مبانی برنامه‌نویسی

(۱۳۹۹-۱۳-۱۱)

جلسه‌ی دهم

لیترال‌ها

عبارت‌های ریاضی

دانشگاه شهید بهشتی

پاییز ۱۳۹۲

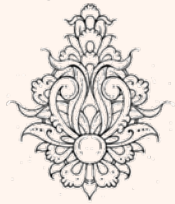
دانشکده‌ی مهندسی برق و کامپیوتر

احمد محمودی ازناوه

http://faculties.sbu.ac.ir/~a_mahmoudi/

فهرست مطالب

- مروری بر جلسه‌ی پیش
- انواع خطا
- داده‌ها
  - لیترال‌ها
- ثابت‌ها
- عبارت‌های ریاضی



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
```

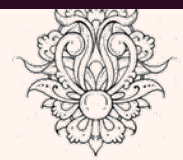
```
gcc -Wall main.c
```

```
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
```



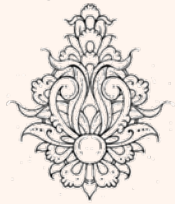
C++ از قراردادن مقادیری بزرگ در متغیری کوچک جلوگیری نمی‌کند....

a | 20000

c: | ???

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

```c
#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
```



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;

}
```

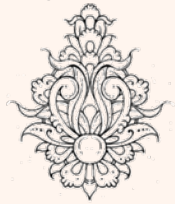
```
main.c: In function ‘main’:
main.c:10: error: assignment of read-only variable ‘PI’
```

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

لیترال

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.

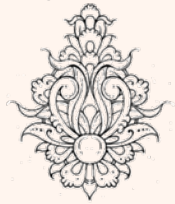
```c
int main()
{// prints "Hello,Wo rld!":
    printf("Hello,W%crld%c\n",'o','!');
    return 0;

}
```

```
Hello,World!
```

```c
#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
```

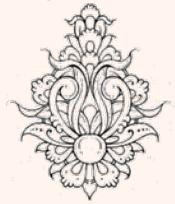


Boolean

```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’
```

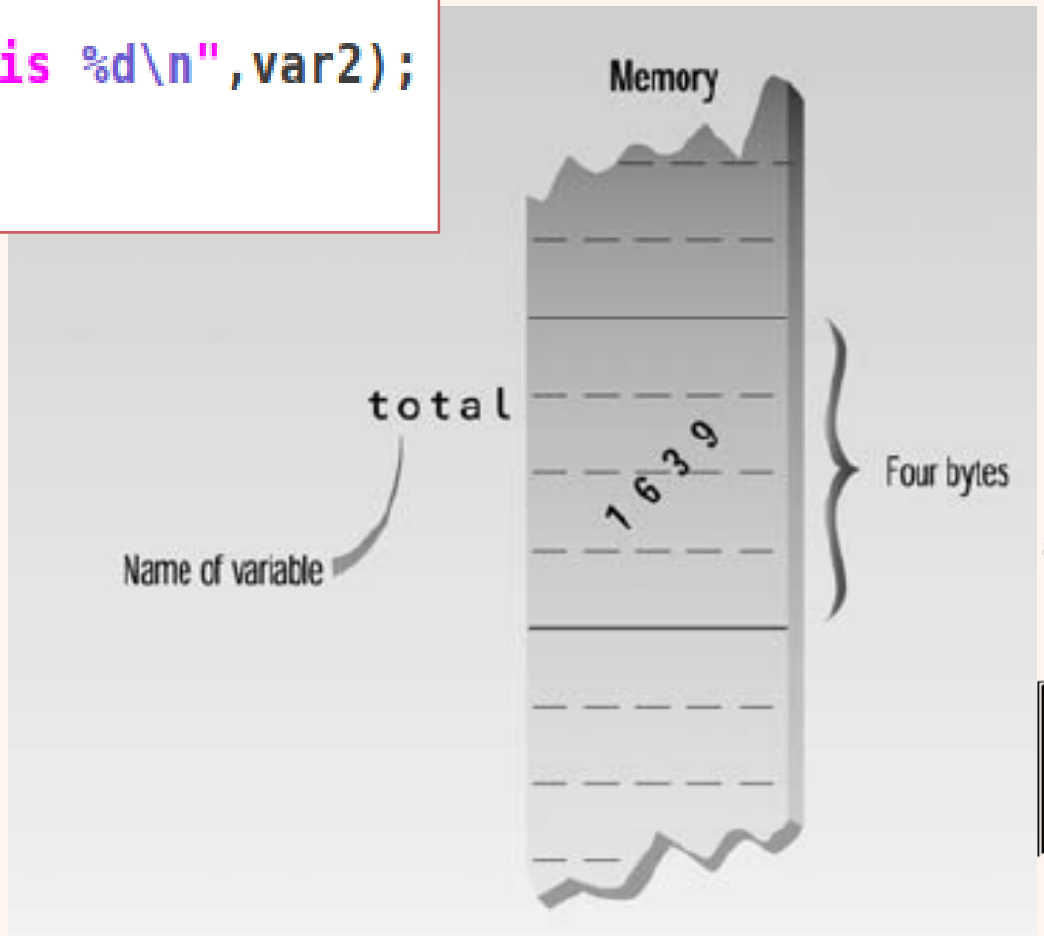
```
ahmad@ubuntu:~/Courses/ITP$ g++  boolean.c
ahmad@ubuntu:~/Courses/ITP$ ./a.out
flag=1
flag=0
```



