## 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? (==, !=, <, >, <=, >=)

Toán tử so sánh dùng để so sánh hai giá trị, trả về kết quả đúng (true) hoặc sai (false).

== (bằng): Kiểm tra hai giá trị có bằng nhau không

Ví dụ: 5 == 5 → true

!= (khác): Kiểm tra hai giá trị có khác nhau không

Ví dụ: 5 != 3 → true

< (nhỏ hơn): Kiểm tra giá trị bên trái có nhỏ hơn giá trị bên phải không

Ví dụ: 3 < 5 → true

> (lớn hơn): Kiểm tra giá trị bên trái có lớn hơn giá trị bên phải không

Ví dụ: 10 > 7 → true

<= (nhỏ hơn hoặc bằng): Kiểm tra giá trị bên trái có nhỏ hơn hoặc bằng giá trị bên phải không

Ví dụ: 5 <= 5 → true

>= (lớn hơn hoặc bằng): Kiểm tra giá trị bên trái có lớn hơn hoặc bằng giá trị bên phải không

Ví dụ: 8 >= 6 → true

1. Toán tử logic là gì? (&&, ||, !)

Toán tử logic dùng để kết hợp nhiều điều kiện hoặc đảo ngược kết quả của điều kiện.

&& (AND - và): Trả về true nếu cả hai điều kiện đều đúng

Ví dụ: (tuoi > 18 && coCMND == true) → chỉ đúng khi đủ tuổi VÀ có CMND

|| (OR - hoặc): Trả về true nếu ít nhất một điều kiện đúng

Ví dụ: (diemToan >= 8 || diemVan >= 8) → đúng nếu Toán hoặc Văn >= 8

! (NOT - phủ định): Đảo ngược kết quả của điều kiện

Ví dụ: !(diem < 5) → đúng khi điểm KHÔNG nhỏ hơn 5 (tức là >= 5)

1. Sự khác biệt giữa for và foreach là gì?

|  |  |
| --- | --- |
| for | foreach |
| Cần biết số lần lặp trước | Tự động lặp qua tất cả phần tử trong tập hợp |
| Có thể điều khiển bước nhảy (i++, i+=2) | Luôn lặp qua từng phần tử một |
| Có thể truy cập bằng chỉ số (i) | Chỉ truy cập được giá trị, không có chỉ số |
| Có thể dùng để thay đổi giá trị phần tử trong mảng | Thường chỉ dùng để đọc giá trị |
| Dùng được cho nhiều mục đích khác nhau | Chuyên dùng cho các tập hợp (mảng, danh sách) |

1. Khi nào dùng break? Khi nào dùng continue?

Dùng break khi:

Đã tìm thấy kết quả cần tìm và muốn dừng hẳn vòng lặp

Gặp điều kiện đặc biệt cần thoát khỏi vòng lặp ngay lập tức

Trong câu lệnh switch để kết thúc mỗi trường hợp

Dùng continue khi:

Muốn bỏ qua bước lặp hiện tại và chuyển sang bước lặp tiếp theo

Gặp điều kiện không cần xử lý trong bước lặp hiện tại

### 📝 VÍ DỤ THỰC HÀNH

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

using System;

namespace GradeClassification

{

class Program

{

static void Main(string[] args)

{

// TODO: Nhập điểm (giả sử: 75)

int score = 75; // Thay đổi giá trị để test khác

// TODO: Phân loại điểm

// 90-100: A (Xuất sắc)

// 80-89: B (Khá)

// 70-79: C (Trung bình)

// 60-69: D (Yếu)

// < 60: F (Không đạt)

string grade = "";

string evaluation = "";

if (score >= 90 && score <= 100)

{

grade = "A";

evaluation = "Xuất sắc";

}

else if (score >= 80 && score <= 89)

{

grade = "B";

evaluation = "Khá";

}

else if (score >= 70 && score <= 79)

{

grade = "C";

evaluation = "Trung bình";

}

else if (score >= 60 && score <= 69)

{

grade = "D";

evaluation = "Yếu";

}

else if (score >= 0 && score < 60)

{

grade = "F";

evaluation = "Không đạt";

}

else

{

grade = "Không hợp lệ";

evaluation = "Điểm phải từ 0-100";

}

// TODO: In kết quả

Console.WriteLine($"Điểm: {score}");

Console.WriteLine($"Xếp loại: {grade}");

Console.WriteLine($"Đánh giá: {evaluation}");

}

}

}

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

using System;

namespace DayOfWeek

{

class Program

{

static void Main(string[] args)

{

// TODO: Nhập số ngày (1-7)

int day = 3; // Thứ Tư

// TODO: Dùng switch để in tên ngày

// 1: Thứ Hai

// 2: Thứ Ba

// ...

