان را برنامه نویس می دهد که برنامه ها را بصورت قطعه های کوچک و مستقل تقسیم کرد که به آن برنامه نویسی تابعی یا functional می گویند
- تابحال فقط از تابع `main()` استفاده کردیم.

- فواید استفاده از تابع
- کاهش پیچیدگی و ساده تر شدن برنامه، افزایش خوانایی برنامه، خطایابی ساده تر، امکان استفاده مجدد از تابع، امکان انجام کار گروهی

- مثال:

`y = sin(x)`

`c = Maximum(a, b)`

`getch()`

نحوه تعریف تابع

- برای تعریف تابع سه مولفه را باید مشخص کرد:
- نوع بازگشتی: نوع بازگشتی تابع که مشخص می کند مقداری که توسط تابع برگردانده می شود از چه نوعی است. اگر تابع خروجی نداشته باشد نوع خروجی آن void در نظر گرفته می شود.
- اسم تابع: که از قواعد نامگذاری شناسه ها پیروی می کند
- لیست پارامترهای ورودی: یک تابع می تواند اصلا ورودی نداشته باشد، یک، دو و یا بیشتر ورودی داشته باشد که توسط کاما از هم جدا می شوند
- نحوه تعریف تابع

(لیست پارامترهای ورودی) نام تابع نوع بازگشتی

{

تعریف متغیرهای محلی تابع

...

بدنه تابع

}

ساختار کلی استفاده از تابع

```
#include <...>
```

```
// ----- functions Declaration -----
```

```
int fcn1 (int a, int b);
```

اعلان یا الگوی تابع

```
// ----- Main Function -----
```

```
int main( )
```

```
{
```

```
int x, y;
```

آرگومان های ورودی تابع

```
z = fcn1(x, y);
```

فراخوانی یا صدا زدن تابع

```
...
```

```
return 0;
```

پارامترهای ورودی تابع

```
}
```

```
// ----- functions implementation -----
```

```
int fcn1(int a, int b)
```

```
{
```

تعریف یا پیاده سازی تابع

```
....
```

بدنه تابع

```
}
```

چند نکته

- الگوی تابع باید قبل از استفاده از تابع یعنی قبل از `main` نوشته شود
- آرگومان های ورودی تابع در هنگام فراخوانی می توانند، متغیر مانند `x`، مقدار ثابت مانند `2` و یا یک عبارت محاسباتی مانند `x+3*y` باشند.
- پیاده سازی تابع می تواند در همان قسمت الگوی تابع نیز انجام شود. این کار برای توابعی که بدنه آن ها یک یا دو خط بیشتر نباشد پیشنهاد می شود ولی برای سایر توابع پیشنهاد نمی گردد.
- در فراخوانی تابع تعداد، ترتیب و نوع آرگومان های ورودی باید با تعداد، ترتیب و نوع پارامترهای ورودی تابع سازگاری داشته باشد. مثلا اگر اولین پارامتر ورودی عدد صحیح است، در هنگام فراخوانی نیز باید اولین آرگومان ورودی عدد صحیح باشد.
- برای برگرداندن یک مقدار از تابع از دستور `return` استفاده می شود و هر زمان که این دستور اجرا شود از تابع خارج شده و کنترل اجرای برنامه به جایی که تابع فراخوانی شده بوده، بازخواهد گشت.

مثال

- تابعی بنویسید که دو عدد اعشاری را به عنوان ورودی دریافت کرده و مجموع آن ها را محاسبه و به عنوان خروجی تابع برگرداند. سپس برنامه اصلی نوشته و از این تابع استفاده کنید.

مثال

- تابعی به نام MyMax بنویسید که دو عدد صحیح را به عنوان ورودی دریافت کرده و ماکزیمم آن ها را محاسبه و به عنوان خروجی تابع برگرداند. سپس برنامه اصلی نوشته و از آن تابع استفاده کنید.

مثال

- تابعی به نام MyPower بنویسید که دو عدد صحیح x و y را به عنوان ورودی دریافت کرده و حاصل x به توان y را محاسبه کرده و برگرداند.
- سپس با به کار گیری تابع فوق برنامه ای بنویسید که سه عدد صحیح a ، b و c را از کاربر دریافت کرده و حاصل عبارت زیر را محاسبه و چاپ کند.

$$a^2 + 2^{(c+1)} + (b - 2)^4$$

تمرین های تکمیلی

توابع

تمرین

- تابعی بنویسید که عدد N در مبنای 10 را به عنوان ورودی دریافت کرده و معادل آن در مبنای 2 را محاسبه و برگرداند
- سپس برنامه ای نوشته و از این تابع استفاده کنید

تمرین

تابعی به نام MySin بنویسید که عدد N و X را به عنوان ورودی از کاربر دریافت کرده و سینوس عدد X را با استفاده از N جمله اول بسط تیلور سینوس که بصورت زیر است محاسبه و برگرداند.