نوع int

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

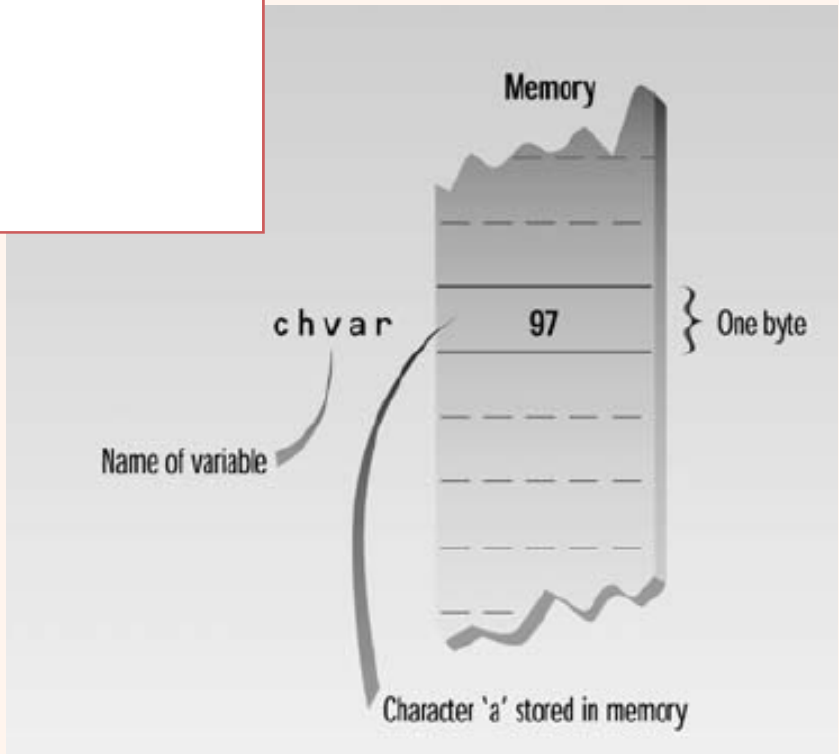
```
var1+10 is 30
```



نوع کاراکتر

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

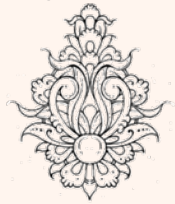
```
A       B
```



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

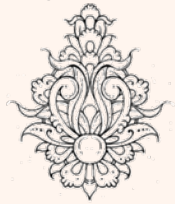
\n	Newline
\t	Tab
\b	Backspace
\a	Beep sound
\”	Double quote
\’	Single quote
\0	Null character
\?	Question mark
\\	Back slash



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
```

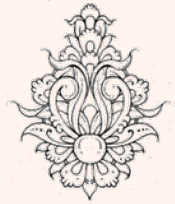


مثال

```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.

