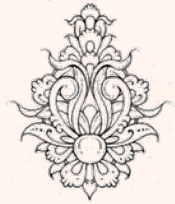
انواع داده
انواع خطا

مبانی برنامه‌نویسی
(۱۳۹۹-۱۳-۱۱)
جلسه‌ی هشتم

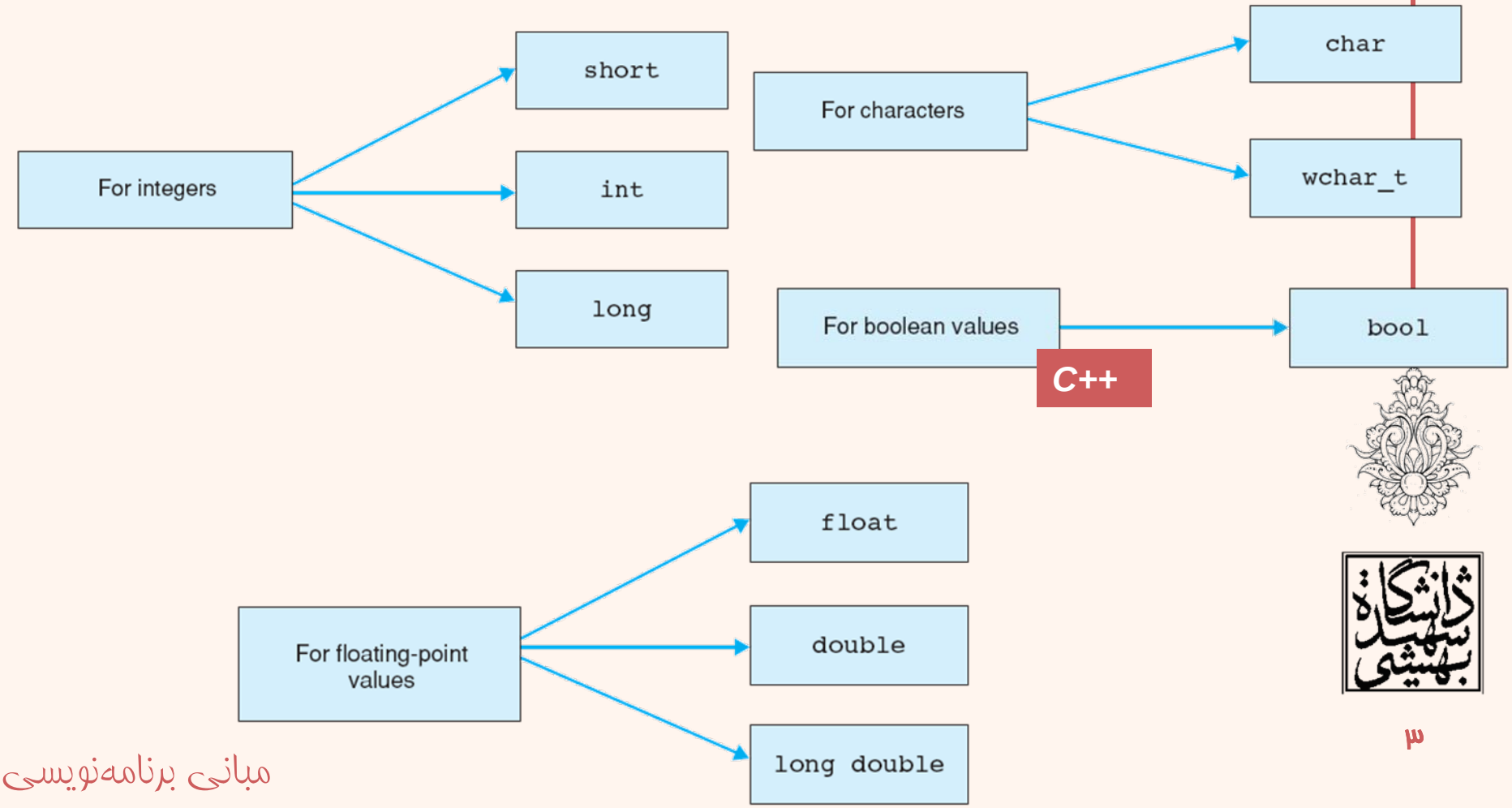
دانشگاه شهید بهشتی
پاییز ۱۳۹۹
دانشکده‌ی مهندسی برق و کامپیوتر
احمد محمودی ازناوه

