طرح درس

عملگرهای محاسباتی

عملگر	مفهوم محاسباتی	مثال با مقادیر صحیح	مثال با مقادیر اعشار	
*	ضرب	4 * 7 ← 28	4.0 * 7.0 ← 28.0	اولویت ۱
/	تقسیم	4 / 7 ← 1	4.0 / 7.0 ← 1.75	
%	باقیمانده تقسیم	4 % 7 ← 3	4.0 % 7.0 ← غیرمجاز	
+	جمع	4 + 7 ← 11	4.0 + 7.0 ← 11.0	اولویت ۲
-	تفریق	4 - 7 ← 3	4.0 - 7.0 ← 3.0	

- پرانتز بر تمامی عملگرها، مقدم است.
- هرگاه در عبارتی چند عملگر با حق تقدم مساوی موجود باشد، ترتیب اجرا از چپ به راست خواهد بود.

مثال ۱) اولویت عملگرهای محاسباتی:

3 * 4 – 2 + 6 / 3 % 5

12 2

10 2

12

(9 – 4) / 5 * (2 + 5)

5 7

1

7

(3 – 4)/ 5 * (2 + 5)

-1 7

0

0

مثال ۲) به ترتیب انجام محاسبات و تبدیل نوع در موقع نیاز توجه کنید.

4 * 5 / 3 - 3.0 / 2 + 2

(1)20

(2)6 (3)1.5

(4)4.5

(5)6.5


```
int k
k = 2 × 3 − 8/5          → 5
k = 1 − 2 + 3 − 4        → −2
```

* پرانتز بالاترین تقدم را دارد.

```
  k = 1 − (2 + 3) − 4    → −8
2 × (3 − 8)/5            → −2
```

```
int i ;
i = 7 ;
i = i + 1 ;  → 8
```

(انتساب) ***assignment***

* **i+1=i** غلط است و با **i=i+1** برابر نیست. سمت چپ باید اسم یک متغیر باشد نه یک عبارت ریاضی.

```
i = i + 1  ≡ i + +
i = i − 1  ≡ i − −
```

```
int k ;
float i, j ;
i = 1.0 ;
j = 3.0 ;
k = i/j ;
```
→ صفر

```
int i, j ;
i = 1 ;
j = 3 ;
float k ;
k = i/j ;
```
→ صفر

* اول محاسبه می کند بعد نگاه می کند کجا باید ذخیره کند.

* نماد e : $۵/۱e۴ = ۵/۱ \times ۱۰^{۴}$

```
float k
k = 1 0000 ... ;
      23
k = 1e23 ;
```

$1.5\,e-4 = 1.5 \times 10^{-4}$

$1.5\,e+4 = 1.5 \times 10^{+4}$

- چنانچه دو طرف عملگر محاسباتی از یک نوع نباشند، ابتدا نوع هر دو یکسان می‌شود و سپس محاسبه انجام می‌گردد.
- توجه شود که چنانچه دو طرف عملگر محاسباتی مقدار صحیح باشد حاصل محاسبه صحیح است و در غیر این صورت حاصل اعشاری می‌شود.
- بطور کلی باید هر دو طرف عملگر هم اندازه و با دقت همسان شوند تا انجام محاسبه امکان‌پذیر شود.

۲) عملگر انتساب

مقدار عبارت سمت راست را در متغیر سمت چپ ذخیره می کند. در سمت چپ عملگر انتساب همواره یک متغیر قرار می گیرد. **X=1**

مقداری که در متغیر قرار می‌گیرد براساس نوع متغیر در سمت چپ = خواهد بود.

