

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

آموزش برنامه نویسی

C++

# C++ Programming Tutorial

گردآورنده : عباس عزیز جلالی

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

# فصل اول

آشنائی با

C++

و محیط کامپایلر

# دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

یک برنامه به زبان C:

```
/* P1.1 First Program */  
  
#include <stdio.h>  
  
main()  
{  
    int a, b;  
  
    scanf("%d %d" , &a, &b);  
    printf("\n%d", a + b);  
}
```

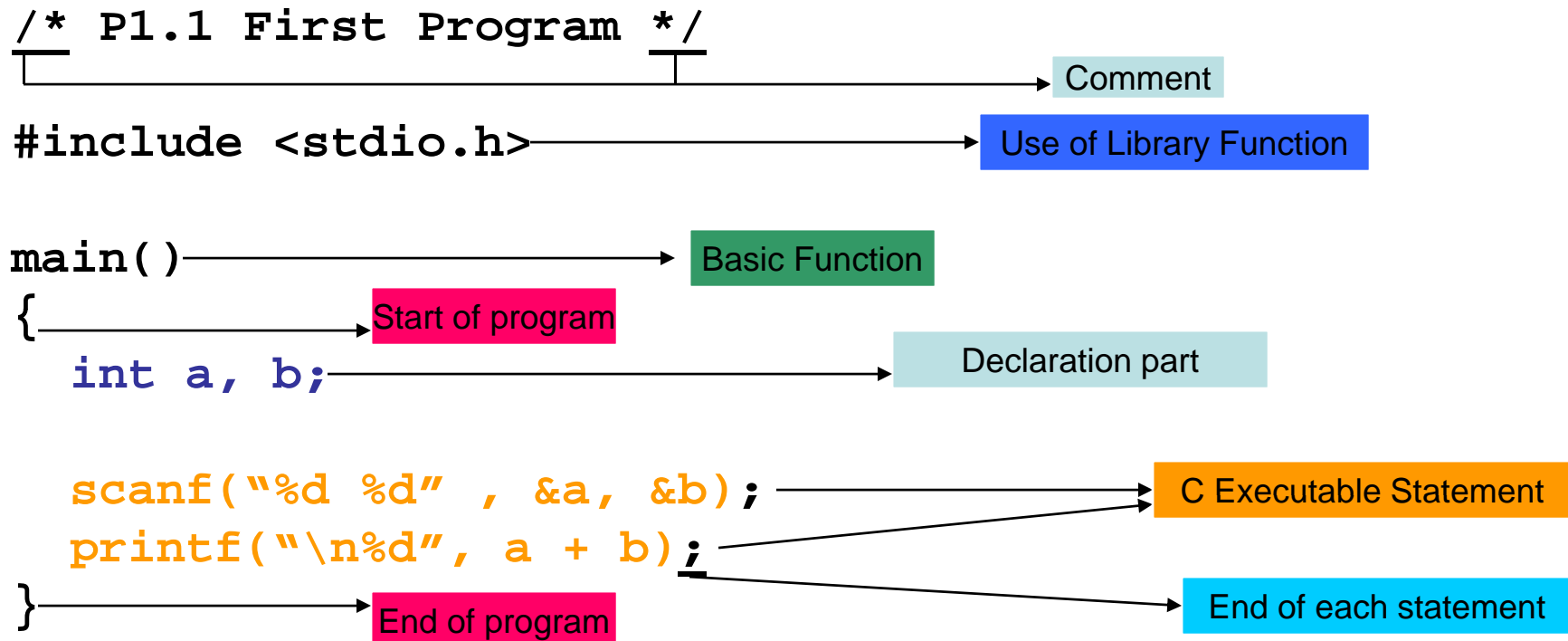
Running program:

12 36

48

# دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

یک برنامه به زبان C:



Running program:

12 36

48

رشته %d %d را الگوی پیمایش مینامیم. الگوی %d نشان دهنده آن است که آنچه که خوانده و یا چاپ میشود یک عدد صحیح است.

## دستور افزایش :

```
++متغیر;
```

```
متغیر++;
```

اگر متغیر با نام **a** تعریف شده باشد، میتوانیم بنویسیم :

```
a++;
```

```
++a;
```

و یا

```
a = a + 1 ;
```

همینطور می توانیم دستور کاهشی داشته باشیم. یعنی :

```
a = a - 1 ; | a--; | --a;
```

# دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

مقداردهی به متغیرها در بخش تعریف متغیر:

```
/* P1.1 First Program */  
  
#include <stdio.h>  
  
main()  
{  
    int a=12, b=36;  
  
    /*scanf("%d %d" , &a, &b);*/  
    printf("\n%d", a + b);  
}
```

Running program:

12 36

48

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## فصل دوم

### آشنائی با

### داده ها و انواع آن ها

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## انواع داده ها :

- ۱- کاراکترها / حروف / نویسه ها.
- ۲- عددهای صحیح.
- ۳- عددهای اعشاری تک دقتی.
- ۴- عددهای اعشاری دو دقتی.
- ۵- مقدار تهی.



## ۱- کاراکترها :

به هر یک از نشانه هائی که در زبان به کار می روند، کاراکتر می گوئیم.  
این کاراکترها عبارتند از :

حروف : a.. z , A..Z

ارقام : 0..9

حروف ویژه : %, \$, #, “, !, &, .....

## ۲- عددهای صحیح :

این اعداد در واقع اعداد طبیعی (بدون اعشار) بوده و بیشتر برای شمارش بکار می روند. عددهای صحیح می توانند با علامت و یا بدون علامت باشند. مثال:

32

+164

-5821

## ۳- عددهای اعشاری تک دقتی :

عددهای اعشاری تک دقتی (یا به اختصار عددهای اعشاری یا شناور)،  
عددهائی هستند که دارای جزء کمتر از یک می باشند و در چهار بایت  
از حافظه ذخیره می شوند. این اعداد در کامپیوتر به شکل دودوئی /  
باینری ضرب در توانی از عدد دو نگهداری می شوند. مثال :

12.5

-132.754

+64.

## ۴- عددهای اعشاری دو دقتی :

عددهای اعشاری دو دقتی هم عددهائی هستند که دارای جزء کمتر از یک می باشند ولی در هشت بایت از حافظه ذخیره می شوند، از این رو این عددها می توانند مقدار بزرگتری را نشان دهند. این اعداد هم در کامپیوتر به شکل دودوئی / باینری، ضرب در توانی از عدد دو نگهداری می شوند.

## ۵- مقدار تهی :

مقدار تهی را معمولا همراه با توابع و نشانگرها به کار می بریم. هر تابع معمولا مقداری را بر می گرداند، چنانکه تابعی هیچ مقداری را برنگرداند، آن را از نوع تهی تعریف می کنیم. همچنین هر نشانگر، نشانی نوع خاصی از داده ها را نشان می دهد. چنان چه نخواهیم نوع داده ای را که نشانگر به آن اشاره می کند تعیین نمائیم، تابع را از نوع تهی تعریف می کنیم.

# انواع تایپ های داده در C

Type	Typical Size In Bits	Minimal Range
char	8	-127 to 127
unsigned char	8	0 to 255
signed char	8	-127 to 127
int	16 or 32	-32,767 to 32,767
unsigned int	16 or 32	0 to 65,535
signed int	16 or 32	same as <b>int</b>
short int	16	-32,767 to 32,767
unsigned short int	16	0 to 65,535
signed short int	16	same as <b>short int</b>
long int	32	-2,147,483,647 to 2,147,483,647
signed long int	32	same as <b>long int</b>
unsigned long int	32	0 to 4,294,967,295
float	32	Six digits of precision
double	64	Ten digits of precision
long double	80	Ten digits of precision

**Table 2-1.** All Data Types Defined by the ANSI/ISO C Standard

## ۵- متغیرها :

متغیرها مکان هائی از حافظه هستند که آن ها را برای نگهداری داده ها بکار می بریم. این محل ها را از آن رو متغیر می نامیم که داده های آنها می توانند تغییر نمایند. برای مراجعه به هر متغیر، برای آن نامی قرار می دهیم که در حقیقت نشان دهنده محتویات آن متغیر است. برای تعریف متغیرها، الگوی زیر را بکار می بریم:

```
char a;
```

```
int i , cont;
```

```
float first,second; /* single precision */
```

```
double sum; /* double precision */
```

```
/* P2.1 Second Program */  
  
#include <stdio.h>  
  
main()  
{  
    int number;  
    float sum;  
    char ch;  
  
    scanf("%d %f %c" , &number, &sum, &ch);  
    printf("\n%d %f %c", number, sum, ch);  
}
```

Running program:

```
12  2.5  x
```

```
12  2.500000  x
```



# دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## ۵- نام گذاری متغیرها :

برای نام گذاری بایستی قواعد زیر را بکار ببریم:

- ۱- نام باید تنها از حروف (a..z و A..Z)، ارقام (0..9) و حرف زیرخط ( \_ ) ساخته شود.
  - ۲- بیش از ۳۱ حرف نداشته باشد.
  - ۳- از واژه های کلیدی C و شناسه های / متغیرهای تعریف شده در توابع کتابخانه ای نباشد.
- اسامی درست :**

```
month  
DAY13  
new_year  
_yes
```

**اسامی نادرست :**

```
month:  
13DAY  
new year  
char  
scanf
```

**یادآوری :**

- ۱- اگر نام بیش از ۳۱ حرف باشد، مترجم فقط ۳۱ حرف اول را در نظر می گیرد.
- ۲- زبان C برای حروف کوچک و بزرگ فرق قائل است. بنابراین **SUM** و **Sum** نام دو متغیر متفاوت می باشد.
- ۳- معمولا نام متغیرها در توابع کتابخانه ای با خط زیر شروع می شود، بنابراین بهتر است این کاراکتر را در آغاز نام ها بکار نبریم.

## بعضی از واژه های کلیدی در C

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

## ثابت ها (constant) :

ثابت ها مقادیری هستند که در طول برنامه تغییر نمی کنند. ثابت ها در زبان C عبارتند از :

**ثابت های صحیح :** معمولا در دو بایت (۱۶ بیت) گذاشته می شوند. این اعداد بین  $+۳۲۷۶۷$  و  $-۳۲۶۷۸$  قرار می گیرند. چرا؟

**ثابت های اعشاری :** ثابت های اعشاری همانند متغیرهای اعشاری می توانند تک دقتی و یا دو دقتی باشند. مثل  $2.35E6$

**ثابت های کاراکتری :** ثابت هائی هستند که برای نمایش یک کاراکتر بکار می روند. این ثابت ها را میان دو گیومه تکی می گذاریم. مثال :

'A'

'+'

'2'

## رشته ها (Strings) :

**متغیرهای رشته ای :** در زبان C متغیر رشته ای مستقیما به کار نمی رود. به جای متغیر رشته ای از آرایه رشته ای استفاده می کنیم. (آرایه ها در فصول بعدی ارائه می شوند)

**ثابت های رشته ای:** برای ساختن ثابت های رشته ای، رشته ها را در میان دو گیومه دوتائی قرار می دهیم. مثال:

“year 1373”

“m”

مترجم C، برای مشخص شدن پایان رشته ها، عدد صفر دودوئی را به طول یک بایت به انتهای هر رشته اضافه می کند.

**سوال :** فرق دو ثابت ‘m’ و “m” چیست؟

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## دستورهای پیمایش و خواندن

```
int a;  
float sum;  
char ch, name[10];  
scanf("%d %f %c %s", &a, &sum, &ch, &name);
```

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## دستورهای پیمایش و چاپ

```
printf("Sum = %f\n", sum);  
printf("Mr. %c %s", ch , name);  
printf("%f\n", sum * 4);
```

## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

**مثال :** برنامه ای بنویسید که نام شخص و شماره تلفن او را بخواند و چاپ نماید.

```
/* P2.3 Name & Tel. No. */
#include <stdio.h>

main()
{
    char name[20], tel[10];
    printf("Enter name and tel# : ");
    scanf("%s %s" , &name , &tel);
    printf("%s %s\n", name, tel);
}
```

**Running program:**

```
Enter name and tel# : Ebrahim 6688007
Ebrahim 6688007
```

**مثال :** برنامه ای بنویسید که نام و چهار نمره دانشجویی را بخواند، نام، نمره ها و میانگین آن ها را چاپ نماید.

## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

**مثال:** برنامه ای بنویسید که نام و چهار نمره دانشجویی را بخواند، نام، نمره ها و میانگین آن ها را چاپ نماید.

```
/* P2.5 Student_Name & Marks */
#include <stdio.h>
main() {
    char name[20];
    float g1, g2, g3, g4, avg;
    printf("Enter name : ");
    scanf("%s" , name );
    printf("Enter 4 marks : ");
    scanf("%f %f %f %f", &g1, &g2, &g3, &g4);
    avg = (g1 + g2 + g3 + g4) / 4;
    printf("\nName : %s\n", name);
    printf("Marks : %.2f %.2f %.2f %.2f\n", g1, g2, g3, g4);
    printf("Average : %.2f\n", avg); }
```

### Running program:

```
Enter name : Ebrahim.Azari
Enter 4 marks : 18 19 17.5 20
```

```
Name : Ebrahim Azari
Marks : 18.00 19.00 17.50 20.00
Average : 18.62
```



## دستورهای ورودی و خروجی `cin` و `cout`

علاوه بر دستورات ارائه شده برای خواندن و یا نمایش/چاپ مقادیر ثابت و یا محتوای متغیرها، دستورات دیگری نیز از قبیل `cin` برای خواندن از صفحه کلید و `cout` برای نمایش روی صفحه مانیتور وجود دارد.

برای اینکه بتوانیم از این دو متد/تابع استفاده کنیم بایستی در ابتدای برنامه فایل هدر زیر را تعریف نمائیم:

```
#include <iostream.h>
```

## دستور خروجی cout

بصورت پیش فرض، خروجی استاندارد یک برنامه صفحه مانیتور است و تابع تعریف شده برای دستیابی به آن cout می باشد. cout به همراه اپراتور << استفاده می‌گردد.

مثال :

```
cout << "Output sentence";    // prints Output sentence on screen
cout << 120;                  // prints number 120 on screen
cout << x;                    // prints the content of x on screen
```

## دستور خروجی cout - ادامه

اپراتور << موجب می‌گردد تا داده ای که بعد از اپراتور قرار گرفته است به cout ارسال شود.

اپراتور << میتواند بیش از یک مرتبه در یک عبارت استفاده شود.

مثال :

```
cout << "Hello, " << "I am " << "a C++ statement";
```

این عبارت موجب می‌گردد تا عبارت زیر روی صفحه مانیتور نمایش داده شود:

Hello, I am a C++ statement

Hello, I am 24 years old and my zipcode is 90064

دانشگاه علم و صنعت ایران – دانشکده مهندسی کامپیوتر

## دستور خروجی cout – ادامه

اپراتور cout میتواند یک عبارت که تلفیقی از متغیرها و ثابت ها باشد را نیز اجرا نماید.

مثال :

```
cout << "Hello, I am " << age << " years old and my zipcode is " << zipcode;
```

اگر فرض کنیم که محتوای `age` مقدار 24 باشد و `zipcode` دارای مقدار 90046 است، آنگاه پیام زیر را روی صفحه مانیتور خواهیم داشت:

Hello, I am 24 years old and my zipcode is 90064

دانشگاه علم و صنعت ایران – دانشکده مهندسی کامپیوتر

## دستور خروجی cout – ادامه

این نکته مهم است که بدانیم عبارات مختلف cout موجب رفتن به خطوط بعدی نمی شود. به عبارت دیگر، هر cout در انتها یک line break نخواهد داشت.

مثال :

```
cout << "This is a sentence.";  
cout << "This is another sentence."
```

موجب نمایش عبارت زیر روی صفحه مانیتور میگردد:

```
This is a sentence. This is another sentence.
```

Hello, I am 24 years old and my zipcode is 90064

دانشگاه علم و صنعت ایران – دانشکده مهندسی کامپیوتر

## دستور خروجی cout – ادامه

اگر بخواهیم جملات را از یکدیگر جدا کنیم و یا به عبارتی کنترل را به خطوط دیگر منتقل نمائیم بایستی از عملگر `\n` استفاده نمائیم.  
مثال :

```
cout << "First sentence.\n";  
cout << "Second sentence.\nThird sentence."
```

موجب نمایش عبارت زیر روی صفحه مانیتور میگردد:

```
First sentence.  
Second sentence.  
Third sentence.
```

Hello, I am 24 years old and my zipcode is 90064

دانشگاه علم و صنعت ایران – دانشکده مهندسی کامپیوتر

## دستور خروجی cout – ادامه

انتقال کنترل به خط بعد برای cout با کلمه رزرو شده endl هم بصورت زیر انجام می پذیرد:

مثال :

```
cout << "First sentence." << endl;  
cout << "Second sentence." << endl;
```

موجب نمایش عبارت زیر روی صفحه مانیتور میگردد:

First sentence.

Second sentence.

دانشگاه علم و صنعت ایران – دانشکده مهندسی کامپیوتر

## دستور ورودی – cin

اسباب ورودی استاندارد معمولاً صفحه کلید میباشد. برای خواندن از صفحه کلید میتوانیم از تابع `cin` و اپراتور `>>` استفاده نمائیم. به دنبال اپراتور، اسامی متغیرهایی که بایستی از ورودی مقدار بگیرند آورده می شود.

مثال :

```
int age;  
cin >> age ;
```

عبارت اول یک متغیر دو بایتی از نوع صحیح را تعریف میکند و عبارت دوم منتظر گرفتن یک عدد از ورودی شده و آن را در متغیر `age` قرار میدهد.

نکته : برخلاف دستور `scanf`، نیاز به استفاده از `&` نداریم.



## دستور ورودی - cin - ادامه

cin تنها میتواند زمانی داده را از صفحه کلید بخواند که کلید **enter** زده شود. قطعا بایستی داده ورودی با متغیر یا متغیرها همخوانی (Data Type) داشته باشد. مثال :

```
// i/o example
#include <iostream.h>
int main ()
{ int i;
  cout << "Please enter an integer value: ";
  cin >> i;
  cout << "The value you entered is " << i;
  cout << " and its double is " << i*2 << ".\n";
  return 0; }
```

Hello, I am 24 years old and my zipcode is 90064

دانشگاه علم و صنعت ایران – دانشکده مهندسی کامپیوتر

## دستور ورودی – cin – ادامه

اجرای برنامه مثال قبل بصورت زیر خواهد بود :

Please enter an integer value: 702

The value you entered is 702 and its double is 1404.

نکته : با استفاده از **cin**، تنها تا اولین کاراکتر بلائک مقدار ورودی خوانده میشود.

## دستور ورودی – cin – ادامه

نکته : با استفاده از **cin**، تنها تا اولین کاراکتر بلاَنک، مقدار ورودی خوانده میشود. این بدان معنا است که با یک عبارت **cin** فقط یک متغیر از ورودی مقدار میپذیرد و یا اینکه میتوانیم بصورت زیر از تابع **cin** استفاده نمائیم.

```
cin >> a >> b;
```

که برابر است با :

```
cin >> a;
```

```
cin >> b;
```

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## دستور ورودی - cin - ادامه

همانگونه که قبلا گفته شد، با استفاده از **cin**، تنها تا اولین کاراکتر بلانک مقدار ورودی خوانده میشود. این مسئله در مورد خواندن رشته ها نیز با استفاده از **cin** صدق میکند. در واقع با استفاده از **cin**، در هر بار خواندن مجموعه ای از کلمات که با کاراکتر بلانک از یکدیگر جدا شده اند، فقط میتوانیم یک کلمه را بخوانیم.

```
// cin with strings
#include <iostream.h>
#include <string>
int main () {
string  mystr;
cout << "What's your name? ";
getline (cin, mystr);
cout << "Hello " << mystr << ".\n";
cout << "What is your favorite team? "; getline (cin, mystr);
cout << "I like " << mystr << " too!\n";
return 0; }
```

Hello, I am 24 years old and my zipcode is 90064

دانشگاه علم و صنعت ایران – دانشکده مهندسی کامپیوتر

## دستور ورودی – cin – ادامه

برای مثال قبل، اجرا بصورت زیر خواهد بود:

What's your name? Juan Soulie

Hello Juan Soulie.