فهرست مطالب

- انواع داده
  - ورودی-خروجی
- انواع خطا
- لیترال‌ها
- عملیات ریاضی



- انواع داده‌ها بسته به مورد استفاده در برنامه می‌توانند شامل موارد زیر باشند:



خلاصه‌ای از انواع اصلی داده

Type	Size	Range of Values (decimal)
`char`	1 byte	-128 to +127 or 0 to 255
`unsigned char`	1 byte	0 to 255
`signed char`	1 byte	-128 to +127
`int`	2 byte resp.	-32768 to +32767 resp.
	4 byte	-2147483648 to +2147483647
`unsigned int`	2 byte resp.	0 to 65535 resp.
	4 byte	0 to 4294967295
`short`	2 byte	-32768 to +32767
`unsigned short`	2 byte	0 to 65535
`long`	4 byte	-2147483648 to +2147483647
`unsigned long`	4 byte	0 to 4294967295

Type	Size	Range of Values	Lowest Positive Value	Accuracy (decimal)
`float`	4 bytes	–3.4E+38	1.2E—38	6 digits
`double`	8 bytes	–1.7E+308	2.3E—308	15 digits
`long double`	10 bytes	–1.1E+4932	3.4E—4932	19 digits

اعلام و مقداردهی متغیرها

Declaration of variables

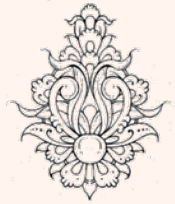
- برای استفاده از متغیرها می‌باید ابتدا آن‌ها را اعلام نمایید:

```
int a;
float mynumber;
int  b, c;
```

- می‌توان در هنگام تعریف متغیر فرآیند مقداردهی اولیه را نیز صورت داد.

Initialization

```
int a=0;
float mynumber=3.67;
int  b=2, c=7;

float x(1.875);
```
C++

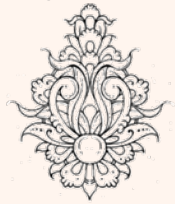


انتساب

- فرآیند انتساب به یک متغیر را گویند.
- مقدار سمت راست ارزیابی و به سمت چپ نسبت داده می‌شود.

```
x=a+b;
x=12;
x=y=z=28 ;
```

- تفاوت مقداردهی اولیه و انتساب چیست؟



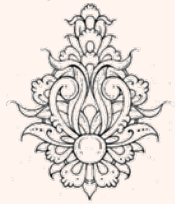
توابع ورودی و خروجی

- برای این که بتوانیم هنگام اجرای برنامه مقادیری را چاپ کنیم، از تابع **printf** استفاده می‌کنیم.

```
printf ( “text  &  Argument  type”, Argument name ) ;
```

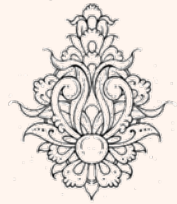
- برای این که بتوانیم هنگام اجرای برنامه مقادیری را وارد کنیم از تابع **scanf** استفاده می‌کنیم.

```
scanf ( “Argument  type”, &Argument name ) ;
```



printf

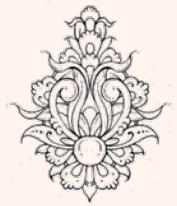
- به منظور مشاهده‌ی داده‌های به دست آمده، از این تابع استفاده می‌شود.
- این تابع می‌تواند مقادیر متغیرها را در خروجی چاپ کند.
- تعداد ورودی‌های این تابع متغیر است.
- ورودی(آرگومان) اول رشته‌ی کنترلی است که چگونگی چاپ متغیرها را مشخص می‌کند.
- کاراکترهای رشته‌ی کنترلی عینا چاپ می‌شود، مگر در مواردی که با «%» همراه شده‌اند.



printf (ادامه...)