// 7: Chủ Nhật

// default: Ngày không hợp lệ

string dayName;

switch (day)

{

case 1:

dayName = "Thứ Hai";

break;

case 2:

dayName = "Thứ Ba";

break;

case 3:

dayName = "Thứ Tư";

break;

case 4:

dayName = "Thứ Năm";

break;

case 5:

dayName = "Thứ Sáu";

break;

case 6:

dayName = "Thứ Bảy";

break;

case 7:

dayName = "Chủ Nhật";

break;

default:

dayName = "Ngày không hợp lệ";

break;

}

Console.WriteLine($"Số ngày: {day}");

Console.WriteLine($"Kết quả: {dayName}");

}

}

}

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

using System;

namespace SumCalculation

{

class Program

{

static void Main(string[] args)

{

// TODO: Tính tổng các số từ 1 đến 100

// Gợi ý: Dùng vòng lặp for

int sum = 0;

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

{

sum += i; // Hoặc sum = sum + i;

}

// TODO: In kết quả

// "Tổng các số từ 1 đến 100: [kết quả]"

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

}

}

}

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

using System;

namespace GuessNumber

{

class Program

{

static void Main(string[] args)

{

// TODO: Tạo chương trình đoán số

// - Số bí mật là 50 (cố định)

// - Người chơi đoán (nhập vào 3 lần: 40, 60, 50)

// - Mỗi lần đoán, chương trình báo "Quá thấp" hoặc "Quá cao"

// - Khi đoán đúng, báo "Chính xác!" và thoát

int secretNumber = 50;

bool isCorrect = false;

int guessCount = 0;

int maxGuesses = 3;

Console.WriteLine("Chào mừng đến với trò chơi ĐOÁN SỐ!");

Console.WriteLine($"Bạn có {maxGuesses} lần đoán. Số bí mật nằm trong khoảng 1-100.");

while (!isCorrect && guessCount < maxGuesses)

{

guessCount++;

Console.Write($"\nLần đoán thứ {guessCount}: ");

// Nhập số từ người dùng

int guess;

bool isValidInput = int.TryParse(Console.ReadLine(), out guess);

if (!isValidInput)

{

Console.WriteLine("⚠️ Vui lòng nhập số hợp lệ!");

guessCount--; // Không tính lần này

continue;

}

if (guess == secretNumber)

{

Console.WriteLine("🎉 Chính xác!");

Console.WriteLine($"Bạn đã đoán đúng sau {guessCount} lần!");

isCorrect = true;

}

else if (guess < secretNumber)

{

Console.WriteLine("⬆️ Quá thấp! Thử số lớn hơn.");

}

else

{

Console.WriteLine("⬇️ Quá cao! Thử số nhỏ hơn.");

}

// Hiển thị số lần đoán còn lại

if (!isCorrect && guessCount < maxGuesses)

{

Console.WriteLine($"Bạn còn {maxGuesses - guessCount} lần đoán.");

}

}

// Kiểm tra nếu hết lượt mà chưa đoán đúng

if (!isCorrect)

{

Console.WriteLine("\n💀 Hết lượt đoán!");

Console.WriteLine($"Số bí mật là: {secretNumber}");

Console.WriteLine("Chúc bạn may mắn lần sau!");

}

Console.WriteLine("\nCảm ơn bạn đã chơi!");

}

}

}

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

using System;

namespace ForeachExample

{

class Program

{

static void Main(string[] args)

{

// TODO: Tạo mảng tên các bạn

string[] friends = { "Mai", "Bình", "Chi", "Danh" };

// TODO: In danh sách bạn bè

// Gợi ý: Dùng foreach

// Output:

// "1. Mai"

// "2. Bình"

// "3. Chi"

// "4. Danh"

string[] friends = { "Mai", "Bình", "Chi", "Danh" };

// Sử dụng foreach

Console.WriteLine("📋 DANH SÁCH BẠN BÈ (DÙNG FOREACH)");

Console.WriteLine(new string('=', 35));

int count = 1;

foreach (string friend in friends)

{

Console.WriteLine($"{count++}. {friend}");

}

Console.WriteLine("\n" + new string('-', 40));

// So sánh với for loop

Console.WriteLine("📋 DANH SÁCH BẠN BÈ (DÙNG FOR LOOP)");

Console.WriteLine(new string('=', 35));

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

{

Console.WriteLine($"{i + 1}. {friends[i]}");

}

}

}

}

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

using System;

namespace BreakContinue

{

class Program

{

static void Main(string[] args)

{

// TODO: In các số lẻ từ 1 đến 20 (dùng continue)

// Output: 1 3 5 7 9 11 13 15 17 19

Console.WriteLine("1. In các số lẻ từ 1 đến 20:");

Console.Write(" ");

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

{

// Nếu là số chẵn thì bỏ qua

if (i % 2 == 0)

{

continue;

}

Console.Write(i + " ");

}

Console.WriteLine("\n");

// TODO: Tìm số 7 trong mảng [2, 5, 7, 1, 9, 7, 3]

// Khi tìm thấy, in "Tìm thấy số 7 tại vị trí [chỉ số]" rồi dừng (break)

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

Console.WriteLine("2. Tìm số 7 trong mảng [2, 5, 7, 1, 9, 7, 3]:");

Console.WriteLine(" Bắt đầu tìm kiếm...");

bool found = false;

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

{

Console.WriteLine($" Đang kiểm tra vị trí {i} (số {numbers[i]})");

if (numbers[i] == 7)

{

Console.WriteLine($" ✅ Tìm thấy số 7 tại vị trí [{i}]");

found = true;

break; // Dừng vòng lặp ngay khi tìm thấy

}

}

if (!found)

{

Console.WriteLine(" ❌ Không tìm thấy số 7 trong mảng");

}

}

}

}

**📸 CHỨNG THỰC [3.1 -> 3.6]:**

**3.1**

A screen shot of a computer

AI-generated content may be incorrect.

3.2A black screen with white text

AI-generated content may be incorrect.

3.3A black screen with white text

AI-generated content may be incorrect.

3.4

A computer screen with white text

AI-generated content may be incorrect.

3.5

A computer screen with white text

AI-generated content may be incorrect.

3.6A computer screen with white text

AI-generated content may be incorrect.