What is your favorite team? The Isotopes

I like The Isotopes too!

**نکته** : به شرط آنکه فایل هدر `string.h` را داشته باشید.

**تمرین** : این کار را با تعریف متغیر بصورت زیر نیز انجام دهید:

```
char mystr[30];
```

## تمرین پایان فصل

۱- برنامه ای بنویسید که طول، عرض و بلندی مکعبی را از کاربر پرسیده، سپس حجم آن را محاسبه و چاپ نماید.

۲- برنامه ای بنویسید که تعداد ساعت ها، دقیقه ها و ثانیه های یک سال را محاسبه و در سه سطر چاپ نماید.

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## فصل سوم

عبارت ها

و

عملگرها

## ۳- عملگرها

### ۱-۳- انواع عملگرها

۱. حسابی
۲. رابطه ای
۳. منطقی
۴. بیتی
۵. جابه جایی
۶. یگانی
۷. واگذاری
۸. شرطی



دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## ۱- عملگرهای حسابی

عملگر	معنای عملگر
+	جمع
-	تفریق
*	ضرب
/	تقسیم
%	باقیمانده تقسیم دو مقدار صحیح

## ۱- اولویت عملگرهای حسابی

اولویت	عملگر	اجرای عملگرهای هم اولویت
۱	% / *	از چپ به راست
۲	- +	از چپ به راست

## ۲- عملگرهای رابطه ای

معنای عملگر	عملگر
بزرگتر	>
بزرگتر یا مساوی	>=
کوچکتر	<
کوچکتر یا مساوی	<=
مساوی	==
نامساوی	!=

## ۲- اولویت عملگرهای رابطه ای

اولویت	عملگر	اجرای عملگرهای هم اولویت
۱	<= < >= >	از چپ به راست
۲	== !=	از چپ به راست

## ۳ - عملگرهای منطقی

معنای عملگر	عملگر
نه یا نقیض	!
و (Logical AND)	&&
یا (Logical OR)	

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## ۳- اولویت عملگرهای منطقی

اولویت	عملگر	اجرای عملگرهای هم اولویت
۱	!	از چپ به راست
۲	&&	از چپ به راست
۳		از چپ به راست

## ۳- نتیجه عملگرهای منطقی

عبارت	مقدار عبارت
a	1 1 0 0
b	1 0 1 0
!a	0 0 1 1
a && b	1 0 0 0
a    b	1 1 1 0

## ۴- عملگرهای بیتی

عملگر	معنای عملگر
&	وی بیتی (AND)
	یای بیتی (OR)
^	یای انحصاری بیتی (Exclusive OR)
~	نقیض بیتی
>>	جابجائی به راست (Shift Right)
<<	جابجائی به چپ (Shift Left)



## ۴- نتیجه عملگرهای بیتی

توجه داشته باشیم که هر بیت  $a$  و یا  $b$  مد نظر است.

عبارت	مقدار عبارت
$a$	1 1 0 0
$b$	1 0 1 0
$\sim a$	0 0 1 1
$a \wedge b$	0 1 1 0
$a \& b$	1 0 0 0
$a   b$	1 1 1 0

## ۵- عملگرهای جابجائی

عملگرهای جابجائی را برای جابجاکردن بیت های یک کاراکتر و یا یک عدد صحیح به کار می بریم. مثال :

'a' >> 1

'E' << 4

0110 0001    آرایش بیتی کاراکتر a

0100 0101    آرایش بیتی کاراکتر E

'a' = 0 1 1 0 0 0 0 1

0 0 1 1 0 0 0 0 = '0' :: ASCII = 48

'E' = 0 1 0 0 0 1 0 1

0 1 0 1 0 0 0 0 = 'P' :: ASCII = ?

## ۵- عملگرهای جابجائی (ادامه)

0110 0001    a    آرایش بیتی کاراکتر

0100 0101    E    آرایش بیتی کاراکتر

a = 'a' ,    E = 'E'

a & E = 0 1 0 0 0 0 0 1 = 'A'

a | E = 0 1 1 0 0 1 0 1 = 'e'

a ^ E = 0 0 1 0 0 1 0 0 = '\$'

a << 6 = 0 1 0 0 0 0 0 0 = '@'

E >> 1 = 0 0 1 0 0 0 1 0 = ""

## ۵- اولویت عملگرهای بیتی

اولویت	عملگر	اجرای عملگرهای هم اولویت
۱	~	از راست به چپ
۲	<< >>	از چپ به راست
۳	&	از چپ به راست
۴	^	از چپ به راست
۵		از چپ به راست

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## ۵- اولویت عملگرهای بیتی (مثال)

$a \& b \gg 3 \mid c \wedge e \ll 2$

If  $a=1$  ,  $b=8$  ,  $c=3$  ,  $e=1$  Then

## ۵- اولویت عملگرهای بیتی (مثال)

$a \& b \gg 3 \mid c \wedge e \ll 2$

If  $a=1$  ,  $b=8$  ,  $c=3$  ,  $e=1$  Then

$1 \& 8 \gg 3 \mid 3 \wedge 1 \ll 2$

$1 \& 1 \mid 3 \wedge 4$

$1 \mid 3 \wedge 4$

$1 \mid 7$

7

## ۶- عملگرهای یگانی

عملگر	معنای عملگر
+	مثبت
-	منفی
++	افزایش
--	کاهش
(نوع)	تبدیل نوع
sizeof	اندازه
&	نشانی
*	نشانگر

## ۶- عملگرهای یگانی (مثال)

If  $i=3$  ,  $j=4$  ,  $k=5$  ,  $m=6$

$++i \rightarrow 4$  ,  $j++ \rightarrow 5$  ,  $--k \rightarrow 4$  ,  $m-- \rightarrow 5$

$i = i + 1;$  ,  $j = j + 1;$  ,  $k = k - 1;$  ,  $x = ++i;$

$(\text{int}) 25.5 \rightarrow 25$  ,  $(\text{float}) 12 \rightarrow 12.0$

$\text{sizeof } 12 \rightarrow 2$  (if shortint) , 4 (if int)

$\text{sizeof } (\text{float}) \rightarrow 4$

$\text{sizeof } (\text{int}) \rightarrow 4$

$\text{sizeof } (\text{double}) \rightarrow 8$



## ۷- عملگرهای واگذاری

عملگرهایی هستند که برای واگذار کردن مقدار یک عبارت به یک متغیر به کار میروند. مثال:

```
a = b + 2;
```

می توانیم عملگرهای دودوئی حسابی را با عملگرهای واگذاری ترکیب نمائیم. این عملگرها عبارتند از:

`+=` `-=` `*=` `/=` `%=`

```
x = x + 3;   :::   x += 3;
```

اولویت عملگرهای واگذاری از همه عملگرهای دیگر کمتر است. خود این عملگرها دارای اولویت یکسانند و در صورت داشتن چند عملگر واگذاری در یک عبارت، اولویت آن ها از راست به چپ می باشد.

## ۱- عملگر شرطی

عملگر شرطی (`?:`)، عملگری است که بسته به برقرار بودن و یا نبودن یک شرط، مقدار یکی از دو عبارت پس از خود را برمی‌گزیند. الگوی به کار بستن این عملگر به صورت زیر است:

عبارت ۲ : عبارت ۱ ? شرط

مقدار این عبار در صورت برقرای شرط عبارت ۱ و در غیر این صورت عبارت ۲ خواهد شد.

```
x = n > 0 ? n : -n;
```

# دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## ترتیب اولویت عملگرها

اولویت	عملگر	اجرای عملگرهای هم اولویت
۱	() → ++ --	از چپ به راست
۲	+ ! ~ - * & (نوع) sizeof	از راست به چپ
۳	* / %	از چپ به راست
۴	+ -	از چپ به راست
۵	<< >>	از چپ به راست
۶	< > <= >=	از چپ به راست
۷	== !=	از چپ به راست
۸	&	از چپ به راست
۹	^	از چپ به راست
۱۰		از چپ به راست
۱۱	&&	از چپ به راست
۱۲		از چپ به راست
۱۳	?:	از راست به چپ
۱۴	= += -= *= /= %= &= ^=  = < <= >> =	از راست به چپ

```
/* P 3.1 Compound assignment operations */
#include <stdio.h>
main()
{
    int p=10 , q=3;
    printf("p = %d , q = %d\n" , p, q);
    printf("p += q → %d\n" , p += q);
    printf("p -= q → %d\n" , p -= q);
    printf("p *= q → %d\n" , p *= q);
    printf("p /= q → %d\n" , p /= q);
    printf("p %= q → %d\n" , p %= q);
    printf("p ^= q → %d\n" , p ^= q);
    printf("p &= q → %d\n" , p &= q);
    printf("p |= q → %d\n" , p |= q);
    printf("p <<= q → %d\n" , p <<= q);
    printf("p >>= q → %d\n" , p >>= q);
}
```

## اجرای برنامه ۱-۳

```
p = 10 , q = 3
p += q → 13
p -= q → 10
p *= q → 30
p /= q → 10
p %= q → 1
p ^= q → 9
p &= q → 2
p |= q → 3
p <<= q → 24
p >>= q → 3
```

## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

مثال (برنامه ۲-۳)

```
/* P 3.2   Compound assignment   operations */
#include <stdio.h>
main()
{
    int  x, n;
    printf("Enter a number : ");
    scanf("%d" , &n);
    x = n > 0 ? n : -n;
    printf("Absolute value of %d is %d\n", n , x);
}
```

### Running Program

```
Enter a number : 100
Absolute value of 100 : is 100
```

### Running Program

```
Enter a number : -100
Absolute value of -100 : is 100
```

## تمرین پایان فصل

۱- برنامه ای بنویسید که مقادیر  $a, b, c, d, i, j, k$  را از ورودی خوانده و عبارات زیر را محاسبه و چاپ نماید.

$a + b / c - d$

$(a + b) / c - d$

$a + b / (c - d)$

$(a + b) / (c - d)$

$i < j$

$i > j$

$i == j$

$i <= j$

$i >= j$

$i != j$

$i > j \ \&\& \ j > k$

$i > j \ \|\ j > k$

$!(i > j \ \|\ j > k)$

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## فصل چهارم

دستورها

**(C Statements)**



## ۱- عبارت کنترلی **if**

```
if (condition) then_clause;  
[else_clause;
```

در صورت درست بودن شرط (condition)، بخش **then\_clause** و در صورت نادرست بودن شرط، بخش **else\_clause** اجرا می شود.  
مثال:

```
if (x > 100)  
    x = 100;  
if (i > 10)  
    x = 100;  
else  
    x = 200;
```

## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

**مثال :** می خواهیم برنامه ای بنویسیم که دو عدد را بخواند، چنان که اولی از دومی بزرگتر باشد تفاضل دو عدد و در غیر این صورت مجموع آن ها را چاپ نماید.

```
/* P4.1 if statement */

#include <stdio.h>
main()
{
    int a, b;

    printf("Enter 2 Integer No. : ");
    scanf("%d %d" , &a, &b);
    if (a>b)
        printf("a - b = %d\n", a - b);
    else
        printf("a + b = %d\n", a + b);
}
```

**Running program:**

```
Enter 2 Integer No. : 12    9
a - b = 3
```

## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

**مثال :** می خواهیم برنامه ای بنویسیم که دو عدد را بخواند، چنان که اولی از دومی بزرگتر باشد تفاضل دو عدد را محاسبه و چاپ نماید، در غیر این صورت یک عدد دیگر را هم خوانده و مجموع سه عدد را چاپ نماید.

```
/* P4.2 compound if statement */

#include <stdio.h>
main()
{
    int a, b, c;
    printf("Enter 2 Integer No. : ");
    scanf("%d %d" , &a, &b);
    if (a>b)
        printf("a - b = %d\n", a - b);
    else {
        printf("Enter third No. : ");
        scanf("%d", &c);
        printf("a + b + c = %d\n", a + b + c); }
}
```

### Running program:

```
Enter 2 Integer No. : 20    32
Enter third No. : 8
a + b + c = 60
```

## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

**مثال :** می خواهیم برنامه ای بنویسیم که دو عدد را بخواند، چنان که اولی از دومی بزرگتر باشد تفاضل دو عدد را محاسبه و چاپ نماید، در صورتی که اولی از دومی کوچکتر باشد مجموع دو عدد و در غیر این صورت حاصل ضرب دو عدد را چاپ نماید.

```
/* P4.3 compound if statement */

#include <stdio.h>
main()
{
    int a, b, c;
    printf("Enter 2 Integer No. : ");
    scanf("%d %d" , &a, &b);
    if (a>b)
        printf("a - b = %d\n", a - b);
    else if (a < b)
        printf("a + b = %d\n ", a + b);
    else
        printf("a * b = %d\n", a * b );
}
```

### Running program:

```
Enter 2 Integer No. : 20    20
a * b = 400
```

## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

### ۲- عبارت کنترلی-انتخابی (Switch)

عبارت انتخابی switch را زمانی به کار می بریم که بخواهیم از میان چند گزینه یکی را انتخاب نمائیم. فرمت عبارت به صورت زیر است:

```
switch (variable/expression) {  
    case value1 : statement;  
    case value2 : statement;  
    ....  
    [default : statement;]  
}
```

#### تذکر:

- در این عبارت، مقادیر بایستی از نوع کاراکتری و یا صحیح باشند.
- مقادیر نباید تکراری باشند.
- برای هر case می توانیم یک یا چند دستور داشته باشیم.
- در هنگام اجرا، به محض برابر شدن با هر یک از حالات، علاوه بر اجرای عبارت [های] متناظر، بقیه عبارات case های به دنبال آن نیز اجرا خواهد شد.
- اگر مقدار عبارت با هیچ یک از مقادیر (value) برابر نباشد، گزینه مربوط به بند پیش فرض (default) اجرا می شود.

## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

**مثال :** می خواهیم برنامه ای بنویسیم که هر کدام از اعداد یک تا سه را که بخواند ، نام آن عدد را چاپ نماید.

```
/* P4.4 switch statement */

#include <stdio.h>
main()
{
    int n;
    printf("Enter a number (1 - 3) : ");
    scanf("%d" , &n);
    switch (n) {
        case 1 : printf("One\n");
        case 2 : printf("Two\n");
        case 3 : printf("Three\n");
        default : printf("Other\n"); }
}
```

### Running program:

```
Enter a number (1 - 3) : 2
Two
Three
Other
```

# دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## ۲- عبارت کنترلی switch - ادامه

برای اینکه بعد از اجرای عبارت / عبارات متناظر با یک case، دستور switch خاتمه یابد و بقیه عبارات به دنبال آن اجرا نشود، باید از دستور break استفاده نمائیم.

```
/* P4.5 switch statement */

#include <stdio.h>
main()
{
    int n;
    printf("Enter a number (1 - 3) : ");
    scanf("%d" , &n);
    switch (n) {
        case 1 :
            printf("One\n");
            break;

        case 2 :
            printf("Two\n");
            break;

        case 3 :
            printf("Three\n");
            break;
        default : printf("Other\n"); }
}
```

### Running program:

```
Enter a number (1 - 3) : 2
Two
```

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## ۲- عبارت کنترلی `switch` - ادامه

اگر گزینه های چند `case` همانند هم باشند، می توانیم آن `case`ها را بدون گزینه زیر هم و یا به دنبال هم نوشته، تنها عبارت / عبارات مربوط به آخرین `case` را بنویسیم.



# دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

مثال :

```
/* P 4.6 switch statement for similar value */
#include<stdio.h>
#include<conio.h>
main()
{
    char ch;
    printf("Enter a letter : ");
    ch = getch();
    switch (ch) {
        case 'A' :
        case 'E' :
        case 'I' :
        case 'O' :
        case 'U' :
        case 'Y' :
        case 'a' :
        case 'e' :
        case 'i' :
        case 'o' :
        case 'u' :
        case 'y' :
            printf("\n%c is a vowel\n", ch);
            break;
        default :
            printf("\n%c is a consonant\n" , ch);
    }
}
```

Running program

```
Enter a letter : E
E is a vowel
```

azizjalali@iust.ac.ir

73

## ۳- عبارات ساخت حلقه

تعدادی از دستورها / عبارات برنامه که قابلیت اجرا برای چند مرتبه را داشته باشند، حلقه نامیده می شوند. در زبان C، عبارات زیر را برای ساخت حلقه ها به کار می بریم.

۱- حلقه های پیش شرط (while).

۲- حلقه های پس شرط (do while).

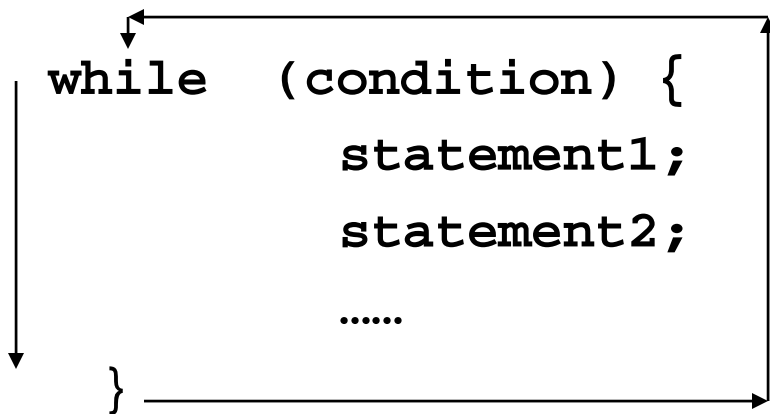
۳- حلقه های با شمارش معین (for).

## ۱-۳- حلقه پیش شرط (while)

این عبارات، عباراتی هستند که میتوانند از صفر تا  $n$  مرتبه تکرار شوند. قالب این دستور به صورت زیر است:

```
while (condition) statement;
```

و یا در صورتی که بخواهیم چند عبارت در حلقه اجرا شود، آن ها را در { و } قرار می دهیم. توجه داشته باشیم که ابتدا شرط بررسی و در صورت درست بودن شرط، عبارات داخل حلقه اجرا میشود :



# دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

۱-۳- حلقه پیش شرط (مثال)

```
/* P4.7 while statement */  
  
#include <stdio.h>  
main()  
{  
    int n=0;  
    while (n < 10) {  
        printf ("%d\n", n);  
        ++n;  
    }  
}
```

Running program

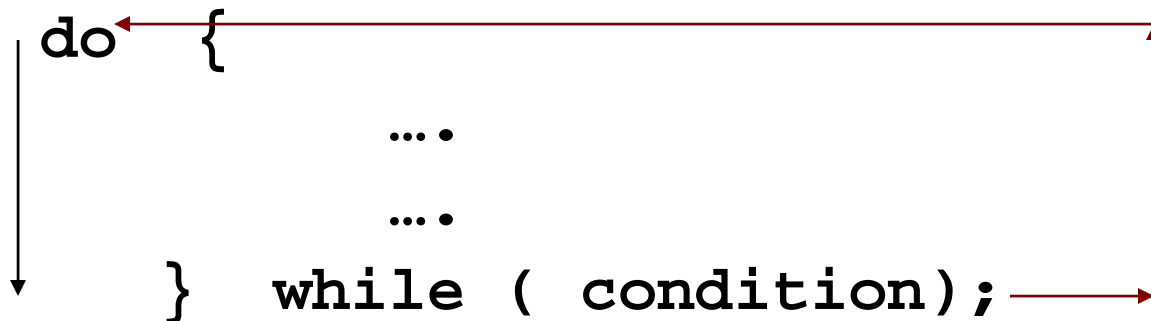
```
0  
1  
2  
3  
4  
5  
6  
7  
8  
9
```

## ۲-۳- حلقه پس شرط (do while)

در این حلقه ها، ابتدا دستورات داخل حلقه اجرا و سپس شرط بررسی می شود که در صورت درست بودن شرط، حلقه مجددا اجرا و در غیر این صورت، کنترل به اولین دستور بعد از حلقه می رود. قالب این دستور به شکل زیر است:

```
do statement while (condition);
```

و اگر بیش از یک عبارت در داخل حلقه داشته باشیم، آن ها را در میان { و } قرار می دهیم.



