**بسم الله الرحمن الرحیم**

**درس اول : آشنایی با زبان برنامه نویسی سی**

زبان برنامه نویسی سی ارتباط نزدیکی با یونیکس دارد

برای مدیریت برنامه ها از محیط هایی استفاده می‌شودکه با آن IDE گویند که مخفف Integrated development Enviroment است که به معنی محیط توسعه جامع است

پسوند تمام فایل های برنامه های زبان سی کلمه C است

عبارتی که در بالای صفحه و با علامت # نمایش داده می شود یک دستور پیش پردازنده است

#include <stdio.h>

این عبارت یک عبارت استاندارد است که به معنی این است که یک ورودی و خروجی را در نظر می گیرد و به صورت std یا استاندارد است این یک هدر است و ربطی به دستورات ندارد

Int main()

یک تابع است که از نوع Integer بوده و نام آن main می باشد

در داخل {} بدنه قرار می گیرد

اساساً در هر برنامه باید یک نتیجه ای برگردانده شود که آن را با Return 0 بر می گردانیم

معمولاً سبک برنامه نویسی به این صورت است که برنامه ای که نوشته می شود معمولا یا خروجی آن درست است یا نادرست

عمدتاً برنامه ها نتیجه صحیح را با صفر بر می گردانند

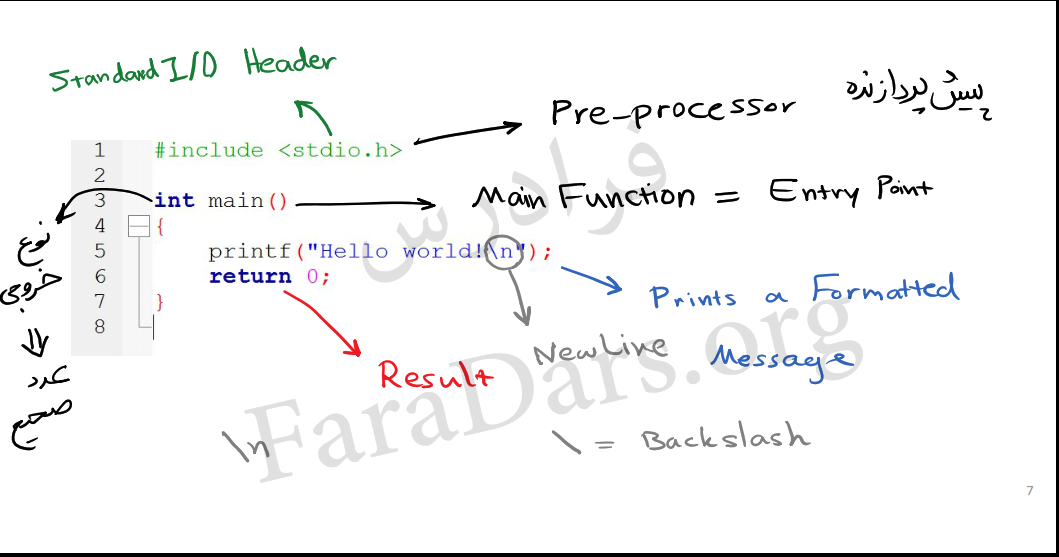
Main باید همیشه در برنامه ها باشد و اصلا هر برنامه ای الزاماً باید دارای تابع main باشد main در حقیقت درگاه ورود ما به برنامه است

در هر برنامه سی باید برای اینکه یک برنامه توسط Compiler اجرا شود حتما باید یک تابع موجود باشد

اگر بخواهیم تابع main بدون خروجی باشد از void استفاده می‌کنیم بعنی به جای int عبارت void می‌گذاریم در این حالت نیار به قرار دادن Return نمی باشد

**بسم الله الرحمن الرحیم**

**درس اول : آشنایی با اجزای سازنده یک برنامه سی**



در داخل پیش پردازنده فایل های مهم تعریف شده است به عنوان مثال دستور Printf در داخل Standard I/o Header تعریف شده است

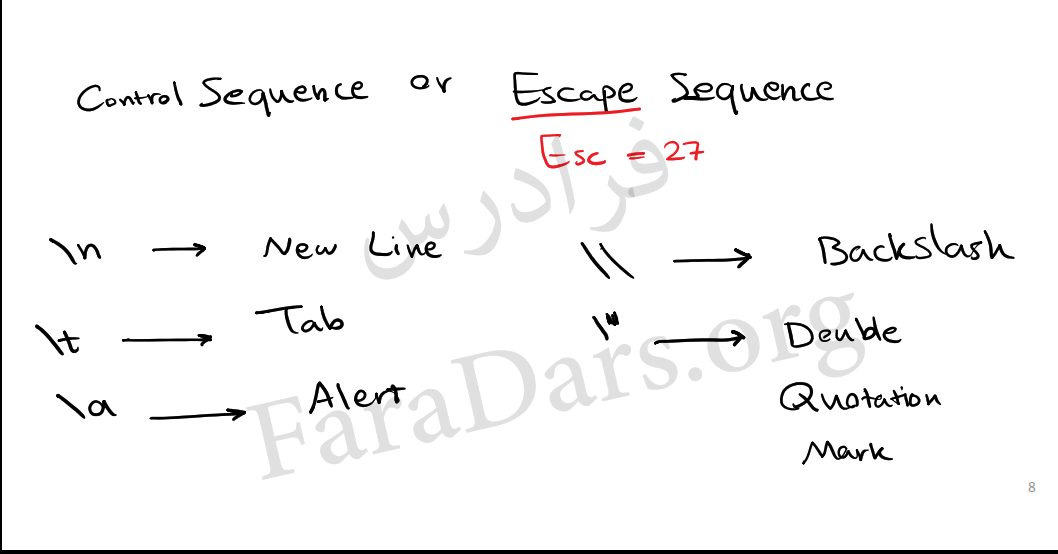
main در حقیقت نقطه ورود به برنامه است که به صورت تابع است توجه به این نکته ضروری است که main باید حتما با حروف کوچک نوشته شود

Printf مخفف print formated است

\ یک کاراکتر کنترلی است که به عنوان مثال n را بهینه می کند

اگر در برنامه ای نیاز به نوشتن کاراکتر \ باشد باید دوبار در کنار هم بنویسیم \\ با یکبار خطا میگیرد

**Control Sequence or Escape Sequence :**



در انتهای تمام دستورات زبان C یک ; باید باشد

اگر در برنامه ای بخواهیم مسیر اجرای برنامه را تغییر دهیم و به عبارتی Intraction داشته باشیم چه عملی باید انجام دهیم در حقیقت اگر بخواهیم یک مقداری را از ورودی بگیریم چگونه عمل میکنیم

برای اینکار از دستور scanf استفاده می‌کنیم

Scanf(“%d”,&نام متغیر )

در این دستور %d یک مقوله پیش فرض است و علامت & به آدرس متغیر اشاره می کند

که در آن %d به معنی یک رقم می باشد

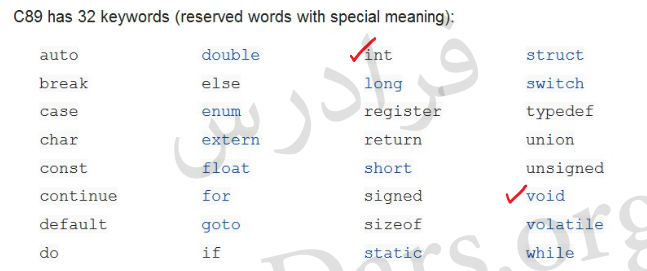
برای نشان دادن یک متغیر مانند n در خروجی به صورت زیر عمل میکنیم

Scanf(“%d”,&n)

که در آن n نام متغیری است که در آن مقادیر ذخیره می شود

**قواعد نام گذاری متغیرها :**

1. نام متغیرها و توابع باید شامل حروف، اعداد و خط زیر (\_) باشد
2. نباید با اعداد شروع شود
3. همیشه سعی می شود اولین حرف از نام توابع با حرف کوچک شروع شود
4. از نظر طول هم باید سعی شود بیشتر از 31 کاراکتر شود
5. اگر اسم متغیر بیشتر از یک کلمه بود با کاراکتر زیر خط (\_) از هم جدا می کنند
6. هر مقداری باید سه مقدار را داشته باشد Type و نام و مقدار و هر متغیر، مجموعه ای از این سه حالت است
7. 32 کی ورد زبان سی که شامل موارد زیر می باشد



از این ها نمی توان برای نام متغیر یا تابع استفاده کرد

**انواع اپراتورها در زبان C :**

1. +
2. -
3. \*
4. / این در حقیقت خارج قسمت است
5. % این علامت باقیمانده است
6. پرانتزها ()

**ترتیب اعمال عملگرها :**

1. پرانتزها
2. \*
3. /
4. %
5. +
6. تساوی ==
7. نامساوی !=
8. &&
9. ||
10. کوچکتر یا مساوی <=
11. بزرگتر مساوی >=
12. بزرگتر <
13. کوچکتر >

**بسم الله الرحمن الرحیم**

**درس سوم : ساختارهای کنترلی در برنامه و زبان سی**

**لازمه پیاده سازی الگوریتم ها :**

1. توالی در اجرای دستورات Sequatial Execution
2. انتخاب ، انشعاب یا تصمیم گیری است Selection
3. حلقه ها و تکرار Repition

با این سه عنصر می توان هر برنامه ای را پیاده کرد

دو عنصر ردیف های 2 و3 به عنوان ساختارهای کنترلی برنامه هستند

**دستور If :**

این دستور ساده ترین دستور کنترلی است که به صورت زیر نوشته می‌شود

If (شرط )

{

دستورها.......  
{

توصیه می شود که {} حتما باشد

معیار برابری در زبان C نماد == می باشد این یعنی دو طرف را بررسی می‌کند که آیا با هم برابرند یا خیر ؟

کلمه کلیدی دیگری به نام else داریم که در صورتیکه شرط برقرار نباشد

در بعضی مواقع که شرط زیاد باشد می توان از else if() در تکرار شروط بهره برداری کرد

این دستور در حقیقت معادل این دستور است

Else

{

If()

{

}

Else

{

}

}

**اپراتورهای مقایسه :**

1. تساوی ==
2. نامساوی !=
3. کوچکتر یا مساوی <=
4. بزرگتر مساوی >=
5. بزرگتر <
6. کوچکتر >

اپراتور مقدار دهی یا assignment به صورت = تعریف می شود

به عنوان مثال a = 10 یعنی در a عدد 10 را قرار بده

کلیه عبارت هایی که در سمت چپ Assignment ها قرار میگیرند باید L-value باشند که این عبارت ها همان متغیرها هستند و یا اینکه آرایه باشد که البته آرایه ها ساده نیستند

حالت دیگری که داریم حالت تساوی است که با عبارت == معین می شود

مانند a == 10 که این بیشتر در عبارات شرطی به کار می رود که در حقیقت می گوید آیا a باید با 10 برابر باشد یا خیر

به تساوی Equality نیز می گویند در این حالت سمت راست و چپ تساوی هر چیزی می توانند باشند

**اولویت اپراتورهای قیاس :**

1. ()
2. + - ++ -- در اینجا اپراتورهای تکی داریم (cast)
3. \* / %
4. + -
5. < < => =<
6. == =!
7. =
8. =+ =- =\* =/ =%

**مباحث پیشرفته Assigment :**

a=a+1 معادل آن a+=1

a=a\*2 معادل آن a\*=2

از این اپراتورها موارد زیر را داریم

=+ =- =\* =/ =%

خلاصه تر a +=1 به صورت a++ است

این به این معنی است که یک واحد به آن اضافه می‌کند

همین مورد در خصوص a=a-1 هم صادق است a—

این اپراتورها ++ و – دو حالت دارند Pre و post

++a را pre گویند

a++ را post گویند

که همین حالت هم برای - - صادق است

Pre یعنی اول مقدار a را افزایش یا کاهش بده بعد استفاده کن

در حالت post یعنی اول a را استفاده کن بعد یک واحد اضافه یا کم کن

**حلقه ها و تکرار Repition**

اگر بخواهیم کاری را به دفعات انجام دهیم از حلقه استفاده می کنیم

حلقه تکرار while :

ساده ترین دستور تکرار است

While (شرط)

{

دستور

}

مادامیکه شرط برقرار است دستور اجرا می شود

این دستور برای مواقعی استفاده می شود که نمیدانیم چند بار این دستور اجرا می شود

**Casting :**

اگر داده های ما به صورت عدد صحیح باشد و خروجی ما به صورت عدد اعشاری باشد خروجی ما به صورت عدد صحیح می باشد برای رفع این مشکل ما داده های اولیه را به اصطلاح Cast یا تبدیل می‌کنیم برای Cast کردن ما در جلوی داده مورد نظر برای Cast کردن متغیر را در داخل () قرار می دهیم

float m = (float)S/i;

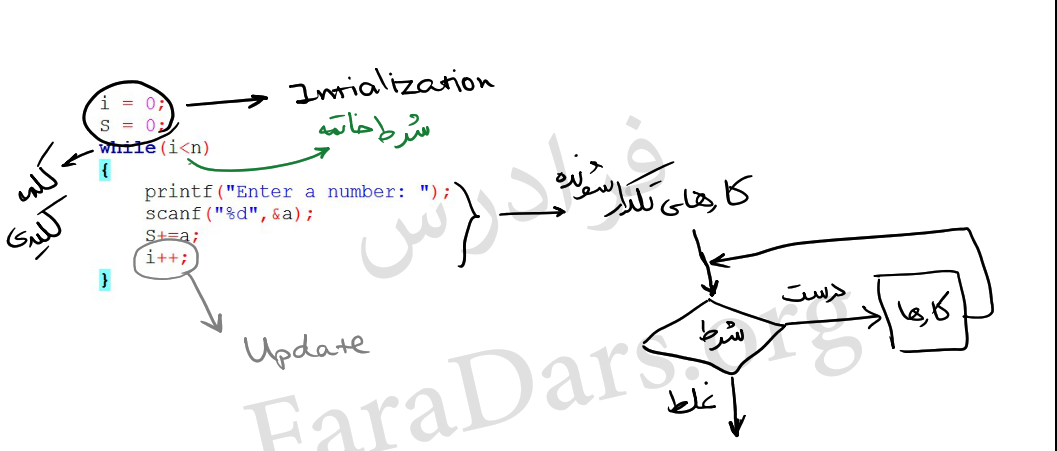
نکته :

برای اینکه مقدار خروجی به صورت اعشاری در بیاید باید از %f در دستور printf به جای %d استفاده کرد

برای تعیین رقم های اعشار در خروجی اعشاری باید به میزان ضرایب مورد نظر در جلوی %f ضریب از 0.1 و 0.2 و ...... قرار داد که در آن 0.1 یک رقم اعشار و 0.2 دو رقم اعشار می باشد

به عنوان مثال %0.1f و یا %0.2f و ........پ

**عناصر و اجزا تشکیل دهنده دستور While :**



می توان به جای شرط و Update از شرط به انضمام یک مورد با Break استفاده نمود

برای اینکه یک حلقه True شود و به صورت همیشگی ادامه یابد باید در شرط while از عدد 1 استفاده کنیم در این حالت الزاما باید در بدنه دستور while از کلمه break استفاده نمود

می توان در این حلقه ها به جای عدد 1 از true و یا false استفاده نمود برای اینکار ابتدا باید آن را تعریف نمود که در اینصورت به صورت زید عمل می‌کنیم

#include <stdio.h>

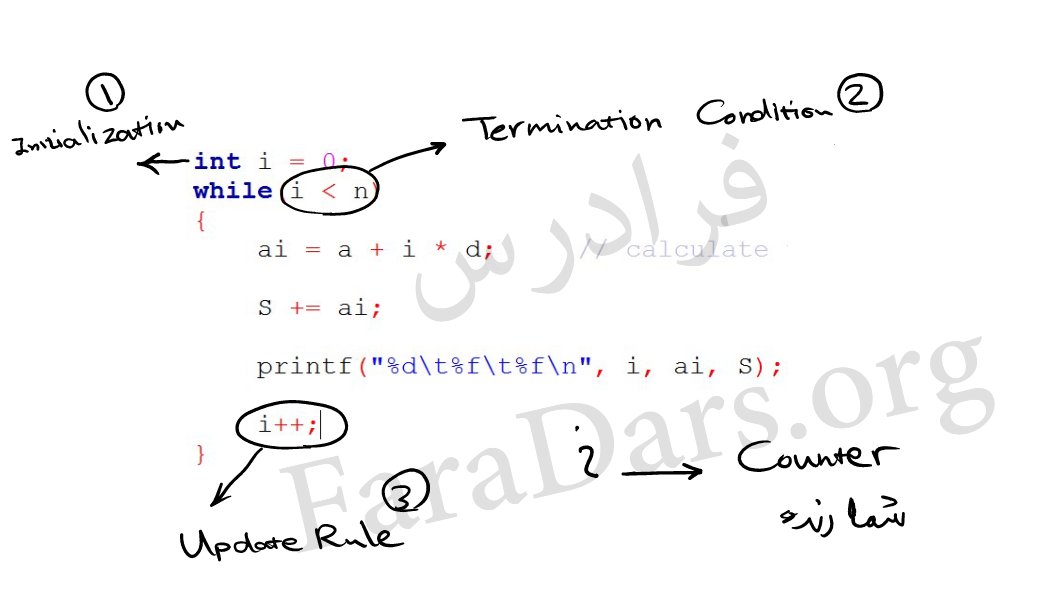
#define TRUE -1

#define FALSE 0

توجه شود که در تعریف این عبارات باید از حروف بزرگ استفاده نمود

**بسم الله الرحمن الرحیم**

**درس چهارم : ساختارهای کنترل برنامه در زبان C قسمت دوم**



**حلقه For :**

این حلقه بر اساس عناصر while به صورت زیر تعریف می شود

For (initialization ; Termination Condition ; Update Rule)

