



ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Expressions

Department of Information System
SoICT, HUST

Expressions

- Combine values using operators and function calls
- Return a value of a known type (int, double, float, pointer)
- Example:
 - $(3+4)/2$ returns an integer value (3).
 - + and / are operators, 3, 4, 2 are operands.

Expressions

- An operator is something which takes one or more values and does something useful with those values to produce a result
- Each thing which is operated upon by an operator is called an operand
- Operation is the action which was carried out upon the operands by the operator

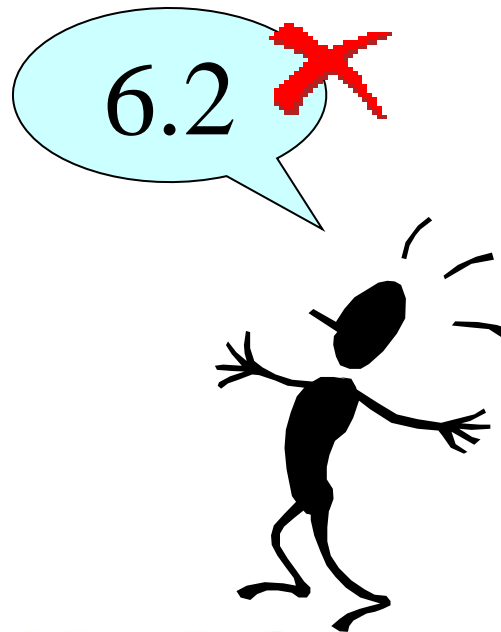
Arithmetic Expressions

- take arithmetic (numerical) values
- return an arithmetic (numerical) value
- Are composed using the following operators:
 - + (unary plus)
 - - (unary minus)
 - + (addition)
 - - (subtraction)
 - * (multiplication)
 - / (division or quotient)
 - % (modulus or remainder)

Example

$$1 + 2 * 3 - 4 / 5$$

$$= 1 + (2 * 3) - (4 / 5)$$



Example (con't)

$$1 + 2 * 3 - 4 / 5 =$$

$$1 + (2 * 3) - (4 / 5)$$

Divide two integers,
the result is also an
integer



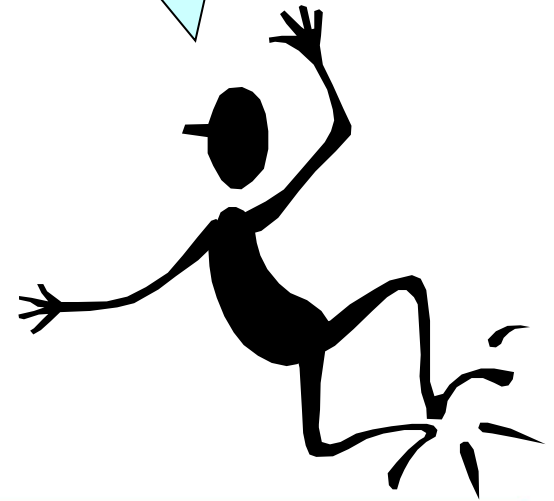
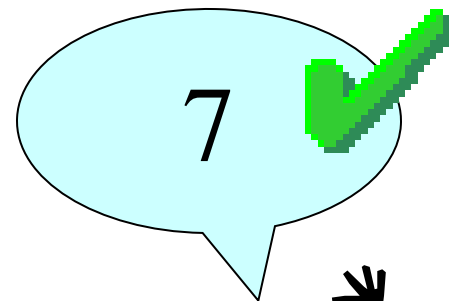
Example (con't)

$$1 + 2 * 3 - 4 / 5 =$$

$$1 + (2 * 3) - (4 / 5)$$



$$= 0$$



Example (con't)

- Use a real number to create an expression that return a real value

$$1 + 2 * 3 - 4.0 / 5$$

$$= 1 + (2 * 3) - (4.0 / 5)$$

$$= 1 + 6 - 0.8$$

$$= 6.2$$

Comparison operators

- $<$ (less than)
- $<=$ (less than or equal)
- $>$ (greater than)
- $>=$ (greater than or equal)
- $==$ (equal)
- $!=$ (in-equal)