توجه داشته باشیم که در هر دو حلقه پیش شرط و پس شرط بایستی شرط خروج از حلقه را محقق نمائیم. در غیر این صورت یک حلقه بی پایان خواهیم داشت.

تذکر : در حلقه پس شرط، عبارات داخل حلقه حداقل یک مرتبه اجرا می شوند.

# دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

۲-۳- حلقه پس شرط (مثال)

```
/* P4.8 do-while statement */  
  
#include <stdio.h>  
main()  
{  
    int n=0;  
    do {  
        printf ("%d\n", n);  
        ++n;  
    } while (n < 10);  
}
```

Running program

```
0  
1  
2  
3  
4  
5  
6  
7  
8  
9
```

## ۳-۳ - حلقه های با شمارش معین (for)

این حلقه ها را زمانی بکار می بریم که بدانیم چند مرتبه بایستی تکرار شود. قالب این دستور به صورت زیر است:

```
for ( statement1 ; condition ; statement2 ) loop_statement
```

عبارت ۱، عبارتی است که در آغاز کار (یعنی در اولین مرتبه که دستور **for** اجرا می شود) اجرا می شود. این دستور را معمولا برای مقداردهی اولیه به متغیر شمارنده بکار می بریم. شرط یا **condition**، عبارتی است منطقی که اگر درست باشد، عبارت **loop\_statement**، که آن را بدنه حلقه می نامیم، اجرا می شود. هنگامی که عبارت نادرست شود، اجرای دستور پایان می یابد. عبارت ۲، معمولا برای تغییر مقدار شمارنده به کار می رود. این عبارت، هر بار که حلقه تکرار شود، پس از آخرین دستور حلقه، اجرا می شود.

**مثال:**

```
for ( n = 0 ; n < 10 ; ++n ) .....
```

# دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

۳-۳- حلقه با شمارش معین (مثال)

```
/* P4.9 for statement */  
  
#include <stdio.h>  
main()  
{  
    int n;  
    for ( n = 0 ; n < 10 ; ++n )  
        printf ("%d\n", n);  
}
```

Running program

```
0  
1  
2  
3  
4  
5  
6  
7  
8  
9
```

تمرین : دستورهایی معادل برای حلقه های پیش شرط و حلقه های با شمارش معین را بنویسید.



## ۳-۳ - حلقه های با شمارش معین (ادامه)

تمرین : دستورهایی معادل برای حلقه های پیش شرط و حلقه های با شمارش معین را بنویسید.

```
for ( statement1 ; condition ; statement2 ) loop statement[s]
```

```
statement1 ;  
while ( condition ) {  
    loop_statement[s] ;  
    statement2 ;  
}
```

### ۳-۳ - حلقه با شمارش معین (ادامه)

در عبارت `for`، به جای عبارت ۱ و عبارت ۲ می توان بیش از یک دستور نوشت که این عبارات را با ویرگول از یکدیگر جدا می کنیم. مثال:

```
for ( n=0 , m=9 ; n<10 ; ++n , --m ) ... .
```

در این مثال، نخست عددهای صفر و نه به متغیرهای `n` و `m` واگذار می شوند. سپس تا زمانی که `n` کوچکتر از ۱۰ است حلقه تکرار می شود. پس از پایان اجرای عبارات داخل حلقه، به `n` یکی اضافه و از `m` یکی کم می شود.

# دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

۳-۳- حلقه با شمارش معین (مثال)

```
/* P4.10 for statement */  
  
#include <stdio.h>  
main()  
{  
    int n, m ;  
    for ( n = 0 , m = 9 ; n < 10 ; ++n , --m )  
        printf ("n = %d          m = %d\n", n , m);  
}
```

## Running program

n = 0	m = 9
n = 1	m = 8
n = 2	m = 7
n = 3	m = 6
n = 4	m = 5
n = 5	m = 4
n = 6	m = 3
n = 7	m = 2
n = 8	m = 1
n = 9	m = 0

تمرین : دستورهایی معادل برای حلقه های پیش شرط و حلقه های با شمارش معین را بنویسید.

## ۴- حلقه های تودرتو (Nested Loop)

حلقه های تودرتو هنگامی ساخته می شوند که در درون یک حلقه، یک عبارت حلقه (پیش شرط ، حلقه پس شرط و یا حلقه با شمارش معین، داشته باشیم. مثال :

```
for ( i = 0 ; i <= 10 ; ++i )  
    for ( j = 0 ; j <= 10 ; ++j )  
        .....
```

تمرین : برنامه ای بنویسید که جدول ضرب را روی صفحه نمایش نشان دهد.

## ۴- حلقه های تودرتو (ادامه)

تمرین : برنامه ای بنویسید که جدول ضرب را روی صفحه نمایش نشان دهد.

```
/* P4.11 Multiplication table */
#include <stdio.h>
main()
{
    int i, j ;
    for ( i = 0 ; i <= 10 ; ++i )
        for ( j = 0 ; j <= 10 ; ++j )
            printf ("%4d" , i * j ) ;
}
```

آیا برنامه فوق بدرستی جدول ضرب را روی صفحه نشان می دهد؟

## ۴- حلقه های تودرتو (ادامه)

تمرین : برنامه ای بنویسید که جدول ضرب را روی صفحه نمایش نشان دهد.

```
/* P4.11 Multiplication table */
#include <stdio.h>
main()
{
    int i , j ;
    for ( i = 0 ; i <= 10 ; ++i )
        for ( j = 0 ; j <= 10 ; ++j )
            printf ("%4d" , i * j ) ;
    printf ("\n");
}
```

آیا برنامه فوق بدرستی جدول ضرب را روی صفحه نشان می دهد؟

## ۴- حلقه های تودرتو (ادامه)

تمرین : برنامه ای بنویسید که جدول ضرب را روی صفحه نمایش نشان دهد.

```
/* P4.11 Multiplication table */
#include <stdio.h>
main()
{
    int i , j ;
    for ( i = 0 ; i <= 10 ; ++i ) {
        for ( j = 0 ; j <= 10 ; ++j )
            printf ("%4d" , i * j ) ;
        printf ("\n");
    }
}
```

آیا برنامه فوق بدرستی جدول ضرب را روی صفحه نشان می دهد؟ بلی

## ۵- دستوره‌های پرش

دستوره‌های پرش، دستورها / عباراتی هستند که مسیر اجرای برنامه را بی هیچ شرطی تغییر می دهند. این عبارات به شرح زیر می باشند.

۱- دستور جهش (goto).

۲- دستور برش/قطع (break).

۳- دستور ادامه (continue).

۴- دستور بازگشت (return).



## تمرین پایان فصل

- ۱- برنامه ای بنویسید که به کمک عبارت حلقه پیش شرط، مجموع عددهای کوچکتر از ۱۰۰ را که بر ۷ بخش پذیر باشند را نمایش دهد.
- ۲- برنامه ای بنویسید که یکصد عدد صحیح را از ورودی خوانده، میانگین، بزرگترین و کوچکترین آن ها را چاپ نماید.
- ۳- برنامه ای بنویسید که دومین کوچکترین را از میان یکصد عدد ورودی صحیح که خوانده می شود را چاپ نماید.
- ۴- برنامه ای بنویسید که فاکتوریل عدد ۷ را محاسبه و نمایش دهد.

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## تمرین پایان فصل (ادامه)

۴- برنامه ای بنویسید که به کمک حرف ستاره، اشکال زیر را چاپ نماید.

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

دانشگاه علم و صنعت ایران – دانشکده مهندسی کامپیوتر

## فصل پنجم

توابع

**(Functions)**

## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

**تعریف تابع :** تابع، برنامه های کوچکی هستند که برای انجام کاری ویژه نوشته می شوند. یک تابع میتواند چند مقدار ورودی در زمان فراخوانی داشته و نیز میتواند یک مقدار را در هنگام پایان در خروجی داشته باشد. تابع ها بر دو نوع هستند :

**۱- توابع کتابخانه ای :** که از پیش آماده شده اند. این تابع ها توسط تولید کنندگان مترجم C و یا نرم افزار نویسان دیگر نوشته و تولید می گردد. از جمله توابع کتابخانه ای میتوانیم `scanf`، `printf`، `getche`، `sqrt` و از این دست را نام برد.

**۲- توابع نوشته شده توسط کاربر :** این توابع توسط کاربران برای انجام کارهای خاص خود، نوشته می شوند.

## فرمت تابع :

```
return_type Function_Name ([argument_type argument][,...])  
{  
    statements;  
}
```

## مثال :

```
int sum ( int x, int y)  
{  
    return x + y;  
}
```

## تذکر :

اگر تابع بعد از تابع **main** تعریف شود، بایستی یک **Prototype** (در واقع خط اول تابع) را بعد از **include** در متن برنامه قرار دهیم.

اگر تابع قبل از تابع **main** نوشته شود، نیازی به نوشتن **Prototype** (در واقع خط اول تابع) در متن برنامه نداریم.

# دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

**برنامه ۱-۵:** وقتی تابع بعد از تابع main نوشته میشود،  
بایستی prototype تابع نوشته شود.

```
/* P5.1 User_Defined function */
#include<stdio.h>

int sum(int x, int y);

main()
{
    int a, b;
    do {
        printf("\nEnter a & b : ");
        scanf("%d %d", &a, &b);
        printf("a + b = %d\n", sum(a, b));
    } while (a);
}

int sum(int x, int y)
{
    return x + y ;
}
```

## اجرای برنامه ۱-۵:

```
Enter a & b : 12 28
```

```
a + b = 40
```

```
Enter a & b : 2000 3210
```

```
a + b = 5210
```

```
Enter a & b : 0 5
```

```
a + b = 5
```



# دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

برنامه ۲-۵:

```
/* P5.2 User_Defined function ver2 */
#include<stdio.h>

int sum(int x, int y);

main()
{
    int a, b;
    do {
        printf("\nEnter a & b : ");
        scanf("%d %d", &a, &b);
        sum (a , b);
    } while (a);
}

int sum(int x, int y)
{
    printf ("a + b = %d\n" , x + y);
}
```

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## اجرای برنامه ۲-۵:

```
Enter a & b : 12 8
```

```
a + b = 20
```

```
Enter a & b : 234 615
```

```
a + b = 849
```

```
Enter a & b : 0 2
```

```
a + b = 2
```

# دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

برنامه ۳-۵:

```
/* P5.3 User_Defined function ver2 */
#include<stdio.h>

fun (int q);

int main()
{
    int p=100;

    printf("\np1 = %d\n", p);
    fun(p);
    printf("p4 = %d\n", p);
    return 0;
}

fun (int q)
{
    printf ("p2 = %d\n" , q);
    q = 333;
    printf("p3 = %d\n", q);
}
```

## اجرای برنامه ۳-۵:

p1 = 100

p2 = 100

p3 = 333

p4 = 100

تمرین ۱: تابعی بنویسید که یک کاراکتر را گرفته، در صورتی که کاراکتر گرفته شده یک حرف بزرگ A تا Z باشد، معدل کوچک آن را برگردانده و در غیر این صورت خود کاراکتر را برگرداند.

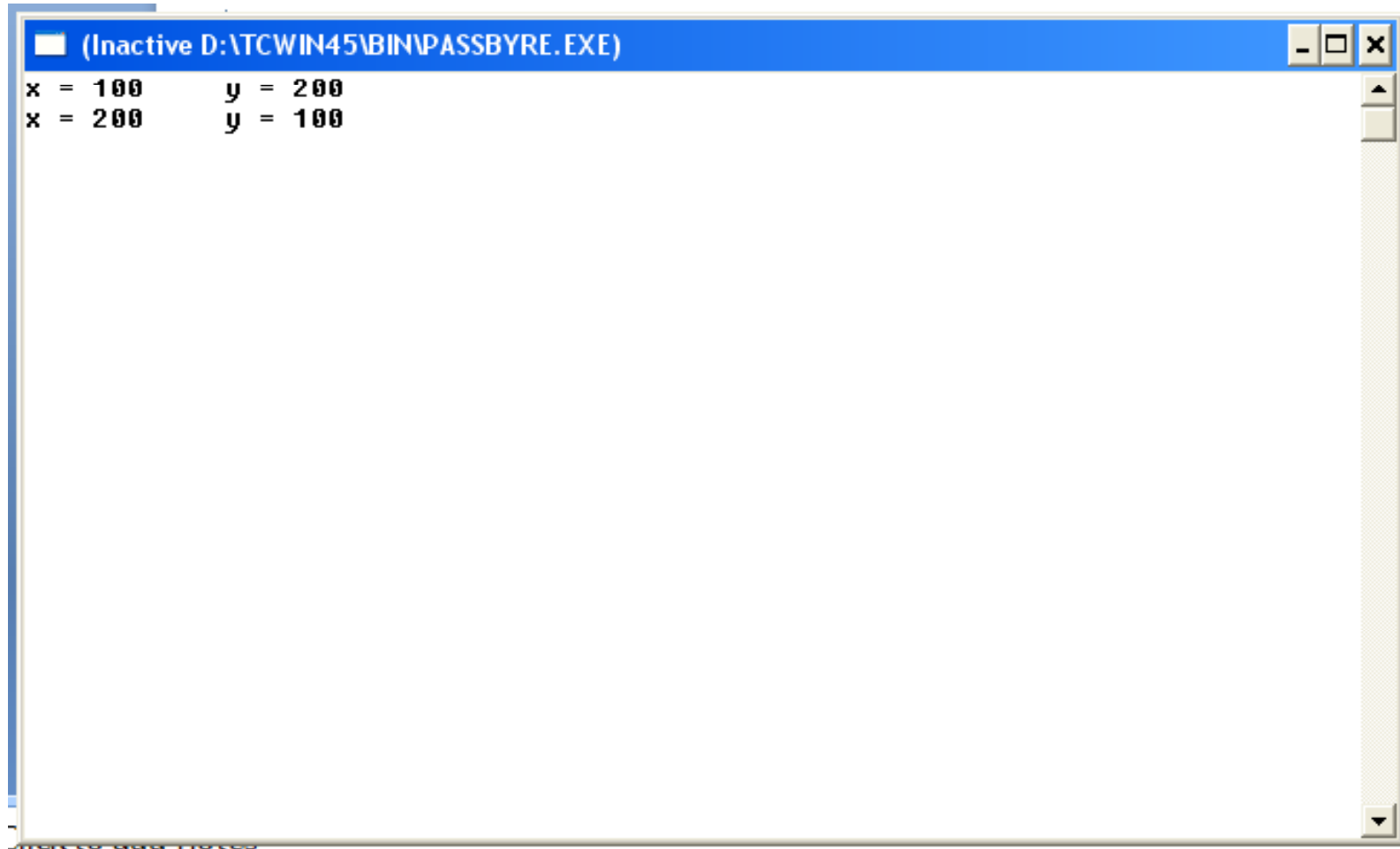
تمرین ۲: تابعی بنویسید که یک کاراکتر را گرفته، در صورتی که کاراکتر گرفته شده رقم باشد، مقدار True و در غیر این صورت مقدار False برگرداند.

- چگونه بصورت **Pass by Ref.** یک تابع را فراخوانی کنیم؟  
به عبارت دیگر داده هائی را که به تابع می فرستیم در هنگام بازگشت مقادیر خود را از پارامترهای متناظر بگیرند؟
- برای اینکار بایستی داده ها را در غالب آدرس ارسال کنیم. به مثال بعد توجه نمائید.
- **تذکر :** در مثال بعد، تابع قبل از تابع **main** تعریف شده است.

## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

- `#include<stdio.h>`
- `int swap(int &a , int &b)`
- `{`
- `int temp=a;`
- `a = b;`
- `b = temp;`
- `return 0;`
- `}`
- `main()`
- `{`
- `int x=100, y = 200;`
- `printf("x = %d y = %d\n",x , y);`
- `swap(x , y);`
- `printf("x = %d y = %d\n",x , y);`
- `}`

# دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر



The image shows a screenshot of a Windows command prompt window. The title bar at the top reads "(Inactive D:\TCWIN45\BIN\PASSBYRE.EXE)". The window contains the following text:

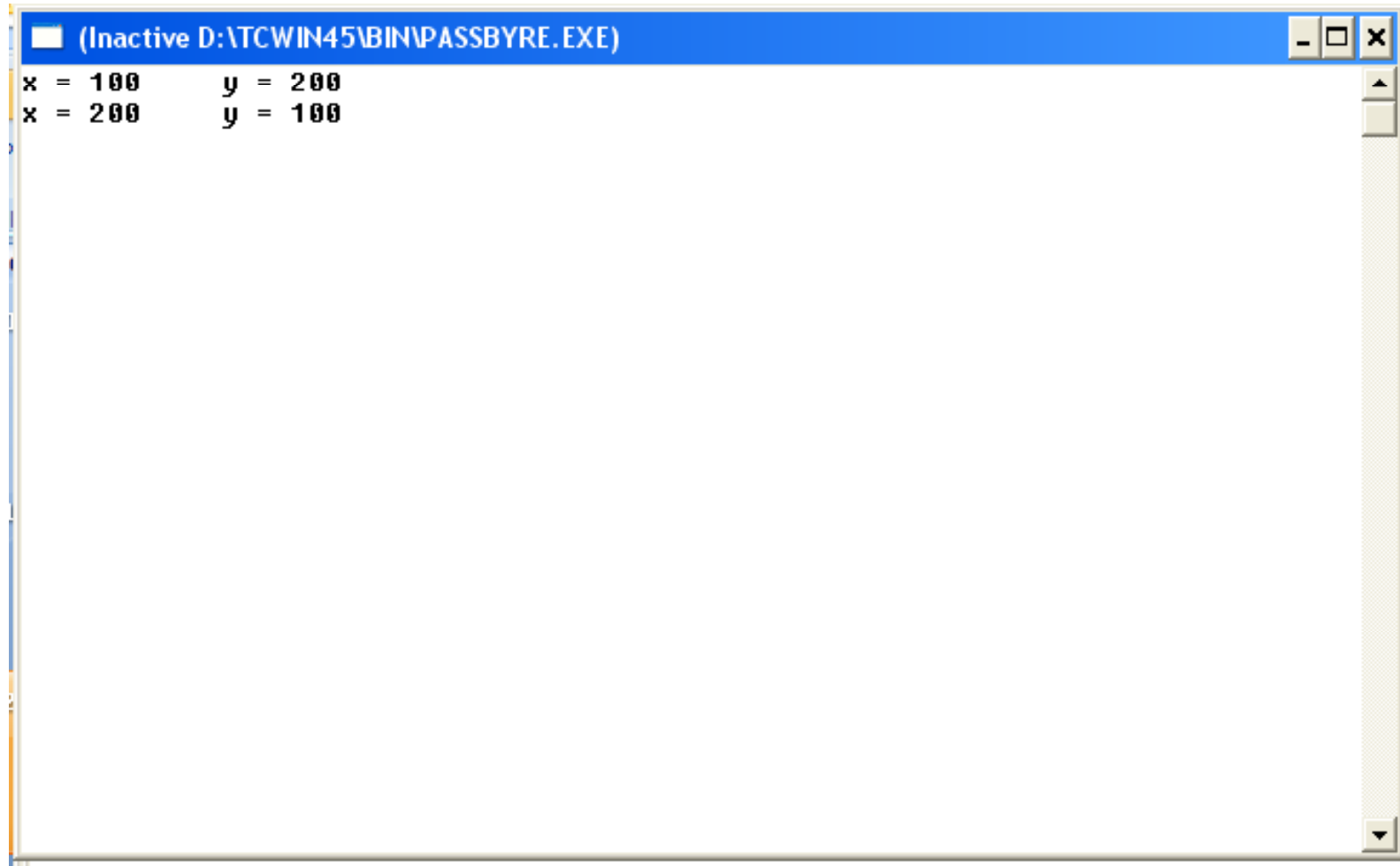
```
x = 100    y = 200  
x = 200    y = 100
```

- میتوانیم متغیرها را بصورت **Global** / سرتاسری تعریف کنیم.

```
#include<stdio.h>
int swap();
int x , y;
main()
{
x=100; y = 200;
printf("x = %d    y = %d\n",x , y);
swap();
printf("x = %d    y = %d\n",x , y);
}
int swap()
{
int temp=x;
x = y;
y = temp;
return 0;
}
C++ Tutorial
```



# دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر



The image shows a screenshot of a Windows command prompt window. The title bar is blue and contains the text "(Inactive D:\TCWIN45\BINPASSBYRE.EXE)". The window content displays two lines of text: "x = 100 y = 200" on the first line and "x = 200 y = 100" on the second line. The window has standard Windows window controls (minimize, maximize, close) in the top right corner and a vertical scrollbar on the right side.

```
(Inactive D:\TCWIN45\BINPASSBYRE.EXE)
x = 100    y = 200
x = 200    y = 100
```

## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

توابع بازگشتی (Recursive): برنامه ۴-۵:

```
/* P5.4 User_Defined function ver2 */
#include<stdio.h>

Double fact (int n);

int main()
{
    int num = 1;
    while (num) {
        printf("\nEnter a number < 0 for quit > : ");
        scanf("%d", &num);
        printf("%d! = %.0f\n", num, fact(num)); }
    return 0;
}

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

## اجرای برنامه ۴-۵:

```
Enter a number < 0 for quit > : 5
```

```
5! = 120
```

```
Enter a number < 0 for quit > : 8
```

```
8! = 40320
```

```
Enter a number < 0 for quit > : 14
```

```
14! = 87178291200
```

```
Enter a number < 0 for quit > : 0
```

```
0! = 1
```

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## روش کار در توابع بازگشتی :

مقدار بازگشتی	مقدار پارامتر	شماره فراخوانی
$4 * \text{fact}(3)$	4	1
$3 * \text{fact}(2)$	3	2
$2 * \text{fact}(1)$	2	3
1	1	4

## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

تمرین در کلاس : تابعی بنویسید که یک عدد صحیح را گرفته و بدون استفاده از آرایه، مقدار دودوئی آن را چاپ نماید. این تابع را بصورت بازگشتی بنویسید. برنامه زیر با **DevC++** تست شده است.

## دانشگاه علم و صنعت ایران – دانشکده مهندسی کامپیوتر

تمرین : تابعی بازگشتی که یک عدد صحیح را به مبنای دو می برد. این برنامه با Dev-C++ تست شده است. این برنامه توسط دانشجویان نوشته شود.

