برنامه‌نویسی

مبانی برنامه‌نویسی

(۱۳۹۹-۱۳-۱۱)

جلسه‌ی دوازدهم

دانشگاه شهید بهشتی

پاییز ۱۳۹۹

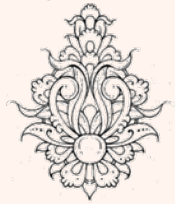
دانشکده‌ی مهندسی برق و کامپیوتر

احمد محمودی ازناوه

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

فهرست مطالب

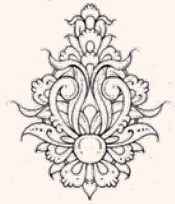
- انواع عملگرها و اولویت‌ها
  - عملگرهای نسبی
  - عملگرهای منطقی
  - عملگرهای بیتی



cin - cout

- در C++ توابع cin و cout برای این کار استفاده می‌شود هر چند می‌توان از توابع کتابخانه‌های زبان C همانند printf و scanf نیز استفاده نمود.
- چرا استفاده از این دو تابع با استفاده از cin و cout جایگزین شده است؟

– استفاده از کتابخانه‌ی iostream به دلیل type safe بودن آن راحت‌تر است.



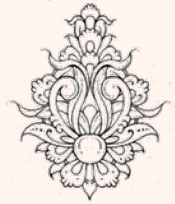
C, C++

- 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;
```



شیوه‌ی جدید و قدیمی

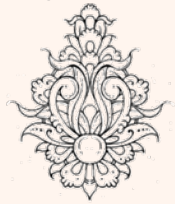
```
/*
   An old-style C++ program.
*/

#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;
}
```



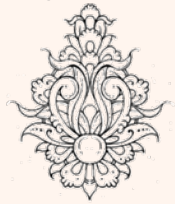
شیوه‌ی جدید و قدیمی (ادامه...)

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

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

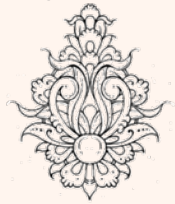
- در شیوه‌ی نوین نام سرآیند لزوماً نشان‌دهنده‌ی یک فایل نبوده، می‌تواند نگاشتی از یک یا چند فایل به‌شمار رود.

```
<iostream>
```



شیوه‌ی جدید و قدیمی (ادامه...)

- هنگامی‌که یک سرآیند، include می‌گردد، محتویات آن در محدوده‌ای واقع شده است که به آن namespace گفته می‌شود.
- مؤلفه‌هایی که در یک namespace وجود دارند، مستقل از مؤلفه‌هایی 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.

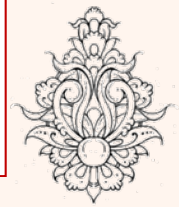


```cpp
#include <iostream>
int main(){
    int m;
    std::cin>>m;
    std::cout<<"m is:"<<m<<"\n";
}
```

```
9
m is:9
```

```cpp
#include <iostream>
using namespace std;
int main(){
    int m;
    cin>>m;
    cout<<"m is:"<<m<<"\n";
}
```

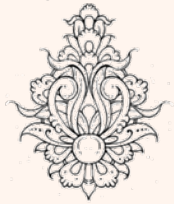
```
8
m is:8
```



مثال

```
int main()
{ // shows the difference between m++ and ++m:
int m,n;
m = 88;
n = ++m; // the pre-increment operator is applied to m
cout << "m = " << m << ", n = " << n << endl;
m = 88;
n = m++; // the post-increment operator is applied to m
cout << "m = " << m << ", n = " << n << endl;
}
```

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



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
```

انواع عملگرها

- عملگرهای دوتایی

Binary Arithmetic Operators

Operator	Significance
+	Addition
-	Subraction
*	Multiplication
/	Division
%	Remainder

- عملگرهای یکانی

Unary Arithmetic Operators

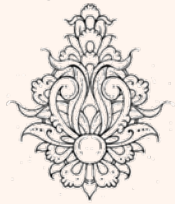
Operator	Significance
+ -	Unary sign operators
++	Increment operator
--	Decrement operator

- انتساب مرکب

```
i += 3;   ⟶   i = i + 3;
i *= j + 2;   ⟶   i = i * (j+2);
```

مثال

```
a  -=  c;   ⟶   a = a-c;
a  +=  5;   ⟶   a = a+5;
a  *=  6;   ⟶   a = a*6;
a  /=  b;   ⟶   a = a/b;
a %=  k;   ⟶   a = a%k;
```



انواع عملگر (ادامه...)