مثال ۳) چند نمونه از عبارت جایگزینی در زیر آورده شده است:

```
i = 50
x = i – 12.5
m = 10 – x * 4 + 2.5
```

مثال ۴) خروجی برنامه زیر چیست؟

```
int main()
{ //tests operators +, -, *, /, and %:
int m=54;
int n=20;
cout << "m = " << m << " and n = " << n << endl;
cout << "m+n = " << m+n << endl;
cout << "m-n = " << m-n << endl;
cout << "m*n = " << m*n << endl;
cout << "m/n = " << m/n << endl;
cout << "m%n = " << m%n << endl;
return 0;
}
```

خروجی:

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


مثال ۵) خروجی برنامه زیر چیست؟

```
int main()
{ //tests operators +, -, *, /, and %:
float x=54.0;
float y=20.0;
cout << "x = " << x << " and y = " << y << endl;
cout << "x+y = " << x+y << endl; // 54.0+20.0 = 74.0
cout << "x-y = " << x-y << endl; // 54.0-20.0 = 34.0
cout << "x*y = " << x*y << endl; // 54.0*20.0 = 1080.0
cout << "x/y = " << x/y << endl; // 54.0/20.0 = 2.7
return 0;
}
```

خروجی:

```
x = 54 and y = 20
x+y = 74
x-y = 34
x*y = 1080
x/y = 2.7
```

مثال ۶) در زیر چند متغیر معرفی و سپس مقدار آن‌ها محاسبه شده است:

```
int i, j;
float x, y;
i = 20;
x  = i / 8;          /* → x = 2.0   */
j  = x;              /* → j = 2     */
y  = i =x = 9/x;     /* → x = 4.5 →     i=4     →     y=4.0   */
i=-10;
j=4;
x=i%j;               /* → x = -2.0   */
```


مثال ۷) در زیر چند متغیر معرفی و سپس متغیرها بر حسب نیاز گسترش یافته‌اند:

```
int main()
{ //prints promoted values of 65 from char to double:
char c='A'; cout << " char c = " << c << endl;
short k=c; cout << " short k = " << k << endl;
int m=k; cout << " int m = " << m << endl;
long n=m; cout << " long n = " << n << endl;
float x=n; cout << " float x = " << x << endl;
double y=x; cout << " double y = " << y << endl;
return 0;
}
```

خروجی:

```
char c = A
short k = 65
int m = 65
long n = 65
float x = 65
double y = 65
```

از آنجا که در C++ کاراکترها دارای ارزش عددی می‌باشند، و این ارزش عددی در کامپیوترهای شخصی معادل کداسکی آن‌ها است، عملیات محاسباتی روی کاراکترها مجاز می‌باشد.

مثال ۸) حاصل عبارات زیر را محاسبه کنید:

'm' + 6 → 109 + 6 → 115
'm' + '6' → 109 + 54 → 163

راهنمایی: کداسکی حرف **m** برابر عدد **109** و کداسکی رقم **6** برابر عدد **54** می‌باشد،

مثال ۹) برنامه‌ای بنویسید که عدد اعشاری **X** را از ورودی گرفته و معکوس آنرا چاپ کند:

```
int main()
{ // prints reciprocal value of x:
double x;
cout << "Enter float: ";
cin >> x;
cout << "Its reciprocal is: " << 1/x << endl;
return 0;
}
```

عملگرهای جایگزینی (مرکب)

عملگر	فرم کلی استفاده	مفهوم معادل	مثال
+=	variable+=expression	variable =variable+expression	i+=1 → i=i+1
-=	variable-=expression	variable =variable-expression	j-=2*k → j= j-2*k
*=	variable*=expression	variable =variable*expression	x*=a+b → x=x*(a+b)
/=	variable/=expression	variable =variable/expression	y/=2-t → y=y/(2-t)
%=	variable%=expression	variable =variable%expression	m%=n → m=m%n

حق تقدم عملگرهای جایگزینی پایین‌تر از عملگرهای محاسباتی است.