```
#include<stdio.h>
#include<conio.h>

long mabna2 (int no)
{
    if (no<=0 )
        return 0;
    else return (no % 2) + (10 * mabna2( (int) no / 2)); }

main()
{
    int n;
    printf("Enter an integer no : ");
    scanf("%d" , &n);
    printf("Base of %d is %d",n , mabna2(n));
    getch(); }
```

## تمرین پایان فصل

- ۱- تابعی بنویسید که یک کاراکتر را گرفته، در صورتی که کاراکتر گرفته شده حرف صدادار باشد مقدار یک و در غیر این صورت مقدار صفر برگرداند.
- ۲- تابعی بنویسید که یک کاراکتر را گرفته، کاراکتر بعدی آن را برگرداند.
- ۳- برنامه ای بنویسید که پنج عدد را خوانده، میانگین آن ها را چاپ نماید. کار خواندن عددها و محاسبه میانگین در تابع و کار چاپ میانگین در تابع اصلی انجام شود.
- ۴- تابعی بنام `printbit` بنویسید که یک عدد صحیح را گرفته و بدون استفاده از آرایه، مقدار دودوئی آن را چاپ نماید.

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## فصل ششم

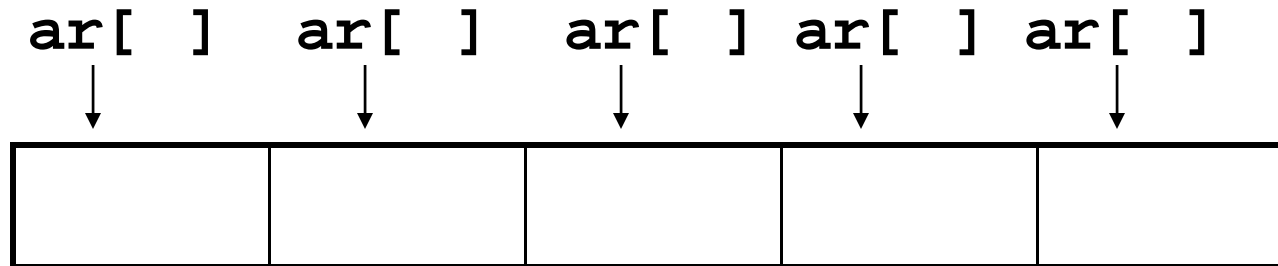
آرایه

(Array)



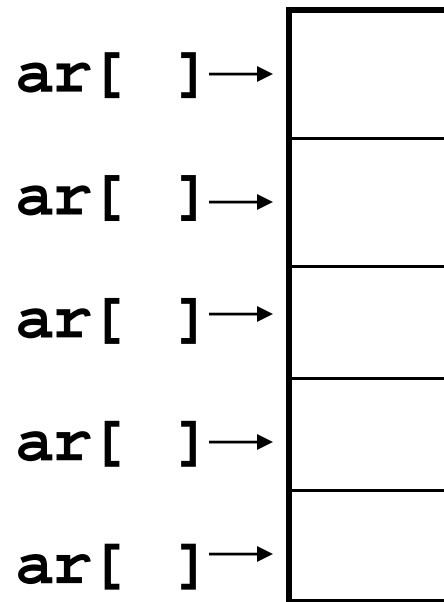
## تعریف آرایه:

به مجموعه ای از داده های هم نوع که زیر یک نام گرد آمده باشند، آرایه گفته می شود.  
برای مثال به مجموعه عناصر زیر، مجموعه متغیرهای آرایه `ar` گفته می شود.



دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

آرایه یک بعدی را به صورت زیر نیز می توان نشان داد:



## تعریف آرایه (ادامه):

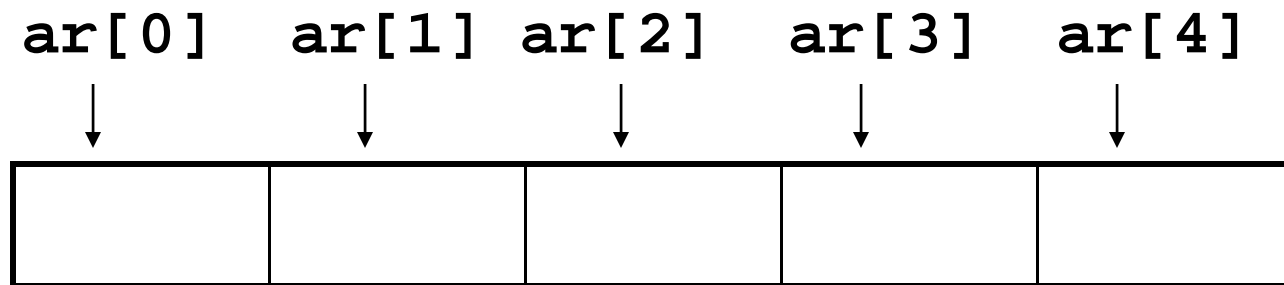
برای تعریف آرایه در زبان C، فرمت زیر را به کار می‌بریم.

```
data_type Array_Name [No_of_Elements]
```

مثال:

```
int ar[5];
```

این تعریف به این معنی است که آرایه `ar` دارای ۵ عنصر می‌باشد که اولین عنصر با اندیس 0 و آخرین عنصر با اندیس 4 مشخص می‌گردد و همگی عناصر مقادیر صحیح می‌گیرند.



دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

هر عنصر آرایه مانند یک متغیر معمولی است. بنابراین تمام قوانین مربوط به متغیرها، در مورد عناصر آرایه نیز صادق است.  
بنابراین :

```
ar[0] = 25;
```

```
b = ar[0];
```

```
ar[1] = ar[0] * 2;
```

```
ar[4] = ar[0] + ar[1];
```

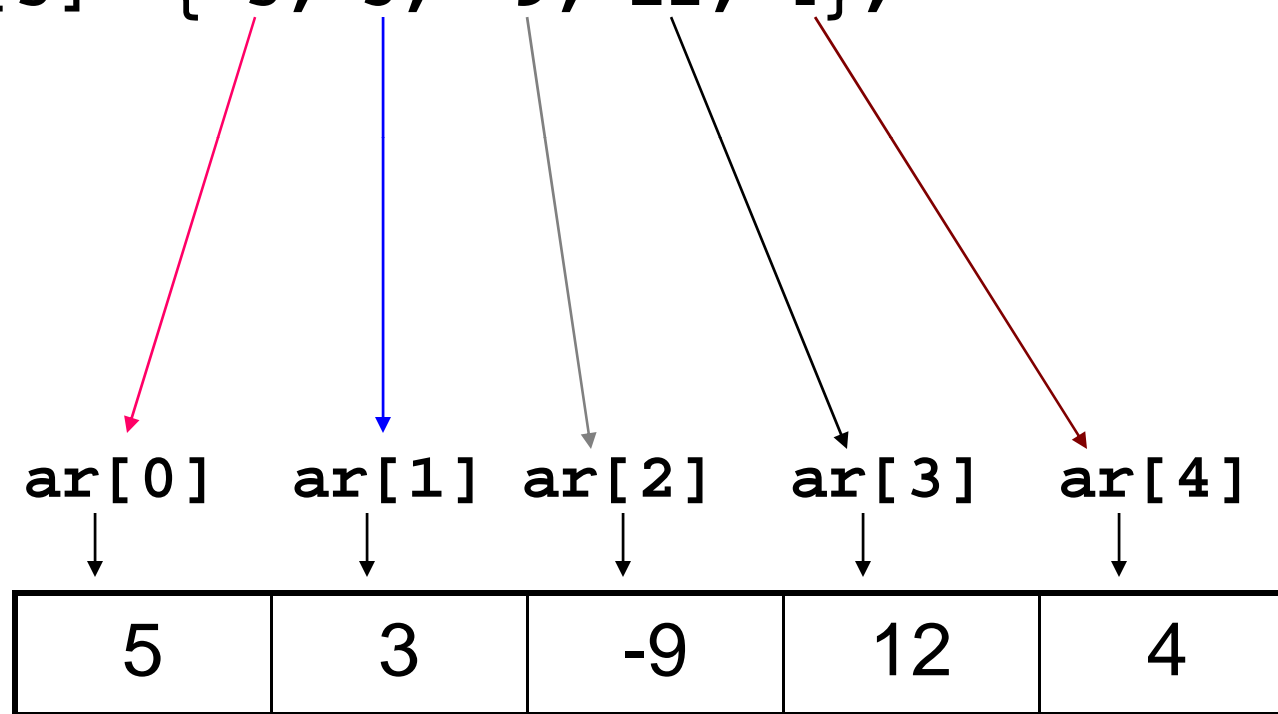
می توانیم در همان بخش تعریف، به عناصر آرایه مقداردهی کنیم.

```
int ar[6] = { 5, 3, 9, 12, 4, 3};
```

## تعریف آرایه (ادامه):

با این تعریف خواهیم داشت:

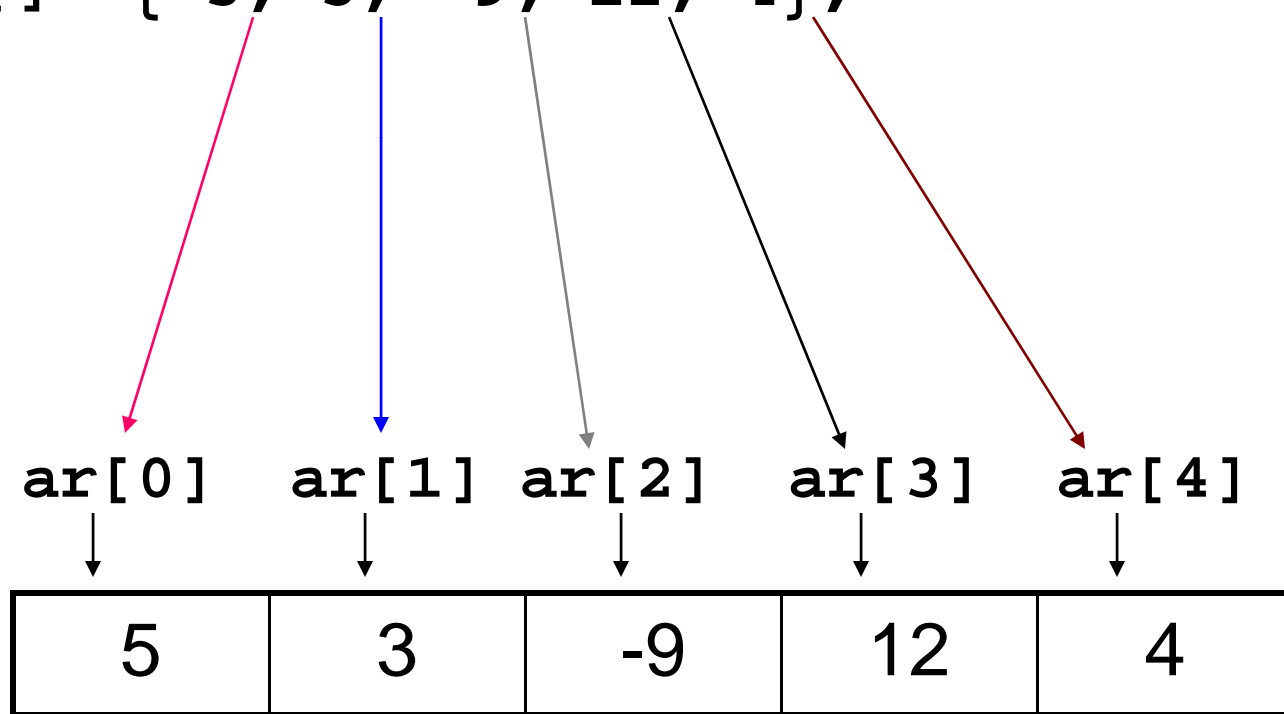
```
int ar[5]= { 5, 3, -9, 12, 4};
```



## تعریف آرایه (ادامه):

در تعریف آرایه و مقداردهی به آن می توان از نوشتن تعداد عناصر آرایه خودداری نمود.

```
int ar[] = { 5, 3, -9, 12, 4};
```



```
/* P6.1 Array */
#include<stdio.h>

void main()
{
    int b, a[6] = {5, 3, 9, 12, 4, 3};
    printf("a[3] = %d\n", a[3]);
    printf("a[5] = %d\n", a[5]);
    b = a[3];
    a[5] = 100;
    printf("b = %d\n", b);
    printf("a[5] = %d\n", a[5]);
}
```

**Running Program:**

```
a[3] = 12
a[5] = 3
b = 12
a[5] = 100
```

## آرایه و حلقه

اگر بخواهیم همه عناصر آرایه را پردازش کنیم، بهتر است آن را در درون حلقه به کار گیریم. در این صورت، شمارنده حلقه می تواند به عنوان اندیس آرایه استفاده شود.

برنامه ۲-۶:

```
/* P6.2 Array & Loops*/
#include<stdio.h>

void main()
{
    int i, a[6] = {5, 3, 9, 12, 4, 3};
    for (i = 0 ; i <= 5 ; ++i)
        printf("a[%d] = %d\n", i , a[i]);
}
```

### Running Program:

```
a[0] = 5
a[1] = 3
a[2] = 9
a[3] = 12
a[4] = 4
a[5] = 100
```



## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

**برنامه ۳-۶:** برنامه ای بنویسید که علاوه بر کار انجام شده در مثال بالا، مجموع و میانگین آنها را نیز چاپ نماید.

```
/* P6.3 Array elements & sum & average*/
#include<stdio.h>

void main()
{
    int i, sum=0, a[] = {5, 3, 9, 12, 4, 3};
    for (i = 0 ; i <= 5 ; ++i)
    {
        sum += a[i];
        printf("a[%d] = %d\n", i , a[i]);
    }
    printf("\nSum = %d\n", sum);
    printf("Average = %f\n", sum / i);
}
```

**آیا برنامه بالا نتیجه مورد نظر و درست را چاپ می کند؟**

# دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

برنامه ۳-۶: (درست)

```
/* P6.3 Array elements & sum & average*/
#include<stdio.h>

void main()
{
    int i, sum=0, a[] = {5, 3, 9, 12, 4, 3};
    for (i = 0 ; i <= 5 ; ++i) {
        sum += a[i];
        printf("a[%d] = %d\n", i , a[i]); }
    printf("Sum = %d\n", sum);
    printf("Average = %.2f\n", (float)sum / i);
}
```

## Running Program:

```
a[0] = 5
a[1] = 3
a[2] = 9
a[3] = 12
a[4] = 4
a[5] = 100
Sum = 36
Average = 6.00
```

## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

برنامه ۴-۶: برنامه زیر چه کاری انجام می دهد / نمایش می دهد؟

```
/* P6.4 */
#include<stdio.h>

void main()
{
    int i, num[10];
    for (i = 0 ; i <= 9 ; ++i) {
        printf("num[%d] = ", i);
        scanf("%d", &num[i]); }
    printf("\n");

    for (i = 0 ; i <= 9 ; ++i)
        printf("%3d", num[i]);
    printf("\n");

    for (i = 9 ; i >= 0 ; --i)
        printf("%3d", num[i]);
    printf("\n");
}
```

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

برنامه ۴-۶: اجرای برنامه :

**Running Program:**

```
nam[0] = 12  
nam[1] = 9  
nam[2] = 6  
nam[3] = 7  
nam[4] = 11  
nam[5] = 15  
nam[6] = 43  
nam[7] = 80  
nam[8] = 1  
nam[9] = 5
```

```
12  9  6  7 11 15 43 80  1  5  
 5  1 80 43 15 11  7  6  9 12
```

## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

برنامه ۵-۶: برنامه زیر چه کاری انجام می دهد / نمایش می دهد؟

```
/* P6.5 */
#include<stdio.h>
int fun1(int x, int y);

main()
{
int i , ar[6] = {8,90,34,-75,55,2,};
/*for (i=0; i<=6 ; ++i)
{
printf("\n%dth No : ",i);
scanf("%d",&ar[i]);
}*/
for (i=0; i<=5 ; ++i)
{
printf("\nAr[%d] = %d",i,ar[i]);
/*printf("\n2 Vahed addition is = %d", fun1(ar[i],2));*/
}
}
int fun1(int x, int y)
{
printf("Input No = %d %d",x , y);
return x + y;
}
}
C++ Tutorial
```

برنامه مرتب کردن مقادیر یک آرایه با استفاده از ارسال یک آرایه به تابع به روش  
Bubble Sort (این برنامه توسط دانشجویان نوشته شود)

```
#include<stdio.h>
#include<conio.h>
int bubble(int no[ ])
{ int k , j;
  for (k=9 ; k>0 ; k--)
    for (j=0 ; j<k ; j++)
      if (no[j] > no[j+1]) {
          int temp = no[j];
          no[j] = no [j+ 1];
          no[j+1] = temp;
      }
}

