## PHT C# [03] - ĐIỀU KIỆN VÀ VÒNG LẶP

### 🎯 KIẾN THỨC CỐT LỐI

| **Cấu Trúc** | **Cú Pháp** | **Khi Dùng** |
| --- | --- | --- |
| **if** | if (condition) { } | Kiểm tra điều kiện đúng/sai |
| **if-else** | if (c1) { } else { } | Có 2 lựa chọn |
| **if-else if-else** | if (c1) { } else if (c2) { } else { } | Nhiều lựa chọn |
| **switch** | switch(x) { case 1: break; } | Kiểm tra giá trị cụ thể |
| **for** | for(i=0; i<n; i++) { } | Lặp với số lần xác định |
| **while** | while(condition) { } | Lặp cho đến khi điều kiện sai |
| **do-while** | do { } while(condition); | Lặp ít nhất 1 lần |
| **foreach** | foreach(var item in array) { } | Lặp qua mảng |
| **break** | Thoát vòng lặp/switch ngay | Dừng sớm |
| **continue** | Bỏ qua phần còn lại, lần lặp kế tiếp | Bỏ qua những trường hợp nhất định |

### 💡 KHÁM PHÁ KIẾN THỨC

**Câu hỏi tự kiểm tra:**

1. Toán tử so sánh là gì? (==, !=, <, >, <=, >=)
   * dùng để so sánh 2 giá trị, kết quả trả về true/false (==, !=, <, >, <=, >=).
2. Toán tử logic là gì? (&&, ||, !)
   * kết hợp nhiều điều kiện (&&: và, ||: hoặc, !: phủ định).
3. Sự khác biệt giữa for và foreach là gì?
   * **for vs foreach**:

for: dùng khi biết số lần lặp, cần chỉ số.

foreach: dùng để duyệt mảng/danh sách, không cần chỉ số.

1. Khi nào dùng break? Khi nào dùng continue?
   * **break**: thoát vòng lặp ngay.  
     **continue**: bỏ qua lần lặp hiện tại, sang lần tiếp theo.

### 📝 VÍ DỤ THỰC HÀNH

**TODO 3.1: [PHÂN LOẠI ĐIỂM - IF/ELSE IF/ELSE]**

if (score >= 90)

Console.WriteLine("A (Xuất sắc)");

else if (score >= 80)

Console.WriteLine("B (Khá)");

else if (score >= 70)

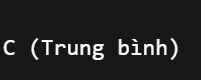
Console.WriteLine("C (Trung bình)");

else if (score >= 60)

Console.WriteLine("D (Yếu)");

else

Console.WriteLine("F (Không đạt)");



**TODO 3.2: [SWITCH - CHỌN NGÀY TRONG TUẦN]**

switch (day)

{

case 1: Console.WriteLine("Thứ Hai"); break;

case 2: Console.WriteLine("Thứ Ba"); break;

case 3: Console.WriteLine("Thứ Tư"); break;

case 4: Console.WriteLine("Thứ Năm"); break;

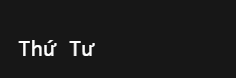
case 5: Console.WriteLine("Thứ Sáu"); break;

case 6: Console.WriteLine("Thứ Bảy"); break;

case 7: Console.WriteLine("Chủ Nhật"); break;

default: Console.WriteLine("Ngày không hợp lệ"); break;

}



**TODO 3.3: [VÒNG LẶP FOR - TÍNH TỔNG]**

int sum = 0;

for (int i = 1; i <= 100; i++)

{

sum += i;

}

Console.WriteLine("Tổng các số từ 1 đến 100: " + sum);

Ảnh có chứa Phông chữ, Đồ họa, văn bản, ảnh chụp màn hình

Nội dung do AI tạo ra có thể không chính xác.

**TODO 3.4: [VÒNG LẶP WHILE - ĐOÁN SỐ]**

int secret = 50;

int[] guesses = { 40, 60, 50 };

int i = 0;

while (i < guesses.Length)

{

if (guesses[i] < secret)

Console.WriteLine("Quá thấp");

else if (guesses[i] > secret)

Console.WriteLine("Quá cao");

else

{

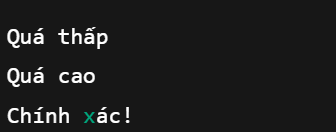
Console.WriteLine("Chính xác!");

break;

}

i++;

}



**TODO 3.5: [FOREACH - IN DANH SÁCH]**

int index = 1;

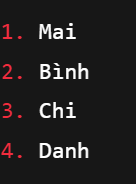
foreach (string name in friends)

{

Console.WriteLine(index + ". " + name);

index++;

}



**TODO 3.6: [BREAK & CONTINUE - TÌM SỐ]**

// In số lẻ

for (int i = 1; i <= 20; i++)