{

Body

}

کلمه Break در حلقه for کارایی دارد و هر جای حلقه استفاده شود همان جا حلقه متوقف می شود و همچنین می توان از دستور Continue نیز استفاده کرد

**تابع توان** **(pow) :**

این تابع به شکل زیر تعریف می شود

Pow(x,y)

که این تابع عدد x را به توان y می رساند

برای استفاده از این تابع باید کتابخانه جدیدی را به برنامه اضافه نمود که با #include <math.h> تعریف می شود

**کتابخانه math.h :**

این کتابخانه شامل توابعی برای محاسبات ریاضی می باشد

1. Pow(x,y) یعنی عدد x به توان عدد y می رسد
2. Sqrt(x,y) یعنی ریشه y عدد x
3. Floor(x) یعنی نزدیک ترین عدد صحیح نزدیک به خود که همان جزء صحیح است یعنی بزرگترین عدد صحیح کوچکتر از x را نشان می دهد
4. Ceil (x) بزرگترین عدد صحیح بزرکتر از x را حساب می‌کند
5. تابع round(x) : این تابع عدد x را گرد می کند اگر بخش اعشاری عدد کمتر از 0.5 باشد مثل floor عمل می کند واگر بیشتر از 0.5 باشد مثل ceil عمل می کند
6. تابع trunc(x) : این تابع برای اعداد مثبت مثل floor و برای اعداد منفی مثل sign عمل می‌کند

چهار تابع Floor و Ceil و roung و trunc توابعی هستند که اعداد غیر صحیح را بع اعداد صحیح تبدیل می کنند

1. تابع fmod(x,y) : این تابع باقیمانده را محاسبه می کند بعنی باقیمانده تقسیم x بر y رامحاسبه می کند
2. تابع fabs(x) : این تابع قدر مطلق عدد x را محاسبه می کند
3. تابع exp(x) : این تابع یعنی
4. Exp 2 (x) : این تابع یعنی
5. Log(x) : این تابع همان ln x است
6. Log 2(x) : این تابع یعنی logx2
7. Log 10(x) : این تابع یعنی log10x
8. Sin (x)
9. Cos(x)
10. Tan(x)
11. asin(x)
12. acos(x)
13. atan(x)
14. sinh(x)
15. cosh(x)
16. tanh(x)
17. asinh(x)
18. acosh(x)
19. atanh(x)
20. تابع خطا erf(x)
21. تابع خطا erfc(x)

**دستور Switch :**

نحوه استفاده از آن به این صورت است که در جلوی switch نام یک متغیر می آید و در بدنه آن مقادیر مختلف آن را با استفاده از

case می سنجیم در انتهای هر case یک درستور Break می گذاریم تا در صورت تامین شرط مورد نظر از حلقه خارج شود همچنین دارای یک حالت default هم هست که اگر هیچ کدام نبود این حالت اتفاق می افتد که پیش فرض است

Switch ( متغیر)

{

Case A :

دستورات

Break;

Case B:

دستورات

Break;

.

.

.

default

}

**نحوه گرفتن یک کاراکتر از کاربر :**

برای گرفتن یک کاراکتر از کاربر از تابع getchar() استفاده می‌کنیم

همچنین برای تعریف کاراکترها ار متغیر char استفاده می کنیم

جهت نشان دادن و خروجی گرفتن از کاراکترها در دستور printf از خروجیc % استفاده می کنیم

کاراکتر eof به معنی end of file است و برای آن از ctrl + z استفاده می کنیم

**حلقه Do – while :**

نحوه نوشتن آن به صورت زیر است :

Do

{

بدنه کدها

}while (condition);

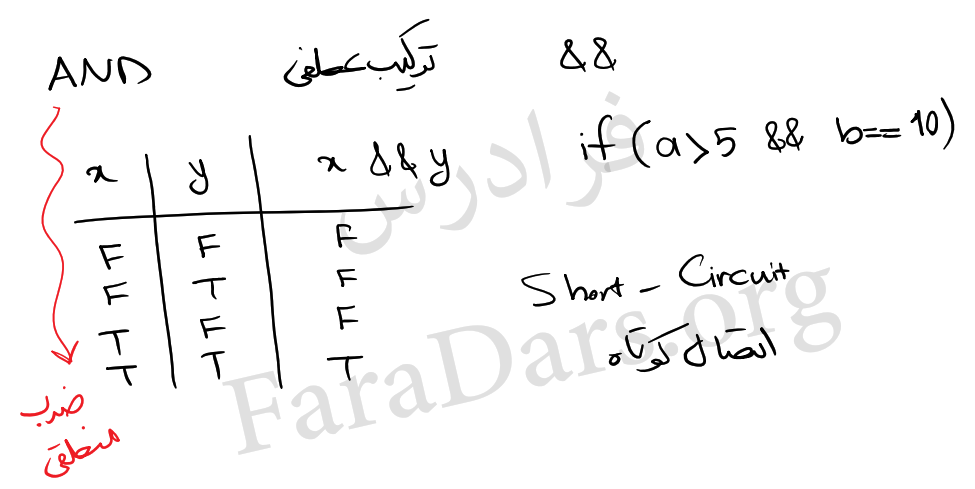
در این شرط حتما یکبار دستور اجرا می شود

**اپراتور and یا && :**

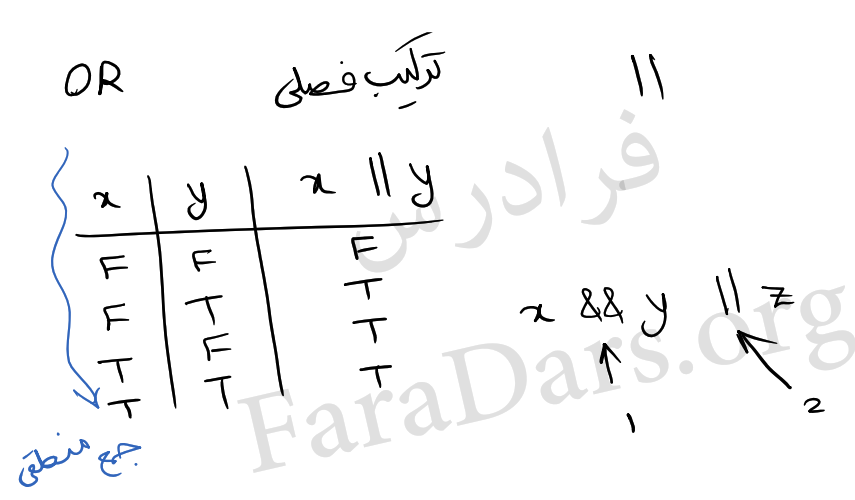
اپراتور and که آن را با && نشان می دهد که به آن ترکیب عطفی گویند

ترکیب جدولی آن به صورت زیر می باشد

از آنجا که این عملگر ممکن است دارای پیچیدگی هایی باشد همیشه عملیاتی که سبک تر است را اول می آوردند



**اپراتور or یا ترکیب فصلی یا || :**



**ترتیب اپراتور های && و || :**

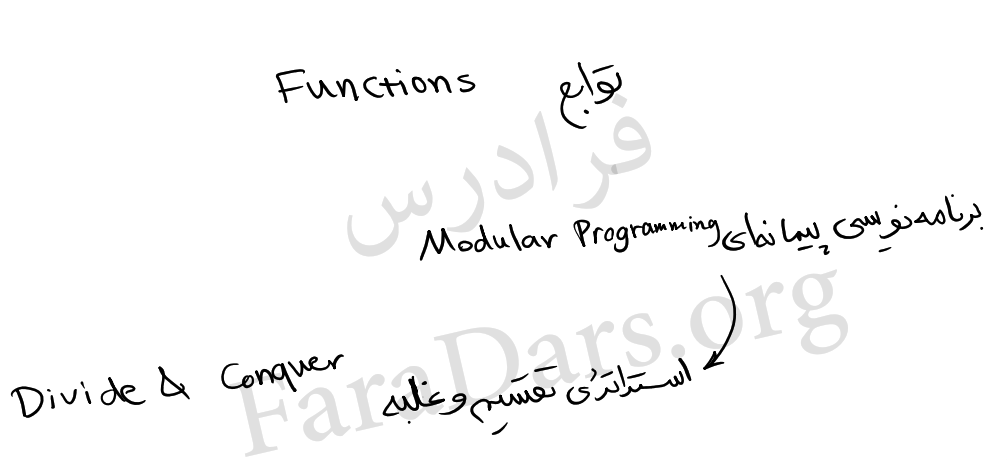
از آنجا که اپراتور && ضرب منطقی است و اپراتور || جمع منطفی است اولویت در اجرا با && است و سپس || می آید

**بسم الله الرحمن الرحیم**

**درس پنجم : پیاده سازی و استفاده از توابع در C**

در مهندسی نرم افزار مبحث برنامه نویسی پیمانه ای (Modular) بسیار مهم است این نوع برنامه نویسی یعنی ریز کردن مسئله به مسائل کوچکتر و نوشتن برنامه های بزرگ در این برنامه نویسی مدل تقسیم و غلبه داریم یعنی مسائل بزرگ را به یک یا چند مسئله کوچکتر تقسیم می‌کنیم

اگر Function ها نبودند خیلی از برنامه هایی که ما تاکنون دیده ایم و خیلی از بسته های نرم افزاری وجود نداشتند



**طریقه نوشتن یک تابع :**

برای نوشتن یک تابع از فرمول زیر استفاده می‌کنیم

Declar variable out put function name (Declar variable input name of variable)

Int is\_prime(int n)

در انتهای تابع مقدار مورد نظر را با return برمی‌گردانیم در این حالت نباید مقدار 0 را برگردانیم

برای اینکه یک تابع داخل یک برنامه کار کند باید آن تابع را قبل از تابع main یک prototype قرار دهیم

که در این صورت ، صورت تابع تعریف شده را قبل از main می آوریم و حتماً در انتهای آن ; می گذاریم به این عنوان اصطلاحاً

Function prototype گویند

در Prototype ها می توان نام متغیر را ننوشت ولی بهتراست آن را بنویسیم

تعدادی هدر های استاندارد هستند که برای توابع به کار می روند

1. Stdio.h که ورودی و خروجی استاندارد را برای ما نشان می دهد printf و scanf از این دسته توابع هستند
2. Stdlib.h توابع استاندارد همه منظوره تبدیل اعداد و رشته ها و .....
3. String.h تابعی برای کار با رشته ها است
4. Time.h تابعی برای کار با زمان و تاریخ
5. Stddef.h توابعی برای کار با یکسری ثوابت و محاسبات استفاده می شود
6. Stdarg.h یکسری توابع مجبور به تعریف می شویم که نوع و تعداد متغیرهای آن نامعلوم می باشد
7. Math.h که توابع ریاضی هستند
8. Local.h که برای تنظیمات محلی و تنظیم زبان و .... می توان استفاده نمود
9. Limits.h محدودیت های اعداد صحیح را در سیستم قابل اجرا تعریف کرده است
10. Float.h برای اعداد غیر صحیح که محدودیت های سیستم را تعریف می‌کند
11. Ctype.h که در حقیقت character type می باشد که این تابع با مواردی که به صورت تابع کاراکتری هستند می توانند کار بکنند
12. Assert.h که برای دیباگ کردن به کار می رودکه برای اشکال یابی خیلی کاربرد دارد
13. Errno.h که برای گزارش خطاها به کار می رود

**تفاوت توابع iteration و Recursive :**

در اجرای توابع می توان به دو صورت برنامه را در تابع پیاده سازی نمود در روش اول می توان از روش آبشاری و یا itetation استفاده کرد و در روش دوم می توان از روش بازگشتی و یا recursive استفاده نمود عمدتا سرعت روش آبشاری از روش بازگشتی بیشتر است روش بازگشتی از پشته ها استفاده می‌کند روش بازگشتی دارای تعداد خط کد کمتری می باشد

**انواع متغیرها از نظر محل قرار گیری :**

متغیرها به دو صورت تعریف می شوند یا به صورت local که در داخل تابع مورد نظر و یا حلقه و یا Scope مورد نظر تعریف می‌شوند و یا به صورت عمومی و Global هستند که در حالت Global در یک فضایی تعریف می‌شوند که تمامی تابع مورد نظر به آن دسترسی دارد

تا آنجا که امکان دارد باید با scope محدود و Local کار کنیم

**شیوه ذخیره سازی متغیرها در یک تابع :**

برای اینکار ما ۴ کلید واژه اساسی داریم

1. Auto حالت Default است
2. Register
3. Extern
4. Static

Auto , Register هر دو به صورت automatic duration هستند یعنی کار ما که با آنها به اتمام رسید از بین می روند

اما exterm و static اینگونه نیستند و Static duration هستند یعنی تا زمانی که برنامه اجرا می شود هستند

از بین اینها auto و Register و static دارای local scope هستند

Exterm به صورت global است

تفاوت بین auto و register در این است که auto به صورت default است ولی register به صورت store in register است که سرعت زیادی در خواندن و نوشتن داده ها به ما می دهد یعنی به جای اینکه ما متغیر را در حافظه سیستم ذخیره کنیم به یکی از register های سیستم اتصال می دهیم

01:04

**بسم الله الرحمن الرحیم**

**درس ششم : آرایه ها**

آرایه ها در پیاده سازی الگوریتم ها بسیار مهم هستند و بدون آرایه های امکان پیاده سازی بسیاری از برنامه ها وجود ندارد

آرایه ها مجموعه ای از اطلاعات و دیتا های مرتبط به هم هستند که با یکدیگر ارتباط دارند و در قالب یک اسم ذخیره می‌شوند و همه اجزا آرایه این خصوصیت مشترک را دارند که نامشان یکی است و نوعشان هم یکی است

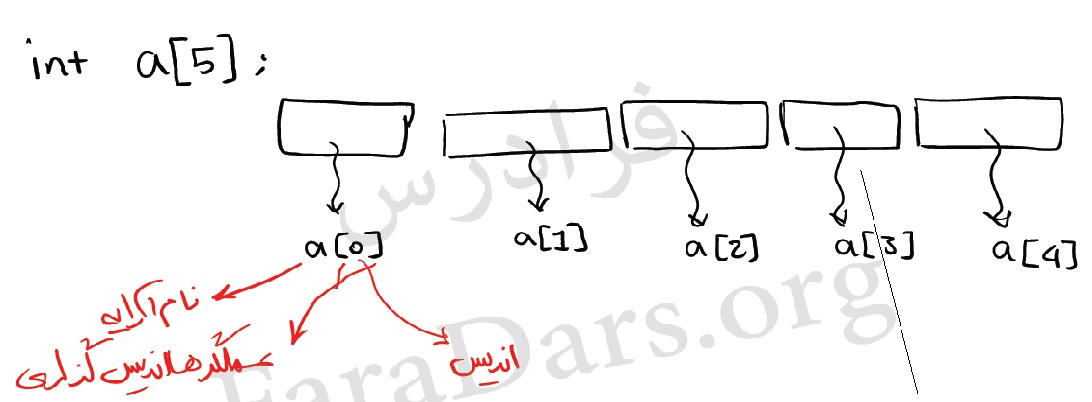
آرایه یکی از ساختارهای داده است که به صورت استاتیک می تواند عمل کند و برای ذخیره سازی انواع داده تحت یک نام به کار رود

**تعریف آرایه در زبان برنامه نویسی C :**

برای تعریف آرایه ها از روش زیر استفاده می شود

Int a[تعداد خانه مورد نظر]