برای چند عملگر جایگزینی: ترتیب اجرا از راست به چپ

مثال ۱۰) نتیجه دستورات زیر چیست؟

```
int n=22;
cout << " n = " << n << endl;
n += 9; // adds 9 to n
cout << "After n += 9, n = " << n << endl;
n -= 5; //subtracts 5 from n
cout << "After n -= 5, n = " << n << endl;
n *= 2; //multiplies n by 2
cout << "After n *= 2, n = " << n << endl;
n /= 3; //divides n by 3
cout << "After n /= 3, n = " << n << endl;
n %= 7; //reduces n to the remainder from dividing by 7
cout << "After n %= 7, n = " << n << endl;
```

نتیجه:

```
n = 22
After n += 9, n = 31
After n -= 5, n = 26
After n *= 2, n = 52
After n /= 3, n = 17
After n %= 7, n = 3
```

مثال ۱۱) نتیجه دستورات زیر چیست؟

```
int main()
{ // adds an int value with a double value:
int n = 22;
double p = 3.1415;
p += n;
cout << "p = " << p << ", n = " << n << endl;
return 0;
}
```

p = 25.1415, n = 22

۳) عملگر مقایسه‌ای

عملگر	در C++	یک مثال	معنی شرط
عملگرهای برابری			
=	==	x==y	x برابر با y است.
≠	!=	x!=y	x با y برابر نیست.
عملگرهای مقایسه ای			
>	>	x>y	x بزرگتر از y است.
<	<	x<y	x کوچکتر از y است.
≥	>=	x>=y	x بزرگتر از یا برابر با y است.
≤	<=	x<=y	x کوچکتر از یا برابر با y است.

مثال ۱۲)

```
X=4;
Y=5;
Z=3;
.
.
.
```

X == 4 , Y + Z == X + Z

T F

X==Y

F

مقایسه ای

۱) =< => < >

۲) == و =!

	اولویت عملگرها				شرکت پذیری در عملیات
۱	()				از چپ به راست
۲	*	/	%		از چپ به راست
۳	+	-			از چپ به راست
۴	<	<=	>	>=	از چپ به راست
۵	==	!=			از چپ به راست
۶	=				از راست به چپ

۴) عملگرهای منطقی:

* عملگر AND :

```
bool t=p && q;
```

p	q	t
T	T	T
T	F	F
F	T	F
F	F	F

* عملگر OR :

```
bool t=p||q;
```

p	q	t
T	T	T
T	F	T
F	T	T
F	F	F

p	q	p&&q	P || q
1	1	1	1
0	1	0	1
1	0	0	1
0	0	0	0

&& || !

AND OR NOT

اولویت : منطقی
۱) !
۲) &&
۳) ||

```
bool x,y;
x = (1 == 2) && (1 < 2);   → F
      F          T
y = !x  → T
x = !(1 == 2)   → T
```

شروط ترکیبی: (x>=0) && (x<=5)

⇒ تنها به کاربردن عملگرها پاسخ را غلط می‌کند

(x<0) || (x>5) ⇒ نقیض عبارت بالا !((x>=0)&&(x<=5))

نقیض یک عبارت را با علامت تعجب نشان می‌دهند

عملگرهای افزایش و کاهش

عملگر افزایش (++) و کاهش (--) می‌تواند قبل یا بعد از یک متغیر قرار گیرد، و موجب می‌شود که به متغیر یک واحد اضافه یا از آن کاسته شود.

قبل: (**++i**) ابتدا به مقدار متغیر یک واحد اضافه می‌شود، سپس از آن در عبارت استفاده می‌گردد.

بعد: (**i++**) اول از مقدار آن در عبارت استفاده می‌گردد، آنگاه به محتوای متغیر یک واحد اضافه می‌شود.

مثال ۱۳) به دستورهای زیر و مقادیر محاسبه شده دقت کنید.

```
i = 6;
j  = ++i;  →      i= 7     ،      j=7
k  = i++;  →      k= 7     ،      i=8

i = 6;
j  = --i;  →      i= 5     ،      j=5
k  = i--;  →      k= 5     ،      i=4
```

```
x =2;
y=x/ x++;   ➡   y=1  x=3

x =2;
y=x/ --x;   ➡   y=2 , x=1
```

توجه!