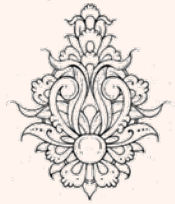


مثال

```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
```

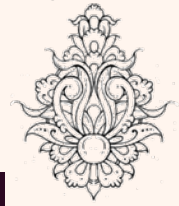


اعمال ریاضی

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
```



شیوه‌ی جدید و قدیمی

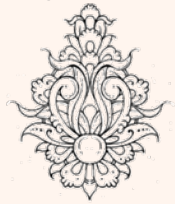
```
/*
   An old-style C++ program.
*/

#include <iostream.h>

int main()
{
  return 0;
}
```

```
/*
   A modern-style C++ program that uses
   the new-style headers and a namespace.
*/
#include <iostream>
using namespace std;

int main()
{
  return 0;
}
```



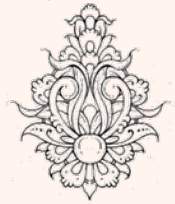
شیوه‌ی جدید و قدیمی (ادامه...)

- سرآیندهایی که با پسوند h. مشخص می‌گردند، حتماً نشان‌گر یک فایل خواهند بود. (شیوه‌ی قدیمی استفاده از سرآیندها)

```
Stdio.h   (file name)
```

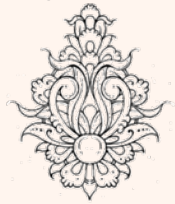
- در شیوه‌ی نوین نام سرآیند لزوماً نشان‌دهنده‌ی یک فایل نبوده، می‌تواند نگاشتی از یک فایل به‌شمار رود.

```
<iostream>
```



شیوه‌ی جدید و قدیمی (ادامه...)

- هنگامی‌که یک سرآیند include می‌گردد، محتویات آن در محدوده‌ای واقع شده است که به آن namespace گفته می‌شود.
- مولفه‌هایی که در یک namespace وجود دارند، مستقل از مولفه‌هایی خواهند بود که در namespace دیگر است.
- C++ Standard Library در namespace با نام std تعریف شده است.
- هر آن‌چیز که در namespace با نام std است می‌باید با پیشوند std آورده شود.



namespace

foo.h

```
// This DoSomething() adds it's parameters
int DoSomething(int nX, int nY)
{
    return nX + nY;
}
```

goo.h

```
// This DoSomething() subtracts it's parameters
int DoSomething(int nX, int nY)
{
    return nX - nY;
}
```

namespace.cpp

```
#include "foo.h"
#include "goo.h"
#include <iostream>

int main()
{
    using namespace std;
    cout << DoSomething(4, 3); // which DoSomething will we get?
    return 0;
}
```

```
ahmad@ubuntu:~/Courses/ITP$ g++ -Wall namespace.cpp foo.h goo.h -o test
In file included from namespace.cpp:2:
goo.h: In function ‘int DoSomething(int, int)’:
goo.h:2: error: redefinition of ‘int DoSomething(int, int)’
foo.h:2: error: ‘int DoSomething(int, int)’ previously defined here
```

Namespace (ادامه...)

foo.h

```
namespace foo{
        // This DoSomething() adds it's paramet
        int DoSomething(int nX, int nY)
        {
            return nX + nY;
        }
}
```

goo.h

```
namespace goo{
        // This DoSomething() subtracts it's parameters
        int DoSomething(int nX, int nY)
        {
            return nX - nY;
        }
}
```

namespace.cpp

```
#include
#include
#include <iostream>

int main()
{
    using namespace std;
    using namespace foo;
    cout <<  DoSomething(4, 3);
    cout << '\n';

    cout << goo::DoSomething(4, 3);
    cout << '\n';
    return 0;
}
```

```
71ahmad@ubuntu:~/Courses/ITPg++ -Wall namespace.cpp foo.h goo.h -o test
ahmad@ubuntu:~/Courses/ITP$ ./test
7
1
```

- برای این که بتوانیم هنگام اجرای برنامه مقادیری را وارد کنیم از عملگر ورودی >> استفاده می‌کنیم.

```
cin >> variable;
```

- برای این که بتوانیم هنگام اجرای برنامه مقادیری را چاپ کنیم.

```
cout << variable;
```