- در واقع متناظر با هر متغیر یک علامت «٪» وجود دارد که توسط کاراکتر بعد از % شیوه‌ی نمایش را مشخص می‌کند:

کاراکتز کنترلی	مفهوم
%d	نمایش یک عدد صحیح در مبنای ده
%u	نمایش یک عدد صحیح بدون علامت در مبنای ده
%f	نمایش یک عدد اعشاری
%c	نمایش به صورت کاراکتری
%%	نمایش نماد (سمبل) %



مثال

```
a=67
b=-1
c=E
x=1.200000
y=3.141500
```

```
a=C
b=4294967295
c=69
x=1073741824
y=3.141500
```

```c
#include <stdio.h>
int main(){
 int a=67;
 int b=-1;
 char c=69;
 float x=1.2;
 double y=3.1415;
 printf("a=%d\n",a);
 printf("b=%d\n",b);
 printf("c=%c\n",c);
 printf("x=%f\n",x);
 printf("y=%f\n",y);
 return 0;
}
```

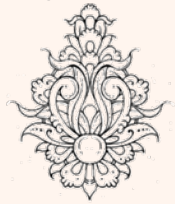
```c
#include <stdio.h>
int main(){
 int a=67;
 int b=-1;
 char c=69;
 float x=1.2;
 double y=3.1415;
 printf("a=%c\n",a);
 printf("b=%u\n",b);
 printf("c=%d\n",c);
 printf("x=%d\n",x);
 printf("y=%f\n",y);
 return 0;
}
```

```
example4.c: In function ‘main’:
example4.c:11: warning: format ‘%d’ expects type ‘int’, but argument 2 has type ‘double’
```

مثال

```c
#include<stdio.h>
int main() {
    int a=0;
    printf("please enter a number:\n");
    scanf("%d",&a);
    printf("a=%d\n",a);
    return 0;
}//main
```

```
please enter a number:
2
a=2
```

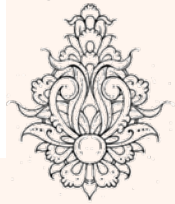


مثال

limit.c

```c
#include<stdio.h>
#include<limits.h> // Definition of INT_MIN, ...
int main()
{
    printf("Range of types int & unsigned int\n\n");
    printf("Type Minimun & Maximum\n");
    printf("-----------------------------\n");
    printf("int %d  %d \n",INT_MIN,INT_MAX);
    printf("unsignedint %d  %u \n",0,UINT_MAX);
    return 0;

}
```

```
Range of types int & unsigned int

Type Minimun & Maximum
-----------------------------
int -2147483648  2147483647
unsignedint 0  4294967295
```



انواع خطا

- خطای زمان کامپایل
  - خطای نحوی

Syntax error

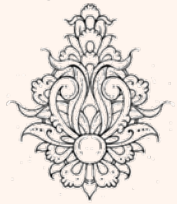
    - هنگامی که کامپایلر نتواند دستوری را تشخیص دهد.
    - به منزله‌ی تخلف از زبان است.
  - خطای نوع داده

Typechecking errors

    - به عنوان مثال تابعی با آرگومان‌های غیر قابل انتظار فراخوانی شود.
- خطای زمان اجرا

Runtime errors

  - فرآیندی در زمان اجرا با مشکل مواجه شود.
    - تقسیم بر صفر
    - کمبود حافظه و....