شکل زیر نمای کلی آرایه و تعریف آن را برای یک آرایه



در صورتیکه آرایه مقدار دهی نشود مقدار آن صفر است

برای مقدار دهی کردن دو روش وجود دارد یکی اینکه آنها را تک تک مقدار دهی کنیم

روش دیگر اینکه آنها را به روش زیر مقدار دهی یا Initialize کنیم

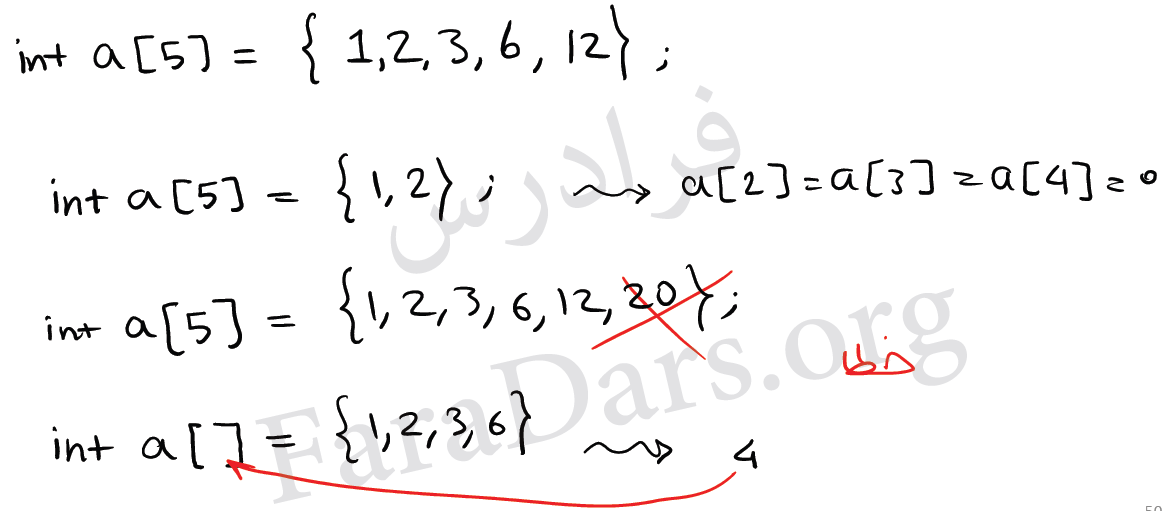
Int a[5] = {1,2,3,4,7,9}

اگر خانه آرایه ای مقدار دهی نشود صفر می‌شود

اگر تعداد خانه های آرایه از مقدار دهی آن کمتر باشد برنامه خطا می‌گیرد

مقدار دهی خانه ها از a[0] آغاز می شود

اگر تعداد خانه های آرایه را ندانیم و بخواهیم به میزانی که مقداردهی می‌کنیم خانه تعیین شود بهتر است که تعداد خانه مورد نظر را خالی بگذاریم تا بعد از مقداردهی، تعداد خانه ها مشخص شود



اگر تعداد آرایه های خانه را ندانیم بهتر است از یک ثابت برای مقداردهی آنها استفاده کنیم در این صورت از روش زیر مقدار ثابت را مشخص می‌کنیم

#define نام ثابت مقدار مورد نظر

در زبان برنامه نویسی C روش مناسبی برای بدست آوردن تعداد المان های یک آرایه وجود ندارد

ولی یک اپراتور به نام sizeof داریم که بر حسب Byte تعداد بایت هایی که این آرایه گرفته است را نشان می‌دهد

از آنجا که یک متغیر int 4 بایت می باشد لذا اگر در sizeof مقدار کل یک آرایه را بدست آوریم می‌توانیم با تقسیم بر عدد 4 تعداد کل آرایه و المان های آن را بدست آوریم

به عنوان مثال داریم

Int N = sizeof(نام آرایه)/sizeof(عنصر اول آرایه)s

**آرایه هایی از جنس String :**

برای تعریف این کاراکترها از متغیر char استفاده می‌کنیم به صورت زیر

Char s[]

برای تعریف آرایه از جنس کاراکتر به دو روش عمل می کنیم

Char s[] = {‘H’,’e’,’l’,’l’,’e’\0}

که در آن \0 کاراکتر صفر یا Null است که در حقیقت یک string Trminator است

روش دیگر که در حقیقت یک Shortcut است به صورت زیر است

Char s[] = “Hello”

در حقیقت روش اول درست است

**روش گرفتن اسم از یک کاربر :**

در این روش باید از یک آرایه برای گرفتن نام یا اسم اقدام کنیم

در این روش در هنگام آدرس دهی از & استفاده نمی شود و خود نام آرایه یک آدرس است

**پیدا کردن تعداد کاراکترهای یک string :**

برای اینکار از یک حلقه و کاراکتر \0 در تعریف آرایه استفاده می کنیم که هر زمان یه \0 رسید همان جا متوقف شود

می توانیم تعداد دلخواهی کاراکتر را بگیریم برای این کار باید در کنار علامت % عدد مورد نظر را قرار دهیم

حتما توجه شود که تعداد کاراکتر مورد نظر در هنگام گرفتن کاراکتر واردشود یعنی کنار علامت درصد بگذاریم

**طریقه به کار بردن آرایه ها در توابع:**

به طور کلّی زبان C هیچ شناختی از آرایه ها و تعیین تعداد آنها ندارد باید زبان C را در این مورد کمک کنیم

Access Modifier function name(Array,Number of Array Element(

برای اینکار دو روش داریم :

1 – تعداد را وارد نکنیم

2- تعداد را وارد کنیم

**به کارگیری Call By refrence :**

در این حالت آرایه ای که در تابع به کار می رود با آرایه ای اصلی در بدنه برنامه یکسان است و درحقیقت آرایه با refrence یا آدرسش فراخوانی می‌شود call by refrence هم خوب است هم بد

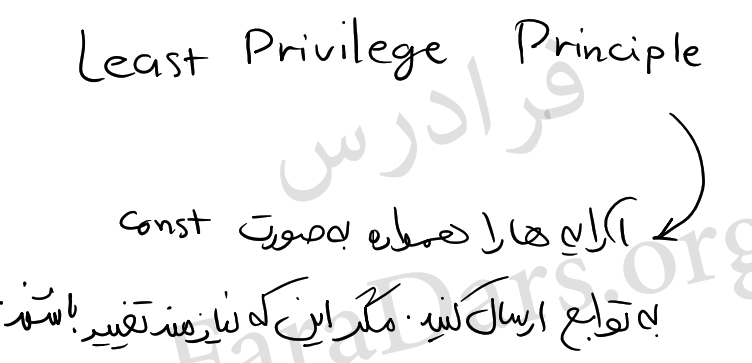
اگر تغییرات و یا مواردی مانند دو برابر شده توان رسانیدن و جذر و یا مرتب سازی باشد را انجام دهیم خیلی مناسب است اما در بعضی مواقع برای ما دچار ایراد می‌کند

**به کارگیری Call By Value:**

اگر تغییرات بر روی یک عدد بدون اشاره به آدرس باشد Call by refrence صورت می‌گیرد

اگر بخواهیم که یک آرایه ای Call by value صورت بگیرد باید کنار نام آرایه در تابع تعریف شده عبارت Const را قرار دهیم

با توجه به اصل اختیارات بهتر است که آرایه ها را با Const به توابع ارسال کنیم که به آن Least privilege principle گویند مگر در مواقعی که مجبور باشیم

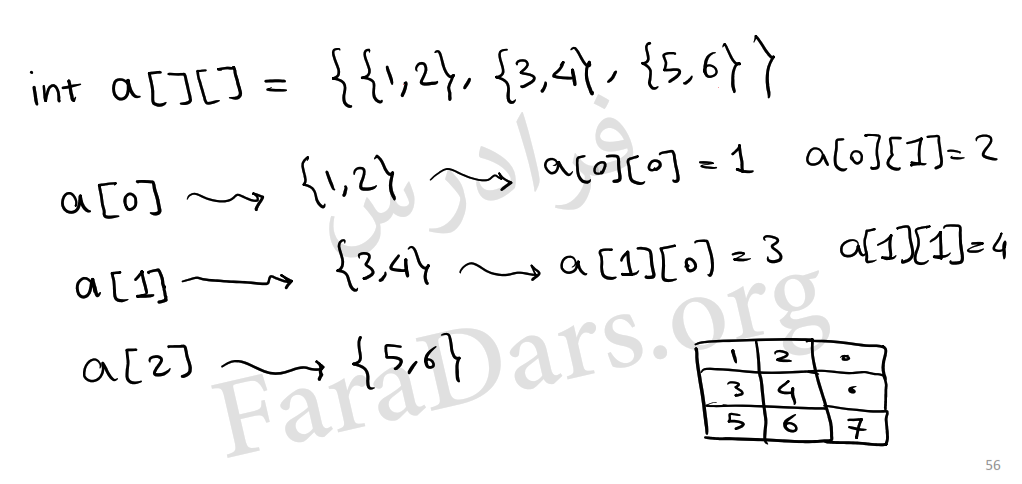


**آرایه های چند بعدی :**

به این نوع از آرایه در حقیقت آرایه ای از آزایه ها می گویند

این آرایه شبیه به ماتریس است

در شکل زیر این آرایه را می‌بینید



تمام این آرایه ها باید ساختار مستطیلی داشته باشند

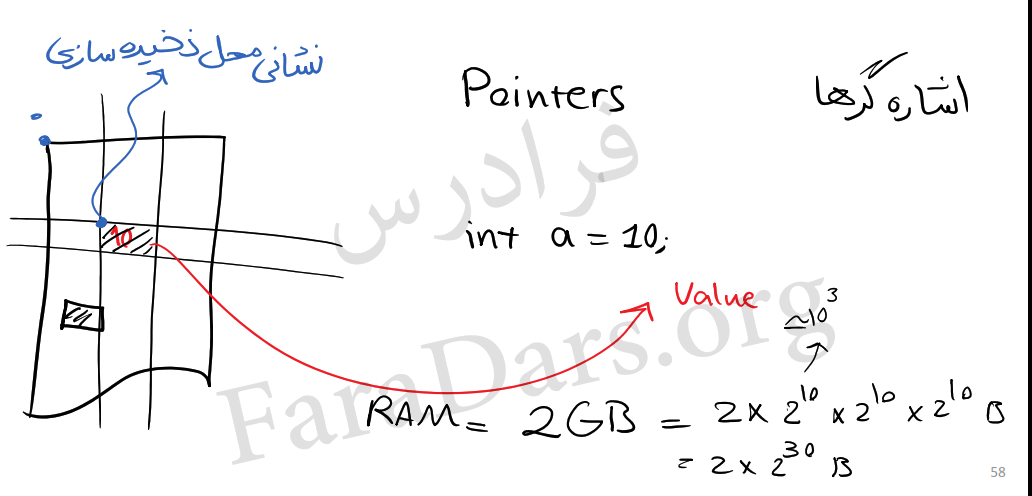
**بسم الله الرحمن الرحیم**

**درس ششم : اشاره گرها**

اشاره گرها یا Pointer ها از امکانات قدرتمند زبان C است اشاره گرها ابزارها ی قدرتمندی هستند که به ما اجازه می دهند که به طور مستقیم با حافظه سروکار داشته باشیم که از نقل و اننقال بی جای اطلاعات و بدون از بین رفتن زمان و سایر موارد جلوگیری کنیم

اشاره گرها به مبحث حافظه بسیار نزدیک است

هر متغیری دارای یک Value است و یک نام دارد و یک آدرس را در حافظه به عنوان مکان ذخیره می‌کند



در حقیقت ما از Pointer ها این توقع را داریم که یک آرگومانی را به تابع که نشان دهنده آدرس است را بدهد

**طریقه نوشتن Pointer :**

Int \*aptr;

عبارت بالا به صورت زیر خوانده می‌شود :

Aptr یک اشاره گر به یک نوع متغیر از نوع ptr است

Aptr آدرس محل زخیره سازی یک متغیر از نوع int است



**چند نکته در خصوص استفاده از Pointer ها :**

1 – از ptr استفاده کنید زیرا ptr نشان دهنده یک Pointer است

Int \*aptr

توجه شود که نام اشاره گر aptr است و \* اشاره به int دارد

2 – حتماً در مقدار دهی \*aptr در اوّلیه مقدار آن را برابر با Null قرار دهیم

که مقدار Null در Header های stdio و stdref قرار دارد

3 – برای چاپ aptr می توان از دستور printf نیز استفاده کرد که آدرس آن را با %p نشان می دهیم

به عنوان مثال

Printf(“%d stored is in %p”,a,&a)

که اوّلی مقدار a و دوّمی آدرس a است

**ثابت کردن محتوا و اشاره‌گر:**

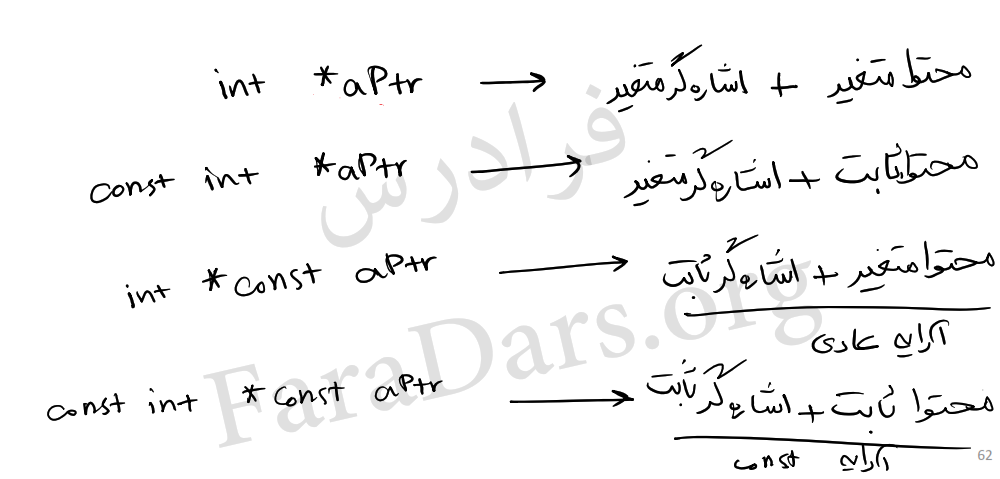
فرض کنیم متغیر a دارای یک مقدار می باشد که در یک قسمت حافظه ذخیره شده است هم مقدار متغیر و هم نشانی می توانند ثابت شوند و هر کدام به صورت یک ثابت می توانند به صورت ثابت فرستاده شوند این بدین معناست که می توانیم یک مقدار را به یک متغیر بدهیم ولی Pointer تغییر نکند و یا یک Pointer به یک تابع بدهیم ولی مقدار تغییر نکند در این حالت چهار ترکیب می توانیم داشته باشیم

|  |  |  |
| --- | --- | --- |
| اشاره گر متغیر | اشاره گر ثابت |  |
| 2 | 1 | محتوا ثابت |
| 4 | 3 | محتوا متغیر |

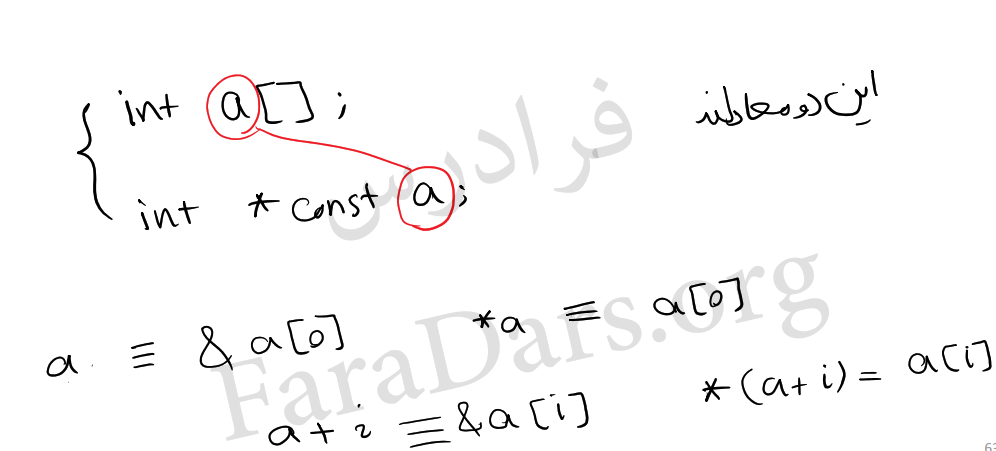
در واقع آرایه ها اشاره‌گر هستن که اشاره‌گر آنها ثابت است

وقتی محتوا متغیر باشد یعنی آرایه به صورت عادی By Refrence ارسال به تابع می شود یعنی حالت 3 اتفاق می افتد