* Chạy 6 chương trình và chụp screenshot output
* Thay đổi input để test các trường hợp khác nhau

### 🤔 CÂU HỎI PHẢN BIỆN

1. **Khi nào nên dùng switch thay vì if/else chain, và ngược lại?**
2. **Trong thực tế, khi nào nên dùng while thay vì for?**
3. **Tại sao foreach không cho phép sửa phần tử collection đang duyệt?**

### 💫 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

## 

## 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. Tại sao phải dùng phương thức? (Lợi ích của phương thức là gì?)

Phương thức mang lại 4 lợi ích chính:

Tái sử dụng code (Reusability): Viết một lần, dùng nhiều nơi

Tổ chức code (Organization): Chia nhỏ chương trình thành các phần độc lập, dễ quản lý

Dễ bảo trì (Maintainability): Sửa ở một chỗ, ảnh hưởng đến mọi nơi dùng phương thức đó

Đóng gói (Encapsulation): Ẩn chi tiết cài đặt, chỉ cần biết cách sử dụng

1. Sự khác biệt giữa tham số và đối số?

|  |  |
| --- | --- |
| Tham số (Parameters) | Đối số (Arguments) |
| Biến khai báo trong phương thức | Giá trị thực tế truyền vào |
| Định nghĩa khi tạo phương thức | Cung cấp khi gọi phương thức |
| Là phần của method signature | Là giá trị cụ thể |

1. Phương thức nào phải có return? Phương thức nào không?

Phương thức CÓ kiểu trả về (không phải void): BẮT BUỘC có return

Phương thức void: KHÔNG CÓ return, hoặc có return; (không giá trị) để thoát sớm

1. Nạp chồng phương thức dựa vào điều gì?

Nạp chồng (overloading) dựa vào SỰ KHÁC NHAU CỦA THAM SỐ, cụ thể:

* Số lượng tham số

int Add(int a, int b)

int Add(int a, int b, int c) // Khác số lượng

* Kiểu dữ liệu tham số

int Add(int a, int b)

double Add(double a, double b) // Khác kiểu

* Thứ tự tham số (khi kiểu khác nhau)

void Display(string name, int age)

void Display(int age, string name) // Khác thứ tự

KHÔNG DỰA VÀO:

* Kiểu trả về (chỉ khác kiểu trả về → LỖI)
* Tên tham số
* Access modifier

### 📝 VÍ DỤ THỰC HÀNH

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

using System;

namespace BasicMethods

{

class Program

{

// TODO: Viết phương thức tính tổng 2 số

// Tên: Add

// Tham số: int a, int b

// Trả về: int

static int Add(int a, int b)

{

return a + b;

}

// TODO: Viết phương thức tính tích 2 số

// Tên: Multiply

// Tham số: double x, double y

// Trả về: double

static double Multiply(double x, double y)

{

return x \* y;

}

static void Main(string[] args)

{

Console.WriteLine("🔢 PHƯƠNG THỨC TÍNH TOÁN CƠ BẢN");

Console.WriteLine(new string('=', 40));

// TODO: Gọi phương thức Add và in kết quả

int sum = Add(15, 27);

Console.WriteLine($"Tổng của 15 và 27 là: {sum}");

// TODO: Gọi phương thức Multiply và in kết quả

double product = Multiply(4.5, 3.2);

Console.WriteLine($"Tích của 4.5 và 3.2 là: {product:F2}");

// Thêm ví dụ khác

double product2 = Multiply(7, 8.5);

Console.WriteLine($"Tích của 7 và 8.5 là: {product2:F2}");

// Gọi kết hợp cả hai phương thức

Console.WriteLine(new string('-', 40));

Console.WriteLine("Kết hợp phương thức:");

int a = 10;

int b = 20;

double x = 2.5;

double y = 4.0;

int sumResult = Add(a, b);

double multiplyResult = Multiply(x, y);

Console.WriteLine($"{a} + {b} = {sumResult}");

Console.WriteLine($"{x} × {y} = {multiplyResult:F2}");

}

}

}

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

using System;

namespace VoidMethods

{

class Program

{

// TODO: Viết phương thức in hộp trang trí

// Tên: PrintBox

// Tham số: string text

// Trả về: void

// Ví dụ: PrintBox("Hello") in ra:

// \*\*\*\*\*\*\*\*\*\*\*

// \* Hello \*

// \*\*\*\*\*\*\*\*\*\*\*

static void PrintBox(string text)

{

// Tính chiều dài của hộp (dài hơn text 8 ký tự)

int boxLength = text.Length + 8;

// In dòng trên cùng

Console.WriteLine(new string('\*', boxLength));

// In dòng chứa text

Console.WriteLine($"\* {text.PadRight(text.Length + 6)} \*");

// In dòng dưới cùng

Console.WriteLine(new string('\*', boxLength));

Console.WriteLine(); // Dòng trống để phân cách

}

static void Main(string[] args)

{

// TODO: Gọi phương thức PrintBox với các giá trị khác nhau

Console.WriteLine("Hộp trang trí cơ bản:");

PrintBox("Hello");

PrintBox("Welcome");

PrintBox("C# Programming");

Console.WriteLine(new string('=', 40));

Console.WriteLine("\nHộp trang trí đặc biệt:");

PrintBox("Chúc mừng!");

PrintBox("Thành công");

PrintBox("Happy Coding!");

}

}

}

**TODO 4.3: [PHƯƠNG THỨC VỚI MẢNG]**

using System;

namespace ArrayMethods

{

class Program

{

// TODO: Viết phương thức tính tổng các phần tử trong mảng

// Tên: SumArray

// Tham số: int[] numbers

// Trả về: int

static int SumArray(int[] numbers)

