ESC101: Introduction to Computing

Conditional Expressions

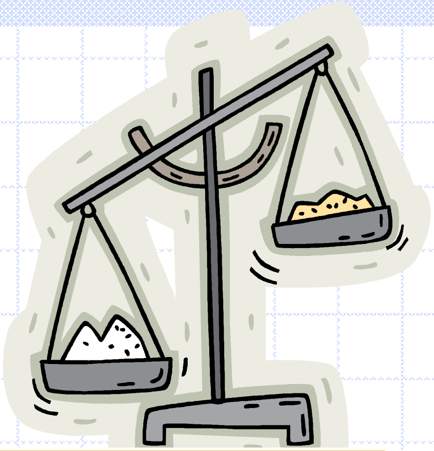
Conditional Expressions

- An expression that evaluates to either true or false.
  - Often known as Boolean expression.
- C **does not** have a separate Boolean data type
  - Value 0 is treated as false.
  - Non-zero values are treated as true.

Conditional Expressions

- If an expression evaluates to true, we get a value 1
  - Think of 1 as *default* true value
- If an expression evaluates to false, we get a value 0

Relational Operators



- Compare two quantities

Operator	Function
>	Strictly greater than
>=	Greater than or equal to
<	Strictly less than
<=	Less than or equal to
==	Equal to
!=	Not equal to

- Work on int, char, float, double...

Examples

Rel. Expr.	Result	Remark
3>2	1	
3>3	0	
‘z’ > ‘a’	1	ASCII values used for char
2 == 3	0	
‘A’ <= 65	1	'A' has ASCII value 65
‘A’ == ‘a’	0	Different ASCII values
(‘a’ – 32) == ‘A’	1	
5 != 10	1	
1.0 == 1	AVOID	May give unexpected result due to approximation

Avoid mixing int and float values while comparing. Comparison with floats is not exact!

Logical Operators

Logical Op	Function	Allowed Types
&&	Logical AND	char, int, float, double
||	Logical OR	char, int, float, double
!	Logical NOT	char, int, float, double

Remember

- value 0 represents false.
- any other value represents true.

Truth Tables

E1	E2	E1 && E2	E1 || E2
0	0	0	0
0	Non-0	0	1
Non-0	0	0	1
Non-0	Non-0	1	1

E	!E
0	1
Non-0	0

Examples

Expr	Result	Remark
2 && 3	1	
2 || 0	1	
‘A’ && ‘0’	1	ASCII value of ‘0’≠0
‘A’ && 0	0	
‘A’ && ‘b’	1	
! 0.0	1	0.0 == 0 is **guaranteed**
! 10.05	0	Any real ≠ 0.0
(2<5) && (6>5)	1	Compound expr

Examples

- You can create very complex expressions, involving arithmetic, logical and relational operators, constants, variables, function calls. e.g:

```
(x + 7 > 93) && !(y + 3 % z)
        || (abs(sqrt(w) – g) < epsilon)
```

Example

- Problem: Input 3 positive integers. Print the count of inputs that are even and odd.
  - Do not use if-then-else

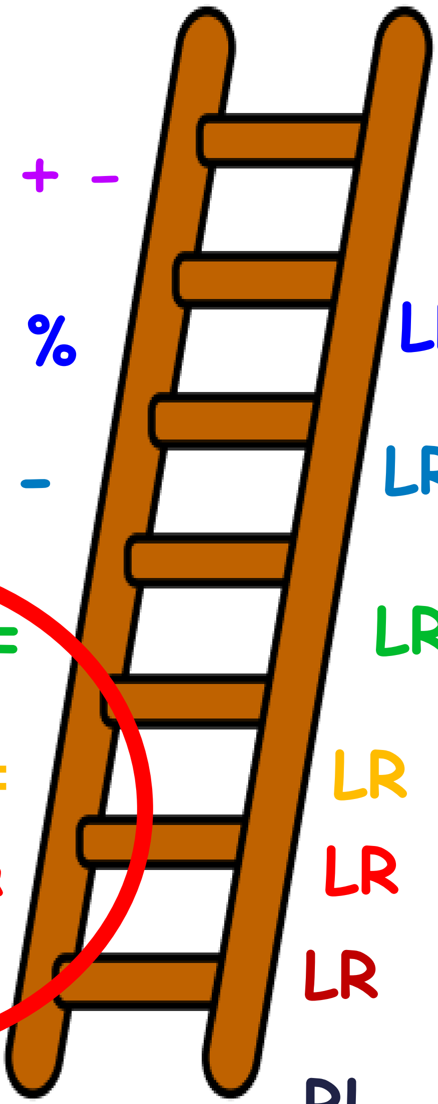
INPUT
10
5
3

OUTPUT
Even=1
Odd=2

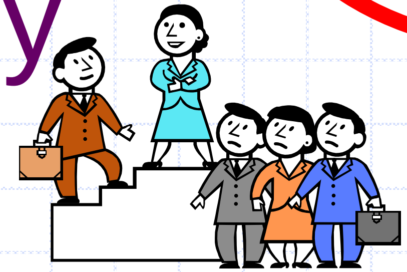
```
int a; int b; int c;
int cEven; // count of even i
scanf("%d%d%d", &a,&b,&c); // input a,b,c

// (x%2 == 0) evaluates to 1 if x is Even,
//                          0 if x is Odd
cEven = (a%2 == 0) + (b%2 == 0) + (c%2 == 0);
printf("Even=%d\nOdd=%d", cEven, 3-cEven);
```