{

if (i % 2 == 0) continue;

Console.Write(i + " ");

}

Console.WriteLine();

// Tìm số 7

int[] arr = { 2, 5, 7, 1, 9, 7, 3 };

for (int i = 0; i < arr.Length; i++)

{

if (arr[i] == 7)

{

Console.WriteLine("Tìm thấy số 7 tại vị trí " + i);

break;

}

}

### 🤔 CÂU HỎI PHẢN BIỆN

1. **Tại sao vòng lặp for lại được dùng nhiều hơn while trong lập trình?**
   * **for dùng nhiều hơn while** vì rõ ràng số lần lặp, code dễ đọc.
2. **Nếu bạn có vòng lặp for lồng trong for, làm cách nào để thoát cả hai vòng lặp cùng lúc?**
   * Thoát 2 vòng for: dùng **break + biến cờ** hoặc **goto**.
3. **So sánh while và do-while - khi nào nên dùng do-while?**
   * **do-while** dùng khi **bắt buộc chạy ít nhất 1 lần** (ví dụ nhập dữ liệu)

### 💫 KẾT NỐI ĐÁNH GIÁ

**Mốc kiểm tra quan trọng:** Bạn phải **thành thạo if/else/switch** và **for/while/foreach** để học phần tiếp theo

* + Để học tốt các phần tiếp theo, người học cần **thành thạo các cấu trúc điều kiện và vòng lặp cơ bản trong C#**. Cụ thể, phải biết sử dụng đúng **if/else, switch** để xử lý điều kiện và **for, while, foreach** để lặp dữ liệu. Đây là nền tảng quan trọng giúp giải quyết bài toán logic, xử lý dữ liệu và phát triển chương trình hiệu quả ở các nội dung nâng cao sau này.

## 

## PHT C# [04] - PHƯƠNG THỨC (FUNCTION/METHOD)

### 🎯 KIẾN THỨC CỐT LỐI

| **Khái Niệm** | **Ý Nghĩa** | **Ví Dụ** |
| --- | --- | --- |
| **Phương thức** | Khối code có tên, có thể tái sử dụng | static int Add(int a, int b) |
| **Tham số (Parameters)** | Biến đầu vào của phương thức | Add(int a, int b) → a, b là tham số |
| **Đối số (Arguments)** | Giá trị cụ thể truyền vào | Add(5, 3) → 5, 3 là đối số |
| **Kiểu trả về** | Kiểu dữ liệu phương thức trả về | int, string, void (không trả về) |
| **Return** | Câu lệnh trả về giá trị | return sum; |
| **Static** | Phương thức thuộc lớp, gọi trực tiếp | Program.Add(5, 3) |
| **Nạp chồng (Overloading)** | Nhiều phương thức cùng tên, tham số khác | Add(int, int) và Add(double, double) |
| **Đệ quy (Recursion)** | Phương thức gọi chính nó | Factorial(n) = n \* Factorial(n-1) |

### 💡 KHÁM PHÁ KIẾN THỨC

**Câu hỏi tự kiểm tra:**

1. **Dùng phương thức** để tái sử dụng code, dễ bảo trì, chương trình rõ ràng.

2,**Tham số**: biến trong khai báo hàm. **Đối số**: giá trị truyền khi gọi hàm.

3. Phương thức **có kiểu trả về ≠ void** thì **bắt buộc return**.  
void thì **không cần return**.

4. **Nạp chồng** dựa vào **số lượng, kiểu tham số** (không dựa vào kiểu trả về).

### 📝 VÍ DỤ THỰC HÀNH

**TODO 4.1: [PHƯƠNG THỨC TÍNH TOÁN CỐ BẢN]**

static int Add(int a, int b)

{

return a + b;

}

static double Multiply(double x, double y)

{

return x \* y;

}

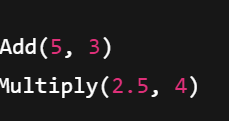
static void Main(string[] args)

{

Console.WriteLine(Add(5, 3));

Console.WriteLine(Multiply(2.5, 4));

}



Ảnh có chứa ảnh chụp màn hình, Phông chữ, màu đen, văn bản

Nội dung do AI tạo ra có thể không chính xác.

**TODO 4.2: [PHƯƠNG THỨC VOID - KHÔNG TRẢ VỀ]**

static void PrintBox(string text)

{

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine("\* " + text + " \*");

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*");

}

static void Main(string[] args)

{

PrintBox("Hello");

PrintBox("C#");

}

**TODO 4.3: [PHƯƠNG THỨC VỚI MẢNG]**

static int SumArray(int[] numbers)

{

int sum = 0;

foreach (int n in numbers)

sum += n;

return sum;

}

static int FindMax(int[] numbers)

{

int max = numbers[0];

foreach (int n in numbers)

if (n > max) max = n;

return max;

}

static void Main(string[] args)

{

int[] scores = { 85, 92, 78, 90, 88 };

Console.WriteLine(SumArray(scores));

Console.WriteLine(FindMax(scores));

}

Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình, màu đen

Nội dung do AI tạo ra có thể không chính xác.