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$$

تمرین

- تابعی بنویسید که عدد N را از کاربر گرفته و N امین جمله دنباله فیبوناچی را محاسبه و به کاربر برگرداند. در سری فیبوناچی هر جمله از جمع دو جمله قبلی حاصل می شود

1, 1, 2, 3, 5, 8, 13,

تمرین

- تابعی بنویسید که یک عدد را از کاربر دریافت کرده و مشخص کند آن عدد اول است یا خیر. بطوریکه اگر اول بود مقدار 1 و در غیر این صورت مقدار 0 را برگرداند.
- سپس با استفاده از تابعی که نوشته اید، برنامه ای بنویسید که N عدد را از کاربر گرفته و مشخص کند هر کدام اول است یا خیر.

آرایه

آرایه

- سوال: فرض کنید بخواهیم نمرات N ($N \leq 50$) دانشجو را از کاربر بگیریم و ابتدا نمرات بزرگتر مساوی 10 و سپس نمرات کوچکتر از 10 را چاپ کنیم.

به چه صورت می توانیم این کار را انجام دهیم؟

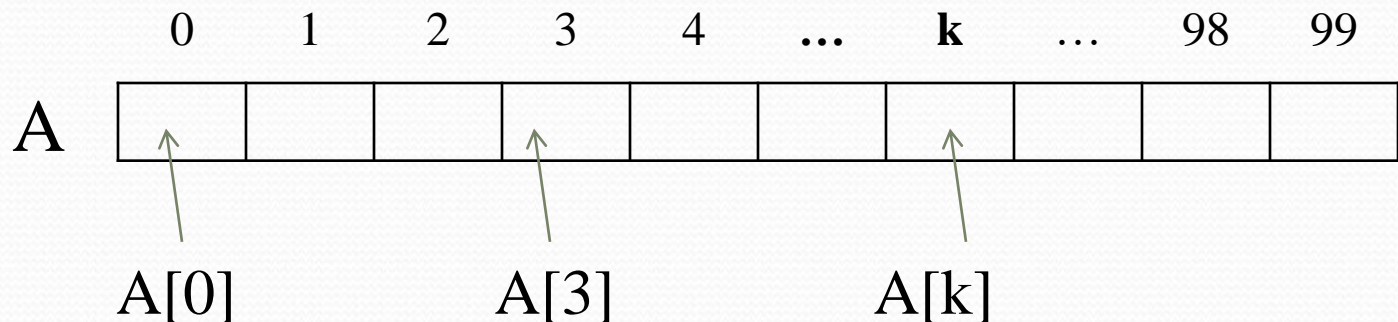
آرایه

- آرایه مجموعه ای از خانه های پشت سر هم از حافظه می باشد که هم نوع و هم اسم بوده و توسط اندیس قابل دسترس می باشد.

- نحوه تعریف آرایه: **[طول آرایه] اسم آرایه** نوع عناصر آرایه

- مثال: تعریف آرایه A از اعداد صحیح به طول 100

`int A[100];`



نکته

- اندیس آرایه در زبان C یا C++ از صفر شروع می شود.

- در تعریف آرایه طول آرایه باید بصورت ثابت تعریف شود. به عنوان مثال:

```
int A[20];
```



```
const int Len=20;  
int B[Len];
```



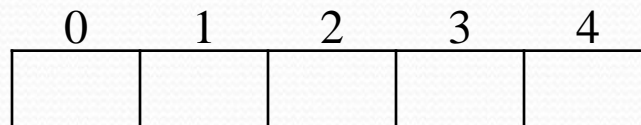
```
int Len=20;  
int C[Len];
```



؟ سوال: اگر طول آرایه را از قبل نمی دانیم
چه کار می توانیم انجام دهیم؟

- در هنگام استفاده از آرایه دقت کنید تا اندیس مورد استفاده، از آرایه خارج نشود در غیر این صورت با خطا مواجه خواهید شد.

```
float D[5];
```