هنگام استفاده از ++ یا -- ، باید یا آن را قبل و یا بعد از متغیر نوشت و نمی توان هم زمان هر دو را استفاده کرد و دو واحد افزایش داشت، یعنی ++x++; یک دستور غلط و تعریف نشده است و نوشتن چنین دستوری باعث بروز پیغام خطا خواهد شد.

عملگر نوع

تغییرنوع متغیر در حین محاسبات

یک عملگر یک طرفه

```
n = int(v);          (int) v
```

وقتی از عملگر ( ) **int** استفاده کنیم، عدد ممیز شناور «بریده» می‌شود، گرد نمی‌شود. یعنی قسمت اعشاری عدد به طور کامل حذف می‌شود و فقط قسمت صحیح آن باقی می‌ماند.

مثال ۱۴) به دستورهای زیر و مقادیر محاسبه شده دقت کنید.

```
int i =4,  j;
float x = 15.0, y;
j  = (int) x %8;        →     j = 7
y  = (float) j/ i;      →     y=1.75
```

- چنانچه در عبارتی عملگرهای محاسباتی و یک طرفه با هم بکار رفته باشند، اجرای عملگرهای یک طرفه اول صورت می‌گیرد. اگر چند عملگر یک طرفه پشت سرهم باشند ترتیب اجرا برای آن‌ها از راست به چپ می‌باشد.
- با قراردادن عملگر یک طرفه (نوع) قبل از متغیر در یک عبارت، فقط مقدار آن در همان عبارت تغییر نوع می‌دهد. بر روی نوع خود متغیر هیچ تأثیری ندارد.

```
float  k;        k = 5/2;     →  2
                 k = float (5)/2;  →  2/5
                 k = float (5/2);  →  2
```

```
C = 'A'
```

مثال :
```
char C;
C = 65;
cout << C;  →  A
```

```
ch = 'A'
cout << int(ch);  →  65  →  (char)
```
راه فهمیدن کد اسکی یک کاراکتر

عملگر یک طرفه sizeof

اندازه متغیر: تعداد بایتهایی که در حافظه به آن تعلق دارد.

اندازه عبارت: تعداد بایتهایی که جهت ذخیره حاصل عبارت در حافظه، مورد نیاز می‌باشد.

عملگر **sizeof** یک عملگر یک طرفه است که اندازه عبارتی که به‌دنبال آن آورده شده است را تحویل می‌دهد.

مثال ۱۵) به دستورهای زیر و مقادیر محاسبه شده دقت کنید.

```
int i , j;
j = sizeof ( int );      →    j = 2
j  = sizeof (float);     →    j = 4
j  = sizeof (double);    →    j = 8
j  = sizeof (char);      →    j = 1
```


مثال ۱۶) به دستورهای زیر و مقادیر محاسبه شده دقت کنید.

```
int i =12345,  j;
float x = 5.2;
j  = sizeof i;          →    j = 2
j  = sizeof x;          →    j = 4
j  = sizeof 12;         →    j = 2
j  = sizeof (i*x);      →    j = 4
j  = sizeof  i*x;       →    j=(sizeof i)*x →    j = 10
```

	نوع عملگر	عملگر	ترتیب
۱	پرانتز	( )	چپ ← راست
۲	یک طرفه	++ -- (type) sizeof	راست← چپ
۳	محاسباتی ضرب، تقسیم و باقیمانده تقسیم	* / %	چپ ← راست
۴	محاسباتی جمع و تفریق	+ -	چپ ← راست
۵	جایگزینی	= += -= *= /= %=	راست← چپ

سرریزی عدد صحیح (Overflow)

مثال ۱۸) خروجی برنامه زیر چیست؟

```
int main()
{ //prints n until it overflows:
int n =1000;
cout << "n = " << n << endl;
n *= 1000; // multiplies n by 1000
cout << "n = " << n << endl;
n *= 1000; // multiplies n by 1000
cout << " n = " << n << endl;
n *= 1000; // multiplies n by 1000
cout << " n = " << n << endl;
return 0;
}
```

