## 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? (==, !=, <, >, <=, >=)
2. Toán tử logic là gì? (&&, ||, !)
3. Sự khác biệt giữa for và foreach là gì?
4. Khi nào dùng break? Khi nào dùng continue?

### 📝 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

string grade;

// TODO: Phân loại điểm

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

{

grade = "A (Xuất sắc)";

}

else if (score >= 80)

{

grade = "B (Khá)";

}

else if (score >= 70)

{

grade = "C (Trung bình)";

}

else if (score >= 60)

{

grade = "D (Yếu)";

}

else

{

grade = "F (Không đạt)";

}

// TODO: In kết quả

Console.WriteLine("Điểm của bạn: " + score);

Console.WriteLine("Xếp loại: " + grade);

}

}

}

**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

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]**

using System;

namespace SumCalculation

{

class Program

{

static void Main(string[] args)

{

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

int sum = 0;

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

{

sum += i;

}

// TODO: In 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)

{

// Số bí mật (cố định)

int secretNumber = 50;

int guess = 0;

// Vòng lặp đoán số

while (guess != secretNumber)

{

Console.Write("Nhập số bạn đoán: ");

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

if (guess < secretNumber)

{

Console.WriteLine("Quá thấp");

}

else if (guess > secretNumber)

{

Console.WriteLine("Quá cao");

}

else

{

Console.WriteLine("Chính xác!");

}

}

}

}

}

**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

int index = 1;

foreach (string friend in friends)

{

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

index++;

}

}

}

}

**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)

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

{

if (i % 2 == 0)

{

continue; // bỏ qua số chẵn

}

Console.Write(i + " ");

}

Console.WriteLine(); // xuống dòng

// TODO: Tìm số 7 trong mảng

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

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

{

if (numbers[i] == 7)