Precedence and Associativity (Refined)

Binary Operations!
Operate on 2 Arguments
Result is either 0 or 1



! + -	RL
* / %	LR
+ -	LR
< <= > >=	LR
== !=	LR
&&	LR
||	LR
=	RL



Class Quiz 2

- What is the value of expression:

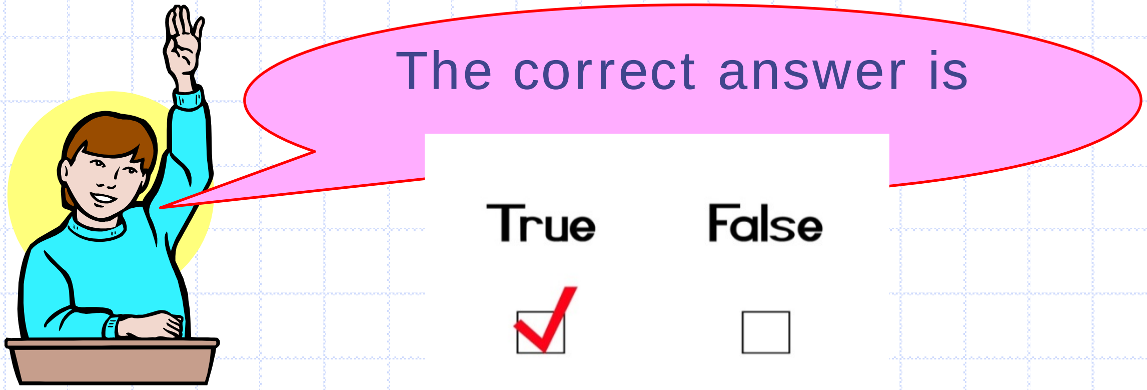
$$0 <= 10 <= 4$$

a) Compile time error

b) Run time crash

c) False (0)

d) True (1)



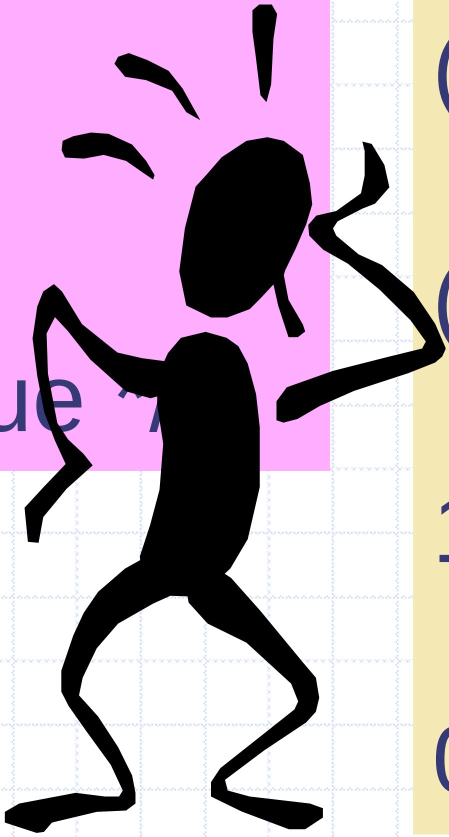
Evaluation

0 <= 10 <= 4

(0 <= 10) <= 4

1 <= 4

1 /* True */



Probably not what you intended. The intended expression is:

0 <= 10 && 10 <= 4

(0 <= 10) && (10 <= 4)

(1) && (10 <= 4)

1 && (0)

0 /*False*/

Conditional Statements

- In daily routine
  - If it is very cold, I will skip class.
  - If there is a quiz tomorrow, I will first study and then sleep. Otherwise I will sleep now.
  - If I have 500 Rs, I will order pizza. If I have 20 Rs, I will eat Maggi. If I have 5 Rs, I will eat biscuits. If I do not have any money, I will eat in hostel mess ☺

Conditional statements in C

- 3 types of conditional statements in C
  - if (cond) action
    else some-other-action
  - if (cond) action
  - switch-case
- Each action is a sequence of one or more statements!

Statements and Blocks

- An expression such as x=0 or printf(...) becomes a statement when it is followed by a semi-colon, as in

```
x = 0;
printf( ... );
5+2;
```

- Braces { and } are used to group variable declarations and statements together into a compound statement or a block
  - Syntactically equivalent to a single statement.
  - Can use it anywhere a single statement can be used

Statements and Blocks

```
{
   int x; float y; /* 2 statements */
   x = 10;
   printf(“x = %d\n”, x);
}
```

A single block

if-else statement

- Read two integers and print the min.

```
# include <stdio.h>
int main() {
    int x, y;
    scanf("%d%d", &x,&y);
    if (x < y) {
        printf("%d", x);
    } else {
        printf("%d", y);
    }
    return 0;
}
```

1. Check if x is less than y.
2. If so, print x
3. Otherwise, print y.

Tracing Execution of if-else

```
# include <stdio.h>
int main() {
      int x; int y;
      scanf("%d%d", &x,&y);
      if (x < y) {
          printf("%d\n",x);
      }
      else {   printf("%d\n",y);}
       return 0;
}
```

6 < 10 so the if-branch is taken

Run the program

Input

6   10

x   y

6   10

Output

6