Mahidol University

ITCS113
Fundamentals of Programming

Lecture 2 - Selection

Instructor: Asst. Prof. Dr. Akara Supratak

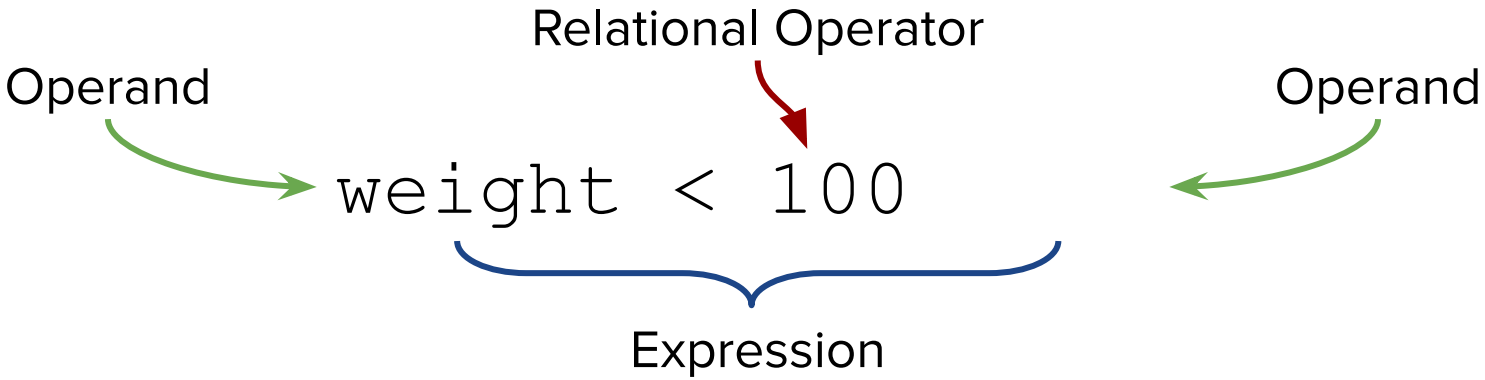
Contact: akara.sup@mahidol.edu

Agenda

- Relational and Logical Expression
- if-else statement
- switch statement
- Symbolic constants

Relational + Logical Expression

Relational Expression



- A relational expression consists of a relational expression comparing two operands
- Produce a **Boolean** result
  - **True**: 1 (or non-zero value)
  - **False**: 0

Relational Operators

Relational Operator	Meaning
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal to
!=	Not equal to

Exercises

Expression	Value	Interpretation
`'A'<'C'`	1	True
`'A'>'a'`	0	False
`1=='1'`		
`(10%2)>=1`		
`(10%2)==(2/3)`		
`(5/9)!=(5.0/9)`		

Logical Expression

- A logical expression is a more complex relational expression created by AND, OR, and NOT.
- Produce a **Boolean** result
  - True: 1 (or non-zero value)
  - False: 0

Operator	Meaning	Example
`&&`	AND	`(25/5 == 5) && (2+3 == 5);` `(3*2 == 6) && (2+3 == 6);`
`||`	OR	`(25/5 == 5) || (2+3 == 5);` `(3*2 == 6) || (2+3 == 6);`
`!`	NOT	`!(3*2 == 6);` `!(2+3 == 6);`

Operator Precedence and Associativity

- **Operator Precedence** indicate which operators has higher *priority* than another
- **Operator Associativity** indicates *the order* in which the operators are evaluated

Logical operator listed from the highest precedence to the lowest precedence.