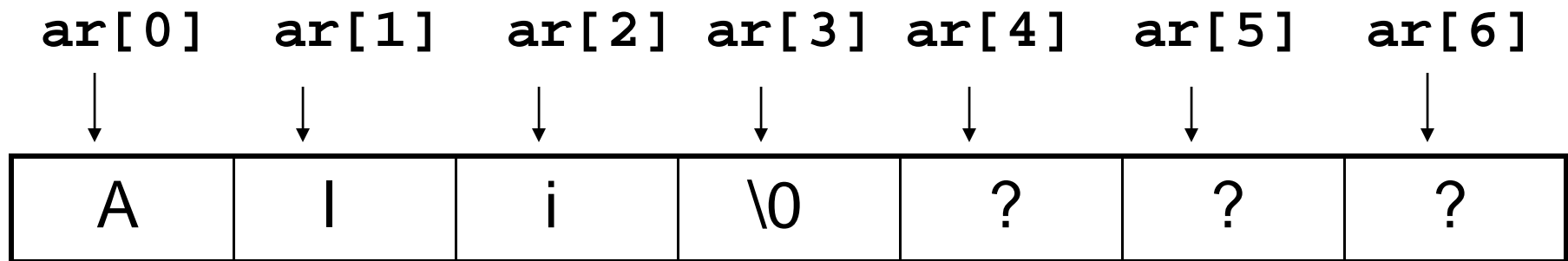
main()
{ int i;
  int num[10];
  for (i=0 ; i <= 9 ; ++i) {
      printf("Enter an integer no for num[%d] : ", i);
      scanf("%d", &num[i]);  }
  bubble(num);
  for (i=0 ; i <=9 ; i++)
      printf("\nnum[%d] = %d", i , num[i]);
  getch();
}
```

## آرایه کاراکتری:

آرایه کاراکتری آرایه ای هستند که عناصر آرایه دارای داده های کاراکتری می باشند. در زبان C، این آرایه ها را به جای رشته به کار می بریم. در این حالت، هر کاراکتر در یک عنصر آرایه جای می گیرد. مثال:

```
char ar[7];
```

```
char ar[7] = { 'A' , 'l' , 'i' };
```

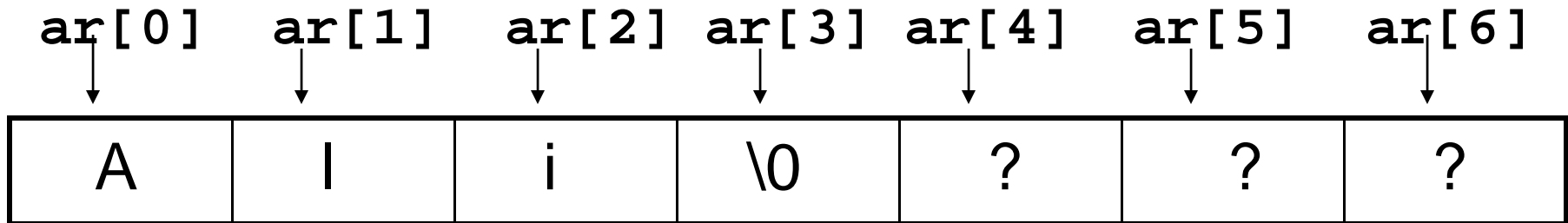


## آرایه کاراکتری: (ادامه)

در عنصر شماره 3، کاراکتر پوچ که دارای کد اسکی 0 (یعنی همه بیت های آن صفر دودوئی است) قرار می گیرد. برنامه هنگامی که به این صفر می رسد میفهمد که به پایان رشته رسیده است. بقیه عناصر دارای مقدار قبلی خود می باشند.

```
char ar[7];
```

```
char ar[7] = { 'A' , 'l' , 'i' };
```





# دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

برنامه ۵-۶: برنامه زیر چه کاری انجام می دهد / نمایش می دهد؟

```
/* P6.5 */
#include<stdio.h>

void main()
{
    char i, num[10] = { 'F' , 'e' , 'r' , 'd' , 'o' , 'w' , 's' , 'i' } ;
    for (i = 0 ; i <= 9 ; ++i)
        printf("%c = %d\n", name[i] , name[i]);
    printf("\nName = %s\n", name);
}
```

## Running Program

```
F = 70
e = 101
r = 114
d = 100
o = 111
w = 119
s = 115
i = 105
= 0
= 0
```

```
Name = Ferdowsi
```

## آرایه دو بعدی:

آرایه دو بعدی را می توان همچون جدولی در نظر گرفت که دارای تعدادی سطر و تعدادی ستون باشد.

	ar[0][0]	ar[0][1]	ar[0][2]	
ar[1][0]				ar[1][2]
ar[2][0]			ar[2][1]	ar[2][2]
ar[3][0]				ar[3][2]

## آرایه دو بعدی: (ادامه)

```
int ar[4][3] = {  
    {15, 6, 13}, {9, 17, 2},  
    {4, 5, 4}, {10, 11, 12} }
```

	ar[0][0]	ar[0][1]	ar[0][2]	
ar[1][0]	15	6	13	ar[1][2]
ar[2][0]	9	17	2	ar[2][2]
ar[3][0]	4	5 ar[2][1]	4	ar[3][2]
	10	11	12	

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

**آرایه دو بعدی: (ادامه) مقداردهی بالا را به صورت زیر نیز می توانیم بنویسیم.**

```
int ar[4][3] = {15, 6, 13, 9, 17, 2, 4, 5, 4, 10, 11, 12};
```

	ar[0][0]	ar[0][1]	ar[0][2]	
ar[1][0]	15	6	13	ar[1][2]
ar[2][0]	9	17	2	ar[2][2]
ar[3][0]	4	5 ar[2][1]	4	ar[3][2]
ar[3][0]	10	11	12	ar[3][2]

## آرایه دو بعدی: (ادامه)

چنانچه تعداد داده ها و عنصرهای آرایه برابر باشند، می توان از نوشتن تعداد سطرها و ستون های آرایه خودداری کرد.

```
float x[][] = {  
    {1.25, 12.2, 42.3}  
    {5.6, 4.11, 98.7}  
};
```

در این مثال، آرایه `x`، آرایه ای دو بعدی و اعشاری است که دارای دو سطر و سه ستون می باشد. برای دستیابی به عناصر آرایه دوبعدی، نخست باید نام آرایه، سپس شماره سطر و پس از آن شماره ستونی که عنصر در آن جای دارد را نوشت.

مثال:

```
u = t[2][1] ;  
t[1][2] = 200 ;
```

## آرایه دو بعدی (ادامه):

مقداردهی به عناصر آرایه دو بعدی از ورودی :

```
/*P6.5 Two_Dimensional Array */
#include<stdio.h>
void main()
{
    int i, j, ar[4][3];
    for (i = 0 ; i <= 3 ; ++i)
        for (j = 0 ; j <= 2 ; ++j) {
            printf("ar[%d][%d] = \", i , j);
            scanf("%d", ar[i][j]);

            .....
            .....
        }
}
```

## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

### آرایه دو بعدی (ادامه):

مقداردهی به عناصر آرایه دو بعدی از ورودی و چاپ مقادیر:

```
/*P6.5 Two_Dimensional Array */
#include<stdio.h>
void main()
{
    int i, j, ar[4][3];
    for (i = 0 ; i <= 3 ; ++i)
        for (j = 0 ; j <= 2 ; ++j)
            {
                printf("ar[%d][%d] = ", i , j);
                scanf("%d", ar[i][j]);
            }

    for (i = 0 ; i <= 3 ; ++i)
        for (j = 0 ; j <= 2 ; ++j)
            printf("ar[%d][%d] = %d", i , j, ar[i][j]);
}
```

## آرایه چند بعدی :

`data-type array-name [dimension-1][dimension-2][.....]`

مثال:

```
int r[2][3][2]={
    {
        {1, 2},
        {3, 4},
        {5, 6}
    },
    {
        {6, 7},
        {8, 9},
        {10, 11}
    }
};
```

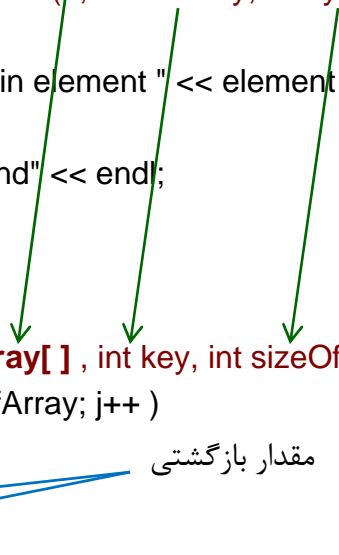
در این مثال آرایه ای سه بعدی و صحیح به نام `r` تعریف کرده ایم که دارای  $2 * 3 * 2 = 12$  عنصر می باشد.



# آرایه - نحوه ارسال آرایه به تابع

- #include <iostream.h>
- #include <conio.h>
- int linearSearch(int array[ ], int, int );
- main() {
- const int arraySize = 7;
- int a[ arraySize ]={2,6,4,3,12,10,5};
- int searchKey;
- cout << "Enter integer search key: ";     cin >> searchKey;
- int element = linearSearch(a, searchKey, arraySize);
- if ( element != -1 )
- cout << "Found value in element " << element << endl;
- else
- cout << "Value not found" << endl;
- char ch=getch();
- }
- int linearSearch(int array[ ], int key, int sizeOfArray ) {
- for ( int j = 0; j < sizeOfArray; j++ )
- if ( array[ j ] == key )
- return j;
- return -1; }

مثالی از نحوه ارسال آرایه به تابع : این برنامه یک آرایه را به یک تابع ارسال میکند و با گرفتن یک عدد، آن را درون آرایه جستجو میکند. در صورت پیدا کردن مقدار ورودی در آرایه، اندیس آن و در غیر اینصورت مقدار -1 را برمیگرداند.



## تمرین پایان فصل

- ۱- برنامه ای بنویسید که تعداد صد عدد اعشاری را از ورودی خوانده، بزرگ ترین، کوچک ترین و اندیس موقعیت این اعداد در آرایه را چاپ نماید.
- ۲- برنامه ای بنویسید که دومین بزرگ ترین را از یک مجموعه عناصر آرایه صد عنصری صحیح که مقادیر آن از ورودی دریافت می شود را به همراه اندیس آن چاپ نماید.
- ۳- برنامه ای بنویسید که کاراکترهای با کد اسکی 0 تا 255 را در یک آرایه 256 عنصره قرار دهد و در نهایت هر عنصر را با مقدارش در هر سطر چاپ نماید.
- ۴- برنامه ای بنویسید که جدول ضرب را چاپ نماید.
- ۵- برنامه ای بنویسید که یکصد عدد صحیح را از ورودی خوانده و آن ها را به صورت حبابی مرتب و چاپ نماید.
- ۶- برنامه ای بنویسید که یکصد عدد صحیح را از ورودی خوانده و آن ها را به صورت درجی-درجا مرتب و چاپ نماید.

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## فصل هفتم

توابع توکار / کتابخانه ای

# Library Function

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

فایل‌هایی که متشکل از پرونده‌ها و توابع کاربردی هستند، در زبان **C++** بسیار وجود دارند. مهم‌ترین آن‌ها عبارتند از:

- ۱- ورودی خروجی استاندارد **stdio.h**
- ۲- رشته **string.h**
- ۳- توابع ریاضی **math.h**
- ۴- کتابخانه استاندارد **stdlib.h**
- ۵- نوع کاراکتری **ctype.h**
- ۶- زمان **time.h**

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## پرونده توابع ریاضی

پرونده توابع ریاضی با نام `math.h` وجود دارد و حاوی توابعی برای عملیات ریاضی میباشد.

-تابع کمان کسینوس

`acos(d) → d`

کمانی که کسینوس آن معلوم است، محاسبه میکند. کسینوس بایستی بین  $-1$  و  $1$  باشد.

-تابع کمان سینوس

`asin(d) → d`

کمانی که سینوس آن معلوم است، محاسبه میکند. سینوس بایستی بین  $-1$  و  $1$  باشد.

## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

-تابع کمان کسینوس

$$\cos(d) \rightarrow d$$

$$\cosh(d) \rightarrow d$$

کسینوس پارامتر را برمیگرداند. در تابع دوم، کسینوس هذلولیائی پارامتر محاسبه می شود.

-توابع سینوس

$$\sin(d) \rightarrow d$$

$$\sinh(d) \rightarrow d$$

سینوس پارامتر را برمی گرداند. شکل دوم تابع برای محاسبه سینوس هذلولیائی بکار می رود.

-تابع کمان کتانژانت

$\text{atan}(d) \rightarrow d$

$\text{atan2}(d1,d2) \rightarrow d$

کمانی که تانژانت آن معلوم است، محاسبه میکند. در تابع دوم، کمان تانژانت  $d1/d2$  را محاسبه میکند.

-تابع حد بالا

$\text{ceil}(d) \rightarrow d$

کوچکترین عدد صحیح بزرگتر از پارامتر را پیدا کرده، آن را به عددی دو دقتی تبدیل و برمی گرداند. مثال:

$\text{ceil}(2.3) \rightarrow 3$

## -توابع تانژانت

$$\tan(d) \rightarrow d$$

$$\tanh(d) \rightarrow d$$

تانژانت پارامتر را محاسبه می کند. تابع دوم برای محاسبه تانژانت هذلولیائی بکار می رود. چنان که گفتیم، در توابع ریاضی، زوایا برحسب رادیان می باشد. برای تبدیل درجه به رادیان، فرمول زیر را بکار میبریم.

$$r = d * \pi / 180$$

در این فرمول،  $r$  زاویه برحسب رادیان،  $d$  زاویه برحسب درجه و  $\pi$  ثابت پی می باشد.  $\pi$  به صورت ثابت نمادین تعریف شده است. از این رو نیازی به تعریف دوباره آن نیست.



دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

-تابع قدرمطلق اعشاری

$\text{fabs}(d) \rightarrow d$

قدرمطلق اعشاری پارامتر را محاسبه می کند.

-تابع حد پائین

$\text{floor}(d) \rightarrow d$

بزرگترین عدد صحیح کوچکتر از پارامتر را پیدا کرده، آن را به عددی دو دقتی تبدیل و برمی گرداند. مثال:

$\text{ceil}(2.3) \rightarrow 2.0$

دانشگاه علم و صنعت ایران – دانشکده مهندسی کامپیوتر

-تابع باقیمانده اعشاری

$fmod(d1,d2) \rightarrow d$

باقیمانده تقسیم  $d1$  و  $d2$  را محاسبه می کند. علامت باقیمانده همانند  $d1$  است.

-لگاریتم طبیعی

$\log(d) \rightarrow d$

لگاریتم طبیعی (در پایه نپر) پارامتر را محاسبه می کند.

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

-لگاریتم در پایه 10

$\log_{10}(d) \rightarrow d$

لگاریتم پارامتر را در پایه 10 محاسبه می کند.

-تابع توان

$\text{pow}(d1,d2) \rightarrow d$

d1 را به توان d2 رسانده، نتیجه را برمیگرداند. اگر d1 برابر صفر باشد، d2 باید عددی مثبت باشد، و اگر d1 منفی باشد، d2 باید عددی صحیح باشد.

-تابع ریشه دوم

$$\text{sqrt}(d) \rightarrow d$$

ریشه دوم پارامتر را محاسبه میکند. پارامتر بایستی عددی مثبت باشد.

-تابع نمائی

$$\text{exp}(d) \rightarrow d$$

عدد نپر (  $e = 2.718282$  ) را به توان پارامتر رسانده، نتیجه را برمیگرداند. با این کار آنتی اگاریتم پارامتر محاسبه می شود. (  $e$  پایه لگاریتم طبیعی است).

## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

```
/* P7.1 Mathematical Function */
#include<stdio.h>
#include<math.h>
void main() {
    printf("acos(.5)           = %f\n",acos(.5));
    printf("asin(.5)          = %f\n",asin(.5));
    printf("atan(.5)         = %f\n",atan(.5));
    printf("ceil(1.2)        = %f\n",ceil(1.2));
    printf("cos(1.047198)    = %f\n",cos(1.047198));
    printf("cosh(.5)         = %f\n",cosh(.5));
    printf("cot(.5)          = %f\n", 1 / tan(.5));
    printf("exp(.5)           = %f\n",exp(.5));
    printf("fabs(-.5)        = %f\n",fabs(.5));
    printf("floor(1.2)       = %f\n",floor(1.2));
    printf("fmod(7.,4.)      = %f\n",fmod(7.,4.));
    printf("log(1.648721)    = %f\n",log(1.648721));
    printf("log10(1.648721)  = %f\n",log10(1.648721));
    printf("pow(2.,3.)       = %f\n",pow(2.,3.));
    printf("sin(.523599)     = %f\n",sin(.523599));
    printf("sinh(.523599)    = %f\n",sinh(.523599));
    printf("sqrt(9.)         = %f\n",sqrt(9.));
    printf("tan(.463648)     = %f\n",tan(.463648));
    printf("tanh(.463648)    = %f\n",tanh(.463648));
}
```

## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

### پرونده توابع رشته ای

پرونده توابع رشته ای `string.h` است که حاوی توابعی جهت عملیات رشته ای میباشد.

### -تابع های پیوند رشته

`strcat(s1,s2) → p`

`strncat(s1,s2,i) → p`

در تابع اول، رشته `s2` به پایان رشته اول پیوند داده می شود و نشانگر، نشانی `s1` را نشان میدهد. در تابع دوم، `i` نشان دهنده تعداد کاراکترهایی است که از `s2` به `s1` الحاق می شود.

### -توابع جستجوی کاراکتر در رشته

`strchr(s,c) → p`

`strrchr(s,c) → p`

در تابع اول، اولین بار که کاراکتر در رشته پیدا شود، نشانی آن در اشاره گر `p` گذاشته می شود. در تابع دوم، آخرین بار که کاراکتر در رشته پیدا شود، نشانی آن در اشاره گر `p` گذاشته می شود. اگر کاراکتر در رشته پیدا نشود، نشانگر پوچ می شود.

### -تابع های مقایسه رشته

`strcmp(s1,s2) → i`

`strncmp(s1,s2,i) → i`

بسته به این که  $s1 > s2$  ،  $s1 = s2$  و یا  $s1 < s2$  باشد، مقدار مثبت، صفر و یا منفی برگردانده می شود. در تابع دوم،  $i$  کاراکتر از دو رشته با هم مقایسه می شوند.

### -توابع کپی برداری از رشته

`strcpy(s1,s2) → p`

`strncpy(s1,s2,i) → p`

در تابع اول، از رشته  $s2$  در  $s1$  کپی برداری شده، نشانگر نشانی  $s1$  را نشان می دهد. در شکل دوم تابع،  $i$  کاراکتر از  $s2$  به  $s1$  منتقل می شود.

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

-تابع طول رشته

$\text{strlen}(s) \rightarrow i$

طول رشته (تعداد کاراکترهای جاری رشته) را برمی گرداند.

-توابع جستجوی رشته در رشته

$\text{strstr}(s1,s2) \rightarrow p$

اگر رشته ای همانند  $s2$  در  $s1$  پیدا شود، نشانی آن در نشانگر گذاشته می شود. در غیر این صورت، نشانگر پوچ می شود.



## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