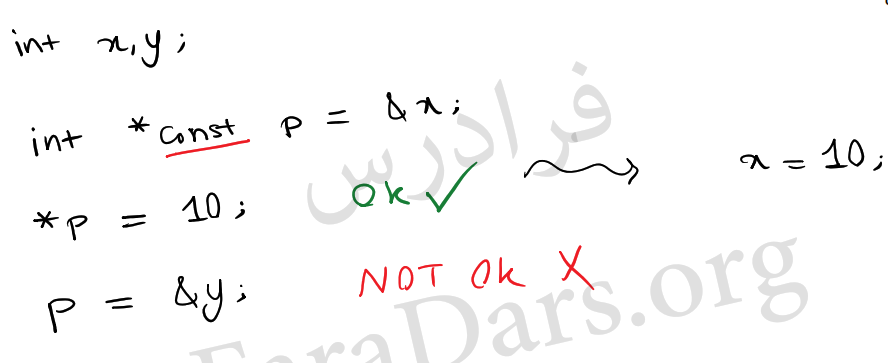
ولی وقتی آرایه را با Const بفرستیم یعنی اشاره‌گر ثابت و محتوا ثابت



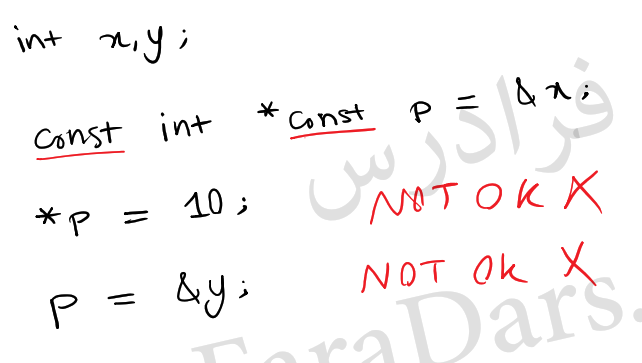
**معادل ها :**



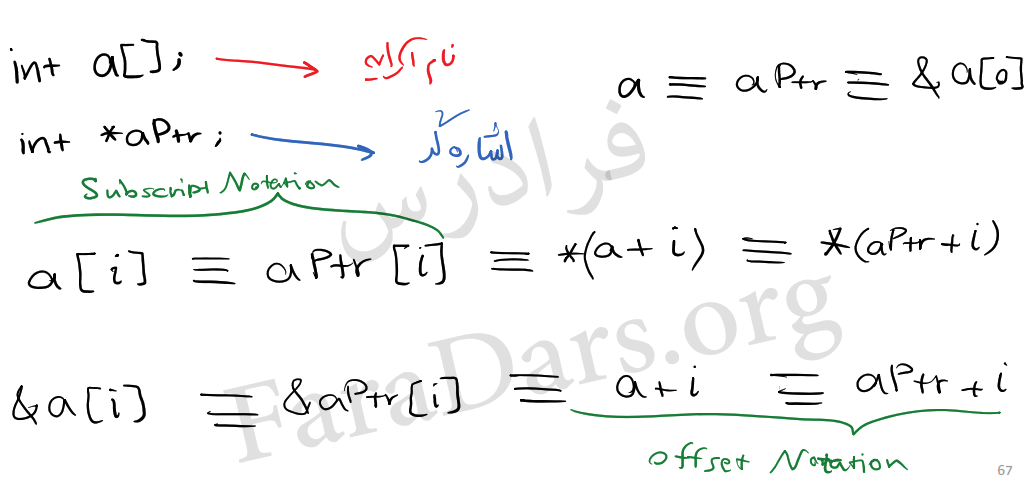
در مثال زیر می توان به یک نوع از محتوا متغیر و محل ثابت را نشان داد



در مثال زیر می توان به یک نوع از محتوا ثابت و محل ثابت را نشان داد

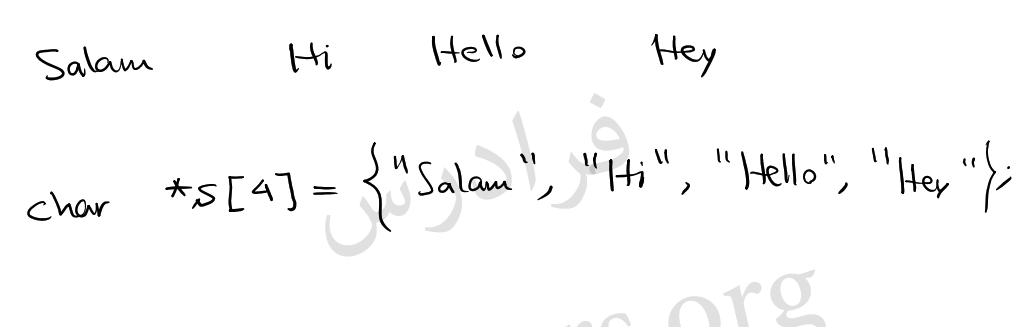






قبلاً با آرایه های مستطیلی آشنا شدیم و دیدیم که یک ضعف بزرگ این آرایه ها این است که الزاماً باید ساختار مستطیلی داشته باشند یعنی اگر بخواهیم فقط یک عنصر به آن اضافه کنیم باید الزاماً ساختاز مستطیلی آن را حغظ کنیم

یکی از کاربردهایی که اشاره گرها دارند این است که می توان آن ها را در به صورت آرایه های مستطیلی در نظر گرفت و موجب می شود که بتوان یک مجموعه ای که دارای طول های اضافی است را با استفاده از اشاره گرها ذخیره نمود



**اشاره گرها به توابع :**

**بسم الله الرحمن الرحیم**

**درس ششم : کاراکترها و رشته ها در زبان C**

رشته ها و کاراکترها اساسی ترین نوع داده برای کار با متن ها هستند

درحقیقت هر برنامه ، الگوریتم و ... که با متن کار می‌کنند از این نوع داده ها استفاده می نمایند برنامه های تایپ و صفحه آرایی را اگر این را ندانیم نمی توانیم با آنها کار کنیم اگر کار با توابع و کاراکترها را ندانیم

رشته ها مجموعه ای از کاراکترها هستند

برای تعریف کاراکتر تنها از ‘ و برای رشته ها از “ استفاده می‌کنیم

در حقیقت وقتی از “ استفاده می‌کنیم یعنی از یک آرایه استفاده می‌کنیم ولی در زمانی که از ‘ استفاده می‌کنیم صرفاً برای معرفی کاراکتر به کار می‌رود

درتعریف آرایه به صورت کاراکتری حتماً باید در انتها یک \0 قرار دهیم که معرف یک کاراکتر Null است علت این Null Character این است که در الگوریتم هایی که در زبان C به کار رفته است زمانی تعریف یک آرایه ختم می شود که الزاماً به Null Character برسد و در حقیقت مشخص کننده انتهای یک String است

آرایه ها می توانند برای اشاره‌گرها هم تعریف شوند به عنوان مثال در زیر

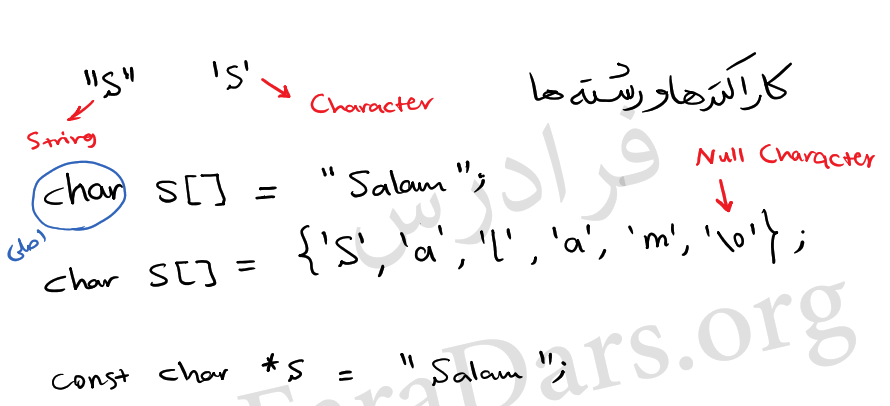
Const char \*s

و این به این معنی است که s یک اشاره‌گر از نوع char است‌ یعنی یک آرایه از نوع کاراکتر است که این عبارت معادل با

Char s[] = “salam”;

است

نوع داده اصلی از نوع character است و در حقیقت از ترکیب این character ها است که باعث ایجاد string می‌شود و لذا در c نوع داده خاصی به نام string نداریم



جهت گرفتن یک رشته از ورودی به صورت زیر عمل می کنیم

Char s[تعداد رشته مورد نظر جهت ذخیره رشته که شامل عنصر صفر هم می شود];

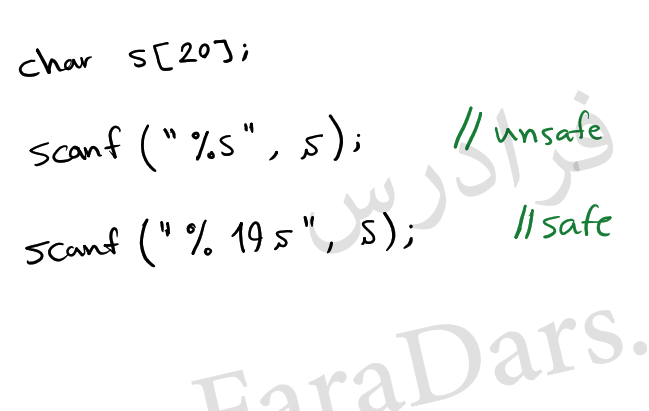
Scanf(“%s,s(نام آرایه مورد نظر));

این حالت ورود ایمن نیست و ما در این حالت بر روی شرایط ورود هیچ تسلطی نداریم

لذا برای یک ورود استاندارد به صورت زیر عمل می‌کنیم

Scanf(“%نام آرایه تعداد عناصر آرایه منهای یک “,نام آرایه);

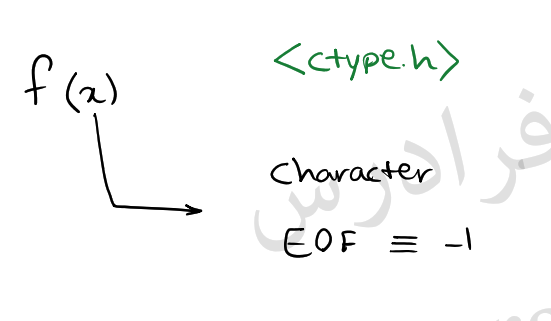
به عنوان مثال :



**کتابخانه ها و توابع برای کار بر روی رشته‌ها :**

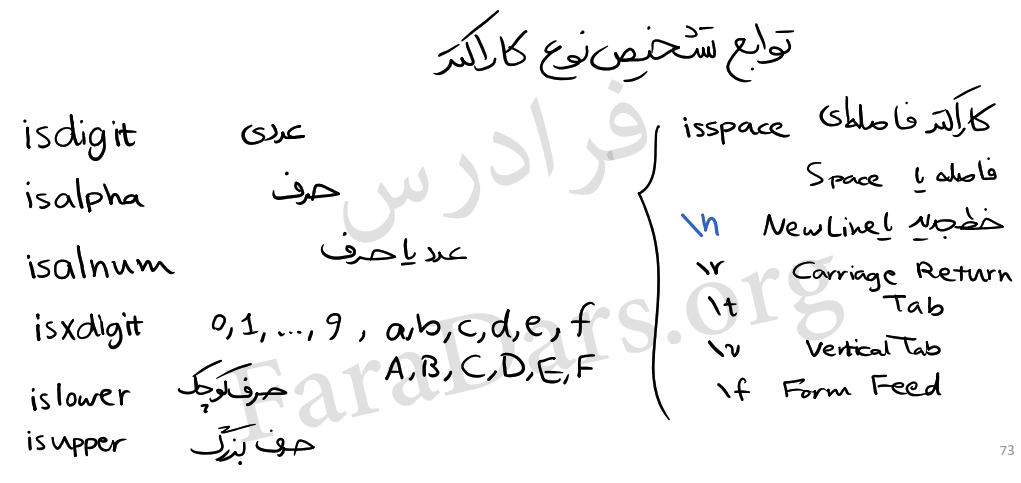


**<ctype.h> :**



این تابع صرفا برای کاراکترها به کار می‌رود مخفف Character type است این توابع ورودی خاصی به نام eof می‌گیرند از جمله این توابع مربوط به تشخیص نوع کاراکتر هستند

که از این توابع میتوان به موارد زیر اشاره کرد



هر چیزی که isprint نباشد کاراکتر کنترلی است مانند \a و عدد اصلی آنها کمتر از 32 است

Isgraph یعنی شکل گرافیکی دارد و تفاوتش با isprint در لحاظ شدن فاصله است

**<string.h> :**

این تابع مهمترین تابع برای رشته ها و کاراکترها می‌باشد

دستور شرطی دیگری داریم که به صورت زیر می‌باشد

(شرط)?A:B

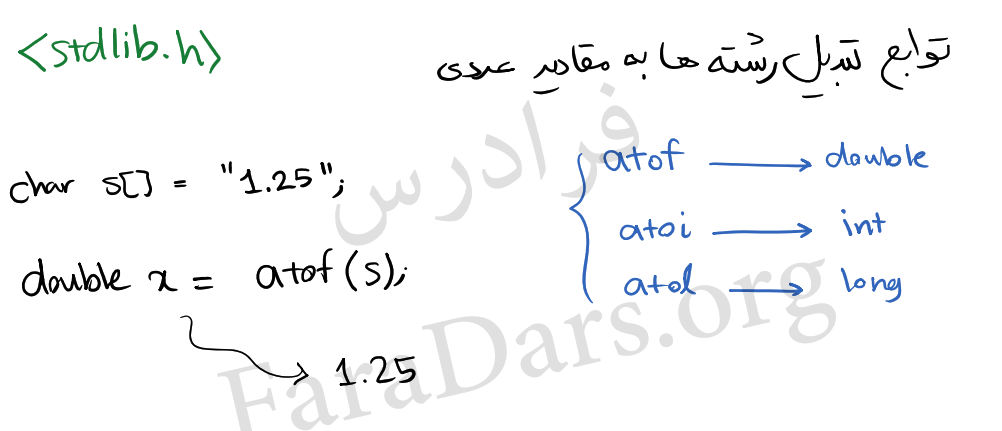
این دستور بدین گونه است که اگر شرط برقرار باشد دستور A و در غیر اینصورت دستور B اجراء می‌شود

با استفاده از دستور ispunch() می توان از ماهیت کاراکتر به یکی از موارد زیر مانند @!#$%^&\*. و غیره اقدام نمود

معرفی تعدادی از توابع که مربوط به stdlib می‌باشند

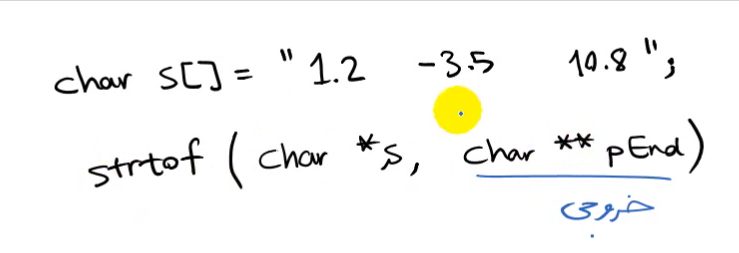
این توابع مربوط به تبدیل رشته ها به اعدادهستند که به آنها parsing گویند

به عنوان مثال اگر بخواهیم یک عدد را تبدیل به یک آرایه با parsing کنیم

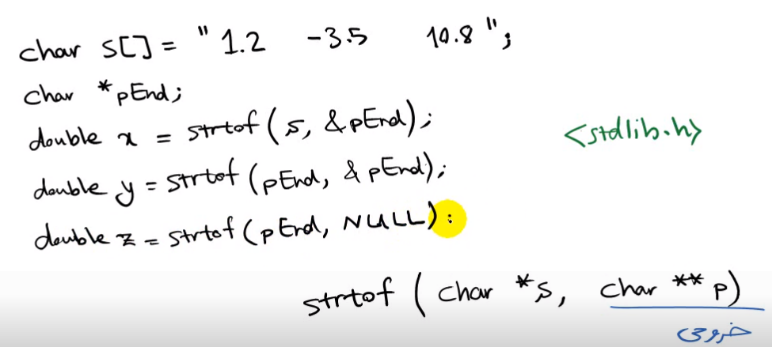


ولی یک حالت پیشرفته تر نیز داریم که مربوط به string هایی است که چند عدد داخل آنها می‌باشد برای اینکار توابع بهتری را درست کرده اند که این توابع علاوه بر اینکه عدد اول را میخوانند شروع بعدی را نیز به ما می‌دهند