{

int sum = 0;

foreach (int num in numbers)

{

sum += num;

}

return sum;

}

// TODO: Viết phương thức tìm số lớn nhất trong mảng

// Tên: FindMax

// Tham số: int[] numbers

// Trả về: int

static int FindMax(int[] numbers)

{

if (numbers.Length == 0)

{

return 0; // Hoặc throw exception nếu mảng rỗng

}

int max = numbers[0];

for (int i = 1; i < numbers.Length; i++)

{

if (numbers[i] > max)

{

max = numbers[i];

}

}

return max;

}

static void Main(string[] args)

{

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

Console.WriteLine("📊 PHƯƠNG THỨC VỚI MẢNG");

Console.WriteLine(new string('=', 40));

// Hiển thị mảng

Console.WriteLine("Mảng điểm số: [" + string.Join(", ", scores) + "]");

Console.WriteLine();

// TODO: Gọi SumArray và in kết quả

int total = SumArray(scores);

Console.WriteLine($"📈 Tổng các phần tử: {total}");

// TODO: Gọi FindMax và in kết quả

int maxScore = FindMax(scores);

Console.WriteLine($"🏆 Điểm cao nhất: {maxScore}");

}

}

}

**TODO 4.4: [NẠP CHỒNG PHƯƠNG THỨC]**

using System;

namespace MethodOverloading

{

class Program

{

// TODO: Viết phương thức Print có 2 phiên bản

// Phiên bản 1: Print(int x) - in một số

// Phiên bản 2: Print(string text) - in một chuỗi

static void Print(int x)

{

Console.WriteLine($"📊 Số nguyên: {x}");

}

static void Print(string text)

{

Console.WriteLine($"📝 Chuỗi: \"{text}\"");

}

// TODO: Viết phương thức Add có 2 phiên bản

// Phiên bản 1: Add(int a, int b) - cộng 2 số nguyên

// Phiên bản 2: Add(double a, double b) - cộng 2 số thực

static int Add(int a, int b)

{

int result = a + b;

Console.WriteLine($"🧮 {a} + {b} = {result} (số nguyên)");

return result;

}

static double Add(double a, double b)

{

double result = a + b;

Console.WriteLine($"🧮 {a:F2} + {b:F2} = {result:F2} (số thực)");

return result;

}

static void Main(string[] args)

{

Console.WriteLine("🎯 NẠP CHỒNG PHƯƠNG THỨC (OVERLOADING)");

Console.WriteLine(new string('=', 50));

Console.WriteLine("\n1️⃣ PHƯƠNG THỨC PRINT:");

Console.WriteLine(new string('-', 30));

// TODO: Gọi Print với int

Console.WriteLine("Gọi Print(42):");

Print(42);

// TODO: Gọi Print với string

Console.WriteLine("\nGọi Print(\"Hello C#\"):");

Print("Hello C#");

Console.WriteLine("\n" + new string('=', 50));

Console.WriteLine("\n2️⃣ PHƯƠNG THỨC ADD:");

Console.WriteLine(new string('-', 30));

// TODO: Gọi Add(int, int)

Console.WriteLine("Gọi Add(10, 20):");

int intResult = Add(10, 20);

// TODO: Gọi Add(double, double)

Console.WriteLine("\nGọi Add(3.14, 2.86):");

double doubleResult = Add(3.14, 2.86);

Console.WriteLine("\n" + new string('=', 50));

}

}

}

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

using System;

namespace Recursion

{

class Program

{

// TODO: Viết phương thức tính giai thừa

// Tên: Factorial

// Tham số: int n

// Trả về: long

// Công thức: n! = n \* (n-1)!

// Điều kiện dừng: 0! = 1

// Gợi ý: Gọi lại chính nó với n-1

static long Factorial(int n)

{

// Điều kiện dừng (base case)

if (n == 0 || n == 1)

{

Console.WriteLine($" Điều kiện dừng: {n}! = 1");

return 1;

}

// Bước đệ quy

Console.WriteLine($" Tính {n}! = {n} × ({n-1}!)");

long result = n \* Factorial(n - 1);

Console.WriteLine($" Đã tính xong: {n}! = {result}");

return result;

}

static void Main(string[] args)

{

Console.WriteLine("🌀 ĐỆ QUY - TÍNH GIAI THỪA");

Console.WriteLine(new string('=', 40));

Console.WriteLine("\n1️⃣ Tính 5! (giai thừa của 5):");

Console.WriteLine(new string('-', 30));

// TODO: Tính 5! bằng Factorial(5) và in kết quả

long factorial5 = Factorial(5);

Console.WriteLine($"\n✅ Kết quả: 5! = {factorial5}");

Console.WriteLine("\n" + new string('=', 40));

Console.WriteLine("\n2️⃣ Tính 10! (giai thừa của 10):");

Console.WriteLine(new string('-', 30));

// TODO: Tính 10! bằng Factorial(10) và in kết quả

long factorial10 = Factorial(10);

Console.WriteLine($"\n✅ Kết quả: 10! = {factorial10:N0}");

// Tính thêm các giá trị khác

Console.WriteLine("\n" + new string('=', 40));

Console.WriteLine("\n3️⃣ CÁC GIÁ TRỊ GIAI THỪA KHÁC:");

Console.WriteLine(new string('-', 30));

}

}

}

**📸 CHỨNG THỰC [4.1 -> 4.5]:**

**4.1**

A computer screen with white text

AI-generated content may be incorrect.

