

# Използване на условни команди

Правилна организация  
на реда на изпълнението



Учителски екип

Обучение за ИТ кариера

<https://it-kariera.mon.bg/e-learning/>



# Съдържание

- Използване на условни команди
- Използване на цикли
- Други структури на реда на изпълнение



# Използване на условни команди

- Винаги ползвайте { и } за тялото на условната команда, дори когато е само на един ред:

```
if (condition)
{
    DoSomething();
}
```



- Пропускането на скобите може да навреди

```
if (condition)
    DoSomething();
    DoAnotherThing();
    DoDifferentThing();
```



- Това е подвеждащ код + подвеждащо форматиране

## Използване на условни команди (2)

- Винаги слагайте обичайното (оачкваното) условие **първо** след **if**-частта

```
var response = GetHttpWebResponse();  
if (response.Code == Code.NotFound)  
{  
    // ...  
}  
else if (response.Code == Code.OK)  
{  
    // ...  
}
```



```
var response = GetHttpWebResponse();  
if (response.Code == Code.OK)  
{  
    // ...  
}  
else if (response.Code == Code.NotFound)  
{  
    // ...  
}
```



- Започнете от **най-често срещаните случаи**, после продължете към по-малко вероятните



# Използване на условни команди (3)

- Избягвайте сравненията с **true** или **false**:

```
if (HasErrors == true)
{
    ...
}
```



```
if (HasErrors)
{
    ...
}
```



- Винаги мислете и за else частта
  - Ако трябва, обяснете защо не е необходима

```
if (parserState != States.Finished)
{
    // ...
}
else
{
    // Ignore all content once the parser has finished
}
```



# Използване на условни команди (4)

- Избягвайте двойното отрицание

```
if (!HasNoError)
{
    DoSomething();
}
```



```
if (HasErrors)
{
    DoSomething();
}
```



- Пишете **if**-частта със смислена команда

```
if (!HasError)
;
else
{
    DoSomething();
}
```



```
if (HasErrors)
{
    DoSomething();
}
```




- Използвайте смислени булеви изрази, звучащи като изречение

# Използване на условни команди (5)


- Внимавайте за сору/paste проблеми в **if-else** телата

```
if (SomeCondition)
{
    var p = GetSomePerson();
    p.SendMail();
    p.SendSms();
}
else
{
    var p = GetOtherPerson();
    p.SendMail();
    p.SendSms();
}
```



```
Person p = null;
if (SomeCondition)
{
    p = GetSomePerson();
}
else
{
    p = GetOtherPerson();
}

p.SendMail();
p.SendSms();
```



# Используйте простые условия

- Не пользуйтесь сложными `if`-условиями
  - Винаги може да ги опростите чрез булеви променливи или булеви методи
  - Лош пример:

```
if (x > 0 && y > 0 && x < Width-1 && y < Height-1 &&  
    matrix[x, y] == 0 && matrix[x-1, y] == 0 &&  
    matrix[x+1, y] == 0 && matrix[x, y-1] == 0 &&  
    matrix[x, y+1] == 0 && !visited[x, y]) ...
```




- Сложните булеви изрази могат да навредят
- Как ще намерите проблема, ако получите **`IndexOutOfRangeException`**?



# Опростяване на булеви условия

- Последният пример може лесно да се промени в **самоописателен** код:

```
bool inRange = x > 0 && y > 0 && x < Width-1 && y < Height-1;
if (inRange)
{
    bool emptyCellAndNeighbours =
        matrix[x, y] == 0 && matrix[x-1, y] == 0 &&
        matrix[x+1, y] == 0 && matrix[x, y-1] == 0 &&
        matrix[x, y+1] == 0;
    if (emptyCellAndNeighbours && !visited[x, y]) ...
}
```



- Сега кодът е:
  - Лесен за четене** – логиката на условието е ясна
  - Лесен за дебъгване** – на **if**-частта може да се сложи точка за прекъсване

## Опростяване на булеви условия (2)

- Използвайте обектно-ориентиран подход

```
public class Maze
{
    public Cell CurrentCell { get; set; }
    public IList<Cell> VisitedCells { get; }
    public IList<Cell> NeighbourCells { get; }
    public Size Size { get; }

    public bool IsCurrentCellInRange()
    {
        return this.Size.Contains(this.CurrentCell);
    }

    public bool IsCurrentCellVisited()
    {
        return this.VisitedCells.Contains(this.CurrentCell);
    }
}
```