**TODO 4.4: [NẠP CHỒNG PHƯƠNG THỨC]**

static void Print(int x)

{

Console.WriteLine(x);

}

static void Print(string text)

{

Console.WriteLine(text);

}

static int Add(int a, int b)

{

return a + b;

}

static double Add(double a, double b)

{

return a + b;

}

static void Main(string[] args)

{

Print(10);

Print("Hello");

Console.WriteLine(Add(3, 4));

Console.WriteLine(Add(2.5, 3.5));

}

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, màu đen

Nội dung do AI tạo ra có thể không chính xác.

**TODO 4.5: [ĐỆ QUY - TÍNH GIAI THỪA]**

static long Factorial(int n)

{

if (n == 0) return 1;

return n \* Factorial(n - 1);

}

static void Main(string[] args)

{

Console.WriteLine(Factorial(5));

Console.WriteLine(Factorial(10));

}

Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình, màu trắng

Nội dung do AI tạo ra có thể không chính xác.

### 🤔 CÂU HỎI PHẢN BIỆN

1. **Tại sao cần nạp chồng phương thức? Không thể dùng tên khác nhau cho từng trường hợp không?**
   * **Nạp chồng** giúp code dễ đọc, thống nhất chức năng; không cần nhớ nhiều tên hàm.
2. **Đệ quy có ưu và nhược điểm gì?**

Ưu: code ngắn, dễ hiểu bài toán.

Nhược: tốn bộ nhớ, dễ tràn stack.

* + Khi nào nên dùng đệ quy?

+ khi bài toán có cấu trúc lặp lại (giai thừa, cây)

* + Khi nào nên dùng vòng lặp?

+ khi cần hiệu năng

1. **Phương thức nên ngắn hay dài? Tại sao?**
   * **Phương thức nên ngắn**, làm **1 nhiệm vụ**, dễ kiểm tra và tái sử dụng.-

### 💫 KẾT NỐI ĐÁNH GIÁ

* + Nắm vững **phương thức, tham số, return, nạp chồng và đệ quy** là nền tảng bắt buộc trước khi học **OOP (Phiếu 05)** – bước chuyển quan trọng từ viết code đơn lẻ sang xây dựng chương trình có cấu trúc.

## PHT C# [06] - XỬ LÝ LỖI VÀ DỰ ÁN TỔNG HỢP

### 🎯 KIẾN THỨC CỐT LỐI

| **Khái Niệm** | **Ý Nghĩa** | **Ví Dụ** |
| --- | --- | --- |
| **Exception** | Lỗi xảy ra lúc chạy chương trình | DivideByZeroException |
| **try-catch** | Bắt lỗi và xử lý | try { } catch { } |
| **finally** | Code luôn chạy, dù có lỗi hay không | Đóng file, giải phóng tài nguyên |
| **Validation** | Kiểm tra dữ liệu đầu vào | Kiểm tra age > 0 |
| **throw** | Ném ra một exception | throw new Exception("Lỗi"); |

### 📝 DỰ ÁN TỔNG HỢP: HỆ THỐNG QUẢN LÝ SINH VIÊN

**Yêu cầu chức năng:**

✅ 1. Danh sách sinh viên (tối đa 50)

✅ 2. Thêm sinh viên mới (có validation)

✅ 3. Xóa sinh viên theo ID

✅ 4. Cập nhật điểm cho sinh viên

✅ 5. Tính điểm trung bình

✅ 6. Tìm điểm cao nhất / thấp nhất

✅ 7. Tìm sinh viên theo ID

✅ 8. Xử lý lỗi (try-catch)

**TODO 6.1: [TẠO LỚP STUDENT]**

public class Student

{

public string StudentId { get; set; }

public string Name { get; set; }

public double Score { get; set; }

public Student(string id, string name, double score)

{

if (string.IsNullOrEmpty(id))

throw new Exception("ID không được rỗng");

if (string.IsNullOrEmpty(name))

throw new Exception("Tên không được rỗng");

if (score < 0 || score > 10)

throw new Exception("Điểm phải từ 0 đến 10");

StudentId = id;

Name = name;

Score = score;

}

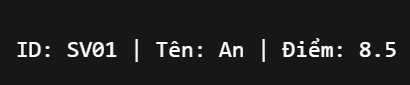
public void Display()

