Chapter 3: Expression

Course: 06016206 – Computer Programming

Asst. Prof. Dr. Kitsuchart Pasupa
Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang

Outline

- Arithmetic Expression
- Assignment Statement
- Unary Operator
- Precedence & Associativity
- Type Conversion
- Boolean Expression

Arithmetic Expression

Operator	Definition	Examples	Results (a=8, b=4)
+	Addition	6+2 a+b	8 12
-	Subtraction	6-2 a-3	4 5
*	Multiplication	6*2 a*b	12 32
/	Division	6/2 a/b	3 2
%	Modulo	6%2 b%a	0 4

Assignment Operator

```
var_name = constant
```

```
var_name = expression
```

- An assignment operation assigns the value of the right-hand operand to the storage location named by the left-hand operand

Example

```c
#include <stdio.h>
void main()
{
   int a,b,ans1,ans2;
   a = 10; b = 20;
// a%b = ans1; //ERROR
   ans1 = a%b; //OK
   ans2 = b%a; //OK
   printf("%d\n",ans1);
   printf("%d",ans2);
}
```

```
10
0
```

Example

```c
#include <stdio.h>
#include <string.h>
int main()
{
   char str[10];
// str = "hello";          //Error
   strcpy(str,"hello");    //OK
   printf("%s",str);
   return 0;
}
```

```
hello
```

Compound Operator

Operator	Definition	Examples	Full Form
+=	Addition	y+=x	y=y+x
-=	Subtraction	y-=x	y=y-x
*=	Multiplication	y*=x	y=y*x
/=	Division	y/=x	y=y/x
%=	Modulo	y%=x	y=y%x

Unary Operator

- As unary operations have only one operand they are evaluated before other operations containing them.
- Negative
  - 3 – –2
    - the first '–' represents the binary subtraction operation,
    - the second '–' represents the unary negation of the 2
    - similar to 3 – (–2) = 5
- Positive
  - There is also a unary positive but it is not needed since we assume a value to be positive: (+2) = 2
  - Unary positive does not change the sign of a negative operation: (+(–2)) = (–2)
  - In this case a unary negative is needed to change the sign: (–(–2)) = (+2)

Increment & Decrement Operator

Operator	Descriptions	Example	Procedure
++	Increment	x++, ++x	Increase x by 1
		y=++x;	Increase x by 1 then set y to the value
		y=x++;	set y to the value of x then increase x by 1
--	Decrement	x--,--x	Decrease x by 1
		y=--x;	Decrease x by 1 then set y to the value
		y=x--;	set y to the value of x then decrease x by 1

Example

```
#include <stdio.h>
int main()
{
   int x=10;
   int y;
   y=++x;
   printf("x=%d\n",x);
   printf("y=%d\n",y);
   return 0;
}
```

```
x=11
y=11
```

```
#include <stdio.h>
int main()
{
   int x=10;
   int y;
   y=x++;
   printf("x=%d\n",x);
   printf("y=%d\n",y);
   return 0;
}
```

```
x=11
y=10
```

sizeof Operator

- the unary operator sizeof is used to calculate the sizes of data types in number of bytes

```
sizeof(type)
```

- Example

```
sizeof(int)
```

Precedence & Associativity

Precedence	Description	Operator	Associativity Left→Right
1	Parenthesis Postfix increment/decrement	( ) ++ --	Y
2	Prefix increment/decrement Unary plus/minus Logical negation/ bitwise complement Cast (change type)	++ -- + - ! ~ (type)	N
3	Multiplication/Division/Modulus	* / %	Y
4	Addition/Subtraction	+ -	Y
5	Relational less than/less than or equal to Relational greater than/greater than or equal to	< <= > >=	Y
6	Relational is equal to/ is not equal to	== !=	Y

Precedence & Associativity

Precedence	Description	Operator	Associativity Left→Right
7	Bitwise AND	&	Y
8	Bitwise Exclusive OR	^	Y
9	Bitwise Inclusive OR	|	Y
10	Logical AND	&&	Y
11	Logical OR	||	Y
12	Assignment Addition/Subtraction Assignment Multiplication/Division Assignment Modulus Assignment	= += -= *= /= %=	N