مثال

```c
#include <stdio.h>
int main(){
    int m;
    int n=44;
    printf("m=%d, n=%d\n",m,n);
    return(0);
}//main
```

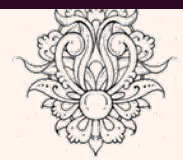
```
gcc -Wall main.c
```

```
main.c: In function ‘main’:
main.c:7: warning: ‘m’ is used uninitialized in this function
```

```
gcc main.c
```

```
./a.out
```

```
m = 8785908 and n = 44
```



C++ از قراردادن مقادیری بزرگ در متغیری کوچک جلوگیری نمی‌کند.

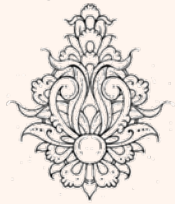
a | 20000

c: | ???

```
oops!: 20000 != 32
```

```
#include <stdio.h>
int main(){
    int a = 20000;
    char c = a;
    int b = c;
    if (a != b)      //  != means “not equal”
        printf("oops!:  %d!=%d",a,b);
    else
        printf("Wow! We have large characters\n");
}//main
```

$(20000)_d=(4E20)_h$



ثابت‌ها

یک **ثابت**، یک متغیر است اما تنها یک بار مقداردهی می‌شود و پس از آن تغییر دادن مقدار آن در ادامه‌ی برنامه ممکن نیست.

تعریف ثابت‌ها مانند تعریف متغیرهاست با این تفاوت که کلمه کلیدی **const** به ابتدای تعریف اضافه می‌شود.

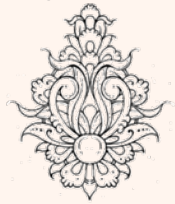
```c
int main()
{
    const char NLINE= '\n';
    const int MAXINT = 2147483647;
    const int N = MAXINT/2;
    const float KM_PER_MI = 1.60934;
    const double PI = 3.141592653589793238 46;
    PI = 5.141592653589793238 46;
    printf("%c\t%d\t%d\t%f\t%f\n",NLINE,MAXINT,N,KM_PER_MI,PI);
    return 0;

}
```

```
main.c: In function ‘main’:
main.c:10: error: assignment of read-only variable ‘PI’
```

بهتر است ثابت‌ها را با حروف بزرگ نام‌گذاری کنید تا در برنامه مشخص باشد.

مقادیری که به انواع مختلف نسبت داده می‌شود

- Boolean literals
  - **true, false**
- Character literals
  - **'a', 'x', '4', '\n', '$'**
- Integer literals
  - **0, 1, 123, -6, 0x34, 0xa3**
- Floating point literals
  - **1.2, 13.345, .3, -0.54, 1.2e3**
- String literals **"asdf"**,
  “This is a test!”



کاراکتر و لیترال رشته‌ای

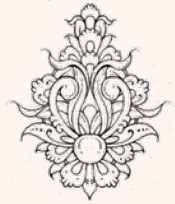
یک لیترال «کاراکتری» می‌تواند یک حرف، رقم یا علامت قابل چاپ باشد که میان دونشانهٔ '' محصور شده باشد. پس 'W' و '!' و '1' هر کدام یک کاراکتر است که در واقع یک کد اسکی را نشان می‌دهد.

```
int main()
{// prints "Hello,Wo rld!":
    printf("Hello,W%crld%c\n",'o','!');
    return 0;

}
```

```
Hello,World!
```

```
#include <stdio.h>
int main()
{ // prints "The Millennium ends Dec 31 2000.":
 printf("The Millennium ends Dec %d%d%c%d\n",3 ,1 ,' ',2000);
 return 0;
}
```

```
The Millennium ends Dec 31 2000
```

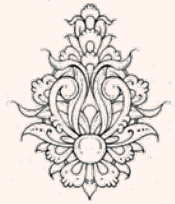


Boolean

```c
#include <stdio.h>
int main(){
// print the value of a boolean variable
 bool flag=true;
 printf("flag=%d\n",flag );
 flag=false;
 printf("flag=%d\n",flag );
}
```

```
ahmad@ubuntu:~/Courses/ITP$ gcc  boolean.c
boolean.c: In function ‘main’:
boolean.c:4: error: ‘bool’ undeclared (first use in this function)
boolean.c:4: error: (Each undeclared identifier is reported only once
boolean.c:4: error: for each function it appears in.)
boolean.c:4: error: expected ‘;’ before ‘flag’
```