4.2

A computer screen shot of a computer

AI-generated content may be incorrect.

4.3

A screenshot of a computer

AI-generated content may be incorrect.

4.4

A screenshot of a computer

AI-generated content may be incorrect.

4.5

A computer screen shot of a black screen

AI-generated content may be incorrect.

* Chạy từng chương trình và chụp output
* Viết comment giải thích logic phương thức

### 🤔 CÂU HỎI PHẢN BIỆN

1. **Phương thức void không trả về giá trị, vậy làm sao biết nó thực hiện thành công hay thất bại?**
2. **Tại sao phải phân biệt tham số (parameter) và đối số (argument)?**
3. **Nếu nạp chồng dựa trên số lượng và kiểu tham số, tại sao không thể nạp chồng chỉ bằng kiểu trả về?**

### 💫 KẾT NỐI ĐÁNH GIÁ

**Chuẩn bị tiếp theo:** Phiếu 05 sẽ dạy **lập trình hướng đối tượng (OOP)** - Đây sẽ là bước nhảy vọt lớn

## 

## PHT C# [05] - LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG (OOP)

### 🎯 KIẾN THỨC CỐT LỐI

| **Khái Niệm** | **Ý Nghĩa** | **Ví Dụ** |
| --- | --- | --- |
| **Lớp (Class)** | Bản thiết kế cho đối tượng | class Student { } |
| **Đối tượng (Object)** | Thể hiện cụ thể của lớp | Student sv1 = new Student(); |
| **Thuộc tính (Property)** | Dữ liệu của đối tượng | sv1.Name, sv1.Age |
| **Phương thức (Method)** | Hành động của đối tượng | sv1.Display(), sv1.Calculate() |
| **Constructor** | Phương thức khởi tạo đối tượng | Student(string name) { ... } |
| **Encapsulation** | Che giấu chi tiết, chỉ mở công khai cần thiết | private, public, protected |
| **Kế thừa (Inheritance)** | Lớp con thừa từ lớp cha | class Dog : Animal { } |
| **Đa hình (Polymorphism)** | Một hành động, nhiều cách thực thi | virtual và override |
| **Giao diện (Interface)** | Hợp đồng quy định phương thức bắt buộc | interface IMovable { } |

### 💡 KHÁM PHÁ KIẾN THỨC

**Câu hỏi tự kiểm tra:**

1. Sự khác biệt giữa lớp và đối tượng?

Lớp (Class): Là một bản thiết kế, một khuôn mẫu để tạo ra các đối tượng. Nó định nghĩa các thuộc tính (biến) và phương thức (hàm) mà đối tượng sẽ có.

Đối tượng (Object): Là một thể hiện cụ thể của lớp, được tạo ra từ lớp đó, có trạng thái (giá trị của thuộc tính) và hành vi (phương thức).

1. Constructor là gì? Tại sao cần constructor?

Constructor: Là một phương thức đặc biệt trong lớp, được tự động gọi khi tạo một đối tượng mới, thường dùng để khởi tạo giá trị ban đầu cho các thuộc tính.

Tại sao cần?: Để đảm bảo đối tượng được khởi tạo đúng cách, thiết lập trạng thái ban đầu và có thể nhận tham số nếu cần.

1. private, public, protected khác gì?

public: Có thể truy cập từ mọi nơi.

private: Chỉ có thể truy cập từ bên trong lớp đó.

protected: Có thể truy cập từ bên trong lớp đó và các lớp con (kế thừa).

1. Kế thừa có tác dụng gì?

Tác dụng chính:

Tái sử dụng mã nguồn.

Tạo mối quan hệ "là một" (is-a) giữa lớp cha và lớp con.

Dễ bảo trì và mở rộng hệ thống.

### 📝 VÍ DỤ THỰC HÀNH

**TODO 5.1: [TẠO LỚP CƠ BẢN - STUDENT]**

using System;

namespace BasicClass

{

// TODO: Tạo lớp Student

// Thuộc tính: StudentId, Name, GPA

// Phương thức: Display() (in thông tin)

class Student

{

public string StudentId;

public string Name;

public double GPA;

public void Display()

{

Console.WriteLine($"ID: {StudentId}, Name: {Name}, GPA: {GPA:F2}");

}

}

class Program

{

static void Main(string[] args)

{

// TODO: Tạo 2 đối tượng Student

// TODO: Gán giá trị thuộc tính

// TODO: Gọi phương thức Display()

Student student1 = new Student();

Student student2 = new Student();

student1.StudentId = "SV001";

student1.Name = "Nguyễn Văn A";

student1.GPA = 3.5;

student2.StudentId = "SV002";

student2.Name = "Trần Thị B";

student2.GPA = 3.8;

student1.Display();

student2.Display();

}

}

}

**TODO 5.2: [CONSTRUCTOR - KHỞI TẠO DỮ LIỆU]**

using System;

namespace Constructor

{

// TODO: Tạo lớp Product với Constructor

// Constructor nhận tham số: productId, productName, price

class Product

{

public string ProductId;

public string ProductName;

public decimal Price;

// Constructor

public Product(string productId, string productName, decimal price)

{

ProductId = productId;

ProductName = productName;

Price = price;

}

}

class Program

{

static void Main(string[] args)

{

// TODO: Tạo đối tượng Product dùng Constructor

// TODO: In thông tin sản phẩm

Product laptop = new Product("P001", "Laptop Dell XPS", 25000000);