D[0] D[1] D[2] D[3] D[4]



D[5]



D[6]



مقدار دهی اولیه آرایه

`int A[4] = {15, 7, 9, -3};`

| | | | | |
|---|----|---|---|----|
| | 0 | 1 | 2 | 3 |
| A | 15 | 7 | 9 | -3 |

`int B[5] = {8};`

| | | | | | |
|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 |
| B | 8 | 0 | 0 | 0 | 0 |

اگر تعداد اعداد از طول آرایه کمتر باشد، آرایه از ابتدا مقداردهی شده و سپس بقیه خانه های آرایه با صفر مقداردهی خواهد شد.

`int C[3] = {4, 9, -2, 12, 10};` ❌

دقت کنید که تعداد اعداد در مقداردهی اولیه از طول آرایه بیشتر نباشد!

خواندن و چاپ کردن آرایه

- خواندن N عدد از صفحه کلید و قرار دادن داخل یک آرایه (با فرض $N \leq 100$)
- عناصر آرایه باید بصورت خانه به خانه از ورودی خوانده شوند.

مثال

```
int Ary[100];  
for(i=0; i<N; i++)  
    cin >> Ary[i];
```

- چاپ کردن عناصر آرایه Ary شامل N عنصر روی صفحه نمایش:

مثال

```
for(i=0; i<N; i++)  
    cout << Ary[i] << endl;
```

مقداردهی و استفاده در محاسبات

● مقداردهی عناصر آرایه

مثال

```
int x=3, A[6], b;
```

```
A[0] = 10;
```

```
A[2] = 4;
```

```
A[5] = x*7;
```

```
A[0] *=2;
```

```
b = A[2]*3 + A[5];
```

```
A[4] = b - A[0];
```

| | | | | | | |
|---|----|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 |
| A | 10 | | 4 | | | |

● استفاده از عناصر آرایه در محاسبات

| | | | | | | |
|---|----|---|---|---|----|----|
| | 0 | 1 | 2 | 3 | 4 | 5 |
| A | 20 | | 4 | | 13 | 21 |

$b = 33$

مثال

- برنامه ای بنویسید که نمره N دانشجو را از کاربر گرفته (تعداد دانشجو از صفحه کلید خوانده شود) و ابتدا نمرات بزرگتر مساوی 10 و سپس نمرات زیر 10 را چاپ نماید. فرض کنید $N \leq 50$.

مثال

• جستجوی خطی

فرض کنید آرایه Ary بصورت زیر در اختیار شما قرار دارد. برنامه ای بنویسید که عدد x را از کاربر گرفته و آن را در آرایه جستجو نماید. بطوریکه اگر x در آرایه وجود داشت، اندیس اولین خانه ای که x در آن واقع شده است را چاپ کند و در غیر این صورت -1 را چاپ کند.

به عنوان مثال اگر $x=4$ باشد 2 چاپ شود، اگر $x=18$ باشد 5 چاپ شود و اگر $x=42$ باشد -1 چاپ شود.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|-----|----|----|---|----|---|----|----|----|---|----|----|----|
| Ary | 10 | 92 | 4 | 85 | 7 | 18 | 10 | 54 | 4 | 26 | 7 | 4 |

مثال

- فرض کنید آرایه Ary به طول N ($N \leq 100$) به شما داده شده است. برنامه ای بنویسید که آرایه را برعکس نماید. به عنوان مثال اگر آرایه بصورت زیر باشد

| | | | | | | | |
|-----|----|----|---|----|---|----|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| Ary | 10 | 92 | 4 | 85 | 7 | 18 | 5 |

برعکس

| | | | | | | | |
|--|---|----|---|----|---|----|----|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| | 5 | 18 | 7 | 85 | 4 | 92 | 10 |

تمرین

- فرض کنید نمرات N دانشجو ($N \leq 50$) در آرایه Grades به شما داده شده است. برنامه ای بنویسید که به نمرات همه دانشجویان d نمره اضافه کند بطوریکه بالاترین نمره برابر 20 شود.

تمرین

- فرض کنید آرایه A شامل N عدد صحیح به شما داده شده است ($N \leq 100$). برنامه ای بنویسید که عدد صحیح x را از کاربر گرفته و مشخص کند این عدد چند مرتبه در آرایه تکرار شده است.

