Flow controls comparison

Overview

- Research the operation mechanism of flow controls.
- Compare the difference between the flow controls.
- Know when to use flow controls.

if – else VS switch

if – else VS switch

```
Console.Write("Please enter your point: ");
int PointValue = int.Parse(Console.ReadLine());
string RankName = "";

if (PointValue >= 9 && PointValue <= 10)
{
    RankName = "Xuat sac";
}
else if (PointValue >= 8 && PointValue < 9)
{
    RankName = "Gioi";
}
else if (PointValue >= 7 && PointValue < 8)
{
    RankName = "Kha";
}
else if (PointValue >= 5 && PointValue < 7)
{
    RankName = "Trung binh";
}
else
{
    RankName = "Yeu";
}
Console.WriteLine(RankName);
```


```
Console.Write("Please enter your point: ");
int PointValue = int.Parse(Console.ReadLine());
string RankName = "";
switch (PointValue) {
    case 1:
    case 2:
    case 3:
    case 4:
        RankName = "Yeu";
        break;
    case 5:
    case 6:
        RankName = "Trung Binh";
        break;
    case 7:
        RankName = "Kha";
        break;
    case 8:
        RankName = "Gioi";
        break;
    case 9:
    case 10:
        RankName = "Xuat sac";
        break;
    default:
        RankName = "Enter point form 0 to 10!";
        break;
}
Console.WriteLine(RankName);
```

if – else VS switch

1. Check the Testing Expression

An **if-else** statement can test expressions based on ranges of values or conditions,

whereas

A **switch** statement tests expressions based only on a single integer, enumerated value, or String object.

if – else VS switch

2. Switch better for Multi way branching

When compiler compiles a **switch** statement, it will inspect each of the case constants and create a “jump table” that it will use for selecting the path of execution depending on the value of the expression. Therefore, if we need to select among a large group of values, a switch statement will run much faster than the equivalent logic coded using a sequence of if-else.
The compiler can do this because **it knows that the case constants** are all the same type and simply must be compared for equality with the switch expression, while in case of if expressions, the compiler has no such knowledge.

if – else VS switch

3. if-else better for boolean values

If-else conditional branches are great for variable conditions that result into a boolean,

whereas

switch statements are great for fixed data values.

if – else VS switch

4. Speed

A **switch** statement might prove to be faster than ifs provided number of cases are good. If there are only few cases, it might not effect the speed in any case. Prefer switch if the number of cases are more than 5 otherwise, you may use if-else too.
If a switch contains more than **five items**, it's implemented using a lookup table or a hash list. This means that all items get the same access time, compared to a list of ifs where the last item takes much more time to reach as it has to evaluate every previous condition first.

if – else VS switch

5. Clarity in readability

A **switch** looks much cleaner when you have to combine cases. Ifs are quite vulnerable to errors too. Missing an else statement can land you up in havoc. Adding/removing labels is also easier with a switch and makes your code significantly easier to change and maintain.

if – else VS switch

6. Conversion

All switch statements can be convert to if-else

BUT

Not all if-else statements can be convert to switch

Give your decision

- Print day of week
- Print month of year
- Branching for check positive or negative number
- Student evaluation by final GPA

for VS foreach

for VS foreach

1. Direction

for: goes in any direction

whereas

foreach: goes in default direction

for VS foreach

2. Syntax

for: complexity in its syntax

whereas

foreach: simple and natural syntax

3. Performance

The foreach-loop is often less efficient than a simple for-loop

foreach loop creates a copy of the collection, so in this loop, the 'item' is not referencing to the array elements; it's just a temporary variable and you cannot assign a value to it.

foreach takes much time as compared to the 'for' loop because internally, it uses extra memory space, as well as, it uses GetEnumarator() and Next() methods of IEnumerables.

for VS foreach

```
static void Forloop()
{
    int[] list = { 1, 2, 3, 4, 5 };
    int len = list.Length;
    for (int i = 0; i < len; i++)
    {
        Console.WriteLine("FOR ITEM: " + ++list[i]);
    }
}
```

```
for items : 2
for items : 3
for items : 4
for items : 5
for items : 6
```

```
0 references
static void ForeachLoop()
{
    int[] list = { 1, 2, 3, 4, 5 };

    foreach (int value in list)
    {
        Console.WriteLine("FOREACH ITEM: " + ++value);
    }
}
```

(local variable) int value

Cannot assign to 'value' because it is a 'foreach iteration variable'

Time to discuss

- Why developers still use foreach?
- Can we set for/foreach loop endless?

Resolve problem

- Find and remove elements in a collection
  - Give list of products with: Name, Manufacture date, Expired Date.
  - Remove all products that Expired Date is in the past
- Should you use for OR foreach?
- Why and How?

for VS while

for VS while

1. Situation

for: when you know how many times the code/block needs to be in loop.
E.g. Fine a phone number in a contacts

whereas

while: it is unsure how many times the code should be in loop
E.g. Read all lines of the text file.

for VS while

2. Conditional

if statement should be include statement to change input of conditional operator.
It usually is not **depend** on result of body

while statement should be change input of conditional operator in code body.
It usually is **depend** on result of body

for VS while

3. Endless situation

Can be: when we don’t change input of conditional operator OR we re-set value of it

VS

Should be: when we don’t change input of conditional operator OR we re-set value of it

for VS while

4. Best practice

With while statement, always set condition to end loop.

```
bool isEndOfFile = false;
//// Set max line to read: 10k lines only
int maxLine = 10 * 1000;
int lineCount = 0;
while(!isEndOfFile && lineCount < maxLine)
{
    var lineContent = ReadLine(out isEndOfFile);
    lineCount += 1;
}
```

while VS do-while

while VS do-while

do-while statement executes the block of code at least **one**

“Tiền trảm hậu tấu”

additional

break vs continue

used to terminate the current loop iteration or terminate the switch statement in which it appears

skip the execution of current iteration only and it does not break the loop/switch

Summary

- ✓ Understand when use flow controls
- ✓ Comparison between pairs of flow controls
- ✓ Know how to use flow control to best performance
- ✓ Know how to convert between pairs of flow controls

Thank you