Console.WriteLine($"Mã SP: {laptop.ProductId}");

Console.WriteLine($"Tên SP: {laptop.ProductName}");

Console.WriteLine($"Giá: {laptop.Price:N0} VND");

}

}

}

**TODO 5.3: [ENCAPSULATION - CHE GIẤU DỮ LIỆU]**

using System;

namespace Encapsulation

{

// TODO: Tạo lớp BankAccount

// - Private field: \_balance (số dư)

// - Public property: Balance (chỉ đọc, phòng trừy cập trái phép)

// - Public method: Deposit(amount) - gửi tiền

// - Public method: Withdraw(amount) - rút tiền (kiểm tra đủ tiền không)

class BankAccount

{

private decimal \_balance;

// - Public property: Balance (chỉ đọc, phòng trừy cập trái phép)

public decimal Balance

{

get { return \_balance; }

}

// Constructor

public BankAccount(decimal initialBalance)

{

\_balance = initialBalance;

}

public void Deposit(decimal amount)

{

if (amount > 0)

{

\_balance += amount;

Console.WriteLine($"Đã gửi: {amount:N0}. Số dư mới: {\_balance:N0}");

}

else

{

Console.WriteLine("Số tiền gửi phải lớn hơn 0");

}

}

public void Withdraw(decimal amount)

{

if (amount <= 0)

{

Console.WriteLine("Số tiền rút phải lớn hơn 0");

}

else if (amount > \_balance)

{

Console.WriteLine($"Không đủ tiền. Số dư hiện tại: {\_balance:N0}");

}

else

{

\_balance -= amount;

Console.WriteLine($"Đã rút: {amount:N0}. Số dư mới: {\_balance:N0}");

}

}

}

class Program

{

static void Main(string[] args)

{

// TODO: Tạo tài khoản và thử các thao tác

BankAccount account = new BankAccount(1000000);

Console.WriteLine($"Số dư ban đầu: {account.Balance:N0}");

account.Deposit(500000);

account.Withdraw(200000);

account.Withdraw(2000000); // Thử rút quá số dư

}

}

}

**TODO 5.4: [KẾ THỪA - ANIMAL & DOG]**

using System;

namespace Inheritance

{

// TODO: Tạo lớp Animal (cha)

// Thuộc tính: Name

// Phương thức: MakeSound() (in ra "Animal makes a sound")

class Animal

{

public string Name;

public virtual void MakeSound()

{

Console.WriteLine("Animal makes a sound");

}

}

// TODO: Tạo lớp Dog (con) kế thừa từ Animal

// Override phương thức MakeSound() (in ra "Woof! Woof!")

class Dog : Animal

{

// Override phương thức MakeSound()

public override void MakeSound()

{

Console.WriteLine("Woof! Woof!");

}

}

class Program

{

static void Main(string[] args)

{

// TODO: Tạo đối tượng Animal và Dog

// TODO: Gọi MakeSound() của cả hai

Animal genericAnimal = new Animal();

genericAnimal.Name = "Generic Animal";

Dog myDog = new Dog();

myDog.Name = "Buddy";

Console.WriteLine($"{genericAnimal.Name}:");

genericAnimal.MakeSound();

Console.WriteLine($"\n{myDog.Name}:");

myDog.MakeSound();

}

}

}

**📸 CHỨNG THỰC [5.1 -> 5.4]:**

**5.1**

A computer screen with white text

AI-generated content may be incorrect.

5.2

A screen shot of a computer

AI-generated content may be incorrect.

5.3

A screenshot of a computer

AI-generated content may be incorrect.

5.4

A screenshot of a computer

AI-generated content may be incorrect.

* Chạy từng chương trình, screenshot output
* Thay đổi dữ liệu để test chức năng

### 🤔 CÂU HỎI PHẢN BIỆN

1. **Constructor chỉ khởi tạo giá trị ban đầu, vậy tại sao không dùng phương thức thường?**
2. **Khi nào thực sự cần setter/getter phức tạp, và khi nào có thể dùng public field đơn giản?"**
3. **Khi nào nên dùng composition (has-a) thay vì inheritance (is-a)?"**

### 💫 KẾT NỐI ĐÁNH GIÁ

**Trọng yếu:** Phần này là **nền tảng OOP** - phải hiểu chắc để học tiếp

## 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]**

// Lớp Student chứa thông tin và điểm của 1 sinh viên

public class Student

{

public string StudentId { get; set; }

public string Name { get; set; }

public double Score { get; set; }

// Constructor

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

{

// TODO: Khởi tạo các thuộc tính với validation

// - StudentId không được rỗng

// - Name không được rỗng

// - Score phải từ 0 đến 10

if (string.IsNullOrWhiteSpace(id))

throw new ArgumentException("Mã sinh viên không được để trống");

// Validation cho Name

if (string.IsNullOrWhiteSpace(name))

throw new ArgumentException("Tên sinh viên không được để trống");

// Validation cho Score

if (score < 0 || score > 10)

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

StudentId = id;

Name = name;

Score = score;

}

// Phương thức in thông tin

public void Display()

{

// TODO: In ra "ID: [StudentId] | Tên: [Name] | Điểm: [Score]"

Console.WriteLine($"🎓 ID: {StudentId} | Tên: {Name} | Điểm: {Score:F2}");

}

}

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

public class StudentManager

{

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

private int count = 0; // Số lượng sinh viên hiện tại

// Thêm sinh viên mới

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

{

try

{

// Kiểm tra nếu đã đầy

if (count >= students.Length)

