# C# Programming

Zheng-Liang Lu

Department of Computer Science & Information Engineering
National Taiwan University

Online Course

```
1 class Lecture3
2 {
3
4                 "Flow Controls: Selection"
5
6 }
7
8 // Keywords:
9 if, else, switch, case, default, break
```

# Flow Controls

- We proceed to introduce the building blocks of algorithms as follows.
- First, most of statements are executed in order.
- A program can handle with multiple situations if the branching (selection) rules are known and written.
- Moreover, the program can repeat actions if necessary.
- For example, remember how to find the maximum in the input list?

# Representation for Branching

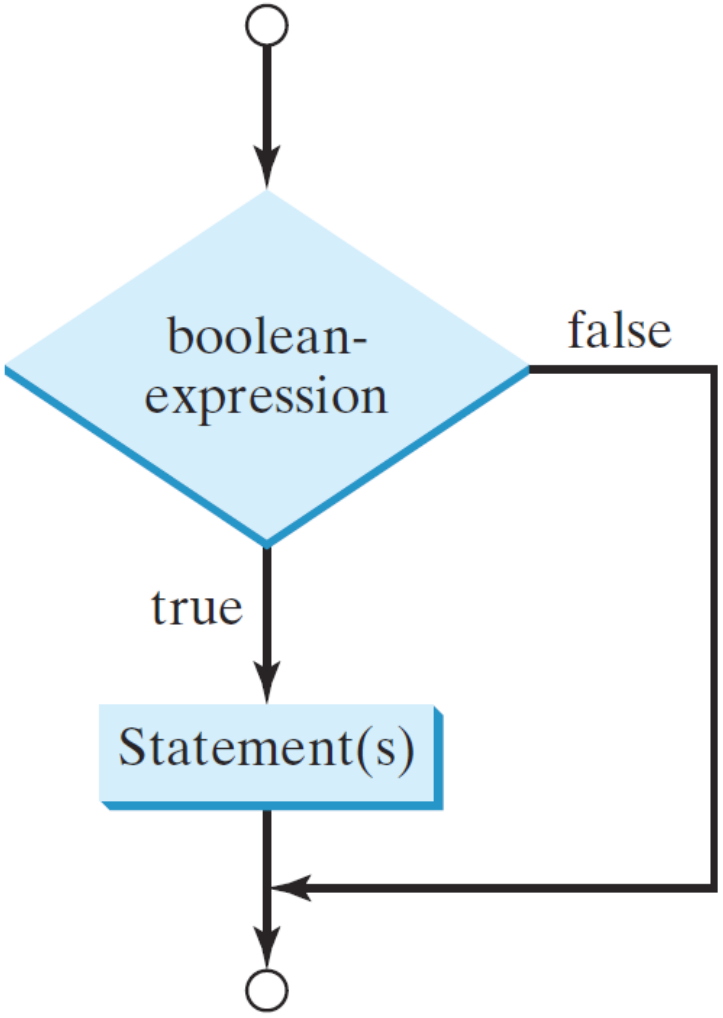
- Conditional statements by if-else.
- Conditional statements by switch-case-break-default.
- Conditional operators.

# Branching Statements by if

- The syntax is simple, shown below.

```
1 ...
2         if (/* Condition: a boolean expression */)
3         {
4             // Selection body: conditional statements.
5         }
6 ...
```

- If the condition is evaluated true, then the conditional statements will be executed once.
- If false, then the selection body will be ignored (or we say that the program jumps to Line 5).
- Note that the braces can be omitted if the body contains only single statement.


boolean-
expression
false
true
Statement(s)