{

Console.WriteLine($"ID: {StudentId} | Tên: {Name} | Điểm: {Score}");

}

}



**TODO 6.2: [TẠO LỚP STUDENT MANAGER]**

public class StudentManager

{

private Student[] students = new Student[50];

private int count = 0;

public void AddStudent(string id, string name, double score)

{

if (count >= 50)

throw new Exception("Danh sách đầy");

if (FindStudentById(id) != null)

throw new Exception("Trùng ID");

students[count++] = new Student(id, name, score);

}

public void RemoveStudent(string id)

{

for (int i = 0; i < count; i++)

{

if (students[i].StudentId == id)

{

for (int j = i; j < count - 1; j++)

students[j] = students[j + 1];

count--;

return;

}

}

throw new Exception("Không tìm thấy sinh viên");

}

public void UpdateScore(string id, double newScore)

{

Student s = FindStudentById(id);

if (s == null) throw new Exception("Không tìm thấy sinh viên");

if (newScore < 0 || newScore > 10)

throw new Exception("Điểm không hợp lệ");

s.Score = newScore;

}

public double GetAverageScore()

{

double sum = 0;

for (int i = 0; i < count; i++)

sum += students[i].Score;

return count == 0 ? 0 : sum / count;

}

public double GetMaxScore()

{

double max = students[0].Score;

for (int i = 1; i < count; i++)

if (students[i].Score > max)

max = students[i].Score;

return max;

}

public Student FindStudentById(string id)

{

for (int i = 0; i < count; i++)

if (students[i].StudentId == id)

return students[i];

return null;

}

public void DisplayAllStudents()

{

for (int i = 0; i < count; i++)

students[i].Display();

}

}

Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình, đồng hồ

Nội dung do AI tạo ra có thể không chính xác.

**TODO 6.3: [MENU CHÍNH - MAIN PROGRAM]**

static void Main(string[] args)

{

StudentManager manager = new StudentManager();

bool running = true;

while (running)

{

Console.WriteLine("\n1.Thêm 2.Xóa 3.Cập nhật 4.In DS 5.Điểm TB 6.Max 7.Tìm 0.Thoát");

Console.Write("Chọn: ");

try

{

int choice = int.Parse(Console.ReadLine());

switch (choice)

{

case 1:

Console.Write("ID: ");

string id = Console.ReadLine();

Console.Write("Tên: ");

string name = Console.ReadLine();

Console.Write("Điểm: ");

double score = double.Parse(Console.ReadLine());

manager.AddStudent(id, name, score);

break;

case 2:

Console.Write("Nhập ID: ");

manager.RemoveStudent(Console.ReadLine());

break;

case 3:

Console.Write("ID: ");

string uid = Console.ReadLine();

Console.Write("Điểm mới: ");

double ns = double.Parse(Console.ReadLine());

manager.UpdateScore(uid, ns);

break;

case 4:

manager.DisplayAllStudents();

break;

case 5:

Console.WriteLine("Điểm TB: " + manager.GetAverageScore());

break;

case 6:

Console.WriteLine("Điểm cao nhất: " + manager.GetMaxScore());

break;

case 7:

Console.Write("ID: ");

Student s = manager.FindStudentById(Console.ReadLine());

if (s != null) s.Display();

else Console.WriteLine("Không tìm thấy");

break;

case 0:

running = false;

break;

}

}

catch (Exception ex)

{

Console.WriteLine("Lỗi: " + ex.Message);

}

}

}

**📸 CHỨNG THỰC DỰ ÁN:**

* Chụp screenshot menu
* Chụp screenshot các chức năng: thêm, xóa, cập nhật, hiển thị
* Chụp screenshot khi có lỗi (input sai), kiểm tra xử lý lỗi
* File source code hoàn chỉnh

### 🤔 CÂU HỎI PHẢN BIỆN

1. **Nếu người dùng nhập dữ liệu sai (ví dụ: điểm là 15), chương trình nên xử lý như thế nào?**
   * **Nhập điểm sai (15)** → validation + throw Exception, bắt bằng try-catch.
2. \*\*Có thể dùng Listthay vì array không? Ưu điểm gì?\*\*
   * Dễ thêm/xóa, không giới hạn kích thước, code gọn hơn.
3. **Nếu muốn lưu dữ liệu xuống file sau khi chương trình thoát, làm sao?**
   * dùng StreamWriter, File.WriteAllText, hoặc JSON.

### 💫 KẾT NỐI ĐÁNH GIÁ

Hoàn thành project này nghĩa là bạn đã:

* Hiểu **OOP + Exception + Validation**
* Sẵn sàng học **ASP.NET Core Backend**
* Chuẩn bị cho **Interface, Abstract, LINQ, Web API**