```
/* P7.2 Mathematical Function */
#include<stdio.h>
#include<string.h>
void main() {
    char s1[80] = "Welcome to c ";
    char s2[80] = "Programming ! ";
    char s3[] = "Welcome to home ";
    char *p;
    strcat(s1,s2);
    puts(s1);
    p = strchr(s1,'m');
    puts(p);
    p = strchr(s2,'m');
    printf("\n%d\n", strcmp(s1,s3));
    printf("\n%d\n", strcmp(s3,s1));
    printf("\n%d\n", strcmp(s1,s3,10));
    strcpy(s2,s3);
    puts(s2);
    strncpy(s2, "This is my school", 11);
    puts(s2);
    printf("\n%d\n\n", strlen(s3));
    p = strstr(s1, "c pro");
    printf("%s\n\n", p);
    puts(strtok(s1, "l")); /*This function split one string to many substring */
}
```

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## توابع کتابخانه استاندارد (**stdlib.h**)

-تابع رها کردن

**abort()**

-تابع های قدر مطلق

**abs(x)**

**labs(i)**

قدر مطلق پارامتر را برمی گرداند. در تابع نخست، پارامتر، عددی صحیح و در تابع دوم، پارامتر، عددی صحیح و بلند می باشد.

## دانشگاه علم و صنعت ایران – دانشکده مهندسی کامپیوتر

-تابع تبدیل رشته به عددی دو دقتی

### **atof(s) → d**

رشته را به عددی دو دقتی تبدیل می کند. رشته باید از کاراکترهای قابل تبدیل به عدد دو دقتی (مثل کاراکترهای 0 تا 9) ساخته شده باشد. کار تبدیل هنگامی به پایان می رسد که کاراکتری جز این کاراکترها در رشته باشد. مثال :

**atof("2.5meter") → 2.5**

-تابع تبدیل رشته به عدد صحیح

### **atoi(s) → i**

رشته را به عددی دو دقتی تبدیل می کند. رشته باید از کاراکترهای قابل تبدیل به عدد دو دقتی (مثل کاراکترهای 0 تا 9) ساخته شده باشد. کار تبدیل هنگامی به پایان می رسد که کاراکتری جز این کاراکترها در رشته باشد. مثال :

**atoi("12-8-74") → 12**

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

-تابع تبدیل رشته به عدد صحیح بلند

**atol(s) → i**

رشته را به عددی دو دقتی تبدیل می کند. رشته باید از کاراکترهای قابل تبدیل به عدد دو دقتی (مثل کاراکترهای 0 تا 9) ساخته شده باشد. کار تبدیل هنگامی به پایان می رسد که کاراکتری جز این کاراکترها در رشته باشد. مثال:

**atol("12-8-74") → 12L**

-تابع خروج

**exit(i)**

برنامه را به صورت طبیعی به پایان می رساند. پارامتر به برنامه فراخواننده-که معمولاً سیستم عامل می باشد- برگشت داده می شود.

## تابع تصادفی

**rand() → i**

هر بار که تابع را فراخوانیم، عددی تصادفی را که مقدار آن صحیح و غیر منفی است، برمی گرداند.

## فایل تابع های زمان (time.h)

-تابع زمان

### time(p) → i

اگر دستگاه دارای ساعت سخت افزاری باشد، زمانی را که آن ساعت نشان می دهد، و در غیر این صورت مقدار ۱- را برمی گرداند. زمان برحسب ثانیه است و از تاریخ معینی محاسبه می شود. نشانگر از نوع صحیح بلند است که نشانی همین زمان را داراست.

-تابع تبدیل زمان به رشته

### ctime(p) → s

زمانی را که نشانی آن در نشانگر قرار دارد از ثانیه به تاریخ و ساعت تبدیل می کند. با این کار تاریخ روز و ساعت به صورت رشته ای همانند رشته زیر به دست می آید.

Sun Jan 01 11:37:13 1995

یعنی : یکشنبه ۱ ژانویه، ساعت ۱۱ و ۳۷ دقیقه و ۱۳ ثانیه سال ۱۹۹۵ .

## فایل تابع های نوع کاراکتری (ctype.h)

فایل نوع کاراکتری را زمانی به کار می بریم که بخواهیم بررسی نمائیم یک کاراکتر چه نوع کاراکتری است: حرف است؟ رقم است؟ و یا از نوع کاراکترهای کنترلی است؟ و ...  
-تابع تعیین اینکه کاراکتر حرفی-رقمی است؟

### isalnum(c) → i

اگر کاراکتر حرف یا رقم باشد، مقدار درست (عددی غیر صفر) و در غیر این صورت، مقدار نادرست (صفر) برمی گرداند.

-تابع تعیین اینکه کاراکتر حرفی است؟

### isalpha(c) → i

اگر کاراکتر حرف باشد، مقدار درست (عددی غیر صفر) و در غیر این صورت، مقدار نادرست (صفر) برمی گرداند.

## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

-تابع تعیین اینکه کاراکتر کنترلی است؟

**iscntrl(c) → i**

اگر کاراکتر کنترلی باشد (کد اسکی آن کوچکتر از 32 یا برابر 127 باشد)، مقدار درست (عددی غیر صفر) و در غیر این صورت، مقدار نادرست (صفر) برمی گرداند.

-تابع تعیین اینکه کاراکتر رقمی است؟

**isdigit(c) → i**

اگر کاراکتر یکی از کاراکترهای 0 تا 9 باشد، مقدار درست (عددی غیر صفر) و در غیر این صورت، مقدار نادرست (صفر) برمی گرداند.



## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

-تابع تعیین اینکه کاراکتر حرف کوچک است؟

**islower(c) → i**

اگر کاراکتر حرف کوچک باشد (یعنی a تا Z)، مقدار درست (عددی غیر صفر) و در غیر این صورت، مقدار نادرست (صفر) برمی گرداند.

-تابع تعیین اینکه کاراکتر حرف بزرگ است؟

**isupper(c) → i**

اگر کاراکتر حرف بزرگ باشد (یعنی A تا Z)، مقدار درست (عددی غیر صفر) و در غیر این صورت، مقدار نادرست (صفر) برمی گرداند.

-تابع تبدیل به حروف کوچک

**tolower(c) → i**

اگر کاراکتر یکی از حروف بزرگ ( A تا Z ) باشد، آن را به حرف کوچک معادل خود (یعنی a تا Z) تبدیل می کند.

-تابع تبدیل به حروف کوچک

**toupper(c) → i**

اگر کاراکتر یکی از حروف کوچک ( a تا Z ) باشد، آن را به حرف بزرگ معادل خود (یعنی A تا Z) تبدیل می کند.

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## فصل هشتم

نشانگر

# Pointer

## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

**نشانگر**: متغیری است که برای نگهداری نشانی داده‌ها در حافظه به کار می‌رود. به عبارن دیگر، نشانگر آدرس و یا مکان داده‌ها در حافظه را نشان می‌دهد. فرمت تعریف نشانگر چنین است:

`/* data_type * pointer_name; */` → نام نشانگر \* نوع داده

مثال:

```
int *p;
```

```
char *ch;
```

در این مثال، `p` نشانگری است که می‌تواند نشانی متغیرهای صحیح را نگهداری نماید و `ch` هم نشانگری است که می‌تواند برای نگهداری آدرس کاراکترها به کار رود.

مثال:

```
x = *p;
```

در این مثال خواسته ایم داده‌ای را که نشانی آن در `p` قرار دارد به `x` واگذار نمائیم.

```
int a , *p;
```

```
a = 100;
```

```
p = &a;
```

اکنون در همه جا می توانیم  $*p$  را به جای  $a$  به کار ببریم.

```
x = a;
```

```
y = *p;
```

با دو دستور فوق، هم  $x$  و هم  $y$  برابر ۱۰۰ می شوند.

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

در شکل زیر فرض کرده ایم که متغیر  $p$  در آدرس 65520 و متغیر  $a$  در آدرس 65522 قرار دارد.



# دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

برنامه ۱-۸:

```
/* P8.1 Pointer Example */
#include <stdio.h>
int main(void)
{
    int a,*p;
    a = 100;
    p = &a;
    printf("address of a = %5u\n", &a); /* u is unsigned Integer & equal d */
    printf("value of a = %d\n\n", a);
    printf("address of p = %5u\n", &p);
    printf("value of p = %5u\n", p);
}
```

## Running Program

```
address of a = 65522
value of a = 100
```

```
address of p = 65520
value of p = 65522
```

# دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

برنامه ۲-۸:

```
/* P8.2 Pointer & variables */
#include <stdio.h>
int main(void)
{
    int a, *p, x, y;
    a = 100;
    p = &a;
    x = a;
    y = *p;
    printf("x = %d\n", x);
    printf("y = %d\n", y);
    printf("\na = %d\n", a);
    ++*p;
    printf("a = %d\n", a);
}
```

## Running Program

```
x = 100
y = 100
```

```
a = 100
a = 101
```



دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

در مثال قبل توجه نمائید که دستور  $++*p$ ، عدد ۱ را به  $a$  افزوده است. اگر بخواهیم عملگر افزایش و یا کاهش را پس از نشانگر به کار ببریم، باید نشانگر را درون پرانتز قرار دهیم.

$( *p ) ++$

$( *p ) --$

همانگونه که در مورد اولویت ها صحبت نمودیم، اولویت عملگرهای  $++$  و  $--$  از  $*$  بیشتر است. بنابراین اگر از پرانتز برای تغییر اولویت عملگرها استفاده نکنیم، نخست مقدار نشانگر افزایش و یا کاهش یافته (به عبارت دیگر، نشانی تغییر می کند)، سپس به محتویات نشانی مراجعه می‌گردد.

# دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

برنامه ۳-۸:

```
/* P8.3 Pointer increment */
#include <stdio.h>
int main(void)
{
    int b = 20, a = 10, *p;
    p = &a;
    printf("address of a = %5u\n", &a);
    printf("address of b = %5u\n\n", &b);
    ++*p;
    printf("a = %d          p = %u\n", a, p);
    *p++;
    printf("a = %d          p = %u\n", a, p);
    ++*p;
    printf("b = %d          p = %u\n", b, p);
}
```

## Running Program

address of a = 65520

address of b = 65522

a = 11 p = 65520

a = 11 p = 65522

b = 21 p = 65522

## نشانگر و محاسبه :

نشانگرها را هم می توان همچون متغیرهای دیگر، در محاسبه به کار برد. ولی تنها کاری که بر روی آن ها می توان انجام داد، جمع و تفریق است. از این رو نشانگرها می توانند با عملگرهای  $+$ ،  $-$ ،  $++$  و  $--$  به کار روند.

هنگامی که به یک نشانگر عدد  $n$  را می افزائیم، بسته به نوع نشانگر،  $n$  یا  $2n$  و یا .... به آن افزوده می شود. مثلا اگر نشانگر از نوع **کاراکتری**، **صحیح**، **اعشاری**، **دو دقتی** و ... باشد، مقدار افزایش به ترتیب برابر  $n$ ،  $2n$ ،  $4n$ ،  $8n$  و ... می شود. بعد خواهیم دید که نشانگر می تواند آدرس **ساختارها** و **انجمن ها** را هم نگهداری کند.

# دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

برنامه ۴-۸:

```
/* P8.3 Pointer Arithmetic */
#include <stdio.h>
int main(void)
{
    char c = '*', *p;
    p = &c;
    printf("c = %c      p = %u\n" , *p, p);

    p += 10;
    printf("c = %c      p = %u\n" , *p, p);

    p -= 10;
    printf("c = %c      p = %u\n" , *p, p);
}
```

## Running Program

```
c = *      p = 65523
c =      p = 65533
c = *      p = 65523
```

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## نشانگر و محاسبه (ادامه):

با افزایش و کاهش نشانگر می توان بر همه داده هائی که در حافظه جای دارند، دست یافت. در برنامه صفحه بعد، به کمک نشانگر عددهای ۱ تا ۱۰ را در حافظه گذاشته، سپس آن ها را از حافظه برداشته و چاپ می کنیم.

# دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

برنامه ۵-۸:

```
/* P8.5 Pointer as Array */
#include <stdio.h>
int main(void)
{
    char n, a[100], *p;
    p = &a[0] + 10;
    for (n = 1; n <= 10 ; ++n) {
        *p = n;
        --p; }
    for (n = 1; n <= 10 ; ++n) {
        ++p;
        printf("%4d", *p); }
    printf("\n");
    for (n = 1; n <= 10 ; ++n) {
        printf("%4d", *p);
        --p;}
}
```

## Running Program

```
10 9 8 7 6 5 4 3 2 1
 1 2 3 4 5 6 7 8 9 10
```

## نشانگر و آرایه :

در زبان C نام آرایه نشانگری است که نشانی نخستین عنصر آرایه را دارا میباشد. در زیر روش دستیابی به عناصر آرایه ای را هم به شیوه نشانگری و هم به روش آرایه ای نشان داده ایم.

`a[0] = *a`

`a[1] = *(a+1)`

`a[2] = *(a+2)`

.....

`a[n] = *(a+n)`

# دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

برنامه ۶-۸:

```
/* P8.6 Arrays as Pointers */
#include<stdio.h>
int main(void) {
    int n,a[10]={1,2,3,4,5,6,7,8,9,10};
    for (n=0; n<10 ; ++n)
        printf("a[%d] = %2d *(a + %d)= %2d\n", n, a[n], n, *(a + n));
}
```

## Running Program

```
a[0] = 1 *(a + 0) = 1
a[1] = 2 *(a + 1) = 2
a[2] = 3 *(a + 2) = 3
a[3] = 4 *(a + 3) = 4
a[4] = 5 *(a + 4) = 5
a[5] = 6 *(a + 5) = 6
a[6] = 7 *(a + 6) = 7
a[7] = 8 *(a + 7) = 8
a[8] = 9 *(a + 8) = 9
a[9] = 10 *(a + 9) = 10
```



## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

باید دانست همان گونه که آرایه را می توان همچون نشانگر به کار برد، نشانگر را نیز می توان همانند آرایه به کار گرفت.

```
/* P8.7 Pointers and Arrays */
#include<stdio.h>
int main(void) {
    int n, *p, a[10]={1,2,3,4,5,6,7,8,9,10};
    p = a;
    for (n=0; n<10 ; ++n)
        printf("a[%d] = %2d    p[%d] = %2d\n", n, a[n], n, p[n]);
}
```

### Running Program

```
a[0] = 1    p[0] = 1
a[1] = 2    p[1] = 2
a[2] = 3    p[2] = 3
a[3] = 4    p[3] = 4
a[4] = 5    p[4] = 5
a[5] = 6    p[5] = 6
a[6] = 7    p[6] = 7
a[7] = 8    p[7] = 8
a[8] = 9    p[8] = 9
a[9] = 10   p[9] = 10
```

## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

البته می توانستیم به جای نام آرایه، نشانی نخستین عنصر آن را به نشانگر **p** واگذار نماییم.

```
/* P8.8 Pointers and Arrays */
#include<stdio.h>
int main(void) {
    int n, *p, a[10]={1,2,3,4,5,6,7,8,9,10};
    p = &a[0];
    for (n=0; n<10 ; ++n)
        printf("a[%d] = %2d      p[%d] = %2d\n", n, a[n], n, p[n]);
}
```

### Running Program

a[0] = 1	p[0] = 1
a[1] = 2	p[1] = 2
a[2] = 3	p[2] = 3
a[3] = 4	p[3] = 4
a[4] = 5	p[4] = 5
a[5] = 6	p[5] = 6
a[6] = 7	p[6] = 7
a[7] = 8	p[7] = 8
a[8] = 9	p[8] = 9
a[9] = 10	p[9] = 10

## دانشگاه علم و صنعت ایران – دانشکده مهندسی کامپیوتر

### نشانگر و رشته :

پیش از این دیدیم که در زبان C، آرایه های کاراکتری را به جای رشته به کار می بریم. از آن جایی که آرایه و نشانگر را می توان به جای هم به کار برد، پس نشانگرهای کاراکتری می توانند به جای رشته به کار گرفته شوند.

مثال :

```
/* P8.9 Pointers and Strings */
#include<stdio.h>
int main(void) {
    char *nam, name[20]="Anahita";
    nam = name;
    printf("%s \n%s \n", nam, name);
}
```

### Running Program

Anahita

Anahita

## دانشگاه علم و صنعت ایران – دانشکده مهندسی کامپیوتر

نشانگر و رشته (ادامه):

یکی از تفاوت های آرایه و نشانگر این است که اگر بخواهیم رشته ای را به کمک دستور واگذاری در آرایه بگذاریم، نشدنی است، ولی در نشانگر این کار را می توان انجام داد.

مثال:

```
/* P8.10 Pointers and Strings */
#include<stdio.h>
int main(void) {
    char *name2, name1[10];
    /* name1 = "Farshad";    Error! */
    name2 = "Farshad";
    printf("%s\n", name2);
}
```

**Running Program**

Farshad

## دانشگاه علم و صنعت ایران – دانشکده مهندسی کامپیوتر

**نشانگر و رشته (ادامه) :**

برای گذاشتن رشته در آرایه کاراکتری، یا باید کاراکترها را یکی یکی به آرایه منتقل کنیم و یا اینکه با استفاده از تابع **strcpy** این کار را انجام دهیم. این تابع در فایل **string.h** قرار دارد.

مثال :

```
/* P8.11 Pointers and Strings */
#include<stdio.h>
#include<string.h>
int main(void) {
    char nam[10];
    strcpy(nam, "Farshad");
    printf("%s\n", nam);
}
```

**Running Program**

Farshad

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

**آرایه نشانگری:** آرایه نشانگری، آرایه ای است که هر عنصر آن یک نشانگر  
میباشد. مثال :

```
char *city[5];
```

در این مثال، چون نشانگرها از نوع کاراکتری می باشند، می توانند به جای رشته به کار  
روند. در نتیجه مثال بالا را می توان به عنوان آرایه رشته ای به کار برد. در مثال صفحه  
بعد، از این روش برای نگهداری نام پنج شهر بزرگ ایران استفاده کرده ایم.

## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

مثال (آرایه نشانگری): تابع `puts` پس از چاپ رشته، چشمک زن را به سطر بعد می برد

```
/* P8.12 Pointers Arrays */
#include<stdio.h>
int main(void) {
    int i;
    char *city[]= {"Tehran","Mashad","Esfahan","Tabriz",Shiraz"};
    puts("Five Great Cities");
    puts("-----");
    for (i=0; i < 5 ; ++i)
        printf("      %s\n", city[i]);
}
```

### Running Program

```
Five Great Cities
-----
Tehran
Mashad
Esfahan
Tabriz
Shiraz
```

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## آرایه نشانگری:

آرایه نشانگری می تواند بیش از یک بعد داشته باشد. در برنامه صفحه بعد، آرایه دو بعدی و نشانگری به نام فرهنگ تعریف کرده ایم که در آن، چند واژه انگلیسی را همراه با معادل های فارسی آن ها نوشته ایم.



## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

مثال ۲ (آرایه نشانگری): (تابع puts پس از چاپ رشته، چشمک زن را به سطر بعد می برد)

```
/* P8.13 English - Persian Dictionary */
#include<stdio.h>
int main(void) {
    int i;
    char *Farhang[][2]= {"Flower","Gol","Apple","Sib","Peach","Holu"};
    puts("\nEnglish      Persian");
    puts("-----");
    for (i=0; i < 3 ; ++i)
        printf("-10s %-10s\n", farhang[i][0], farhang[i][1]);
}
```

### Running Program

```
English      Persian
-----
Flower       Gol
Apple        Sib
Peach        Holu
```

## پارامتر نشانگری :

اگر پارامترهای تابع از نوع نشانگری باشند، آن ها را پارامتر نشانگری می نامیم. پارامتر نشانگری را هنگامی به کار می بریم که بخواهیم به جای یک داده، نشانی آن را به تابع بفرستیم.  
مثال:

```
fun (int *p);
```

در این مثال، fun تابعی است که دارای یک پارامتر نشانگری از نوع صحیح به نام n می باشد.  
**تذکر :** تابع ها نمی توانند پارامترهای خود را تغییر دهند، مگر این که از نوع آرایه ای یا نشانگری باشند و یا آدرس آن ها را به تابع بفرستیم. در مثال بالا، چون p پارامتر نشانگری است، مقدار آن در تابع (در واقع مقداری که این نشانگر نشانی آن را دارد) می تواند در تابع تغییر کند.

# دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

برنامه ۸-۱۴:

```
/* P8.14 Pointer as Parameter */
#include<stdio.h>

void fun (int *q);

int main()
{
    int a=100, *p = &a;

    printf("\np1 = %d\n", *p);
    fun(p);
    printf("p4 = %d\n", *p);
    return 0;
}

void fun (int *q)
{
    printf ("p2 = %d\n" , *q);
    *q = 333;
    printf("p3 = %d\n", *q);
}
```

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## اجرای برنامه ۸-۱۴:

**p1 = 100**

**p2 = 100**

**p3 = 333**

**p4 = 333**

## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

### نشانگر تابعی :

نشانگر تابعی، نشانگری است که برای نگهداری نشانی تابعها به کار می رود. از این نشانگر بیشتر در کارهای نرم افزار نویسی، مثل سیستم عامل، مترجم ها، بازی های کامپیوتری و ... استفاده می شود. فرمت نوشتاری آن به شکل زیر است:

**;(پارامترها)(نشانگر \*) نوع**

### مثال :