# Example: Circle Area (Revisited)

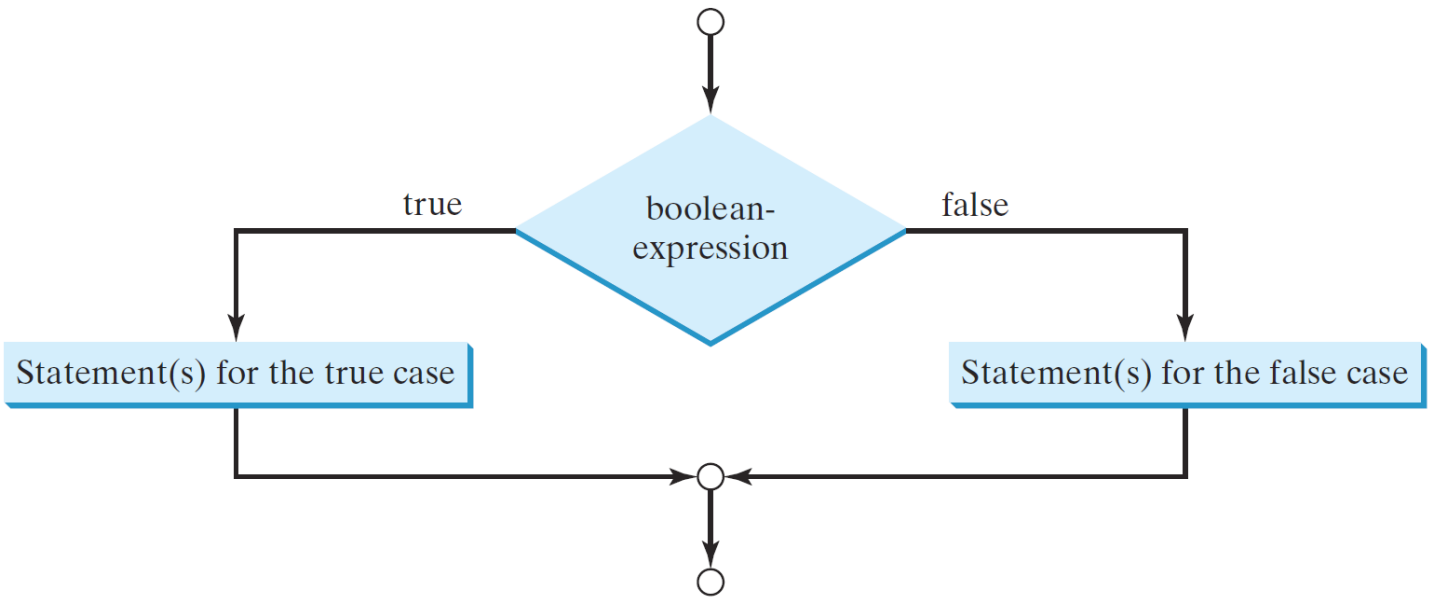
- Write a program which receives a positive number as the circle radius and outputs the resulting area.

```
1 ...
2         if (r > 0)
3         {
4             double A = r * r * 3.14;
5             Console.WriteLine(A);
6         }
7 ...
```

- What if the false case?

# The if-else Statements

```
1  ...
2          if (/* Condition: a boolean expression */)
3          {
4              // Body for the true case.
5          }
6          else
7          {
8              // Body for the false case.
9          }
10 ...
```



# Example: Circle Area (Revisited)

- Now add a conditional statement for the false case.

```
1  ...
2          if (r > 0)
3          {
4              double A = r * r * 3.14;
5              Console.WriteLine(A);
6          }
7          else
8          {
9              Console.WriteLine("Not a circle.");
10         }
11 ...
```

# Nested Conditional Statements

```
1  ...
2          if (score >= 90)
3              Console.WriteLine("A");
4          else
5          {
6              if (score >= 80)
7                  Console.WriteLine("B");
8              else
9              {
10                 if (score >= 70)
11                     Console.WriteLine("C");
12                 else
13                 {
14                     if (score >= 60)
15                         Console.WriteLine("D");
16                     else
17                         Console.WriteLine("F");
18                 }
19             }
20         }
21 ...
```

# Multiple Branches

```
1  ...
2          if (score >= 90)
3              Console.WriteLine("A");
4          else if (score >= 80)
5              Console.WriteLine("B");
6          else if (score >= 70)
7              Console.WriteLine("C");
8          else if (score >= 60)
9              Console.WriteLine("D");
10         else
11             Console.WriteLine("F");
12 ...
```

- Easier to read!
- We should avoid deep indentation.

- The alternative to the previous program looks like:

```
...
        if (score >= 90 && score <= 100)
            Console.WriteLine("A");
        else if (score >= 80 && score < 90)
            Console.WriteLine("B");
        else if (score >= 70 && score < 80)
        ...
...
```

- However, the order of conditions may be relevant. (Why?)
- The performance may degrade due to the order of conditions. (Why?)

# Common Bugs

```
1 ...
2         double A;
3         if (r > 0);
4             A = r * r * 3.14;
5             Console.WriteLine(A);
6 ...
```

- Don't add a semicolon to the condition (in Line 3).
  - If you do so in Line 3, this statement is not effective (useless).
- Multiple conditional statements should be grouped by braces.