{

Console.WriteLine("❌ Danh sách đã đầy! Không thể thêm sinh viên mới.");

return false;

}

// Kiểm tra trùng ID

if (FindStudentById(id) != null)

{

Console.WriteLine($"❌ Đã tồn tại sinh viên với ID: {id}");

return false;

}

// TODO: Phương thức AddStudent(string id, string name, double score)

// Thêm sinh viên mới, kiểm tra trùng lặp

Student newStudent = new Student(id, name, score);

students[count] = newStudent;

count++;

Console.WriteLine($"✅ Đã thêm sinh viên: {name} (ID: {id})");

return true;

}

catch (ArgumentException ex)

{

Console.WriteLine($"❌ Lỗi: {ex.Message}");

return false;

}

}

// TODO: Phương thức RemoveStudent(string id)

// Xóa sinh viên theo ID

public bool RemoveStudent(string id)

{

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

{

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

{

// Dịch chuyển các phần tử phía sau lên

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

{

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

}

students[count - 1] = null;

count--;

Console.WriteLine($"✅ Đã xóa sinh viên có ID: {id}");

return true;

}

}

Console.WriteLine($"❌ Không tìm thấy sinh viên với ID: {id}");

return false;

}

// TODO: Phương thức UpdateScore(string id, double newScore)

// Cập nhật điểm

public bool UpdateScore(string id, double newScore)

{

try

{

// Validation cho điểm mới

if (newScore < 0 || newScore > 10)

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

Student student = FindStudentById(id);

if (student != null)

{

double oldScore = student.Score;

student.Score = newScore;

Console.WriteLine($"✅ Đã cập nhật điểm cho {student.Name}: {oldScore:F2} → {newScore:F2}");

return true;

}

Console.WriteLine($"❌ Không tìm thấy sinh viên với ID: {id}");

return false;

}

catch (ArgumentException ex)

{

Console.WriteLine($"❌ Lỗi: {ex.Message}");

return false;

}

}

// TODO: Phương thức GetAverageScore()

// Tính điểm trung bình

public double GetAverageScore()

{

if (count == 0)

return 0;

double total = 0;

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

{

total += students[i].Score;

}

return total / count;

}

// TODO: Phương thức GetMaxScore()

// Tìm điểm cao nhất

public (Student student, double maxScore) GetMaxScore()

{

if (count == 0)

return (null, 0);

double max = students[0].Score;

Student topStudent = students[0];

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

{

if (students[i].Score > max)

{

max = students[i].Score;

topStudent = students[i];

}

}

return (topStudent, max);

}

// TODO: Phương thức FindStudentById(string id)

// Trả về đối tượng Student hoặc null

public Student FindStudentById(string id)

{

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

{

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

return students[i];

}

return null;

}

// TODO: Phương thức DisplayAllStudents()

// In danh sách tất cả sinh viên

public void DisplayAllStudents()

{

if (count == 0)

{

Console.WriteLine("📭 Danh sách sinh viên trống!");

return;

}

Console.WriteLine($"\n📋 DANH SÁCH SINH VIÊN ({count} sinh viên)");

Console.WriteLine(new string('=', 50));

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

{

Console.Write($"{i + 1}. ");

students[i].Display();

}

}

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

using System;

namespace StudentManagementSystem

{

class Program

{

static void Main(string[] args)

{

StudentManager manager = new StudentManager();

bool running = true;

while (running)

{

try

{

// TODO: In menu

Console.WriteLine("\n========== MENU ==========");

Console.WriteLine("1. Thêm sinh viên");

Console.WriteLine("2. Xóa sinh viên");

Console.WriteLine("3. Cập nhật điểm");

Console.WriteLine("4. In danh sách");

Console.WriteLine("5. Tính điểm trung bình");

Console.WriteLine("6. Tìm điểm cao nhất");

Console.WriteLine("7. Tìm sinh viên");

Console.WriteLine("0. Thoát");

Console.WriteLine("========================");

// TODO: Nhận lựa chọn từ người dùng

// TODO: Dùng switch xử lý từng lựa chọn

// TODO: Thêm try-catch để xử lý lỗi

string choice = Console.ReadLine();

switch (choice)

{

case "1": // Thêm sinh viên

Console.Write("Nhập mã sinh viên: ");

string id = Console.ReadLine();

Console.Write("Nhập tên sinh viên: ");

string name = Console.ReadLine();

Console.Write("Nhập điểm (0-10): ");

if (double.TryParse(Console.ReadLine(), out double score))

manager.AddStudent(id, name, score);

else

Console.WriteLine("❌ Điểm không hợp lệ!");

break;

case "2": // Xóa sinh viên

Console.Write("Nhập mã sinh viên cần xóa: ");

string removeId = Console.ReadLine();

manager.RemoveStudent(removeId);

break;

case "3": // Cập nhật điểm

Console.Write("Nhập mã sinh viên: ");

string updateId = Console.ReadLine();

Console.Write("Nhập điểm mới (0-10): ");

if (double.TryParse(Console.ReadLine(), out double newScore))

manager.UpdateScore(updateId, newScore);

else

Console.WriteLine("❌ Điểm không hợp lệ!");

break;

case "4": // In danh sách

manager.DisplayAllStudents();

break;

case "5": // Tính điểm trung bình

double average = manager.GetAverageScore();