ارسال آرایه به عنوان پارامتر ورودی تابع

- آرایه ها را نیز می توان مشابه متغیرها به توابع ارسال نمود. در این جا نیز طول آرایه باید یک مقدار ثابت باشد. به عنوان مثال در دستور زیر آرایه Ary به طول 100 و شامل اعداد صحیح به تابع Test ارسال شده است.

```
int Test( int Ary[100], int x)
{
    ....
}
```

- در هنگام فراخوانی تابع، فقط کافی است اسم آرایه مورد نظر را به تابع پاس داده شود. به عنوان مثال:

```
int main()
{
    int Nums[100], a=10, b;
    ...
    b = Test (Nums, a);
    ...
}
```


مثال

- تابعی بنویسید که آرایه NumS از اعداد اعشاری و به طول N ($N \leq 100$) را به عنوان ورودی دریافت کرده و ماکزیمم عناصر آرایه را پیدا کرده و به عنوان خروجی تابع برگرداند.

توجه: طول آرایه یعنی N را می بایست به عنوان یکی از پارامترهای ورودی های تابع در نظر بگیرید.

مثال

- تابعی بنویسید که آرایه A به طول 100 شامل اعداد صحیح و همچنین عدد صحیح x را به عنوان ورودی دریافت کرده و عدد x را در آرایه جستجو نماید. بطوریکه اگر x در آرایه وجود داشت، اندیس اولین مکان وقوع x و در غیر این صورت -1 را به کاربر برگرداند.

ماتریس

ماتریس

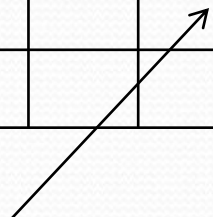
- به آرایه های دو بعدی ماتریس گفته می شود.
- نحوه تعریف ماتریس:

[تعداد ستون] [تعداد سطر] اسم ماتریس نوع عناصر ماتریس

• مثال

```
int M[4][5];  
cin >> M[0][2];  
M[1][3] = 5;  
M[3][3] = M[1][3]*2;  
cout << M[3][3];
```

| M | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | | | | | |
| 1 | | | | | |
| 2 | | | | | |
| 3 | | | | | |

 M[2][3]

مقداردهی اولیه ماتریس

- یک ماتریس را می توان به صورت زیر در هنگام تعریف مقداردهی کرد:

```
int A[3][4] = { {2, 15, 7, 8}, // Row 1  
               {16, 0, -6, 3}, // Row 2  
               {0, 9, -4, 17} }; // Row 3
```

| A | 0 | 1 | 2 | 3 |
|----------|----|----|----|----|
| 0 | 2 | 15 | 7 | 8 |
| 1 | 16 | 0 | -6 | 3 |
| 2 | 0 | 9 | -4 | 17 |

خواندن ماتریس از صفحه کلید

- خواندن یک ماتریس با ابعاد N سطر و M ستون از صفحه کلید (با فرض $N, M \leq 100$)

مثال

```
int M[100][100];
for(i=0; i<N; i++)
    for( j=0; j<M; j++)
        cin >> M[i][j];
```

پر کردن ماتریس بصورت سطری
(ابتدا سطر اول، سپس سطر دوم،)

مثال

```
int M[100][100];
for( j=0; j<M; j++)
    for(i=0; i<N; i++)
        cin >> M[i][j];
```

پر کردن ماتریس بصورت ستونی
(ابتدا ستون اول، سپس ستون دوم،)

چاپ کردن ماتریس روی صفحه نمایش

- چاپ کردن عناصر یک ماتریس با N سطر و M ستون روی صفحه نمایش بصورت زیر قابل انجام می باشد:

مثال

```
for(i=0; i<N; i++)  
{  
    for( j=0; j<M; j++)  
        cout << M[i][j] << "\t";  
    cout << endl;  
}
```


مثال

- برنامه ای بنویسید که یک جدول ضرب را داخل یک ماتریس 10×10 ذخیره کرده و سپس آن را چاپ نماید.

مثال

- برنامه ای بنویسید که یک ماتریس $N \times N$ که $N < 100$ را از کاربر دریافت کرده و سپس عناصر بالای قطر اصلی ماتریس را صفر نماید.