# Example feat. Uncertainty

- Write a program which first shows a math question, say sum of two random integers ranging from 0 to 9, and asks the user to type an answer.
- How to generate random numbers?
  - We need first a (pseudo) random number generator[1] by asking for a **Random** object.[2]
  - Then invoke the method Next(10) for a random integer in the aforesaid range.
- For example, consider $2 + 5 =?$.
- If the input is correct, then report "Correct."; otherwise, report "Wrong. It is 7."

---

[1] See `https://en.wikipedia.org/wiki/Pseudorandom_number_generator`.

[2] See `https://docs.microsoft.com/en-us/dotnet/api/system.random.next?view=netcore-3.1`.

```
...
        // (1) Generate two random integers.
        Random rng = new Random();
        int x = rng.Next(10); // Ranging from 0 to 9.
        int y = rng.Next(10);

        // (2) Display the math question.
        Console.WriteLine("{0} + {1} = ?", x, y);

        // (3) Ask the user to type his/her answer.
        int g = int.Parse(Console.ReadLine());

        // (4) Judge the input.
        if (g == x + y)
        {
            Console.WriteLine("Correct.");
        }
        else
        {
            Console.WriteLine("Wrong. It is {0}.",  x + y);
        }
...
```

- Can you extend this program for all arithmetic operators $(+ - \times \div)$?

*"Exploring the unknown requires tolerating uncertainty."*

– Brian Greene

*"I can live with doubt, and uncertainty, and not knowing. I think it is much more interesting to live not knowing than have answers which might be wrong."*

– Richard Feynman

# Exercise

- Write a program which outputs the maximum value in 3 random integers ranging from $-50$ to 50. (Try.)
- Recall the first algorithm in the preliminary lecture.

```
1  ...
2          Random rng = new Random();
3          int x = rng.Next(-50, 51); // Why 51?
4          int y = rng.Next(-50, 51);
5          int z = rng.Next(-50, 51);
6
7          int max = x;
8          if (y > max) max = y;
9          if (z > max) max = z;
10         Console.WriteLine("max = {0}", max);
11 ...
```

- Note that the method Next() could take two int values, one for the lower bound of the random number (included), and the other for the upper bound (excluded).
- As a remark, this program is limited by the number of data.
- To develop a reusable solution, we need arrays and loops.

# The switch-case-break-default Statements

```
1  ...
2          switch (target)
3          {
4              case v1:
5                  // Conditional statements.
6                  break; // Leaving (jump to Line 16).
7              case v2:
8              .
9              .
10             .
11             case vk:
12                 // Conditional statements.
13                 break; // Leaving (jump to Line 16).
14             default:
15                 // Default statements.
16         }
17 ...
```

- The structure is convenient for finite and discrete conditions.
- The variable *target* must be a value of char, byte, short, int, or **String** type.
- The type of $v_1, \ldots,$ and $v_k$ must be identical to *target*.
- A break statement may be needed to leave the construct.[3]
- The default case is used to perform default actions when none of cases matches *target*.
  - This acts like the else statements.

[3] If not, there will be a fall-through behavior.

# Example

```
1  ...
2          string status = Console.Readline();
3
4          switch (status)
5          {
6              case "RED":
7                  Console.WriteLine("Stop!!!");
8                  break;
9              case "YELLOW":
10                 Console.WriteLine("Slow down!!");
11                 break;
12             case "GREEN":
13                 Console.WriteLine("Go!");
14                 break;
15             default:
16                 Console.WriteLine("Never happen?");
17         }
18 ...
```

# Conditional Operators

```
1  ...
2          if (num1 > num2)
3          {
4              max = num1;
5          }
6          else
7          {
8              max = num2;
9          }
10
11         // The above statement is equivalent to the following:
12         max = num1 > num2 ? num1 : num2;
13 ...
```

- If num1 $>$ num2, then do $\boxed{\text{max} = \text{num1}}$.
- Instead, do $\boxed{\text{max} = \text{num2}}$.

*"We must all face the choice between what is right and what is easy."*

– Prof. Albus Dumbledore,
*Harry Potter and the Goblet of Fire*, J.K. Rowling

*"To be or not to be, that is the question."*

– Prince Hamlet, *Hamlet*, William Shakespeare