Console.WriteLine($"📊 Điểm trung bình của tất cả sinh viên: {average:F2}");

break;

case "6": // Tìm điểm cao nhất/thấp nhất

var (topStudent, maxScore) = manager.GetMaxScore();

var (bottomStudent, minScore) = manager.GetMinScore();

if (topStudent != null)

{

Console.WriteLine("🏆 ĐIỂM CAO NHẤT:");

Console.WriteLine($" Sinh viên: {topStudent.Name} (ID: {topStudent.StudentId})");

Console.WriteLine($" Điểm: {maxScore:F2}");

Console.WriteLine("\n📉 ĐIỂM THẤP NHẤT:");

Console.WriteLine($" Sinh viên: {bottomStudent.Name} (ID: {bottomStudent.StudentId})");

Console.WriteLine($" Điểm: {minScore:F2}");

}

else

{

Console.WriteLine("📭 Không có sinh viên nào trong danh sách!");

}

break;

case "7": // Tìm sinh viên theo ID

Console.Write("Nhập mã sinh viên cần tìm: ");

string findId = Console.ReadLine();

Student found = manager.FindStudentById(findId);

if (found != null)

{

Console.WriteLine("✅ Tìm thấy sinh viên:");

found.Display();

}

else

{

Console.WriteLine($"❌ Không tìm thấy sinh viên với ID: {findId}");

}

break;

case "8": // Thống kê

Console.WriteLine("📈 THỐNG KÊ HỆ THỐNG");

Console.WriteLine(new string('=', 30));

Console.WriteLine($"Số lượng sinh viên: {manager.GetStudentCount()}/50");

double avg = manager.GetAverageScore();

var (top, max) = manager.GetMaxScore();

var (bottom, min) = manager.GetMinScore();

Console.WriteLine($"Điểm trung bình: {avg:F2}");

if (top != null)

{

Console.WriteLine($"Điểm cao nhất: {max:F2} ({top.Name})");

Console.WriteLine($"Điểm thấp nhất: {min:F2} ({bottom.Name})");

Console.WriteLine($"Khoảng cách điểm: {max - min:F2}");

}

break;

case "0": // Thoát

Console.WriteLine("👋 Cảm ơn đã sử dụng hệ thống!");

running = false;

break;

default:

Console.WriteLine("❌ Lựa chọn không hợp lệ! Vui lòng chọn từ 0-8.");

break;

}

}

catch (Exception ex)

{

Console.WriteLine($"💥 Đã xảy ra lỗi: {ex.Message}");

Console.WriteLine("Vui lòng thử lại!");

}

}

}

}

}

**📸 CHỨNG THỰC DỰ ÁN:**

**menu**

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

* 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?**

Dùng double.TryParse() và kiểm tra range (0-10). Nếu sai, thông báo lỗi và yêu cầu nhập lại.

1. \*\*Có thể dùng Listthay vì array không? Ưu điểm gì?\*\*

Có, ưu điểm:

* Tự động mở rộng: Không lo đầy mảng
* Dễ dàng thêm/xóa: Không cần tự quản lý chỉ số
* LINQ support: Tìm kiếm, lọc, sắp xếp dễ dàng
* Code sạch hơn: Giảm boilerplate code

1. **Nếu muốn lưu dữ liệu xuống file sau khi chương trình thoát, làm sao?**

Dùng JSON serialization:

// Lưu

string json = JsonSerializer.Serialize(students);

File.WriteAllText("data.json", json);

// Đọc

string json = File.ReadAllText("data.json");

students = JsonSerializer.Deserialize<List<Student>>(json);

### 💫 KẾT NỐI ĐÁNH GIÁ

**Bước tiếp theo:**

* Nếu hoàn thành được dự án này, bạn đã sẵn sàng cho **ASP.NET Core** Backend
* Phiếu tiếp theo sẽ dạy **advanced OOP** (interfaces, abstract class, LINQ)
* Tiếp theo là **xây dựng Web API** thực tế

## 🎓

## HƯỚNG DẪN NỘP BÀI

### Cấu Trúc Thư Mục Nộp Bài:

StudentName\_CSHarp/

├── PHT01\_Introduction/

│ ├── HelloWorld.cs

│ └── HelloWorld\_Output.png

├── PHT02\_Variables/

│ ├── ShoppingCalculator.cs

│ ├── StudentGrades.cs

│ ├── UserProfile.cs

│ └── [screenshots]

├── PHT03\_Conditions/

│ ├── GradeClassification.cs

│ ├── DayOfWeek.cs

│ └── [screenshots]

├── PHT04\_Methods/

│ ├── BasicMethods.cs

│ └── [screenshots]

├── PHT05\_OOP/

│ ├── BasicClass.cs

│ └── [screenshots]

└── PHT06\_Project/

├── Student.cs

├── StudentManager.cs

├── Program.cs

└── [screenshots]

### Yêu Cầu Nộp Bài:

* ✅ **Hoàn thành tất cả TODO** trong mỗi phiếu
* ✅ **Screenshot chạy chương trình** cho mỗi bài
* ✅ **Comment code** rõ ràng
* ✅ **Nộp file .cs** + file .md (tài liệu)
* ✅ **Dự án 06** phải **chạy trơn tru**, **xử lý lỗi tốt**

### Tiêu Chí Đánh Giá:

| **Tiêu Chí** | **Điểm** |
| --- | --- |
| Hoàn thành 100% TODO | 40% |
| Code chạy đúng, không lỗi | 30% |
| Xử lý lỗi (try-catch, validation) | 15% |
| Comment, giải thích logic | 10% |
| Nộp đầy đủ file, cấu trúc rõ ràng | 5% |