Example

$$1 + 2 < 3$$

$$= (1 + 2) < 3$$

$$= 3 < 3 = 0$$

- Not to be confused between `==` and `=` (assignment)

Example

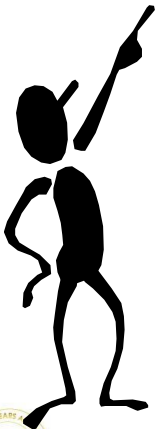
Prevent
misconsidering
as assign
operator (=)

3 == 4 → 0

3 != 4 → 1

3 < 4 → 1

3 < 4 && 5 > 2 → 1



Logic

- A special data type that has only two values:
 - true
 - false
- It is used to create the selection of conditions or the loop for an algorithm
- Boolean expression: is an expression that return only true/false

Use **int** as logic

- In C, logic values are represented by integer
 - 0 is false
 - any non-zero value is taken interpreted as true (often use 1)
- All expressions in C return a number
- A “true” logic expression will return 1, otherwise 0

Logic operators

- ... is used to built logic expression
- && (and)
- || (or)
- ! (not)
- comparison (**==** , **!=** , **<** , **>** , **<=** , **>=**)

Example

$$\begin{aligned} & (3 == 3) \ \&\& \ (1 + 2) < 3 \\ & = 1 \ \&\& \ (3 < 3) \\ & = 1 \ \&\& \ 0 = 0 \end{aligned}$$

Example

Prevent
misconsidering
as **and** bit (&)

Prevent
misconsidering
as **reverse** bit
(~)

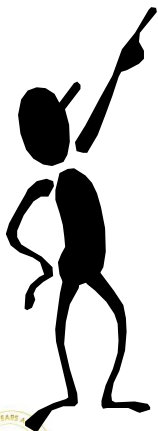
Prevent
misconsidering
as **or** bit (|)

5 && 4 → 1

1 || 4 → 1

! 0 → 1

! 0 || 0 && 2 → 1



Bit operators

An expression that only uses bit operators is not logic expression. Result of this expression is an integer.

& (and bit)

| (or bit)

~ (negation)

>> (shift right)

<< (shift left)

Bit operators

- Not to be confused with boolean operators: $\&\&$, \parallel , $!$
- Example:

$$\begin{array}{r} 5 = 101 \\ \& 4 = 100 \\ \hline = 4 = 100 \end{array}$$

$$1 \mid 4 \rightarrow ?$$

$$5 \& (4 \gg 1) \rightarrow ?$$

Common errors

```
#include <stdio.h>
```

```
/* Common errors */
```

```
int main()
```

```
{
```

```
    int score;
```

```
    scanf("%d", &score);
```

```
    if ( score == 9 || 10 )
```

```
    {
```

```
        printf("Excellent\n");
```

```
    }
```

```
    return 0;
```

Return value is
always 1

Return value is
0 or 1

Common errors (con't)

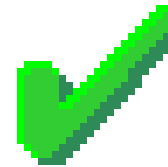
```
#include <stdio.h>

/* Correct program */

int main()
{
    int score;

    scanf("%d", &score);

    if ( score == 9 || score == 10 )
    {
        printf("Excellent\n");
    }
    return 0;
```



Common errors (con't)

```
#include <stdio.h>

/* Common errors */

int main()
{
    int score;

    scanf("%d", &score);

    if ( 8 <= score <= 10 )
    {
        printf("Good\n");
    }

    return 0;
}
```

Return value is
always 1

Return value is
0 or 1

Common errors (con't)

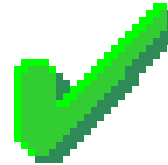
```
#include <stdio.h>

/* Correct program */

int main()
{
    int score;

    scanf("%d", &score);

    if ( 8 <= score && score <= 10 )
    {
        printf("Good\n");
    }
    return 0;
```



Assignment expressions

- Assignment = is also an operator that returns the assignment value.
- This operator can be used to create an expression that return a value: result of the assignment is the right value of the expression
- Example:

$(x = 4) \rightarrow 4$

$(y = 0) \rightarrow 0$

$a = b = 5 \rightarrow a = (b = 5) \rightarrow a = 5$

- Can create an expression with a series of assignment
 $x = y = z = 4$

Common errors (con't)

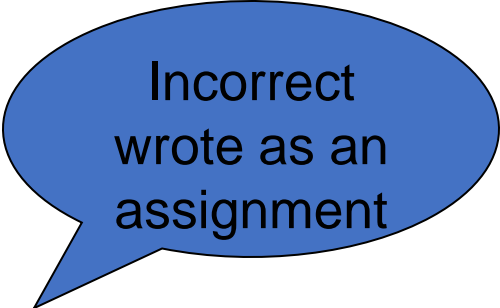
```
#include <stdio.h>

/* Common errors */

int main()
{
    int score;

    scanf("%d", &score);

    if (score = 9 || score = 10)
    {
        printf("Good!\n");
    }
    return 0;
}
```



Incorrect
wrote as an
assignment

Common errors (con't)

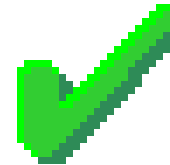
```
#include <stdio.h>

/* Probably the most common C error. */

int main()
{
    int score;

    scanf("%d", &score);

    if (score == 9 || score == 10)
    {
        printf("OK!\n");
    }
    return 0;
}
```



Some extend assignment operators

Operator	Example	Equal expression
+=	$x += 5$	$x = x + 5$
-=	$x -= 5$	$x = x - 5$
*=	$x *= 5$	$x = x * 5$
/=	$x /= 5$	$x = x / 5$
%=	$x \% = 5$	$x = x \% 5$

Increment, decrement operators

- ++ is the *increment* operator
- ++i is equivalent to $i = i + 1$
- -- is the *decrement* operator
- --j is equivalent to $j = j - 1$
- Two ways of writing: prefix (++i) and suffix (i++)
- They are different in return values of expressions.

Example, if $i = 5$

- Prefix return value after adding 1, $(++i) \rightarrow 6$
- Posfix return value before adding 1, $(i++) \rightarrow 5$
- In both cases, value of i increases by 1

Example

```
int i = 5;  
++i;  
printf(“%d”, i);
```

- Output: 6

Conditional Expressions

- ... a ternary operator

Condition ? Expr2 : Expr3

- Example:

```
int max,a,b;
```

```
...
```

```
max = ( a > b ) ? a : b;
```

Casting data type

- Assignment is only carried out in variables and values in the same data type
- C can automatically convert data type for assignment if this conversion **do not loose information**. Example, convert from int to float

```
int a;  
float f;  
f = a; /* OK */  
a = f; /* not OK */
```

- In case of loosing information, casting data type is needed. Example, convert from float to int.

```
a = (int) f;
```

Precedences

- Unary operators (!, -)
- Multiply, divide (*, /, %)
- Addition, subtraction (+, -)
- Comparison 1 (<, <=, >, >=)
- Comparison 2 (==, !=)
- And (&&)
- Or (||)

Example

- $7+5 \& \& 4 < 2+3-2/3 || 5 > 2+1$
- $(7+5) \& \& 4 < 2+3-(2/3) || 5 > (2+1)$
- $12 \& \& 4 < (2+3-0) || (5 > 3)$
- $12 \& \& (4 < 5) || 1$
- $(12 \& \& 1) || 1$
- $1 || 1 = 1$

Exercise

- $3\&\&7+4/3-2>6+-3*10\%2$
- $2+3/5>6-10/2\|3/7\&\&4$
- $(3<<1)\&(4>>2)|5$
- $(1>4)\&\&(2\|(3<4))$

(!, -)

(*, /, %)

(+, -)

(<, <=, >, >=)

(==, !=)

And (&&)

Or (||)



25 YEARS ANNIVERSARY
SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

**Thank you
for your
attentions!**



soict.hust.edu.vn/



fb.com/groups/soict