اگر فرض کنیم سه عدد داشته باشیم که بخواهیم در آرایه ای باشد و آن را Parse کنیم اگر تابع atof را به کار ببریم فقط عدد اوّل تغییر می‌کند برای این کار یک سری توابع بهتری را وضع کرده اند این توابع علاوه بر خواندن عدد اول شروع عدد بعدی را هم به عنوان خروجی می‌دهند به عنوان مثال تابع strtof را داریم که این تابع دو ورودی می‌‌گیرد که اوّلین ورودی همان آرایه مورد نظر می باشدکه به صورت یک اشاره گر به یک کاراکتر تعریف می‌شود و دوّمی یک اشاره‌گر به اشاره‌گر اوّل می‌باشد که یک Pointer است که به انتهای عدد اوّل و یا کاراکتر اوّل اشاره می‌کند و در واقع قسمت دوّم یک خروجی تابع است

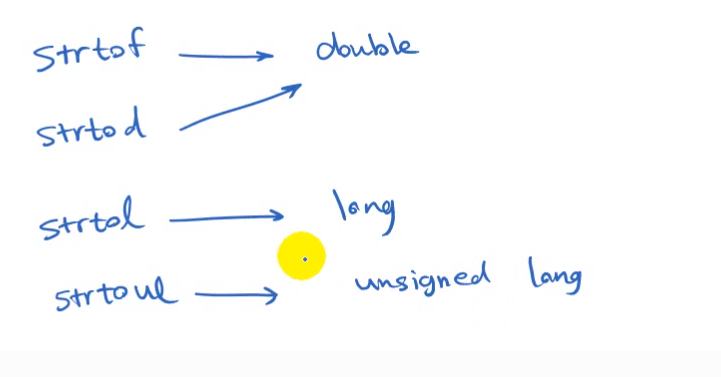


که این مورد در کتابخانه stdlib.h است به عنوان مثال داریم



به جز strtof موارد دیگر را هم داریم که در زیر به آن اشاره می‌کنیم

Strtof , strtod برای اعداد اعشاری و strtol , strtoul برای اعداد صحیح به کار می‌روند



**بسم الله الرحمن الرحیم**

**درس ششم : کاراکترها و رشته ها در زبان C قسمت دوم**

**توابع مورد استفاده در کتابخانه stdio.h**

**تابع Getchar :** برای دریافت کاراکتر از ورودی استاندارد است

**تابع Putchar :** برای چاپ کاراکتر در خروجی استاندارد است

**تابع Gets() :** برای رشته ها به کار می روند دریاغت رشته از ورودی و یا چاپ رشته از خروجی در این مورد زمانی اتمام زمانی که یک خط جدید و یا eof را وارد کنیم و همچنین طول مشخصی را خواهیم داشت

C gets() function

The gets() function enables the user to enter some characters followed by the enter key. All the characters entered by the user get stored in a character array. The null character is added to the array to make it a string. The gets() allows the user to enter the space-separated strings. It returns the string entered by the user.

**Declaration**

1. **char**[] gets(**char**[]);

**تابع fgets() :** برای رشته ها به کار می روند حالت خاصی از gets() است و برای فایل‌ها و رشته های غیر استاندارد هستند از تابع gets امن تر است و مشخص می‌کند از کجا خوانده شود که ممکن است رشته فایل و یا موارد دیگر باشد

char\* fgets(char\* str, int n, FILE\* stream);

* **Arguments**
  + char\* str: pointer to an initialized string in which characters are copied.
  + int n: number of characters to copy.
  + FILE\* stream: pointer to the file stream, this can be replaced by stdin when reading from standard input.
* **Return Type**: On successful read, the pointer to str is returned. If there was an error or the end of file character is encountered before any content could be read, a NULL pointer is returned.

**تابع puts() :** برای رشته ها به کار می روند دریاغت رشته از ورودی و یا چاپ رشته از خروجی

**تابع fputs() :** برای رشته ها به کار می روند حالت خاصی از puts() است و برای فایل‌ها و رشته های غیر استاندارد هستند

دو تابع داریم که خیلی شبیه به توابع printf و scanf می باشند

**Sprintf :** نتیجه را در یک رشته ذخیره می‌کند تنها چیزی که دریافت می‌کند به عنوان اولین پارامتر یک String است و بقیه اش مانند printf است

# sprintf

int sprintf ( char \* str, const char \* format, ... );

**Write formatted data to string**

Composes a string with the same text that would be printed if *format* was used on [printf](https://cplusplus.com/printf), but instead of being printed, the content is stored as a *C string* in the buffer pointed by *str*.  
  
The size of the buffer should be large enough to contain the entire resulting string (see [snprintf](https://cplusplus.com/snprintf) for a safer version).  
  
A terminating null character is automatically appended after the content.  
  
After the *format* parameter, the function expects at least as many additional arguments as needed for *format*.

**Sscanf :** به جای دریافت رشته از ورودی از یک رشته دریافت می‌کند

**تابع strlen :** برای محاسبه طول رشته است

**تابع strerr :** که برای دریافت متن خطا با توجه به شماره خطا می‌باشد

**نکته :** برای معکوس کردن یک رشته از تابع بازگشتی استفاده می‌شود

و به این ترتیب است که اگر یک رشته را در نظر بگیریم مانند abcdef ابتدا a را کنار می‌گذاریم و سپس بقیه را معکوس می‌کنیم بعد b را در نظر می‌گیریم و بقیه را معکوس می‌کنیم و به این ترتیب آنقدر ادامه می‌دهیم تا به اخرین کاراکتر برسیم و آن را نوشته و بقیه را اقدام می‌کنیم برای این کار از یک تابع بازگشتی استفاده می‌کنیم

**تابع getchar() :**

int getchar ( void );

**Get character from stdin**

Returns the next character from the standard input ([stdin](https://cplusplus.com/stdin)).  
  
It is equivalent to calling [getc](https://cplusplus.com/getc) with [stdin](https://cplusplus.com/stdin) as argument.

**تابع Puts :**

این تابع شبیه به printf عمل می کند

int puts ( const char \* str );

**Write string to stdout**

Writes the *C string* pointed by *str* to the *standard output* ([stdout](https://cplusplus.com/stdout)) and appends a newline character ('\n').  
  
The function begins copying from the address specified (*str*) until it reaches the terminating null character ('\0'). This terminating null-character is not copied to the stream.  
  
Notice that puts not only differs from [fputs](https://cplusplus.com/fputs) in that it uses [stdout](https://cplusplus.com/stdout) as destination, but it also appends a newline character at the end automatically (which [fputs](https://cplusplus.com/fputs) does not).

**توابع موجود در کتابخانه string.h :**

این کتابخانه یک کتابخانه اختصاصی بار کار با رشته ها و کاراکترها است

یک سری از توابع جهت کپی و الحاق رشته ها به یکدیگر است که با نام Copy & cat معنا پیدا می‌کند

این توابع عبارتند از strcpy - strncpy – strcat و strncat را داریم که strcpy و strncpy برای کپی به کار می‌رود که کپی کردن یک رشته در یک رشته دیگر به کار می‌رود و strcat و strncat برای الحاق رشته ها به کار می‌رود دو تابع strcmp و strncmp را داریم که برای مقایسه به کار می‌روند و توابعی که برای جستجو به کار می روند مانند strchr که اولین وقوع از یک کاراکتر در یک رشته را نمایش می‌دهد strspn می‌گوید که یک رشته اوّلین بار در یک رشته دیگر کجا دیده می‌شودstrcpn که می گوید آیا یک رشته در یک رشته دیگر دیده نمی‌شود strpbrk که این مجموعه ای از کاراکترها را از یک آرایه و یا رشته ای می‌گیرد و در واقع یکی از آنها را پیدا می‌کند و strrchr آخرین وقوع یک کاراکتر در یک رشته را داریم strtok که رشته را در وافع به چند توکن می‌شکند توکن ها عبارت هایی هستند که یک رشته کلی را می سازند و در واقع کلماتی هستند که با space از هم جدا می‌شوند و یا هر جدا کننده دیگری که کار جدا کردن آنها را این تابع انجام می‌دهد

1. **Strcpy :** خروجی این تابع یک رشته است

# strcpy

char \* strcpy ( char \* destination, const char \* source );

**Copy string**

Copies the C string pointed by *source* into the array pointed by *destination*, including the terminating null character (and stopping at that point).  
  
To avoid overflows, the size of the array pointed by *destination* shall be long enough to contain the same C string as *source* (including the terminating null character), and should not overlap in memory with *source*.

destination

Pointer to the destination array where the content is to be copied.

source

C string to be copied.

### Return Value

*destination* is returned.

*2-***strncpy :**که یک سایز را مشخص می کند

# strncpy

char \* strncpy ( char \* destination, const char \* source, size\_t num );

**Copy characters from string**

Copies the first *num* characters of *source* to *destination*. If the end of the *source* C string (which is signaled by a null-character) is found before *num* characters have been copied, *destination* is padded with zeros until a total of *num* characters have been written to it.  
  
No null-character is implicitly appended at the end of *destination* if *source* is longer than *num*. Thus, in this case, *destination* shall not be considered a null terminated C string (reading it as such would overflow).  
  
*destination* and *source* shall not overlap (see [memmove](https://cplusplus.com/memmove) for a safer alternative when overlapping).

### Parameters

destination

Pointer to the destination array where the content is to be copied.

source

C string to be copied.

num

Maximum number of characters to be copied from *source*.  
[size\_t](https://cplusplus.com/cstring:size_t) is an unsigned integral type

بهتر است در استفاده از این function حتماً آخرین عضو را برابر با Null د رنظر بگیریم

**3 - Strcat :** این تابع یک Source را در Destination ادغام می‌کند

function

<cstring>

# strcat

char \* strcat ( char \* destination, const char \* source );

**Concatenate strings**

Appends a copy of the *source* string to the *destination* string. The terminating null character in *destination* is overwritten by the first character of *source*, and a null-character is included at the end of the new string formed by the concatenation of both in *destination*.  
  
*destination* and *source* shall not overlap.

### Parameters

destination

Pointer to the destination array, which should contain a C string, and be large enough to contain the concatenated resulting string.

source

C string to be appended. This should not overlap *destination*.

**Strcmp :** این تابع جهت مقایسه بین دو رشته به کار می‌رود

function

<cstring>

# strcmp

int strcmp ( const char \* str1, const char \* str2 );

**Compare two strings**

Compares the C string *str1* to the C string *str2*.  
  
This function starts comparing the first character of each string. If they are equal to each other, it continues with the following pairs until the characters differ or until a terminating null-character is reached.  
  
This function performs a binary comparison of the characters. For a function that takes into account locale-specific rules, see [strcoll](https://cplusplus.com/strcoll).

### Parameters

str1

C string to be compared.

str2

C string to be compared.

خروجی این تابع به صورت int است اگر این دو تابع با هم برابر باشند مقدار صفر را می‌دهد اگر str1 از نظر ارزش بزرگتر باشد آن مقدار مثبت خواهد داد و اگر str1 کوچکتر باشد آن مقدار منفی خواهد داد

### Return Value

Returns an integral value indicating the relationship between the strings:

|  |  |
| --- | --- |
| **return value** | **indicates** |
| <0 | the first character that does not match has a lower value in *ptr1* than in *ptr2* |
| 0 | the contents of both strings are equal |
| >0 | the first character that does not match has a greater value in *ptr1* than in *ptr2* |

**Strchr :**

این تابع نشان می‌دهد کجا اوّلین وقوع از یک کاراکتر اتفاق می‌افتد

function

# strchr

const char \* strchr ( const char \* str, int character );

char \* strchr ( char \* str, int character );

**Locate first occurrence of character in string**

Returns a pointer to the first occurrence of *character* in the C string *str*.  
  
The terminating null-character is considered part of the C string. Therefore, it can also be located in order to retrieve a pointer to the end of a string.

### Parameters

str

C string.

character

Character to be located. It is passed as its int promotion, but it is internally converted back to *char* for the comparison.

### Return Value

A pointer to the first occurrence of *character* in *str*.  
If the *character* is not found, the function returns a null pointer.

**تابع strspn :**

این تابع یک رشته را در یک رشته دیگر جستجو می کند طول قسمتی از str1 را برمی‌گرداند که فقط شامل کاراکترهایی است که در str2 است

# strspn

size\_t strspn ( const char \* str1, const char \* str2 );

**Get span of character set in string**

Returns the length of the initial portion of *str1* which consists only of characters that are part of *str2*.  
  
The search does not include the terminating null-characters of either strings, but ends there.

### Parameters

str1

C string to be scanned.

str2

C string containing the characters to match.

### Return value

The length of the initial portion of *str1* containing only characters that appear in *str2*.  
Therefore, if all of the characters in *str1* are in *str2*, the function returns the length of the entire *str1* string, and if the first character in *str1* is not in *str2*, the function returns zero.  
[size\_t](https://cplusplus.com/cstring:size_t) is an unsigned integral type.

**تابع strcspn :**

اولین وقوعی که در دو رشته یکسان اتفاق می افتد در کدام محل است

function

<cstring>

# strcspn

size\_t strcspn ( const char \* str1, const char \* str2 );

**Get span until character in string**

Scans *str1* for the first occurrence of any of the characters that are part of *str2*, returning the number of characters of *str1* read before this first occurrence.  
  
The search includes the terminating null-characters. Therefore, the function will return the length of *str1* if none of the characters of *str2* are found in *str1*.

### Parameters

str1

C string to be scanned.

str2

C string containing the characters to match.

### Return value

The length of the initial part of *str1* **not** containing any of the characters that are part of *str2*.  
This is the length of *str1* if none of the characters in *str2* are found in *str1*.  
[size\_t](https://cplusplus.com/cstring:size_t) is an unsigned integral type.

**تابع strstr :**

این تابع اولین وقوع یک رشته را نشان می‌دهد

### Portability

In C, this function is only declared as:  
  
char \* strstr ( const char \*, const char \* );  
  
instead of the two overloaded versions provided in C++.

### Example

|  |  |  |
| --- | --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 | /\* strstr example \*/  #include <stdio.h>  #include <string.h>  int main ()  {  char str[] ="This is a simple string";  char \* pch;  pch = strstr (str,"simple");  if (pch != NULL)  strncpy (pch,"sample",6);  puts (str);  return 0;  } | [Edit & Run](https://cplusplus.com/reference/cstring/strstr/?kw=strstr) |

This example searches for the "simple" substring in *str* and replaces that word for "sample".  
  