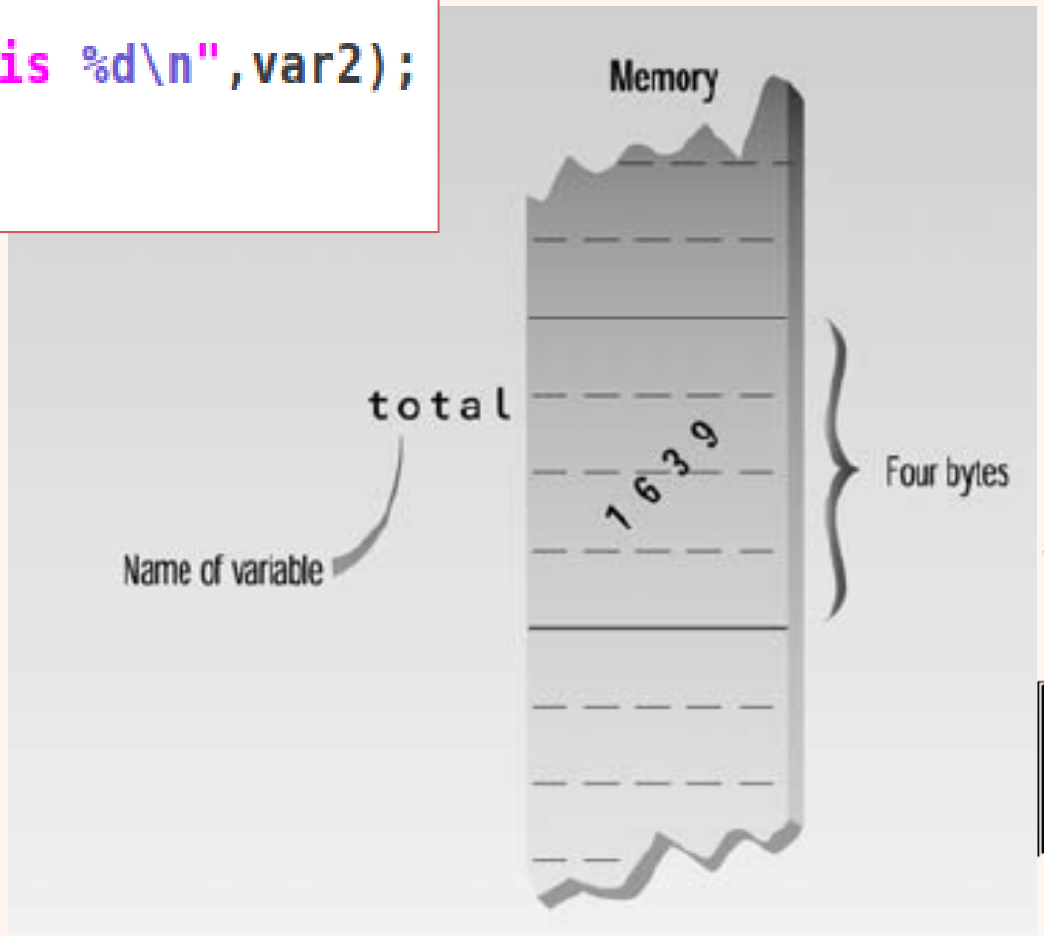
```
ahmad@ubuntu:~/Courses/ITP$ g++  boolean.c
ahmad@ubuntu:~/Courses/ITP$ ./a.out
flag=1
flag=0
```



نوع int

```c
int main()
{
    int var1;
    int var2;
    var1=20;
    var2=var1+10;
    printf("var1+10 is %d\n",var2);
    return 0;
}
```