خروجی:

```
n = 1000
n = 1000000
n = 1000000000
n = -727379968
```

سرریزی عدد ممیز شناور

مثال ۱۹) خروجی برنامه زیر چیست؟

```
int main()
{ //prints x until it overflows:
float x=1000.0;
cout << "x = " << x << endl;
x *= x; //multiplies n by itself; i.e., it squares x
cout << "x = " << x << endl;
x *= x; //multiplies n by itself; i.e., it squares x
cout << "x = " << x << endl;
x *= x; //multiplies n by itself; i.e., it squares x
cout << "x = " << x << endl;
x *= x; //multiplies n by itself; i.e., it squares x
cout << "x = " << x << endl;
return 0;
}
```

خروجی:

```
x = 1000
x = 1e+06
x = 1e+12
x = 1e+24
x = inf
```

خطای گرد کردن

مثال ۲۰) خروجی برنامه زیر چیست؟

```
int main()
{ //illustrates round-off error:
double x = 1000/3.0;
cout << "x = " << x << endl; // x = 1000/3
double y = x-333.0;
cout << "y = " << y << endl; // y = 1/3
double z = 3*y-1.0
cout << "z = " << z << endl; // z = 3(1/3) - 1
return 0;
}
```

خروجی:

```
x = 333.333
y = 0.333333
z = -5.68434e-14
```

مثال ۲۱) خروجی برنامه زیر چیست؟

```
#include <cmath> //defines the sqrt() function
#include <iostream>
using namespace std;
int main()
{ //implements the quadratic formula
float a, b, c;
cout << "Enter the coefficients of a quadratic equation:"<< endl;
cout << "\ta: ";
cin >> a;
cout << "\tb: ";
cin >> b;
cout << "\tc: ";
cin >> c;
cout << "The equation is: " << a << "*x*x + " << b<< "*x + " << c << " = 0" << endl;
float d = b*b - 4*a*c; // discriminant
float sqrtd = sqrt(d);
float x1 = (-b + sqrtd) /(2*a);
float x2 = (-b – sqrtd) /(2*a);
cout << "The solutions are:" << endl;
cout << "\tx1 = " << x1 << endl;
cout << "\tx2 = " << x2 << endl;
cout << "check:" << endl;
cout << "\ta*x1*x1 + b*x1 + c = " << a*x1*x1 + b*x1 + c<< endl;
cout << "\ta*x2*x2 + b*x2 + c = " << a*x2*x2 + b*x2 + c<< endl;
return 0;}
```

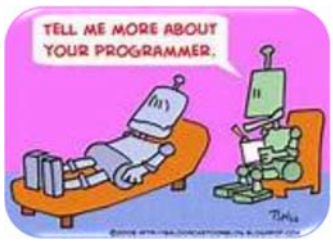
```
Enter the coefficients of a quadratic equation:
   a: 2
   b: 1
   c: -3
The equation is: 2*x*x + 1*x + -3 = 0
The solutions are:
   x1 = 1
   x2 = -1.5
check:
   a*x1*x1 + b*x1 + c = 0
   a*x2*x2 + b*x2 + c = 0
```



```
Enter the coefficients of a quadratic equation:
   a: 2
   b: 8.001
   c: 8.002
The equation is: 2*x*x + 8.001*x + 8.002 = 0
The solutions are:
   x1 = -1.9995
   x2 = -2.00098
check:
   a*x1*x1 + b*x1 + c = 5.35749e-11
   a*x2*x2 + b*x2 + c = -2.96609e-1
```

```
Enter the coefficients of a
quadratic equation:
    a: 1
    b: 2
    c: 3
The equation is: 1*x*x + 2*x + 3 = 0
The solutions are:
    x1 = nan
    x2 = nan
check:
    a*x1*x1 + b*x1 + c = nan
    a*x2*x2 + b*x2 + c = nan
```


```
Enter the coefficients of a
quadratic equation:
    a: 0
    b: 2
    c: 5
The equation is: 0*x*x + 2*x + 5 = 0
The solutions are:
    x1 = nan
    x2 = -inf
check:
    a*x1*x1 + b*x1 + c = nan
```