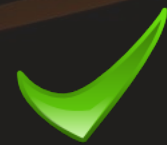


*(continues on the next slide)*

# Опростяване на булеви условия (3)

```
public bool AreNeighbourCellsEmpty()
{
    ...
}

public bool ShouldVisitCurrentCell()
{
    return
        this.IsCurrentCellInRange() &&
        this.CurrentCell.IsEmpty() &&
        this.AreNeighbourCellsEmpty() &&
        !this.IsCurrentCellVisited()
}
}
```



- Сега кодът:
  - Отразява реалния сценарий
  - Остава близо до областта на проблема

# Используйте таблица на решенията (Decision Table)

- Понякога за простота може да ползвате таблица за решения

```
var table = new Hashtable();  
table.Add("A", new AWorker());  
table.Add("B", new BWorker());  
table.Add("C", new CWorker());  
  
string key = GetWorkerKey();  
  
var worker = table[key];  
if (worker != null)  
{  
    ...  
    worker.Work();  
    ...  
}
```





# Положителни булеви изрази

- Започването с **положителен израз** подобрява четливостта

```
if (IsValid)
{
    DoSomething();
}
else
{
    DoSomethingElse();
}
```



```
if (!IsValid)
{
    DoSomethingElse();
}
else
{
    DoSomething();
}
```



- Ползвайте законите на ДеМорган за отрицателни проверки

```
if (!IsValid || !IsVisible)
```



```
if (!(IsValid && IsVisible))
```

# Опростете със скоби

- Избягвайте сложни булеви условия без скоби

```
if (a < b && b < c || c == d)
```



- Употребата на скоби подобрява четливостта и подsigурява верността на изчислението

```
if ((a < b && b < c) || c == d)
```



- Твърде много скоби обаче също трябва да се избягват
  - В такива случаи обмислете отделни булеви методи или променливи

# Пресмятане на булевите изрази

- В повечето езици изразите се пресмята отляво надясно
  - Спира се пресмятането веднага щом булевата операция е с ясна стойност

```
if (FalseCondition &&OtherCondition) = false
```

```
if (TrueCondition ||OtherCondition) = true
```

- Някои езици не следват това „правило за бързо пресмятане“
- Удобно е при проверка за **null**

```
if (list != null && list.Count > 0) ...
```
- TODO: проверете дали индекса е в дадения обхват преди проверката на стойността му
- **НЕ** извиквайте методи в if-условие или инициализация на цикъл for

# Числови изрази като операнди

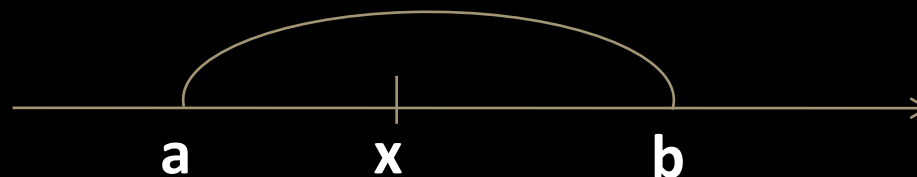
- Пишете числовите булеви изрази така, както стоят на числовата ос

- В даден интервал

```
if (x > a && b > x)
```



```
if (a < x && x < b)
```



- Извън даден интервал

```
if (a > x || x > b)
```



```
if (x < a || b < x)
```





# Избягвайте многократното влагане на блокове

- Многократното влагане на условни команди и цикли прави кода неясен
  - Повече от 2-3 нива е твърде много
  - Многократно вложения код е сложен и труден за четене и разбиране
  - Обикновено може да преместите части от кода в отделни методи
    - Това опростява логиката на кода
    - Добрите имена на методи правят кода самоописателен

# Многократно влагане – пример

```
if (maxElem != Int32.MaxValue)
{
    if (arr[i] < arr[i + 1])
    {
        if (arr[i + 1] < arr[i + 2])
        {
            if (arr[i + 2] < arr[i + 3])
            {
                maxElem = arr[i + 3];
            }
            else
            {
                maxElem = arr[i + 2];
            }
        }
        else
        {
            if (arr[i + 1] < arr[i + 3])
            {
                maxElem = arr[i + 3];
            }
            else
            {
                maxElem = arr[i + 1];
            }
        }
    }
}
```