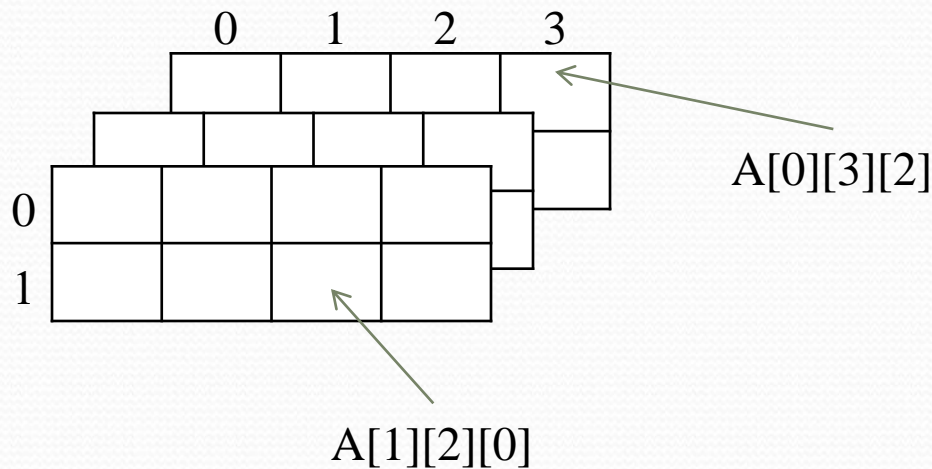
آرایه ها چند بعدی

- می توان آرایه های با ابعاد سه یا بیشتر نیز داشت

- مثال:

```
int A[2][4][3];
```

آرایه چند بعدی شامل 3 صفحه 2x4



آرایه های کاراکتری یا رشته ها

آرایه های کاراکتری یا رشته ها

- به آرایه های از نوع کاراکتر رشته گفته می شود. از رشته ها برای نشان دادن اسامی، جملات، آدرس و غیره می توان استفاده نمود. مانند "Ali" یا "Hello World"

- تعریف آرایه از نوع کاراکتر:

```
char Name[10];
```

```
char Address[100];
```


مقداردهی اولیه کاراکتری

● مثال

```
char Name[10] = "ALI";
```

```
char Test [14] = "Hello world";
```

| | | | | | | | | | | |
|------|---|---|---|----|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Name | A | L | I | \0 | | | | | | |

| | | | | | | | | | | | | | | |
|------|---|---|---|---|---|---|---|---|---|---|----|----|----|----|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| Test | H | e | l | l | o | | w | o | r | l | d | \0 | | |

نکته: در زبان C و C++ انتهای رشته ها با کاراکتر تهی یا '\0' مشخص می شود. که این کاراکتر در مقداردهی اولی یا خواندن از ورودی بصورت خودکار توسط کامپایلر در انتهای رشته قرار داده می شود.

ورودی و خروجی رشته

- کامپایلر زبان C و زبان C++ قوانین خاصی را برای آرایه های از نوع کاراکتر تعریف کرده است که در سایر انواع آرایه برقرار نیست. از جمله:
 - پایان رشته یا آرایه کاراکتر با کاراکتر '\0' می باشد.
 - آرایه کاراکتری را می توان مستقیماً توسط دستور cin از ورودی خواند. در این حالت به آخر رشته خوانده شد یک کاراکتر تهی یا '\0' بصورت خودکار اضافه خواهد شد.

```
char Name[10];  
cin >> Name;
```

- آرایه کاراکتری را می توان مستقیماً توسط دستور cout در خروجی چاپ نمود. در این حالت از ابتدای رشته تا کاراکتر '\0' روی صفحه نمایش چاپ خواهد شد.

```
cout << Name;
```

دسترسی به عناصر آرایه از نوع کاراکتر

- در آرایه های کاراکتری می توان مشابه آرایه های دیگر بصورت مجزا به هر یک از خانه های آرایه دسترسی داشت.
- مثال:

```
char str[10] = "AliReza";
```

```
cout << str;    // AliReza
```

```
cout << str[0]; // A
```

```
cout << str[2]; // i
```

```
str[3] = '*';
```

```
cout << str;    // Ali*eza
```


اعمال روی رشته ها

• برخی از توابع کار بر روی رشته ها در کتابخانه `<string.h>` قابل دسترس می باشد. از جمله:

- `strcpy(str1, str2)` برای کپی کردن رشته `str2` در رشته `str1`
- `strcmp(str1, str2)` برای مقایسه رشته `str1` و رشته `str2`
- `strcat(str1, str2)` برای چسباندن رشته `str2` به انتهای رشته `str1`

پایان

موفق و سربلند باشید