حوزه متغیرها

مثال ۲۲)

```
int main()
{ //illustrates the scope of variables:
      x = 11; // ERROR: this is not in the scope of x
      int x;
      {
            x = 22; // OK: this is in the scope of x
            y = 33; // ERROR: this is not in the scope of y
            int y;
            x = 44; // OK: this is in the scope of x
            y = 55; // OK: this is in the scope of y
      }
      x = 66; // OK: this is in the scope of x
      y = 77; // ERROR: this is not in the scope of y
      return 0;
}
```


متغیرهای تو در تو

مثال ۲۳)

```
int x = 11; // this x is global
int main()
{ //illustrates the nested and parallel scopes:
      int x = 22;
      { //begin scope of internal block
            int x = 33;
            cout << "In block inside main() : x = " << x << endl;
      } //end scope of internal block
      cout << "In main() : x = " << x << endl;
      cout << "In main() : ::x = " << ::x << endl;
      return 0;
} //end scope of main()
```

```
In block inside main() : x = 33
In main() : x = 22
In main() : ::x = 11
```

استفاده از بلوک‌ها به عنوان محدودهٔ حوزه

مثال ۲۴) در این برنامه سه متغیر مختلف با نام n استفاده شده است:

```
int main()
{     int n=44;
      cout << "n = " << n << endl;
      {     int n; // scope extends over 4 lines
            cout << "Enter an integer: ";
            cin >> n;
            cout << "n = " << n << endl;
      }
      {    cout << " n = " << n << endl; // n that was declared first
      }
      {     int n; // scope extends over 2 lines
            cout << "n = " << n << endl;
      }
      cout << "n = " << n << endl; // n that was declared first
}
```

```
n=44
Enter an integer: 111
n=111
n=44
n=4251897
n=44
```


پرسش و پاسخ ؟

http://www.cplusplus.com/reference/clibrary/cmath

	ارائه قدر مطلق آرگومان صحیح	int	**abs(int)**
به صورت اعشاری مضاعف	ارائه کوچکتر عدد صحیح بزرگتر یا برابر آرگومان	double	**ceil(double)**
آرگومان رادیان	محاسبه کسینوس آرگومان	double	**cos(double)**
	محاسبه کسینوس هیپربولیک	double	**cosh(double)**
e=2.7182818...	محاسبه e به توان آرگومان	double	**exp(double)**
	ارائه قدر مطلق آرگومان اعشار	double	**fabs(double)**
به صورت اعشاری مضاعف	ارائه بزرگترین عدد صحیح کوچکتر یا برابر آرگومان	double	**floor(double)**
باقیمانده پس از محاسبه خارج قسمت صحیح	محاسبه باقیمانده تقسیم آرگومان اول بر دوم	double	**fmod (double ,double)**
	محاسبه لگاریتم طبیعی آرگومان	double	**log(double)**
	محاسبه لگاریتم در مبنای ۱۰	double	**Log10(double)**
	آرگومان اول به توان آرگومان دوم	double	**pow (double ,double)**
عدد صحیح و مثبت	تولید یک عدد تصادفی صحیح	int	**rand( )**
آرگومان رادیان	محاسبه سینوس آرگومان	double	**sin(double)**
	محاسبه سینوس هیپربولیک	double	**sinh(double)**
آرگومان غیرمنفی	محاسبه جذر آرگومان	double	**sqrt(double)**
خروجی ندارد	دریافت مقدار اولیه مولد اعداد تصادفی	void	**srand(unsigned)**
آرگومان رادیان	محاسبه تانژانت آرگومان	double	**tan(double)**
	محاسبه تانژانت هیپربولیک	double	**tanh(double)**