(продължава на другия слайд)

# Многократно влагане – пример (2)

```
else
{
    if (arr[i] < arr[i + 2])
    {
        if (arr[i + 2] < arr[i + 3])
        {
            maxElem = arr[i + 3];
        }
        else
        {
            maxElem = arr[i + 2];
        }
    }
    else
    {
        if (arr[i] < arr[i + 3])
        {
            maxElem = arr[i + 3];
        }
        else
        {
            maxElem = arr[i];
        }
    }
}
}
```



# Избягване на многократно влягане – пример

```
private static int Max(int i, int j)
{
    if (i < j)
    {
        return j;
    }
    else
    {
        return i;
    }
}

private static int Max(int i, int j, int k)
{
    if (i < j)
    {
        int maxElem = Max(j, k);
        return maxElem;
    }
    else
    {
        int maxElem = Max(i, k);
        return maxElem;
    }
}
```



*(продължава на другия слайд)*



# Избягване на многократно влагане – пример (2)

```
private static int FindMax(int[] arr, int i)
{
    if (arr[i] < arr[i + 1])
    {
        int maxElem = Max(arr[i + 1], arr[i + 2], arr[i + 3]);
        return maxElem;
    }
    else
    {
        int maxElem = Max(arr[i], arr[i + 2], arr[i + 3]);
        return maxElem;
    }
}

if (maxElem != Int32.MaxValue) {
    maxElem = FindMax(arr, i);
}
```



# Използване на команда за избор на вариант (Case)

- Изберете най-ефективната подредба на случаите
  - Сложете нормалният (обичайният) случай пръв
  - Подредете случаите по вероятност да са верни
  - Сложете най-необичайният (извънреден) случай последен
  - Подредете случаите по азбучен ред или по номера
- Нека действията за всеки отделен случай да са прости
  - Сложни логически изрази извадете в отделни методи
- За прихващане на грешки използвайте default-частта в командата **case** или последният **else** в **if-else** поредицата

# Невярна команда за избор на вариант (Case)

```
void ProcessNextChar(char ch)
{
    switch (parseState)
    {
        case InTag:
            if (ch == ">")
            {
                Console.WriteLine("Found tag: {0}", tag);
                text = "";
                parseState = ParseState.OutOfTag;
            }
            else
            {
                tag = tag + ch;
            }
            break;
        case OutOfTag:
            ...
    }
}
```



# Подобрена команда за избор на вариант (Case)

```
void ProcessNextChar(char ch)
{
    switch (parseState)
    {
        case InTag:
            ProcessCharacterInTag(ch);
            break;
        case OutOfTag:
            ProcessCharacterOutOfTag(ch);
            break;
        default:
            throw new InvalidOperationException(
                "Invalid parse state: " + parseState);
    }
}
```





# Команда Case – най-добри практики

- Избягвайте пропадания (като в примера отдолу) и goto case
- Когато все пак ги ползвате, документирайте ги добре

```
switch (c)
{
    case 1:
    case 2:
        DoSomething();
        // FALLTHROUGH
    case 17:
        DoSomethingElse();
        break;
    case 5:
    case 43:
        DoOtherThings();
        break;
}
```



# Обобщение

- За прости **if-else** случаи внимавайте за реда на **if** и **else**-частите
  - Уверете се, че номиналният случай е ясен
- За поредни **if-then-else** и **case** команди от тип изберете най-четливата поредност на разглеждане на случаите
- Опростете булевите изрази, за да подобрите четливостта
- За да прихванете грешки използвайте **default** частта на командата **case** или последният **else** от поредица **if**-ове



# Използване на условни команди



Въпроси?



# Министерство на образованието и науката (МОН)

- Настоящият курс (презентации, примери, задачи, упражнения и др.) е разработен за нуждите на Национална програма **"Обучение за ИТ кариера"** на МОН за подготовка по професия "Приложен програмист"



Министерство  
на образованието  
и науката



Национална  
програма  
„Обучение за  
ИТ кариера“

- Курсът е базиран на учебно съдържание и методика, предоставени от **фондация "Софтуерен университет"** и се разпространява под свободен лиценз **CC-BY-NC-SA**



SoftUni  
Foundation