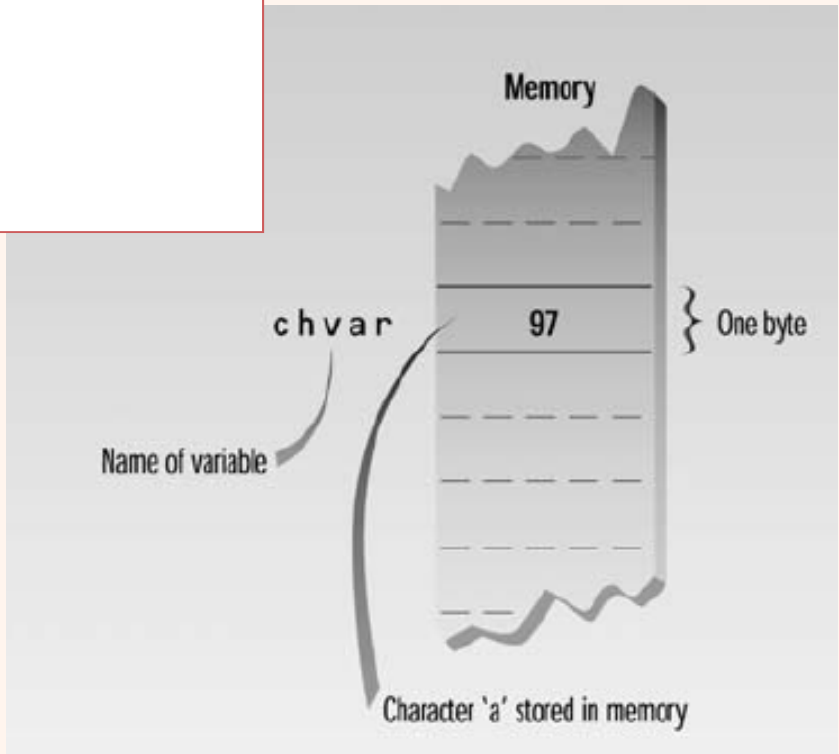
```
var1+10 is 30
```



نوع کاراکتر

```c
int main()
{
    char charvar1 = 'A';
    char charvar2 = '\t';
    printf("%c%c",charvar1,charvar2);
    charvar1 = 'B';
    printf("%c\n",charvar1);
    return 0;
}
```

```
A       B
```



کاراکترها و کاراکترهای کنترلی

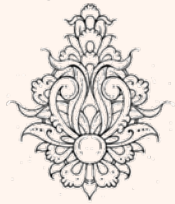
Constant	Character	Constant Value (ASCII code decimal)
'A'	Capital A	65
'a'	Lowercase a	97 || ' '	Blank	32
'.'	Dot	46
'0'	Digit 0	48
'\0'	Terminating null character	0

\n	Newline
\t	Tab
\b	Backspace
\a	Beep sound
\”	Double quote
\’	Single quote
\0	Null character
\?	Question mark
\\	Back slash

مثال

```c
#include<stdio.h>
int main() {
    char c='A';
    printf("c = %c,\t int(c)=%d\n",c, c);
    c='t';
    printf("c = %c,\t int(c)=%d\n",c, c);
    c='\t';
    printf("c = %c, int(c)=%d\n",c, c);
    c='!';
    printf("c = %c, int(c)=%d\n",c, c);
    return 0;
}//end main
```

```
c = A,    int(c)=65
c = t,    int(c)=116
c =      , int(c)=9
c = !, int(c)=33
```

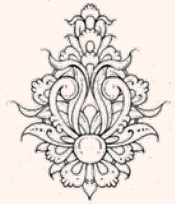


مثال

```c
#include<stdio.h>
int main(){
    printf("\nThis is \t a string\n\t\t"
        "with many \"escape\" sequences!\n");
    return(0);
}//end main
```

```
This is          a string
                with many "escape" sequences!
ahmad@ubuntu:~/MyData/courses/ITP$
```

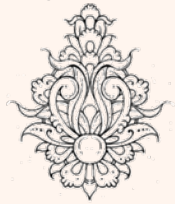
چنان‌چه پیش از این اشاره شد، در مواردی که می‌خواهیم " چاپ شود از «\» به عنوان پیشوند استفاده می‌شود.