Relational Operators

- عملگرهای نسبی

Operator	Significance
<	less than
<=	less than or equal to
>	greater than
>=	geater than or equal to
==	equal
!=	unequal

- مثال

Comparison	Result
5 >= 6	
1.7 < 1.8	
4 + 2 == 5	
2 * 4 != 7	

اولویت‌ها

Precedence	Operator
High	arithmetic operators
	< <= > >=
	== !=
Low	assignment operators

- مثال

```
bool flag = index < max - 1;
```

ابتدا محاسبات صورت می‌گیرد، سپس مقایسه و در نهایت

مثال

```
int result;
result = length + 1 == limit;
```

ابتدا length + 1 محاسبه می‌شود و نتیجه با limit مقایسه می‌شود و نهایتاً مقدار درست یا نادرست به result منتسب می‌گردد.
چون result عدد صحیح است صفر یا یک منتسب می‌گردد.

```
int result;
(result = length + 1) == limit
```

ابتدا length + 1 محاسبه می‌شود و نتیجه در متغیر result ریخته می‌شود. در نهایت مقدار به دست آمده با limit مقایسه می‌شود و نهایتاً مقدار درست مقایسه می‌گردد.

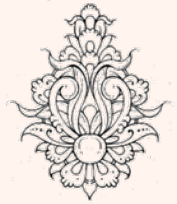
عملگرهای منطقی

LogicalOperators

A	B	A && B	A || B
true	true	true	true
true	false	false	true
false	true	false	true
false	false	false	false

A	!A
true	false
false	true

x	y	Logical Expression	Result
1	-1	x <= y || y >=0	
0	0	x > -2 && y == 0	
-1	0	x && !y	
0	1	!(x+1) || y - 1 > 0	



اولویت‌ها

Precedence	Operator
High	arithmetic operators
	< <= > >=
	== ! =
Low	assignment operators

- اولویت در اپراتورهای منطقی

اپراتور های یکانی از بالاترین اولویت برخوردارند

نقیض !

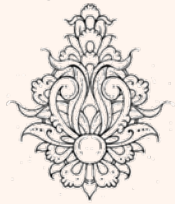
و &&

یا ||

محاسبات

مقایسه

انتساب



مثال

```
int main()
{
bool res = false;
int y = 5;
res = 7 || (y = 0);
cout << "Result of (7 || (y = 0)): " << res
<< endl;
cout << "Value of y: " << y << endl;
int a, b, c;
a = b = c = 0;
res = ++a || ++b && ++c;
cout << '\n'
<< " res = " << res
<< ", a = " << a
<< ", b = " << b
<< ", c = " << c << endl;
a = b = c = 0;
res = ++a && ++b || ++c;
cout << " res = " << res
<< ", a = " << a
<< ", b = " << b
<< ", c = " << c << endl;
return 0;
}
```

```
Result of (7 || (y = 0)): 1
Value of y: 5

 res = 1, a = 1, b = 0, c = 0
 res = 1, a = 1, b = 1, c = 0
```

به نظر شما این پاسخ درست است؟؟

نکته

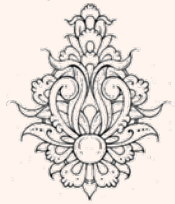
در عبارت منطقی **یا** به دلیل این که با درست بودن اولین قسمت پاسخ کلی درست خواهد بود از ارزیابی قسمت های دیگر صرف نظر می شود.

در عبارت منطقی **و** به دلیل این که با نادرست بودن اولین قسمت پاسخ کلی نادرست خواهد بود از ارزیابی قسمت های دیگر صرف نظر می شود.

عملگرهای بیتی

&	**بیتی AND**
|	بیتی OR
~	بیتی NOT
^	بیتی XOR
<<	شیفت به چپ
>>	شیفت به راست

```
C = a & b;
C = a | b;
C = ~a;
C = a^b;
C = a << 2
C= a >> 1
```



شیفت به چپ معادل ضرب در عدد ۲ و شیفت به راست معادل تقسیم بر عدد دو خواهد بود

عملگرهای بیتی (ادامه...)

- اگر a=10 و b=15 باشد حاصل عبارت زیر چیست؟

C=a&b

	بایت پر ارزش								بایت‌کم ارزش							
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
a	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
b	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
C	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0

→ C=10

عملگرهای بیتی (ادامه...)

- اگر a=10 و b=15 باشد حاصل عبارت زیر چیست؟

C=a | b

	بایت پر ارزش								بایت کم ارزش							
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
a	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
b	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
C	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

→ C=15

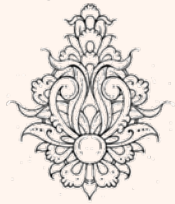
عملگرهای بیتی (ادامه...)

- اگر b=15 باشد حاصل عبارت زیر چیست؟

C=b<<2 →

	بایت پر ارزش								بایت کم ارزش							
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
b	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
C	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0

C=2*2*15=60



مثالی از کاربرد عملگرهای بیتی

- به عنوان مثال اگر اطلاعاتی از پورت موازی کامپیوتر خوانده شوند با اپراتورهای بیتی می‌توان آنها را تفسیر کرد. مثلا اگر بیت سوم عدد خوانده شده نشان‌دهنده‌ی وجود نداشتن کاغذ در پرینتر باشد و بخواهیم تنها بیت سوم را چک کنیم، می توانیم به صورت زیر عمل کنیم:

```
if((n&4) != 0)
    cout<<“Paper out!”;
```

(فرض کرده ایم عدد خوانده شده در n قرار گرفته و 1 بودن بیت سوم نشان دهنده نبودن کاغذ در پرینتر است)