Mathematics & C Expression

Maths	C
$x+y-z$	`x+y-z`
x^2+4yz	`x*x + 4*y*z`
b^2-4ac	`(b*b)-4*a*c`
$2r$	`2*r`

Examples

Sequence	Expression	First Computed Expression
1	x = 10 * 4 * 2 + 21 / 3	10 * 4 , 21 / 3
2	x = 40 * 2 + 7	40 * 2
3	x = 80 + 7	80 + 7
4	x = 87	

Sequence	Expression	First Computed Expression
1	x = 3 * 4 + 4 / 2 – 5 * 4	3 * 4 , 4 / 2 , 5 * 4
2	x = 12 + 2 - 20	12 + 2
3	x = 14 - 20	14 - 20
4	x = -6	

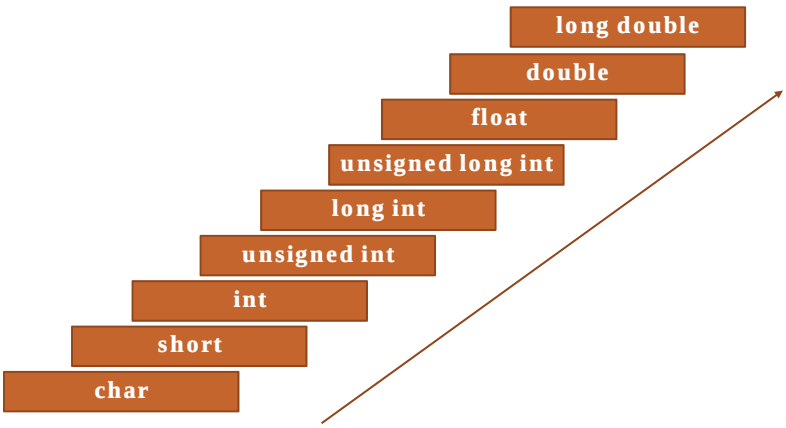
Type Conversion

- Implicit Type Conversion
  - automatic type conversion by the compiler
  - data of one or more subtypes can be converted to a supertype as needed at runtime
- Explicit Type Conversion
  - explicitly defined within a program

Implicit Type Conversion

Type 1	Operator	Type 2	Result	Example	
int	+, -, *, /	int	int	3*3 19/2	9 9
int	+, -, *, /	float	float	3*3.0 19/2.0	9.000000 9.500000
float	+, -, *, /	int	float	3.0/3 19.0/2	9.000000 9.500000
float	+, -, *, /	float	float	3.0/3.0 19.0/2.0	1.000000 9.500000
int	%	int	int	7%4 8%4	3 0

Promotion Hierarchy



Example

```
#include <stdio.h>
void main()
{
   float floatNum = 25.21;
   int intNum = 300;
   short shortNum = 10;

   printf("%d\n", intNum * shortNum);
   printf("%.2f\n", floatNum * intNum);
}
```

```
3000
7563.00
```

Example

```
#include <stdio.h>
void main()
{
   int iValue = 10;
   float fValue = 15.5;
   float fResult;                  //Correct
// int fResult;                   //Wrong

   fResult = iValue + fValue;
   printf("%d+%.2f=%.2f", iValue, fValue, fResult);
}
```

```
10+15.50=25.50
```

Explicit Type Conversion

```
(type) variable
```

```
(type) expression
```

- We can call this “Casting”

```
int iNum = 5;
float fNum;
fNum = (float) iNum;
```

Example

```
#include <stdio.h>
void main()
{
   int i1 = 10;
   int i2 = 3;
   float f;

   f = i1/i2;
   printf("%.2f\n", f);

   f = (float) (i1/i2);
   printf("%.2f\n", f);

   f = (float) i1/i2;
   printf("%.2f\n", f);
}
```

```
3.00
3.00
3.33
```

Boolean Expression

- True or False
- 2 types
  - Relational operator
  - Logical operator

Relational Operator

Operator	Descriptions	Examples	Results x=12, y=-5
>	Greater than	x > y	1
>=	Greater than or equal to	x >= y	1
<	Less than	x < y	0
<=	Less than or equal to	x <= y	0
==	Equal to	x == y	0
!=	Not equal to	x != y	1

Logical Operator

Operators	Descriptions
&& || !	And Or Not

Examples

A	B	A&&B	A||B	!A
T	T	T	T	F
T	F	F	T	F
F	T	F	T	T
F	F	F	F	T