```
int (*p) (int n);
```

در این مثال، **p** نشانگر تابعی است که می تواند برای نگهداری نشانی تابع های نوع صحیح که دارای یک پارامتر صحیح باشند به کار رود. برای مثال، اگر تابعی را به شکل زیر تعریف کنیم:

```
int fun (int n)
```

آن گاه می توانیم آن را به کمک نشانگر به صورت زیر تعریف کنیم:

```
p = fun;
```

```
x = (*p) (y);
```

این دو دستور، روی هم کار دستور زیر را انجام می دهند:

```
x = fun (y);
```

## نشانگر تهی :

نشانگر تهی، نشانگری است که از نوع تهی (void) تعریف شده باشد. مثال :

```
void *vp;
```

در این مثال، vp نشانگر نوع تهی است، یعنی در حقیقت نوع داده ای را که این نشانگر نشان می دهد را تعیین نکرده ایم. نشانگرهای تهی را برای نشان دادن هر گونه داده می توان به کار برد، در حالی که نشانگرهای نوع دیگر معمولاً برای نشان دادن داده های هممنوع خود به کار میروند.

در برنامه صفحه بعد، نشانگر تهی را یک بار برای نگهداری نشانی عدد صحیح  $n$ ، و بار دیگر برای نگهداری نشانی عدد اعشاری  $m$  به کار برده ایم. البته برای چاپ اعداد، دوباره نشانی آن ها را در نشانگرهای هممنوع خود آن ها گذاشته ایم.

## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

مثال :

```
/* P8.15 void pointers */
#include<stdio.h>
int main() {
    int i , *ip;
    float m = 20.5, *fp;
    void *vp;
    vp = &n;
    ip = vp;
    printf("n = %d\n" , *ip);
    vp = &m;
    fp = vp;
    printf("m = %.1f\n", *fp);
    return 0;
}
```

### Running Program

```
n = 10
m = 20.5
```

## تمرینات پایان فصل

۱- برنامه ای بنویسید که به کمک دستور حلقه (while) و نشانگر، اعداد یک تا ده را چاپ نماید. این کار را با for نیز انجام دهید.

۲- اگر cp، ip، fp و dp به ترتیب نشانگرهای کاراکتری، صحیح، اعشاری و دو دقتی باشند، دستورهای زیر چه کار را انجام می دهند؟

cp++;

ip++;

fp++;

dp++;

۳- تابع مقایسه رشته ها (strcmp) را یک بار به کمک رشته و یک بار به کمک نشانگر بنویسید.

۴- تابع کپی رشته ها (strcpy) را یک بار به کمک پارامتر آرایه ای و یک بار به کمک پارامتر نشانگری بنویسید.



دانشگاه علم و صنعت ایران – دانشکده مهندسی کامپیوتر

## فصل نهم

فایل ها

**Files**

## انواع فایل ها:

- File of Character**
- Text File**
- File of Record**

## فایل های کاراکتری

فایل های کاراکتری را فایل های دودوئی نیز می نامند . برای نوشتن یک کاراکتر داخل آن ها می توان از تابع **fputc** و برای خواندن یک کاراکتر از داخل آن می توان از **fgetc** استفاده کرد.

## فایل های متنی (Text Files)

فایل های متنی، فایل هایی هستند که از تعدادی سطر تشکیل شده اند و اندازه هر سطر می تواند متفاوت باشد. برای کار با فایل های متنی، معمولا یک سطر از فایل را می خوانیم و یا داخل فایل می نویسیم. در این فایل ها، سطرها در هنگام ذخیره شدن به دنبال هم قرار می گیرند و سطرها با استفاده از دو کاراکتر **CR** (با کد ۱۳) و کاراکتر **Line Feed** (با کد ۱۰) از یکدیگر جدا می شوند.

برای نوشتن در پرونده های متنی از تابع **fprintf** و برای خواندن/پیمایش از پرونده های متنی از تابع **fscanf** استفاده می کنیم.

## فایل های رکوردی

### - تعریف رکورد ؟

فایل های رکوردی شامل تعدادی رکورد هستند که همگی آن ها دارای یک ساختار (هم اندازه، دارای فیلدهای یکسان و هم تایپ) می باشند. در این فایل ها، رکوردها بی هیچ نشانه ای به دنبال یکدیگر نوشته می شوند.

برای کار با فایل های رکوردی، یا یک یا چند رکورد را در آن می نویسیم و یا اینکه یک یا چند رکورد را از آن می خوانیم.

برای کار با فایل های رکوردی از توابع **fread** (خواندن یک رکورد از فایل) و **fwrite** (نوشتن یک رکورد در فایل) استفاده می کنیم.

**تذکر : به فایل های رکوردی نیز فایل های دودوئی گفته می شود.**

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## روش های دستیابی به اطلاعات داخل فایل ها

۱- ترتیبی (Sequential)

۲- مستقیم / تصادفی (Random)

برای پردازش فایل های تصادفی و خواند یک رکورد مشخص، می توانیم از تابع **fseek** استفاده کنیم.

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## روش های پردازش فایل ها

۱- نوشتن

۲- خواندن

۳- الحاق کردن / پیوست کردن

۴- به روز کردن / اصلاح کردن

پردازش پرونده ها به کمک تعدادی تابع کتابخانه ای انجام میشوند که در فایل ورودی = خروجی استاندارد (**stdlib.h**) وجود دارند.

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## توابع پردازش فایل های سطری (متنی)

- ۱- بازکردن پرونده (**fopen**)
- ۲- نوشتن در پرونده (**fprintf**)
- ۳- پیمایش پرونده (**fscanf**)
- ۴- پایان پرونده (**feof**)
- ۵- بستن پرونده (**fclose**)



دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## تابع باز کردن پرونده (**fopen**)

فرمت نوشتاری این تابع به صورت زیر است:

```
variable_file_name = fopen (file_name , pattern);
```

مثال :

```
main( )  
{  
...  
FILE *out;  
...  
out = fopen ("daneshju" , "w");  
...  
}
```

## شیوه های پردازش استاندارد

شیوه پردازش	باز کردن برای ....
w	نوشتن در فایل متنی
r	خواندن از فایل متنی
a	الحاق کردن در فایل متنی
wb	نوشتن در فایل کاراکتری و یا رکوردی (دودوئی =b)
rb	خواندن از فایل کاراکتری و یا رکوردی (دودوئی)
ab	الحاق کردن به فایل کاراکتری و یا رکوردی (دودوئی)

## شیوه های پردازش استاندارد (ادامه)

شیوه پردازش	باز کردن برای ....
<b>w+</b>	نوشتن و به روز کردن فایل متنی
<b>r+</b>	خواندن و به روز کردن فایل متنی
<b>a+</b>	الحاق کردن در فایل متنی و به روز کردن آن
<b>w+b</b>	نوشتن در و به روز کردن فایل کاراکتری و یا رکوردی
<b>r+b</b>	خواندن از فایل کاراکتری و یا رکوردی و به روز کردن
<b>a+b</b>	الحاق کردن به فایل کاراکتری و یا رکوردی و به روز کردن

## تابع نوشتن در پرونده متنی (**fprintf**)

فرمت نوشتاری این تابع به صورت زیر است:

```
[no_of_char=] fprintf (file_name , print_pattern,variables);
```

مثال :

```
main()  
{  
...  
FILE *out;  
...  
out = fopen ("daneshju" , "w");  
...  
fprintf(out, "%s %5.2f\n", name, mark);  
...  
}
```

## الگوهای نوشتن در پرونده متنی

الگو	برای چاپ یک ....
<code>%c</code>	کاراکتر
<code>%i , %d</code>	عدد صحیح در مبنای ده
<code>%u</code>	عدد صحیح بدون علامت و در مبنای ده
<code>%o</code>	عدد صحیح در مبنای هشت ( <b>octal</b> )
<code>%X , %x</code>	عدد صحیح در مبنای شانزده ( <b>Hex</b> )
<code>%f</code>	عدد اعشاری

## الگوهای نوشتن در پرونده متنی (ادامه)

الگو	برای چاپ یک ....
<code>%E , %e</code>	عدد اعشاری به صورت نمائی
<code>%G , %g</code>	عدد اعشاری به صورت نمائی یا معمولی
<code>%s</code>	رشته
<code>%p</code>	اشاره گر / نشانگر
<code>%%</code>	کاراکتر %

## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

مثال :

```
/* file example */
#include<stdio.h>
void main()
{
    char name[20];
    float mark;
    FILE *out;
    out = fopen ("Daneshju" , "w");
    if (out == NULL) /*the file is not open for a reason */ {
        puts ("Cannot open output file !!!);
        return; }
    puts("\nEnter 'end' to EXIT ")
        "\n----- ");
    printf(" Name : ");
    scanf("%s", name);
    while (strcmp(name , "end" )) {
        printf (" Mark : "); scanf ("%f", &mark);
        fprintf(out, "%s %5.2f\n", name , mark);
        printf("\n Name : ");
        scanf("%s", name);
    }
}
```

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

اجرای برنامه مثال قبل :

Enter 'end' to EXIT

-----

Name : Reza

Mark : 19

Name : Ahmad

Mark : 18

Name : Mehdi

Mark : 16.5

Name : end



دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

حال می توانیم فایل **Daneshju** را با یک واژه پرداز (همانند **notepad**) باز کرده و یا با استفاده از برنامه **type** سیستم عامل، محتوای آن را روی صفحه مانیتور ببینیم.

- دستور **TYPE** ؟

## تابع خواندن از فایل متنی (**fscanf**)

**[no\_of\_char=] fscanf (file\_name , read\_pattern,variables);**

اگر فایل به پایان برسد، یا در خواندن اشکالی بوجود آید، مقدار این متغیر برابر **EOF** و یا -۱ می شود.  
مثال :

```
main( )
{
...
FILE *in;
...
in = fopen ("daneshju" , "r");
...
fscanf(in, "%s %f\n", name, &mark);
...
}
```

## الگوهای خواندن از پرونده متنی

الگو	برای خواندن یک ....
<code>%c</code>	کاراکتر
<code>%i</code>	عدد صحیح در مبنای ده، شانزده و هشت
<code>%d</code>	عدد صحیح در مبنای ده
<code>%o</code>	عدد صحیح در مبنای هشت (octal)
<code>%x</code>	عدد صحیح در مبنای شانزده (Hex)
<code>%g , %e , %f</code>	عدد اعشاری به صورت معمولی و یا نمائی

## الگوهای خواندن از پرونده متنی (ادامه)

الگو	برای خواندن یک ....
%s	رشته
%p	اشاره گر / نشانگر
%[]	مجموعه ای از کاراکترها
%ld	اعداد صحیح بلند
%lf	اعداد اعشاری دو دقتی بلند

الگوی **[%]**، را برای خواندن رشته ها به کار می بریم. کاراکترهایی را که رشته باید تنها از آن ها ساخته شوند را در میان دو کاراکتر **[ و ]** قرار می دهیم.

- **[%abc]**: نشان دهنده آن است که رشته بایستی از کاراکترهای **a** و **b** و **c** ساخته شده باشد. اگر رشته دارای کاراکتری جز این باشد، کار خواندن رشته به پایان رسیده و خواندن داده بعدی شروع می شود.

- **[%a-k]**: یعنی رشته باید از کاراکترهای **a** تا **k** ساخته شده باشد.

- **[%^ijk]**: یعنی رشته می تواند از هر کاراکتری جز کاراکترهای **i** و **j** و **k** تشکیل شده باشد.

## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

مثال :

```
/* file reading example */
#include<stdio.h>
void main()
{
    char name[20];
    float mark;
    FILE *in;
    in = fopen ("Daneshju" , "r");
    if (out == NULL) /*the file is not open for a reason */ {
        puts ("Cannot open output file !!!);
        return; }
    puts("\nName          Mark ")
        "\n----- ");

    while (!feof(in)) {
        fscanf(in , "%s %f\n" , name , &mark);
        printf("%-20s %6.2f\n", name , mark);
    }
}
```

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

اجرای برنامه مثال قبل :

<b>Name</b>	<b>Mark</b>
-----	
<b>Reza</b>	<b>19.00</b>
<b>Ahmad</b>	<b>18.00</b>
<b>Mehdi</b>	<b>16.50</b>

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

**تابع بستن فایل**

**مثال :**

**fclose(in)**



## تابع خواندن کاراکتر از فایل :

```
[Variable = ] fgetc(file_name);
```

در این عبارت، `variable` متغیری است که کاراکتر خوانده شده در آن قرار می گیرد. اگر در خواندن فایل اشکالی پیش آید و یا فایل به پایان برسد، مقدار متغیر برابر با مقدار ثابت پایان پرونده (EOF) می شود.

**تذکر :** پرونده بایستی برای خواندن و یا به روز شدن باز شده باشد.

## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

**مثال : خواندن نام فایل از ورودی، خواندن اطلاعات فایلی که با notepad ایجاد شد و نمایش آن روی صفحه مانیتور. (برنامه اجرا شده کپی شد)**

```
#include<stdio.h>
int main()
{
    int ch;
    char fname[20];
    FILE *tele;
    printf("\nFile Name : ");
    gets(fname);
    puts("-----");
    tele = fopen(fname , "rb");
    if (tele==NULL) {
        puts("Cannot open Input File !!!");
        return 0;
    }
    while ((ch=fgetc(tele)) != EOF)
        putchar(ch);
}
```

## تابع نوشتن رشته در فایل :

تذکر : برای استفاده از این تابع، فایل بایستی به صورت نوشتن، الحاق کردن و یا به روز کردن، باز شده باشد.

**fputs (str, fstr)**

در این مثال، رشته **str** در فایل **fstr** نوشته می شود.

**fgets (str , 80 , fstr)**

عبارت بالا، تعداد حداکثر ۸۰ کاراکتر از فایل **fstr** خوانده و در متغیر **str** قرار می دهد.

# دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

مثال :

```
#include<stdio.h>
int main()
{
    int ch;
    char str[81];
    FILE *fstr;
    fstr = fopen("string" , "wb");
    if (fstr == NULL) {
        puts("Cannot open Input File !!!");
        return 0;
    }
    puts("Enter some string"
        "\n <^z to exit >"
        "\n-----");
    while (gets(str) != NULL)
        fputs(str, fstr);
    fputs("\n" , fstr);
}
```

```
Running Program
Enter some string
<^z to exit >
-----
Welcome
To
C
Textbook
!
^z
```

## توابع پردازش فایل های رکوردی

### ۱- نوشتن در فایل ( `fwrite` )

فرمت نوشتاری این تابع به صورت زیر است:

(نام فایل ، تعداد رکوردها ، طول رکورد ، اشاره گر ) `fwrite` [= متغیر ]

```
fwrite (&danesh , sizeof danesh , 1 , out)
```

**تذکر:** این فایل بایستی به شیوه نوشتن، الحاق کردن و به روز کردن باز شده باشد.

**اشاره گر:** همان نشانی واسطی است که رکوردها در آن قرار دارد.

### ۲- خواندن از فایل ( `fread` )

فرمت نوشتاری این تابع به صورت زیر است:

(نام فایل ، تعداد رکوردها ، طول رکورد ، اشاره گر ) `fread` [= متغیر ]

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## توابع دیگر کار با فایل :

- جستجوی فایل (`fseek`)

- گزارش فایل (`ftell`)

- نمایانگر اشکال در فایل (`ferror`)

- پاک کردن فایل (`remove`)

- تغییر نام فایل (`rename`)

# حرکت اشاره گر فایل با استفاده از `fseek`

- `int fseek ( FILE * stream, long int offset, int origin );`

- **Parameters**

stream Pointer to a [FILE](#) object that identifies the stream.

offset Number of bytes to offset from *origin*.

origin Position from where *offset* is added.

It is specified by one of the following constants defined in `<stdio.h>`:

<code>SEEK_SET</code>	Beginning of file
<code>SEEK_CUR</code>	Current position of the file pointer
<code>SEEK_END</code>	End of file

# یک مثال از `fseek`

این برنامه با `dev cpp` پیاده سازی و تست شده است. نام فایل اجرایی `fileseek` است.

```
/* fseek example */
#include <stdio.h>
int main ()
{
    FILE * pFile;
    pFile = fopen ( "example.txt" , "w" );
    fputs ( "This is an apple." , pFile );
    fseek ( pFile , 9 , SEEK_SET );
    fputs ( " sam" , pFile );
    fclose ( pFile );
    return 0;
}
```



# توابع دیگر کار با فایل

- **See also**

- ftell** Get current position in stream (function)
- fsetpos** Set position indicator of stream (function)
- rewind** Set position indicator to the beginning (function)

**#include <stdio.h>**

int **fseek** (FILE \*stream, long offset, int whence)

long **ftell** (FILE \*stream)

void **rewind** (FILE \*stream)

int **fgetpos** (FILE \* restrict stream, fpos\_t \* restrict pos)

int **fsetpos** (FILE \*stream, const fpos\_t \*pos)

## تمرینات پایان فصل

۱- برنامه ای بنویسید که اطلاعات شماره دانشجویی، نام، نام خانوادگی و نمره میان ترم، پایان ترم و تکالیف را از یک فایل خوانده (اطلاعات هر شخص در یک سطر از فایل نوشته شده است) و آن را در یک جدول با شکلی مناسب روی مانیتور نمایش دهید.

۲- یک دفترچه تلفن بسازید.

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## فصل ده

### ساختار

# Structure

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## ساختار چیست؟

قبل از تعریف ساختار، ابتدا تعریفی از یک موجودیت و ویژگی های آن ارائه می کنیم.

**موجودیت** : به هر شئی، موجود و یا مفهوم جهان اطراف خود، یک موجودیت می گوئیم.

مثال :

۱- موجودیت **صندلی**

۲- موجودیت **انسان**

۳- موجودیت **دانشجو**

۴- موجودیت **صفحه نمایش**

۵- موجودیت **درس**

## ویژگی های یک موجودیت :

هر موجودیت دارای یک سری ویژگی است که از نظر ما می تواند بعضی از آن ها در ارتباط با کاری که انجام می دهیم، حائز اهمیت باشد.

## ویژگی های سندلی :

- ۱- کاربرد / نوع سندلی (اداری، آموزشی و .....
- ۲- رنگ سندلی
- ۳- جنس سندلی (فلزی، چوبی و .....
- ۴- .....

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## موجودیت درس :

۱- نام درس

۲- کد درس

۳- کد درس پیش نیاز

۴- تعداد واحد

۵- نوع درس (تئوری / عملی)

۶- .....

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## ویژگی های موجودیت دانشجو (از نگاه اطلاعات فردی)

۱- شماره دانشجویی

۲- نام

۳- نام خانوادگی

۴- شماره شناسنامه

۵- سال تولد

۶- کد رشته تحصیلی

۷- معدل دوره دبیرستان

۸- رتبه کنکور

۹- قد

۱۰- وزن

۱۱- .....

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## ویژگی های دانشجو (از نظر اطلاعات ترمی)

۱- شماره دانشجو

۲- کد درس

۳- نمره میان ترم اول

۴- نمره میان ترم دوم

۵- نمره تکالیف

۶- نمره پایان ترم

۷- نمره نهائی

۸- وضعیت دانشجو (عادی، محروم و ....)



همانگونه که می بینیم با توجه به کاربرد یک موجودیت، ویژگی های خاصی از آن را باید در نظر بگیریم. ویژگی های مورد نظر از یک موجودیت می توانند دارای تایپ های (**Types**) داده ای گوناگون باشند. به طور مثال :

### موجودیت درس :

- ۱- نام درس (رشته)
- ۲- کد درس (رشته / صحیح چهار بایتی بدون علامت)
- ۳- کد درس پیش نیاز (رشته / صحیح چهار بایتی بدون علامت)
- ۴- تعداد واحد (صحیح)
- ۵- نوع درس (تئوری / عملی) (مقدار دودوئی)

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

**و یا موجودیت دانشجو در ترم :**

۱- شماره دانشجو (رشته / صحیح چهار بایتی بدون علامت)

۲- کد درس (رشته / صحیح چهار بایتی بدون علامت)

۳- نمره میان ترم اول ( صحیح)

۴- نمره میان ترم دوم ( صحیح)

۵- نمره تکالیف ( صحیح)

۶- نمره پایان ترم ( صحیح)

۷- نمره نهائی (اعشاری)

۸- وضعیت دانشجو (عادی، محروم و ....) ( صحیح)

دانشگاه علم و صنعت ایران – دانشکده مهندسی کامپیوتر

حال مجموعه ای از داده ها را در اختیار داریم که متعلق به یک موجودیت بوده و ضمناً دارای تایپ های داده ای متفاوت می باشند.

در زبان C، به موجودیت، ساختار (Structure) گفته می شود و به هر یک از ویژگی های یک موجودیت / ساختار، عنصر (element) گفته می شود.

تذکر: به ساختار، رکورد هم گفته می شود.

برای تعریف ساختار در زبان C، داریم:

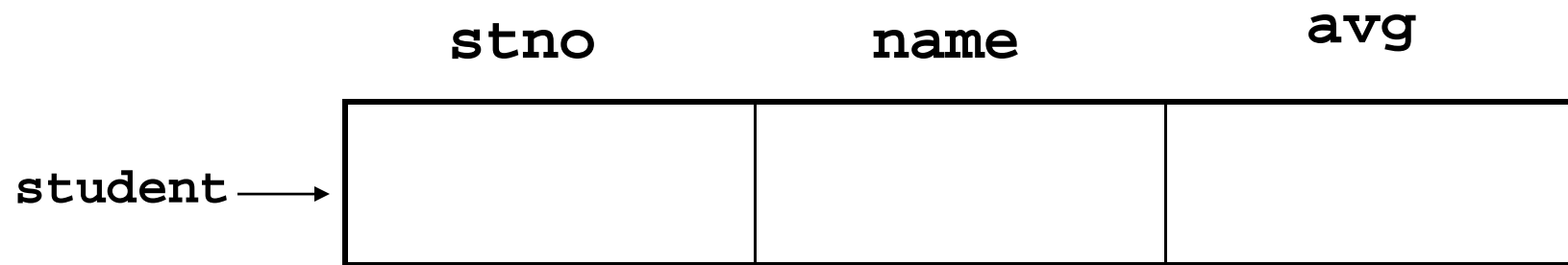
```
struct [ struct_code] {  
    data_type1    element1;  
    data_type2    element2;  
    .....  
} struct_name;
```