Output:

|  |
| --- |
| This is a sample string |

**بسم الله الرحمن الرحیم**

**درس دهم : فالب بندی ورودی و خروجی**

مهمترین عملیاتی هستند که در خصوص ارتباط بین کامپیوتر با دنیای بیرون از ا« که چه سخت افزار باشد و چه نرم افزار با آن سرو کار داریم

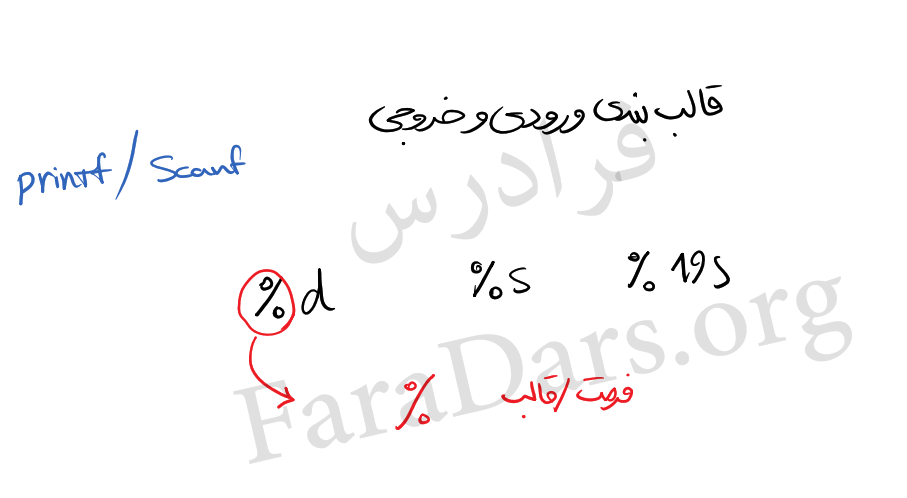
Printf وscanf دو تابع مهم برای ورودی و خروجی استاندارد هستند و یکی از مهمترین ویژگی های آنها Formated بودن آنهاست

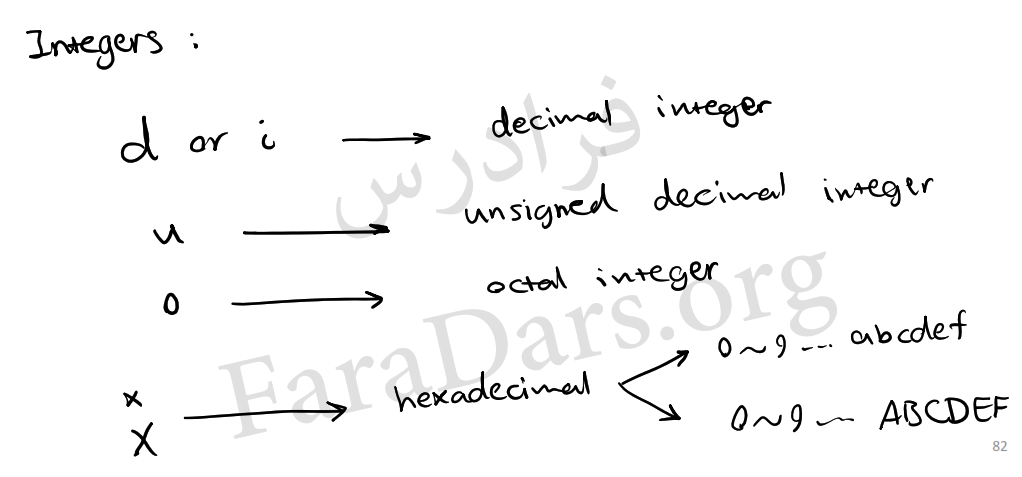
Format Specifier :

این فرمت ها مانند %d و %s هستند علامت % در آنها خیلی مهم است که در انتهای آن فرمت یا قالب را نمایش می‌دهد

**انواع فرمت ها :**

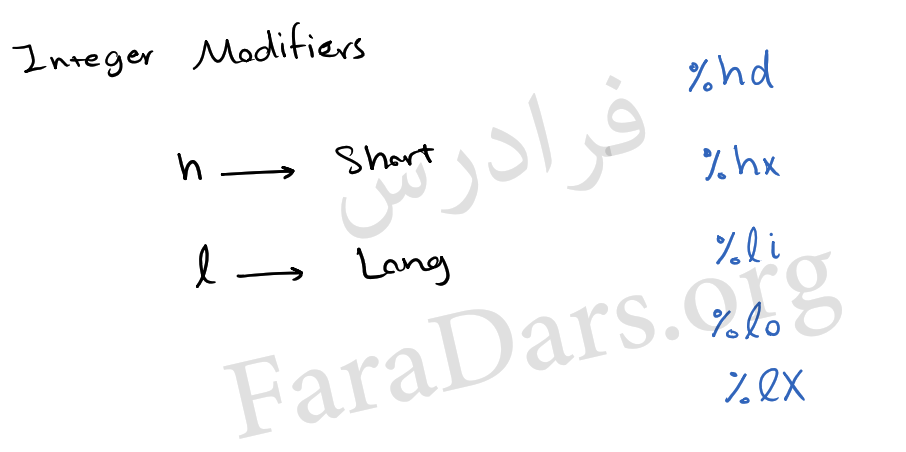
**Integers -1 :**





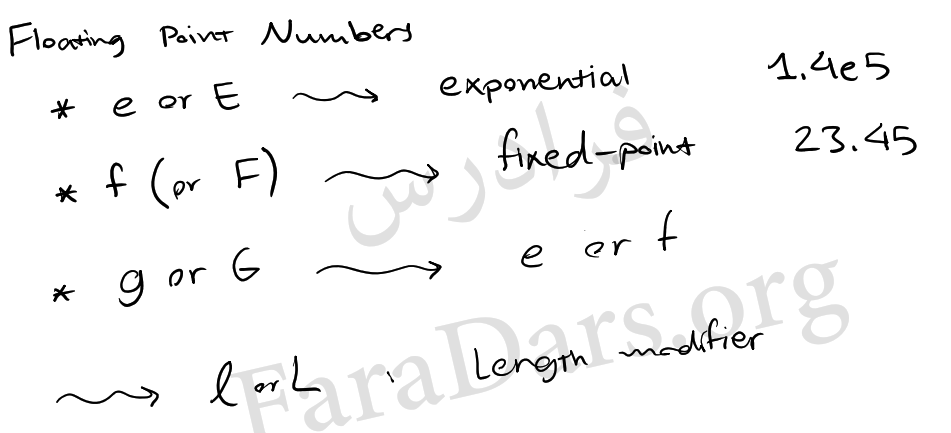
این‌ موارد نوع عدد و شیوه نمایش آن را نشان می‌دهند

دو flag تکمیلی دیگر هم داریم که اینها بیشتر تکمیلی هستند



این دو flag تکمیلی هستند که در آن h برای کوتاه کردن و l برای بزرگ کردن به کار می رود

**2 – اعداد غیر صحیح :**



در اعداد بالا g یا G بسته به نوع عدد از e یاf استفاده می‌کند

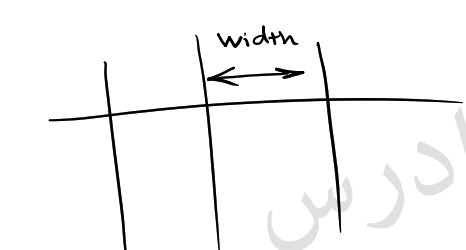
**3 – string or character :**



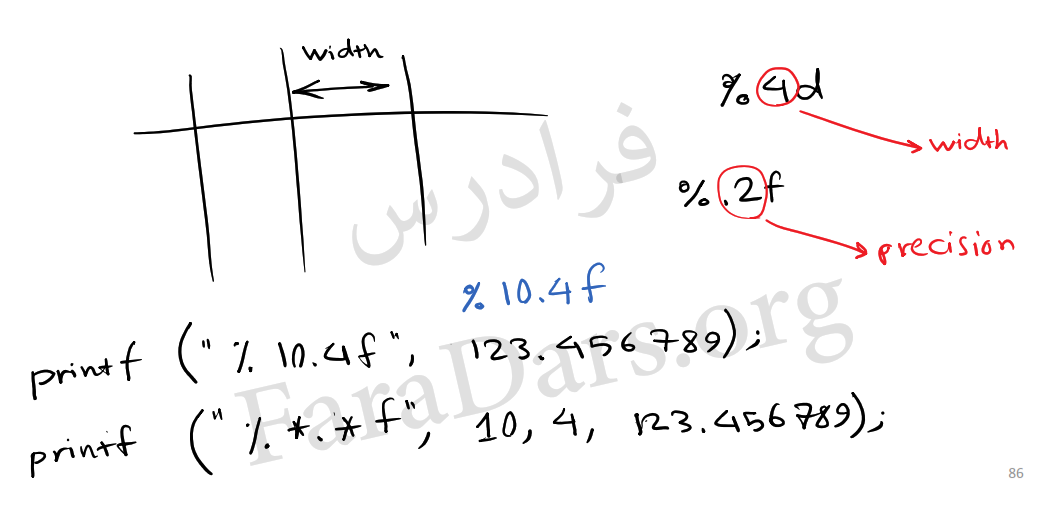
برای چاپ علامت % نیز از دو بار علامت % استفاده می‌کنیم

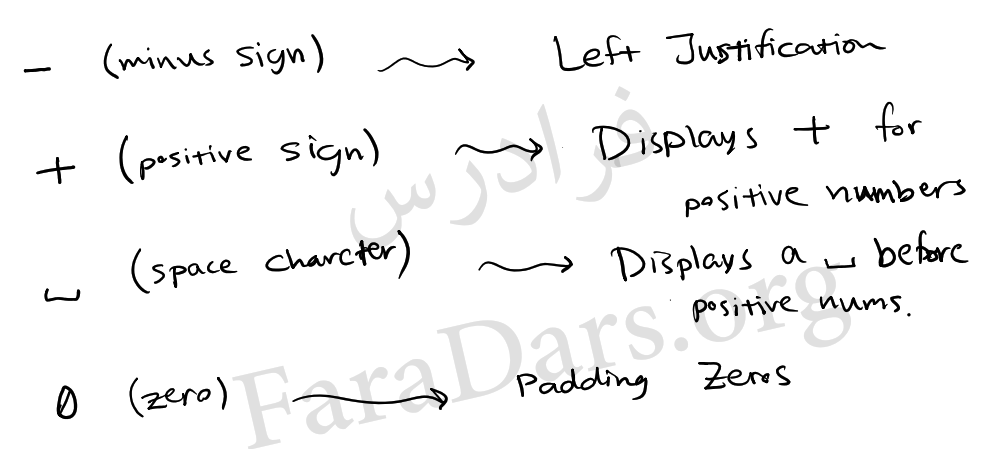
Poll و دقت :

اگر یک ستون داشته باشیم می‌توانیم عرض ستون ها را به صورت دقیق تعیین کنیم



به عنوان مثال %4d واحدی که در داخل ستون نمایش داده می‌شود به مقدار 4 کاراکتر است در کنار اینها ما دقت را هم داریم که به عنوان مثال %.2f یعنی تا دو رقم اعشار را نشان بدهد که می توان آنها را با هم ترکیب نمود بع عنوان مثال %10.4f یعنی عرض 10 و با دقت 4 رقم اعشار آن را نشان بده

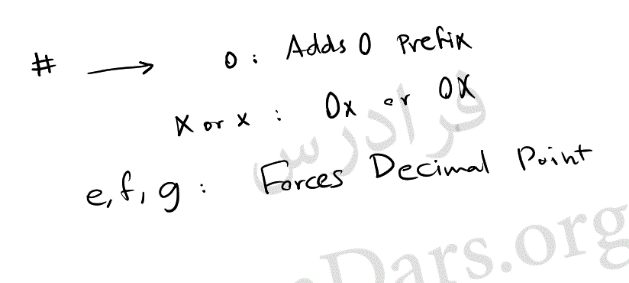




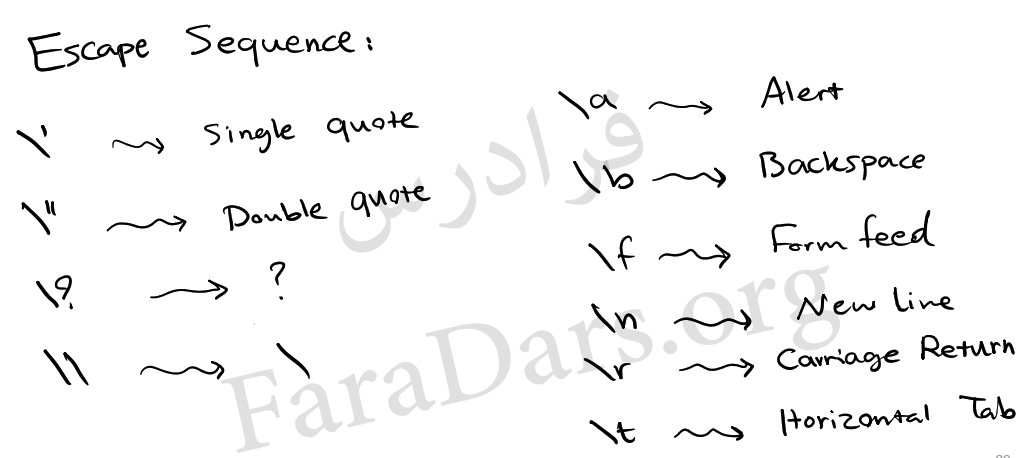
**علامت # :**

اگر در حالت octal باشد یک o در اوّل اضافه می‌کند

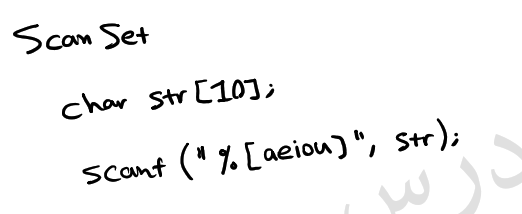
اگر X یا x باشد آن را به OX یا ox تبدیل می‌کند

اگر در e و f و g و یا متناظرهای آن را داشته باشیم الزاماً باعث ایجاد کاراکتر ممیز می‌شود

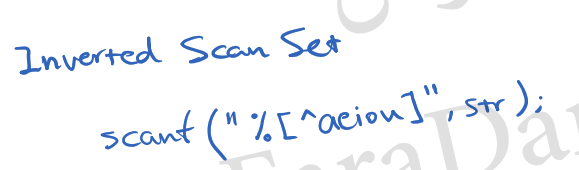
**Escape secuence :**



**Scan set :**



**Inverted scan set :**

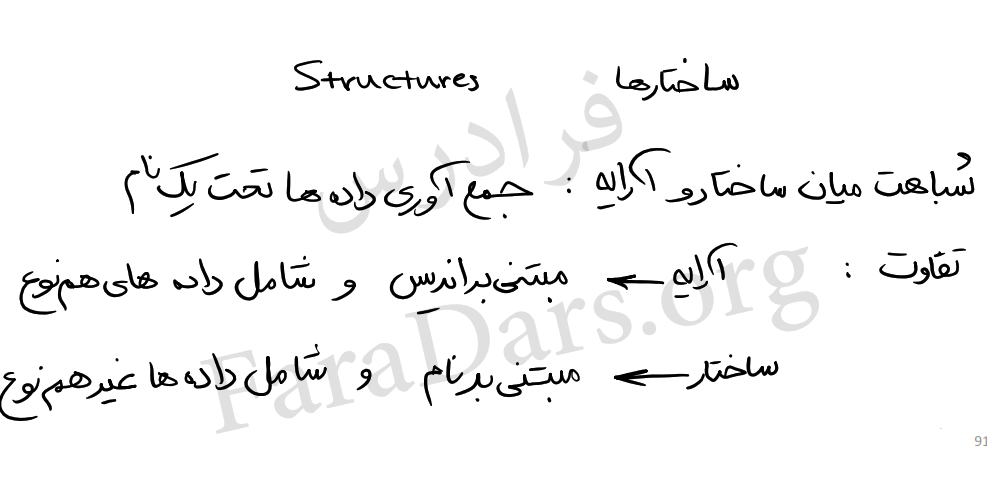


**بسم الله الرحمن الرحیم**

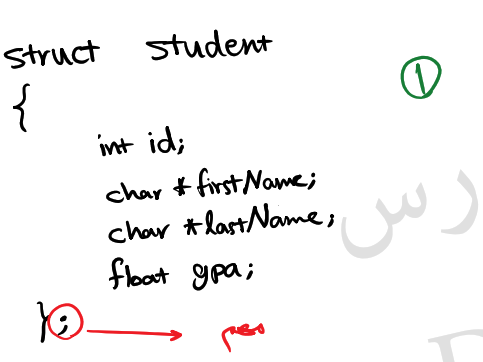
**درس یازدهم : ساختار (Structurs) و یونیون (union) :**

ساختارها کارکردی مشابه آرایه ها دارند که داده ها را یک جا جمع می‌کنند که ما بتوانیم به یک اسم به آنها ارجاع کنیم اما یک تفاوتی دارند که ما در آرایه ها داده‌های هم نوع را یک جا جمع می‌کنیم و اینکه اندیس آنها یکسان است و انواع مختلف داده نمی توانیم داشته باشیم