Operator	Associativity
!	right to left
& &	left to right
| |	left to right

Example: Logical Expression

Assume `a = 5.0, b=2.0, c=4, d=6, flag=0`

Expression	Value	Interpretation
`a>b && c<=d`		
`flag || c/d`		
`!(a!=b)`		

Writing an Expression

- When the problem has a *specific condition*, programmers need to analyze and translate that condition to be a C expression.
- Relational expressions are also known as conditions, which can be used to select the prefered criteria
- **Example**:

> "Write an expression to check if an input number, **`x`**, is a positive number and also an odd number"

```
x > 0 && x%2 == 1
```

Exercise: Writing an expression

Given a variable x, write relational expression to check an input number, X, to satisfy the following conditions

Condition	C Expression
Between 0 and 100	
A negative number	
An even number between 0-100	
Greater than or equal to 24 or less than 16	
End with 7	
Not a negative even number (= positive odd/even and negative odd)	

Exercise: Writing an expression

Note: Use variable names of your own choosing

Condition	C Expression
The input is a letter 'p' (lowercase)	`input == 'p'`
A Temperature is *greater than* 37.5	
A weight is *below* 10 kg.	
A number of characters is *at least* 5 char	
A class with at least 20 students but not more than 40 students.	
A character is any characters from 'A' to 'Z'	
A number that is divisible by 3 or 5	

Operator Precedence and Associativity

	Operator	Associativity
	`!, (unary) -, ++, --`	right to left
Arithmetic	`*, /, %`	left to right
	`+,- (subtraction)`	left to right
Relational	`<, <=, >, >=`	left to right
	`==, !=`	left to right
Logical	`&&`	left to right
	`||`	left to right
Assignment	`+=, -=, *=, /=`	right to left

`if` and `if-else` Statement

Flow Controls

- Flow controls (in programming): the management of program statements that are executed.
- Classified into 4 standardized structures
  - **Sequential**: normal flow of control
  - **Selection**: select which statement to be performed next
  - **Repetition**: repeat a set of statement
  - **Invocation**: call a function

`if` statements

`if-else` statements

Exercise: Membership

Write a program to receive the total price and member-ship status. Calculate the total price based on the membership status and print it out. Every member gets 10% discount.

Step 1: Analyze the problem

- What is the input?
- What is the output?
- What is the formula/algorithm to relate the inputs and the output?

Exercise: Membership

Nested `if-else`

Chained `if-else`

Exercise: Membership II

- Extend the membership exercise
  - **Member**: Continue to ask for his/her age. If the age is greater than 60, give the 20% discount. Otherwise, the discount is 10%.
  - **Non-member**: Continue to ask whether he/she would like to pay by cash or credit-card. Paying by cash will get 5% discount. Paying by credit-card will not get any discount.

Exercise: Membership II

`if-else` chain (nested)

The `else-if` statement is useful when there are **multiple** conditions that may all be evaluated to `true`

```
if (expression1)
  statement1;
else if (expression2)
  statement2;
else if (expression3)
  statement3;
…
else if (expressionN)
  statementN;
else
  lastStatement;
```

Example: Grading System

Score	Grade
>= 80	A
>= 70	B
>=60	C
>=50	D
<50	F

```
#include <stdio.h>
int main()
{
    int score;
    printf("Input a score: ");
    scanf("%d", &score);









    printf("Thank you");
    return 0;
}
```

`switch` Statement

`switch` statement

Example: Select One Option

`switch` vs. chained `if-else` statement

Symbolic Constants

Symbolic Constants

- `#define` can be used to define constant variables, i.e., you cannot change the value
- `#define CNAME value`
- Example

```
#define PI 3.14
#define G 9.81
#define DEBUG 0
```

What is the different?

```
#define PI 3.14
```

- Constant
- This `PI` cannot be changed

```
float PI=3.14;
```

- Variable
- This `PI` can be changed

Symbolic Constants

```c
#include <stdio.h>

#define PI 3.14
#define G 9.81
#define DEBUG 0
#define CURRENCY '$'
#define GREETING "Hello World"

int main()
{
    printf("circle area: %.2f\n", PI * 2 * 2);
    printf("gravity: %.2fm/s^2\n", G);

    if (DEBUG)
        printf("Print something to check bugs. \n");

    printf("%s, currency: %c\n", GREETING, CURRENCY);

    return 0;
}
```

```
circle area: 12.56
gravity: 9.81m/s^2
Hello World, currency: $
```

Lab Exercises