```
struct {  
    int stno;  
    char name[20];  
    float avg;  
} student;
```

- در این مثال ساختاری با نام student تعریف کردیم که شامل عناصر:
- شماره دانشجویی با نوع داده ای صحیح.
  - نام دانشجو، که ممتغیری ۲۰ کاراکتری می باشد.
  - میانگین، که از نوع اعشاری است.

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

برای تعریف صفحه قبل، ساختاری در حافظه با نام **student**،  
به صورت زیر و با عناصر مربوطه ایجاد نمودیم.



دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## مقدار دهی به عناصر یک ساختار :

برای مراجعه به عناصر یک متغیر ساختاری، ابتدا نام متغیر ساختار، سپس عملگر نقطه و پس از آن نام عنصر را می نویسیم.

تذکر : هر آن چه در مورد متغیرهای معمولی داشته ایم، در مورد این نوع از متغیرها نیز صادق است. تنها تفاوت در نحوه ارجاع به آن ها است.

مثال:

```
student.stno = 7318;  
student.name = "Ali Mousavi";  
student.avg = 18.25;
```

سوال : آیا عبارات بالا صحیح است؟

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

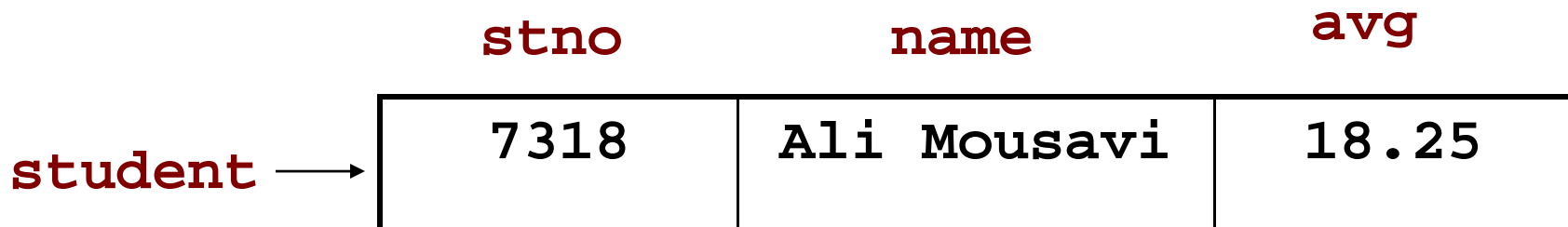
## عبارات صحیح مثال قبل :

```
student.stno = 7318;
```

```
strcpy(student.name, "Ali Mousavi");
```

```
student.avg = 18.25;
```

با توجه به مثال بالا، مقادیر در متغیر ساختاری `student` به صورت زیر خواهد بود :



## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

مثال : در برنامه زیر ساختار مثال قبل را چاپ کرده ایم.

```
/* P10.1 Structures */
#include <stdio.h>
main() {
    struct {
        int  stno;
        char name[20];
        float avg;
    } student;
    student.stno = 7318;
    strcpy(student.name, "Ali Mousavi");
    student.avg = 18.25;
    printf("%d %s %.2f\n", student.stno, student.name, student.avg);
}
```

### Running Program

7318 Ali Mousavi 18.25



## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

می توانیم به عناصر یک ساختار در همان زمان تعریف، مقدار بدهیم. مثال :

```
/* P10.2 Structures */
#include <stdio.h>
main() {
    struct {
        int    stno;
        char   name[20];
        float  avg;
    } student = {
        7318,
        "Ali Mousavi",
        "18.25"
    }
    printf("%d %s %.2f\n", student.stno, student.name, student.avg);
}
```

### Running Program

```
7318 Ali Mousavi 18.25
```

## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

عناصر یک ساختار می توانند از ورودی مقدار بگیرند. مثال:

```
/* P10.3 Structure reading from keyboard */
#include <stdio.h>
main() {
    struct {
        int    stno;
        char   name[20];
        float  avg;
    } student;
    printf("Enter Num  Name  Average : ");
    scanf("%d %s %f", &student.stno, student.name, &student.avg);
    printf("%d %s %.2f\n", student.stno, student.name, student.avg);
}
```

### Running Program

```
Enter Num  Name  Average : 7318 Ali.Mousavi 18.25
7318 Ali.Mousavi 18.25
```

تذکر : به کاراکتر نقطه بین نام و نام خانوادگی توجه نمائید.

## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

بهتر است که برای خواندن مقادیر، با توجه به الگوی پیمایش ( در صفحه قبل، در الگوی پیمایش برای جداسازی مقادیر ورودی برای عناصر، از کاراکتر فاصله استفاده کردیم)، **به جای حرف فاصله از کاراکتر با کد اسکی ۲۵۵ استفاده کنیم و یا این که مقادیر را به صورت منفرد بخوانیم:**  
مثال:

```
...  
printf("\nEnter Student Specification\n Number :");  
scanf("%d" , &student.stno);  
printf(" Name : ");  
scanf("%s", student.name);  
printf(" Average : ");  
scanf("%f" , &student.avg);  
...  
...
```

## تعریف ساختار به گونه دیگر:

```
struct student {  
    int stno;  
    char name[20];  
    float avg;  
};  
struct student stud;
```

در این مثال، **stud** متغیری است ساختاری، که کد آن **student** میباشد. پس از این تعریف می توانیم همانند مثال هائی که پیش از این دیدیم، متغیر ساختاری **stud** را به کار ببریم.

## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

استفاده از متغیرهای ساختاری در حلقه ها

```
/* P10.4 Structure & Loop */
#include <stdio.h>
#include<string.h>
void main() {
    struct telefon {
        char  name[20];
        char  num[12];
    };
    struct telefon tel;
    puts("\nEnter 'end' for quit");
    puts("-----");
    do {
        printf("Enter Name : ");
        scanf("%s", tel.name);
        if (!strcmp(tel.name,"end")) break;
        printf("Enter tel# : ");
        scanf("%s",tel.num);
        printf("\n%-20s  %-12s\n\n", tel.name, tel.num);
    } while (1);
}
```

# دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

اجرای برنامه مثال قبل :

```
Enter 'end' for quit
```

```
-----
```

```
Enter Name : Reza
```

```
Enter tel# : 238071
```

```
Reza                238071
```

```
Enter Name : Ahmad
```

```
Enter tel# : 452201
```

```
Ahmad                452201
```

```
Enter Name : end
```

## تعریف ساختار با استفاده از **typedef**

یک دیگر از روش های تعریف ساختار، تعریف آن در دستور تعریف نوع (**Type Definition**) است. دستور تعریف نوع را زمانی به کار می گیریم که بخواهیم بر یک نوع داده، نامی جدید نسبت دهیم.

مثال :

```
typedef int sahih;
```

پس از این دستور، همه جا می توانیم به جای **int** از **sahih** استفاده کنیم:

```
sahih a, b;
```

همین کار را برای تعریف ساختار می توانیم به کار ببریم.

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## تعریف ساختار با استفاده از **typedef** - ادامه

```
typedef struct {  
    char name[20];  
    char num[12];  
} telefon;  
telefon tel;
```



## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

استفاده از متغیرهای ساختاری در حلقه ها

```
/* P10.5 Structure & Loop */
#include <stdio.h>
#include<string.h>
void main() {
    typedef struct {
        char  name[20];
        char  num[12];
    } telefon;
    telefon tel;
    puts("\nEnter 'end' for quit");
    puts("-----");
    do {
        printf("Enter Name : ");
        scanf("%s", tel.name);
        if (!strcmp(tel.name,"end")) break;
        printf("Enter tel# : ");
        scanf("%s",tel.num);
        printf("\n%-20s %-12s\n\n", tel.name, tel.num);
    } while (1);
}
```

# دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

اجرای برنامه مثال قبل :

```
Enter 'end' for quit
```

```
-----
```

```
Enter Name : Reza
```

```
Enter tel# : 238071
```

```
Reza                238071
```

```
Enter Name : Ahmad
```

```
Enter tel# : 452201
```

```
Ahmad               452201
```

```
Enter Name : end
```

اگر متغیرهای ساختاری از یک نوع باشند، می توان مقادیر آن ها را همانند متغیرهای معمولی و نشانگرها به یکدیگر واگذار کرد:

```
struct {  
    char name[20];  
    char num[12];  
} telefon;  
telefon tel1, tel2;  
tel1 = tel2;
```

اگر با استفاده از `typedef` یک ساختار تعریف کرده باشیم، باز به همین ترتیب می توانیم متغیرهای هم نوع تعریف کنیم.

## ساختار و آرایه :

ساختار و آرایه از این نظر که هر دو مجموعه ای از داده ها هستند که زیر یک نام گرد آمده اند، با هم همانند هستند، ولی در میان آن ها تفاوت هائی وجود دارد که در زیر به آن ها اشاره شده است:

- ۱- هم نوع بودن عناصر آرایه، و نا هم نوع بودن عناصر ساختار.
- ۲- اندیس دار بودن عناصر آرایه، و نام داشتن عناصر ساختار.
- ۳- اشاره گر بودن نام آرایه، و معمولی بودن نام ساختار.
- ۴- نشدنی بودن واگذاری دو آرایه به هم، و شدنی بودن واگذاری دو ساختار به هم.

با درهم آمیختن ساختار و آرایه، ترکیب های زیر بدست می آید:

- ۱- آرایه ساختاری.
- ۲- ساختار آرایه ای.

## آرایه ساختاری :

آرایه ای که هر عنصر آن یک ساختار باشد، آرایه ساختاری نامیده میشود.  
مثال :

```
struct {  
    char name[20];  
    char num[12];  
} tel[100];
```

در این مثال، `tel` آرایه ای است که دارای یکصد عنصر بوده و هر یک از عناصر یک ساختار می باشد. برای رسیدن به هر یک از عناصر، خواهیم داشت :

`tel[i].name`

`tel[i].num`

## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

استفاده از متغیرهای ساختاری در حلقه ها

```
/* P10.6 Structure & Array */
#include <stdio.h>
#include<string.h>
void main() {
    struct {
        char  name[20];
        char  num[12];
    } tel[100];
    int i = 0, n;
    puts("\nEnter 'end' for quit");
    puts("-----");
    do {
        printf("Enter Name : ");
        scanf("%s", tel[i].name);
        if (!strcmp(tel[i].name,"end")) break;
        printf("Enter tel# : ");
        scanf("%s",tel[i].num); puts(" ");
    } while (++i);
    puts("\n\n Name                Tel # ");
    puts("-----");
    for (n=0 ; n < i; ++n)
        printf("\n%-20s  %-12s\n\n", tel.name, tel.num); }
}
```

# دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

اجرای برنامه مثال قبل :

```
Enter 'end' for quit
```

```
-----
```

```
Enter Name : Reza
```

```
Enter tel# : 238071
```

```
Enter Name : Ahmad
```

```
Enter tel# : 452201
```

```
Enter Name : Mahdi
```

```
Enter tel# : 761132
```

```
Enter Name : end
```

```
      Name                Tel #
-----
Reza                238071
Ahmad               452201
Mahdi               761132
```

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## ساختار آرایه ای :

ساختار آرایه ای، متغیر ساختاری است که بعضی از عناصر آن یک آرایه است.

## مثال :

```
struct {  
    char name [15];  
    char ketab [10] [15];  
} author;
```

در این ساختار، بری نام نویسنده تا ۱۵ کاراکتر، و برای نام کتاب تا ۳۰ کاراکتر پیش بینی شده است. در برنامه صفحه بعد، این ساختار را برای خواندن و چاپ نام و نام کتاب های نویسندگان مختلف به کار برده ایم.



## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

استفاده از متغیرهای ساختاری در حلقه ها

```
/* P10.7 Structure & Array */
#include <stdio.h>
#include<string.h>
void main() {
    struct {
        char  name[15];
        char  Ketab[10][30];
    } author;
    int i;
    puts("\nEnter 'end' for quit");
    puts("-----");
    do {
        printf("Enter Name : ");
        scanf("%s", author.name);
        if (!strcmp(author.name,"end")) break;
        i = 0;
        do {
            printf("        book %d : ", i + 1);
            scanf("%s" , author.ketab[i]);
            if (!strcmp(author.ketab[i],"end")) break;
            ++i
        } while(1); / C
```

## ادامه برنامه اسلاید قبل :

```
i = 0;
puts( "\n\nName                               Book(s)");
puts( "-----");
while (strcmp(author.ketab[i], "end"))
{
    printf( "\n%-15s  %-30s\n", author.name,    author.ketab[++i]);
    strcmp(author.name, " ");
}
} while(1);
}
```

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

اجرای برنامه صفحه قبل :

```
Enter 'end' for quit
```

```
-----
```

```
Enter Name : Molavi
```

```
    book 1 : Masnavi
```

```
    book 2 : Divan.e.Shams
```

```
    book 3 : end
```

```
Name                Book(s)
```

```
-----
```

```
Molavi              Masnavi
```

```
                    Divane.e.Shams
```

```
Enter Name : end
```

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

**ساختار زیر را شرح دهید:**

```
struct {  
    char stno[8];  
    char name[30];  
    int grade[3];  
} Student;
```

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

**ساختار زیر را شرح دهید:**

```
struct {  
    char stno[8];  
    char name[30];  
    int grade[3];  
} Student[100];
```

## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

نحوه استفاده از cout : مثال

```
2: /* This Program Describe the use of COUT */
3: #include <iostream.h>
4: int main()
5: {
6: cout << "Hello there.\n";
7: cout << "Here is 5: " << 5 << "\n";
8: cout <<"The manipulator endl writes a new line to the screen."<<endl;
9: cout << "Here is a very big number:\t" << 70000 << endl;
10: cout << "Here is the sum of 8 and 5:\t" << 8+5 << endl;
11: cout << "Here's a fraction:\t\t" << (float) 5/8 << endl;
12: cout <<"And a very very big number:\t"<<(double)7000 * 7000 <<endl;
13: cout << "Don't forget to replace Baski with your name...\n";
14: cout << "Baski is a C++ programmer!\n";
15: return 0;
16: }
```

# دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## نحوه عملکرد cout - توضیح مثال صفحه قبل

On line 3, the statement `#include <iostream.h>` causes the `iostream.h` file to be added to your source code. This is required if you use `cout` and its related functions.

On line 6 is the simplest use of `cout`, printing a string or series of characters. The symbol `\n` is a special formatting character. It tells `cout` to print a newline character to the screen.

Three values are passed to `cout` on line 7, and each value is separated by the insertion operator. The first value is the string "Here is 5: ". Note the space after the colon. The space is part of the string. Next, the value 5 is passed to the insertion operator and the newline character (always in double quotes or single quotes). This causes the line Here is 5: 5 to be printed to the screen. Because there is no newline character after the first string, the next value is printed immediately afterwards. This is called concatenating the two values.

On line 8, an informative message is printed, and then the manipulator `endl` is used. The purpose of `endl` is to write a new line to the screen. (Other uses for `endl` are discussed on Day 16.)

On line 9, a new formatting character, `\t`, is introduced. This inserts a tab character and is used on lines 8-12 to line up the output. Line 9 shows that not only integers, but long integers as well can be printed. Line 10 demonstrates that `cout` will do simple addition. The value of `8+5` is passed to `cout`, but 13 is printed.

On line 11, the value `5/8` is inserted into `cout`. The term `(float)` tells `cout` that you want this value evaluated as a decimal equivalent, and so a fraction is printed. On line 12 the value `7000 * 7000` is given to `cout`, and the term `(double)` is used to tell `cout` that you want this to be printed using scientific notation. All of this will be explained on Day 3, "Variables and Constants," when data types are discussed.

On line 14, you substituted your name, and the output confirmed that you are indeed a C++ programmer. It must be true, because the computer said so!

## دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

مثالی از کار با تابع **strcmp** و تابع **strncmp** - این برنامه اجرا و به خروجی توجه شود

```
#include <iostream.h>
#include <string.h>
void main()
{
    char *s1 = "Happy New Year";
    char *s2 = "Happy New Year";
    char *s3 = "Happy Holidays";
    cout << "s1 = " << s1 << "\ns2 = " << s2 << "\ns3 = " <<
s3 << "\n\nstrcmp(s1, s2) = " << " " << strcmp( s1, s2 )
<< "\nstrcmp(s1, s3) = " << " " << strcmp( s1, s3 ) <<
"\nstrcmp(s3, s1) = " << " " << strcmp( s3, s1 );

    cout << "\n\nstrncmp(s1, s3, 6) = " << " " <<
strncmp(s1,s3,6) << "\nstrncmp(s1,s3,7) = " << " " <<
strncmp( s1, s3, 7 ) << "\nstrncmp(s3, s1, 7) = " << " "
<< strncmp( s3, s1, 7 ) << endl;
}
```



دانشگاه علم و صنعت ایران – دانشکده مهندسی کامپیوتر

## ساختار و اشاره گر

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## ساختار بازگشتی

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## ساختار و تابع - تابع ساختاری

دانشگاه علم و صنعت ایران - دانشکده مهندسی کامپیوتر

## پارامتر ساختاری