**شباهت و تفاوت بین آرایه و Struct :**



**نحوه تعریف struct :**

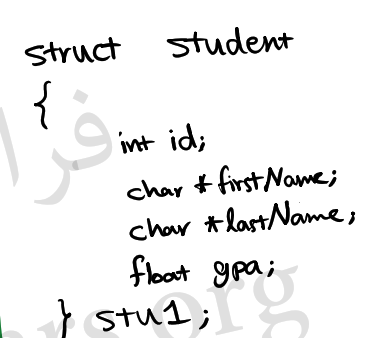


نکته مهم این است که باید آن را با ; ببندیم

بعد از تعریف آن باید یک متغیر از نوع Student را تعریف می‌کنیم

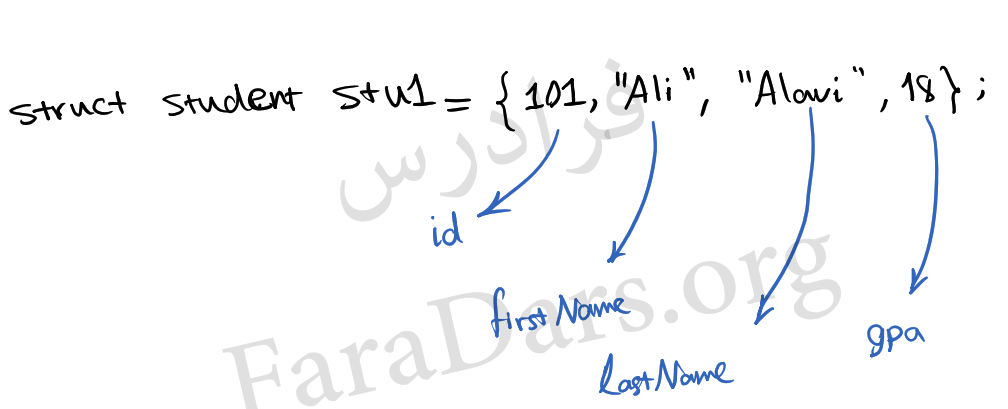


یک روش دیگر تعریف این متغیر به صورت زیر است

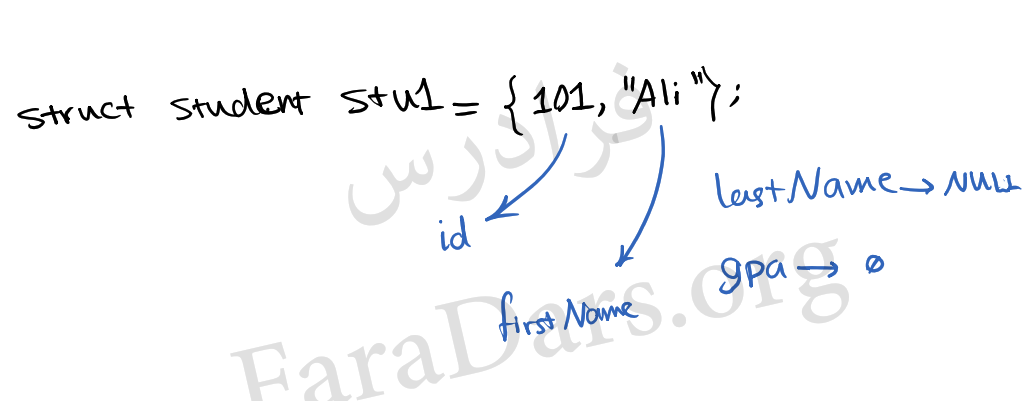


**روش های مقدار دهی به Struct ها :**

**روش اوّل : در این روش به صورت مستقیم مقدار دهی می شود**

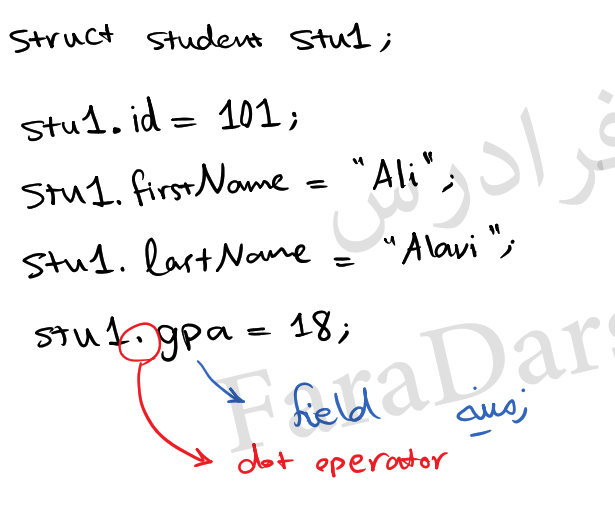


اگر مقدار دهی برای بعضی از متغیرها قرار نگیرد در این حالت مقدار Null برای مقدار Refrence و مقدار صفر برای مقادیر عددی درنظر گرفته می شود



**روش دوّم با استفاده از مقدار دهی از اپراتور . می‌باشد :**

که در این حالت به مقادیر بعد از . Field گفته می‌شود که از اپراتور . برای دسترسی به این فیلدها استفاده می‌شود



**تعریف اشاره گر برای Struct :**

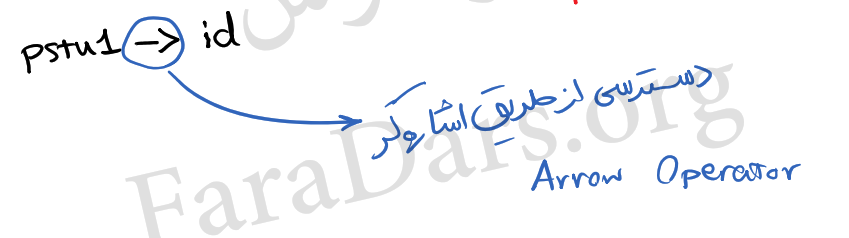


برای دسترسی به فیلدها در Struct دو روش وجود دارد

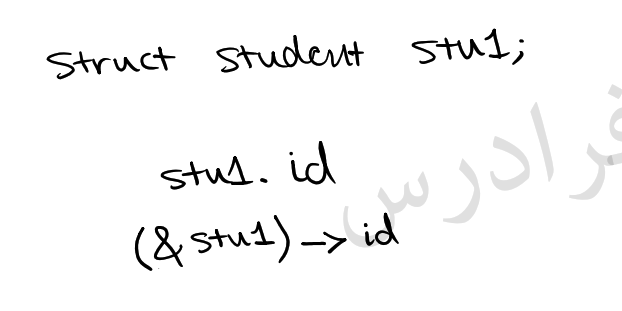
در روش اوّل که روش مستقیم است با استفاده از . می توان به فیلد با همان نام Struct دسترسی پیدا کرد



روش دیگر با استفاده از اپراتور Arrow یا فلش می باشد

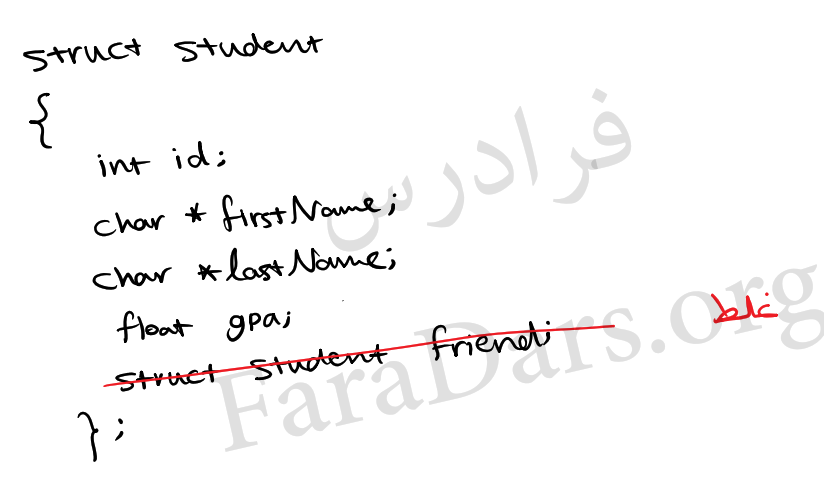


که معادل آن به صورت زیر می‌باشد

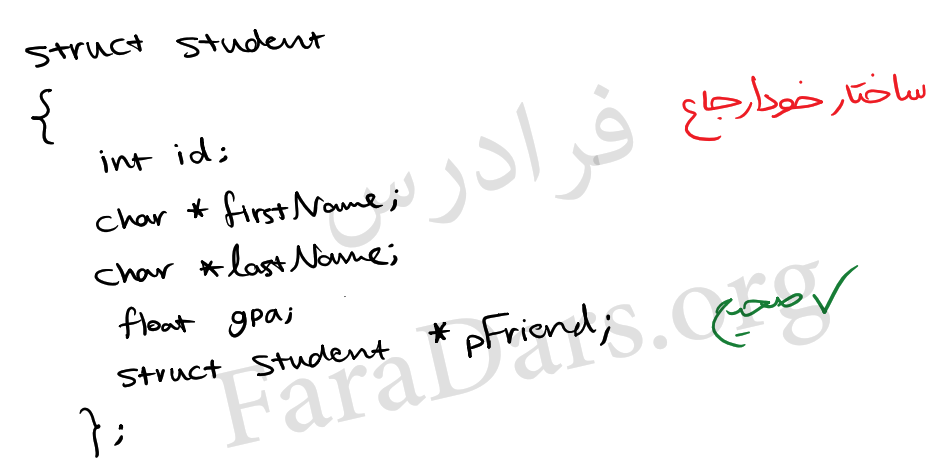


که در آن &stu1 محل stu1 است در حافظه

نمی توان Struct را در داخل یک Struct دیگر به صورت مستقیم نوشت

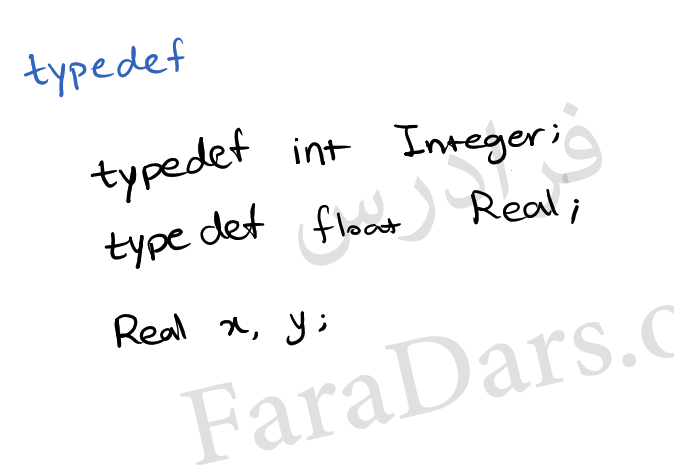


برای اینکار باید از یک اشاره‌گر استفاده نمودکه در این حالت خود ارجاع می باشد



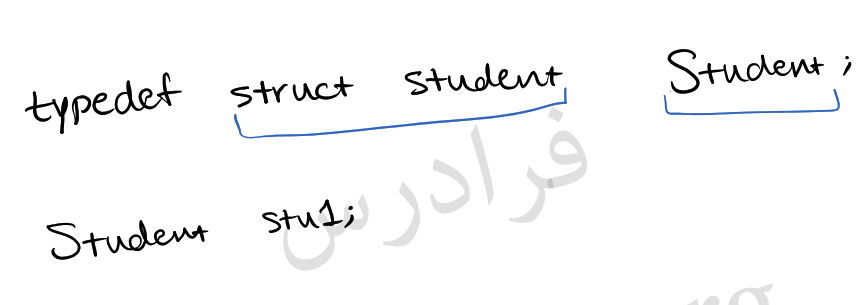
**دستور Type def :**

این دستور برای انتقال پذیری در برنامه به کار می‌‌رود با استفاده از این دستور می توان یک متغیر را تعویض کرد و یا نام دیگری داد

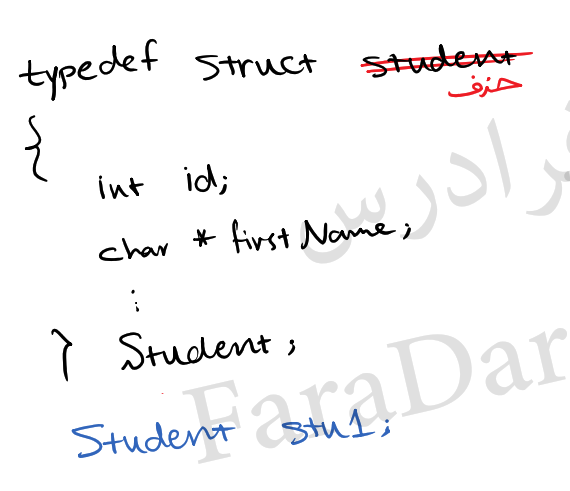


**استفاده typedef در ساختمان داده ها :**

برای اینکار از روش زیر استفاده می‌کنیم



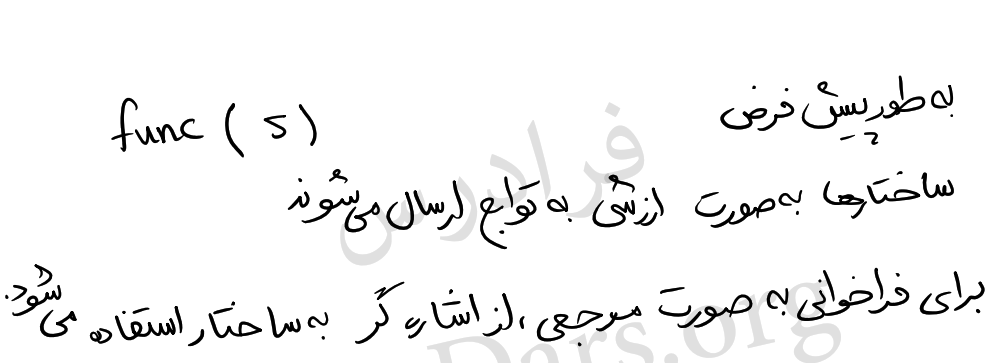
روش بهتری برای نوشتن وجود دارد که به صورت زیر می باشد



**نحوه ارسال Struct ها به توابع :**

به صورت پیش فرض Structure به صورت By Value به توابع ارسال می شوند

برای ارسال به صورت ByRefrence با استفاده از اشاره‌گر اقدام می‌نماییم

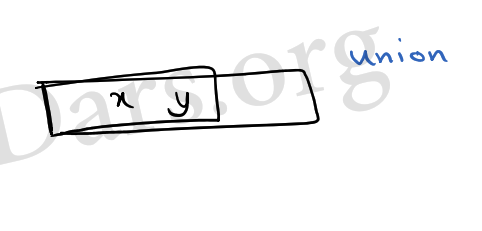


**مبحث یونیون (ساختارهای متحد ) :**

یکی از کاربردهای Union ها جهت استفده در ایجاد انواع داده‌های منعطف است یعنی داده‌ای که بتواند انواع داده‌ها را پوشش بدهد

تفاوت Union با Structurs در این است که در Structure به ازاء هر متغیر یک بلوک داده اشغال می‌شود ولی در Union این گونه نیست ولی در Union کلاً یک بلوک را اشغال می‌کند و از یک فضای مشترک استفاده می‌کنند