مثال

```c
int main()
{
    double area, circuit, radius = 1.5;
    const double pi = 3.141593;
    area = pi * radius * radius;
    circuit = 2 * pi * radius;
    printf("To Evaluate a Circle\n\n"
           "Radius: %f\n"
           "Circumference: %f\n"
           "Area: %f\n",radius,circuit,area);
    return 0;
}
```

```
To Evaluate a Circle

Radius: 1.500000
Circumference: 9.424779
Area: 7.068584
```



اعمال ریاضی

فرآیند	عملگر ریاضی	عبارت جبری	نمونه
جمع	+	f+7	F+7
تفریق	-	p-c	p-c
ضرب	×	bm	b*m
تقسیم	/	$x \div y$ $\frac{x}{y}$ x/y	x/y
پیمانه	٪	r mod s	r%s

```
int main()
{// test operators +,-,*,/,and %:
    int m=54;
    int n=20;
    printf("m = %d end n = %d \n",m,n);
    printf("m+n = %d \n",m+n);
    printf("m-n = %d \n",m-n);
    printf("m*n = %d \n",m*n);
    printf("m/n = %d \n",m/n);
    printf("m%cn = %d \n",'%',m%n);
    return 0;
}
```

```
m = 54 end n = 20
m+n = 74
m-n = 34
m*n = 1080
m/n = 2
m%n = 14
```

مثال

```c
#include<stdio.h>
int main(){
    int num=5;
    float num=5.1;
    printf("num=%d\tnum=%f",num,num);
    return(0);
}//end main
```

```
ahmad@ubuntu:~/MyData/courses/ITP$ gcc example14.c
example14.c: In function ‘main’:
example14.c:4: error: conflicting types for ‘num’
example14.c:3: note: previous definition of ‘num’ was here
example14.c:5: warning: format ‘%d’ expects type ‘int’, but argument 2 has type
‘double’
```

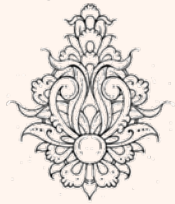
نمی‌توانیم دو متغیر هم‌نام تعریف کنیم!

اولویت‌ها

- تقدم عملیات ریاضی به صورت زیر است:

عملگر	ترتیب
پرانتزها	()
عملگر های ضرب و تقسیم و باقی مانده	* / %
عملگرهای جمع و تفریق	+ -

- در صورت وجود بیش از یک نمونه در یک عبارت به ترتیب از چپ به راست ارزیابی می‌شوند.



اپراتورهای افزایش و کاهش

- عملگر ++:
  - پیشوندی

```
++P
```

    - یک واحد افزایش یافته سپس فرآیند صورت می‌گیرد.
  - پسوندی

```
P++
```

    - فرآیند صورت می‌گیرد، سپس یک واحد افزایش می‌یابد.
- عملگر --:
  - پیشوندی

```
--P
```

    - یک واحد کاهش یافته سپس فرآیند صورت می‌گیرد.
  - پسوندی

```
P--
```

    - فرآیند صورت می‌گیرد، سپس یک واحد کاهش می‌یابد.

این عملگرها بالاترین اولویت را دارند

مثال

```c
#include <stdio.h>
int main(){ // shows the difference between m++ and ++m:
    int m,n;
    m = 88;
    n = ++m; // the pre-increment operator is applied to m
    printf("m=%d, n=%d\n",m, n);
    m = 88;
    n = m++; // the post-increment operator is applied to m
    printf("m=%d, n=%d\n",m, n);
    return 0;
}// end of main
```

```
m=89, n=89
m=89, n=88
```