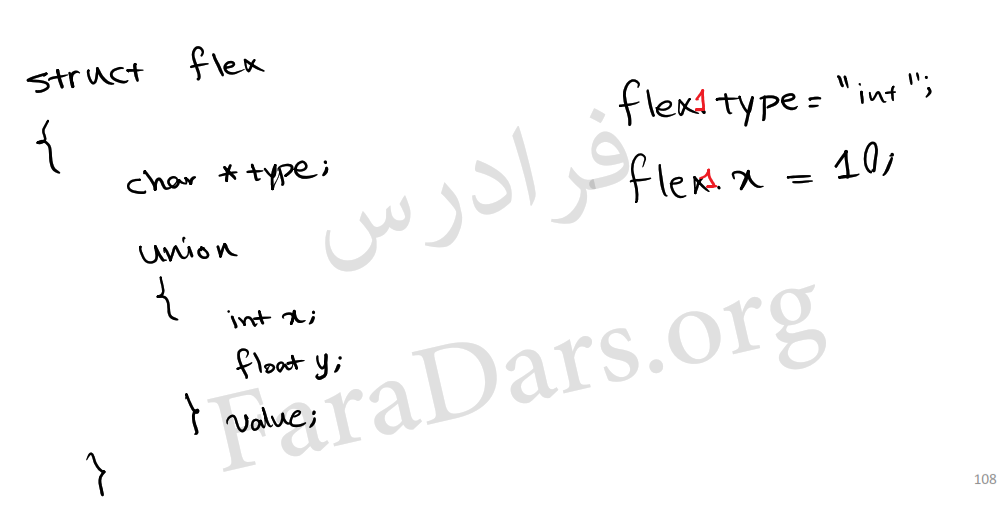




**نحوه تعریف Union :**



**نمونه ای Union :**



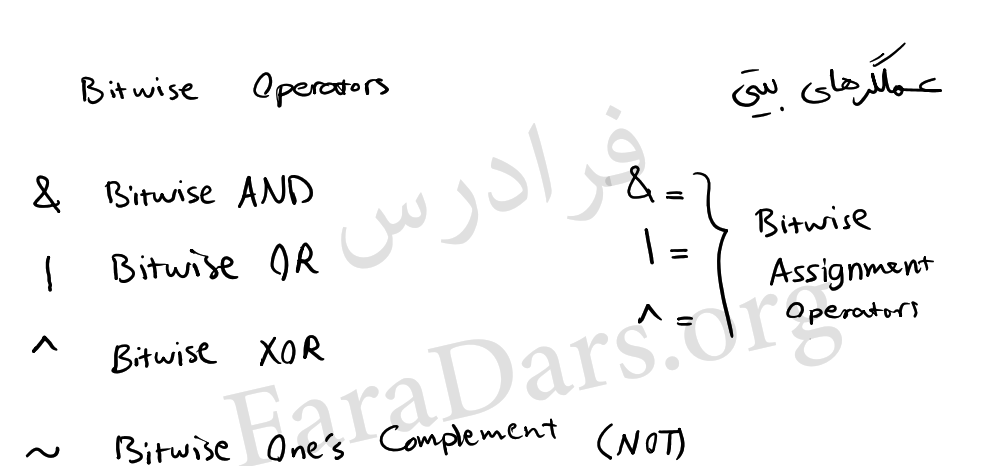
**بسم الله الرحمن الرحیم**

**درس یازدهم : عملگرهای بیتی ، فیلدهای بیتی ، ثوابت نام دار و شمارشی:**

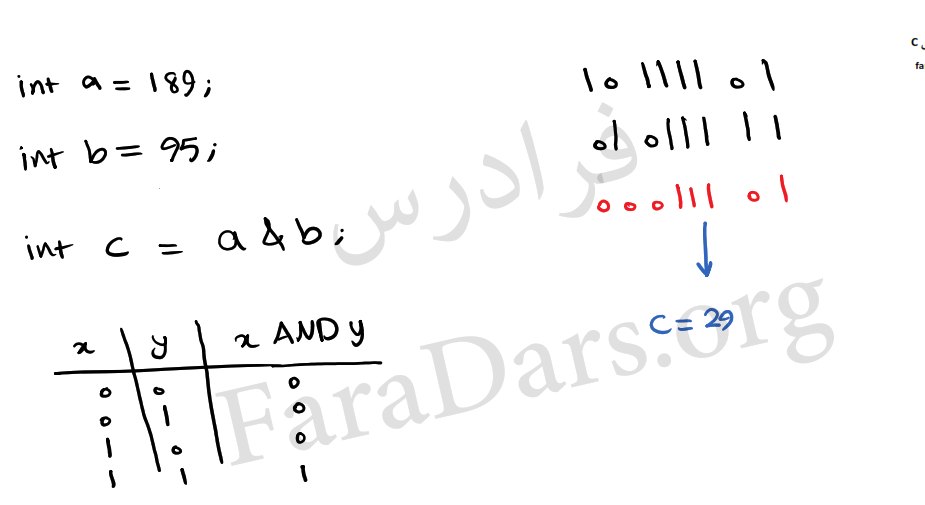
**عملگرهای بیتی :**

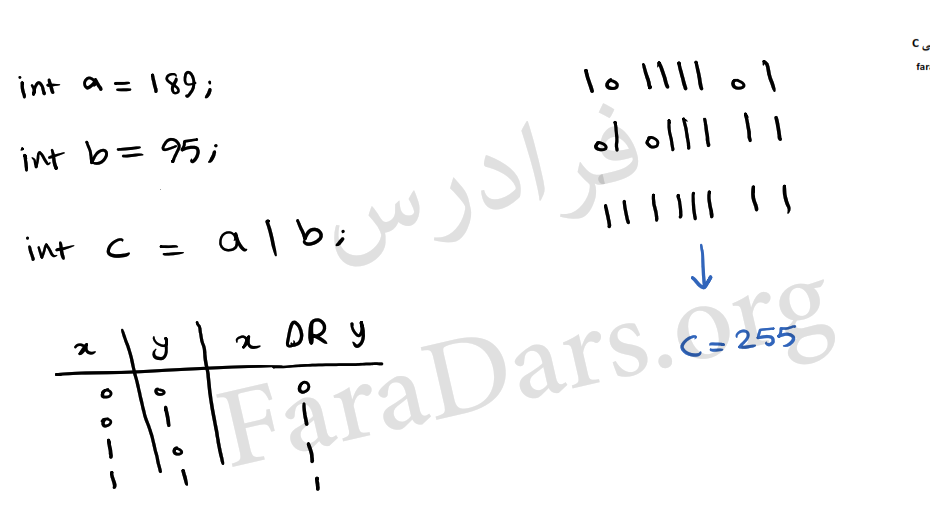
عملگرهای بیتی مشابه با عملگرهای منطقی هستند که بر روی اطلاعات می‌توانند اعمال شوند

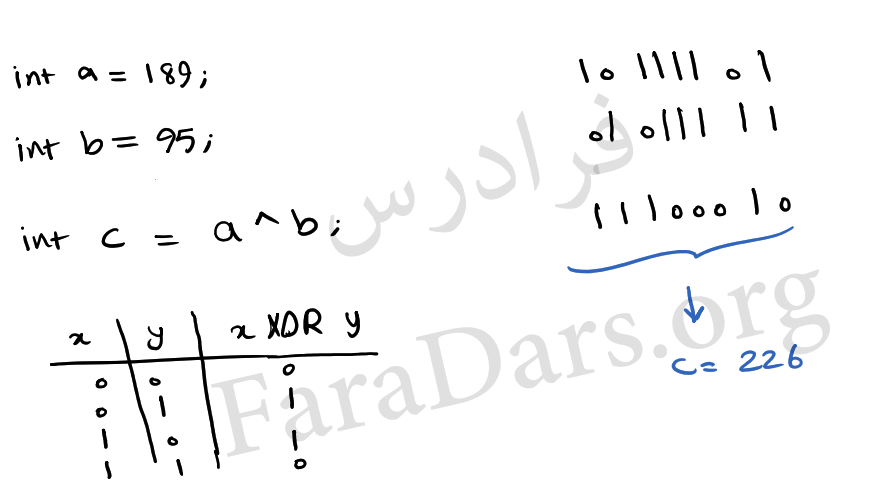
**انواع عملگرهای بیتی :**

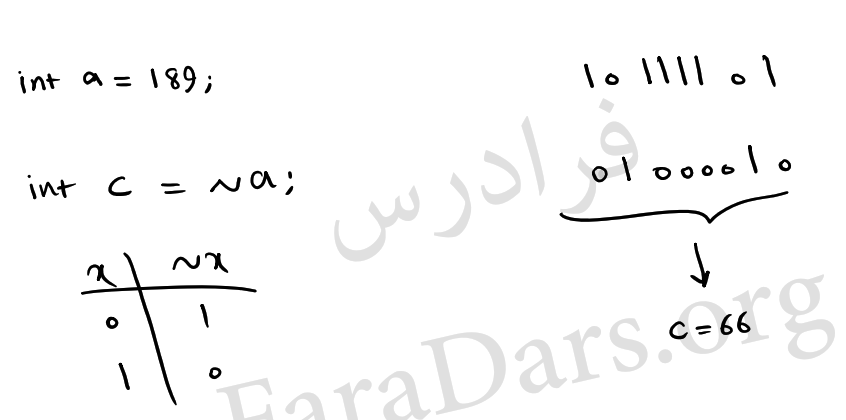


**مثالی از عملگرهای بیتی :**

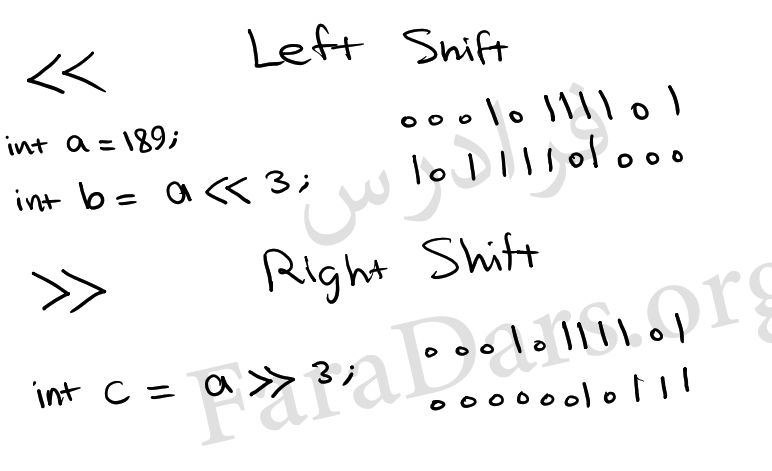




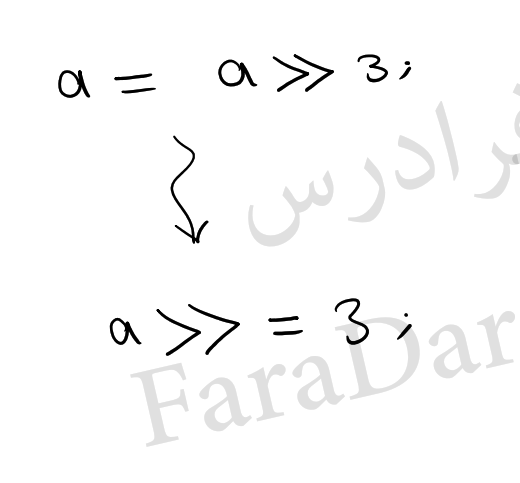




**عملگرهای بیتی Left shift و Right shift :**

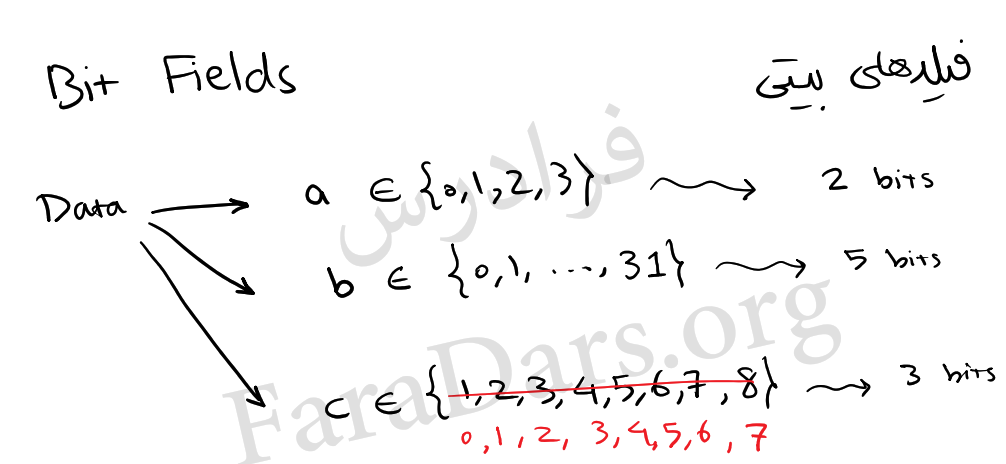


**اپراتورهای معادل Shift :**

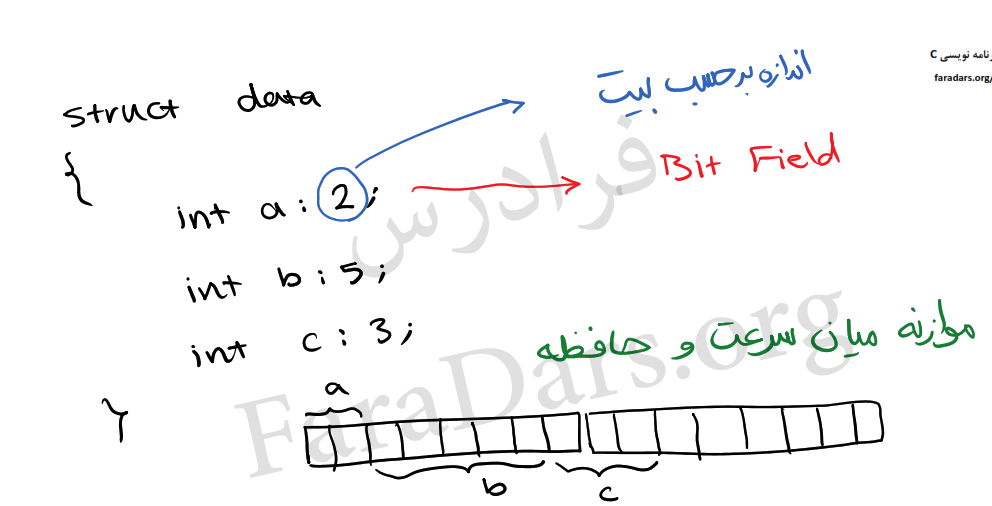


**فیلدهای بیتی Bit field :**

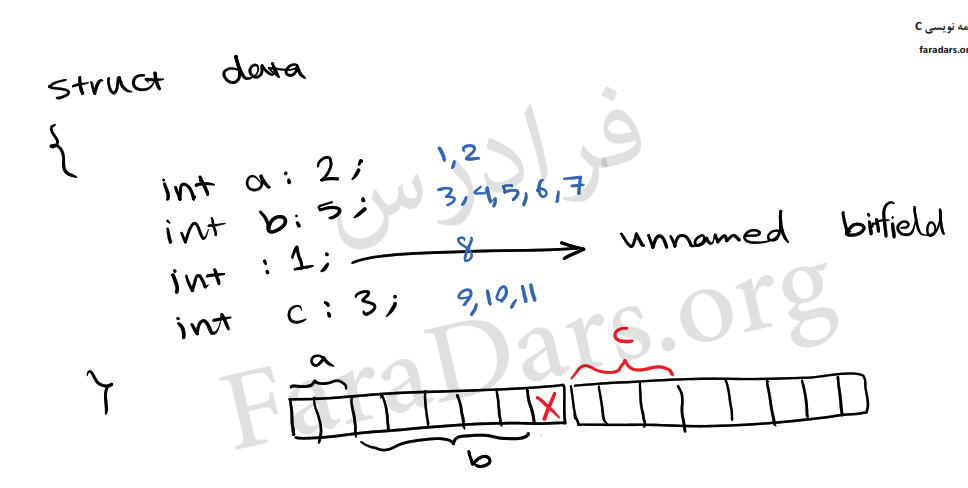
**این بیت ها باعث استفاده بهینه در Structur ها می شوند**



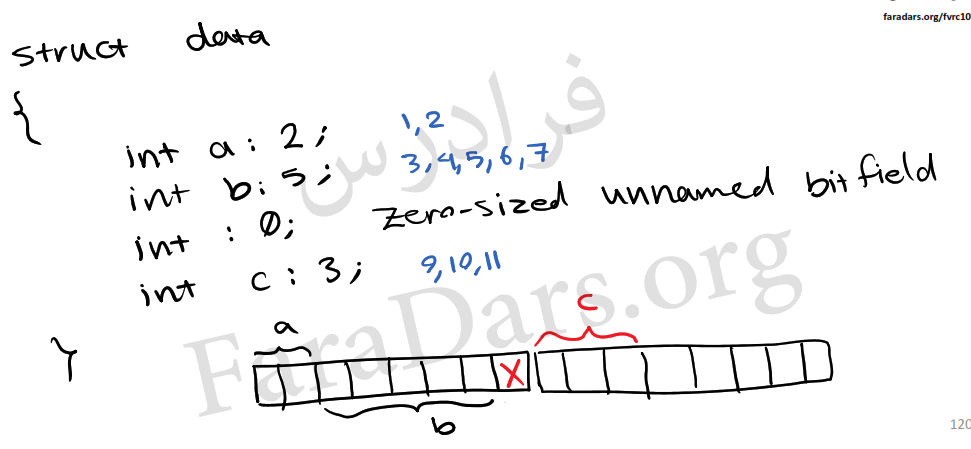
**در صورتی که بخواهیم مدیریت حافظه کنیم و مقدار کمی از یک بایت را اشغال کنیم به صورت زیر عمل می‌کنیم**



**اگر در نظر داشته باشیم که به عنوان مثال از بایت دوم شروع کنیم**



**اگر ندانیم بین دو بایت جقدر فاصله است به صورت زیر عمل می‌کنیم**

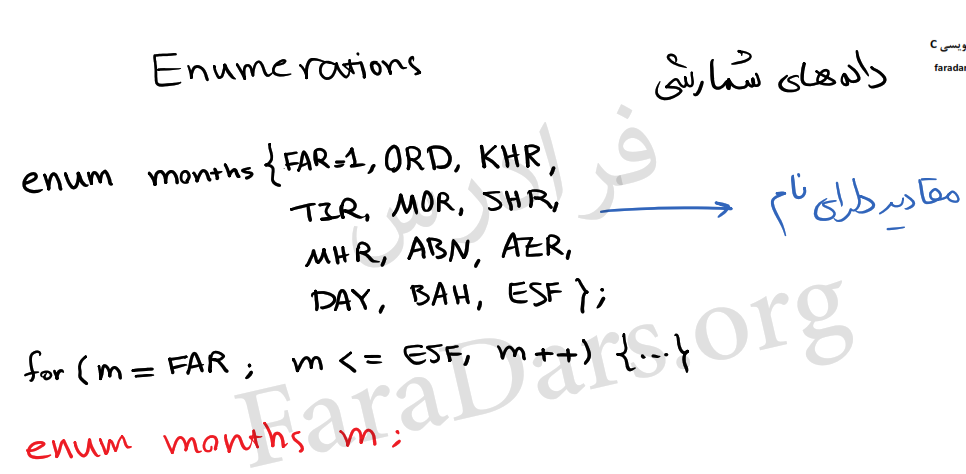


**\**

**Emumaration (داده های شمارشی) :**

ثوابتی هستند که می توانند داده های شمارشی را متناظر کنند و می توان روی اعداد اسم گذاشت

به عنوان مثال اگر بخواهیم ماه‌های سال را با داده های شمارشی نشان دهیم به صورت زیر اقدام می‌کنیم



استفاده از دستور Typedef در داده های شمارشی :