{

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

break; // dừng vòng lặp khi tìm thấy số 7 đầu tiên

}

}

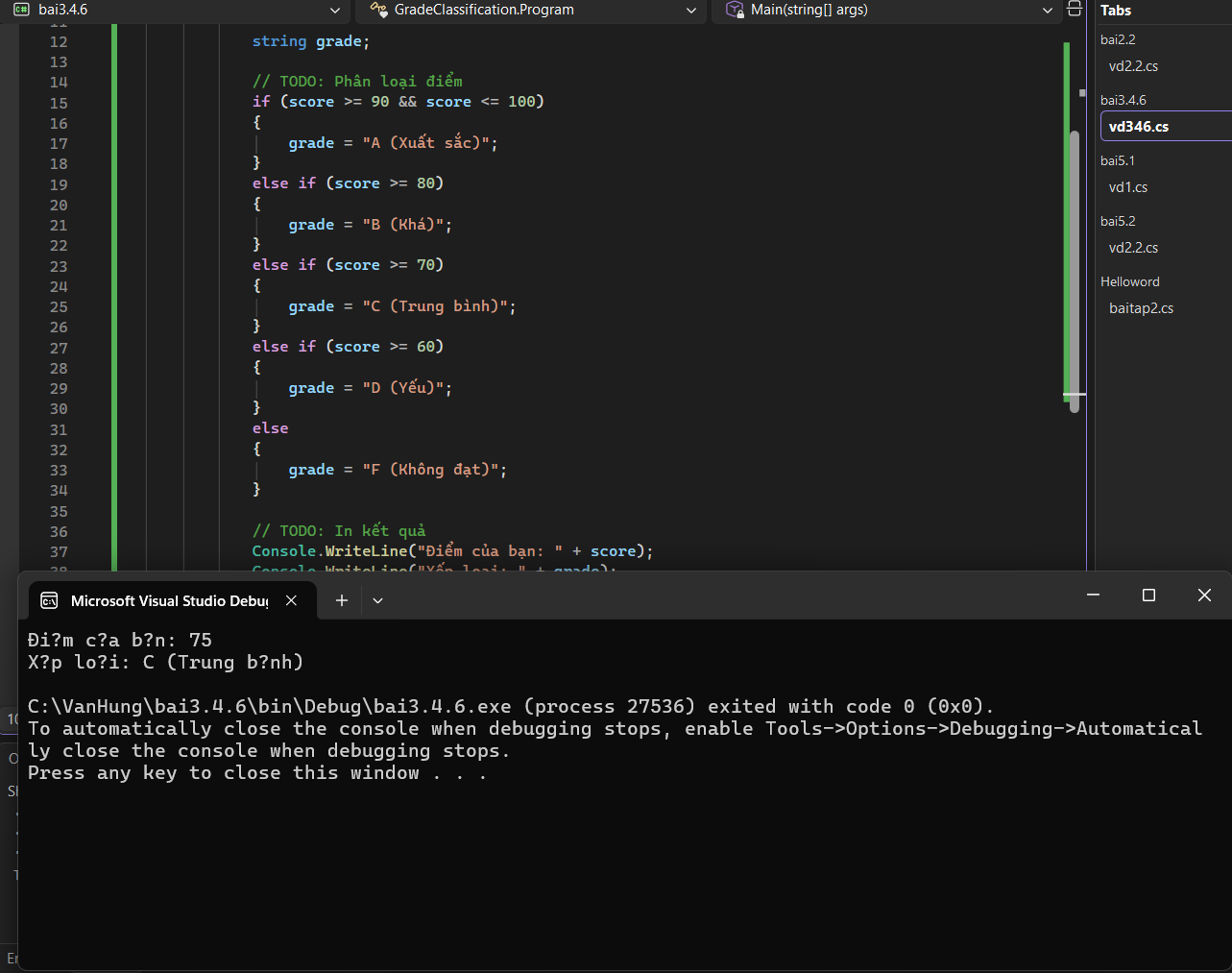
}

}

}

**📸 CHỨNG THỰC [3.1 -> 3.6]:**

3.1 :



3.2Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, màn hình

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

3.3 :

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

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

3.4 :

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

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

3.5 :

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, phần mềm

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

3.6 :

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

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

### 🤔 CÂU HỎI PHẢN BIỆN

1. **Tại sao vòng lặp for lại được dùng nhiều hơn while trong lập trình?**

**Lý do**

* for **gộp đủ 3 phần** trong một dòng:
  + Khởi tạo biến
  + Điều kiện lặp
  + Bước tăng/giảm
* Nhìn vào là **biết ngay vòng lặp chạy bao nhiêu lần**

**🔹 Lợi ích**

* Code **ngắn gọn, dễ đọc, dễ kiểm soát**
* Ít gây lỗi **vòng lặp vô hạn** hơn while
* Rất phù hợp khi:
  + Lặp theo số lần xác định
  + Duyệt mảng, danh sách

1. **Nếu bạn có vòng lặp for lồng trong for, làm cách nào để thoát cả hai vòng lặp cùng lúc?**

Dùng **biến cờ (flag)** để báo dừng vòng lặp ngoài

Hoặc dùng **return** để thoát khỏi toàn bộ hàm

1. **So sánh while và do-while - khi nào nên dùng do-while?**

Khi **bắt buộc vòng lặp chạy ít nhất 1 lần**

Thường dùng cho:

* + Menu chương trình
  + Nhập dữ liệu người dùng
  + Các thao tác cần thực hiện trước rồi mới kiểm tra điều kiện

### 💫 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?)
2. Sự khác biệt giữa tham số và đối số?
3. Phương thức nào phải có return? Phương thức nào không?
4. Nạp chồng phương thức dựa vào điều gì?

### 📝 VÍ DỤ THỰC HÀNH

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

using System;

namespace BasicMethods

{

class Program

{

// Phương thức tính tổng 2 số

static int Add(int a, int b)

{

return a + b;

}

// Phương thức tính tích 2 số

static double Multiply(double x, double y)

{

return x \* y;

}

static void Main(string[] args)

{

// Gọi phương thức Add và in kết quả

int sum = Add(5, 3);

Console.WriteLine("Tổng = " + sum);

// Gọi phương thức Multiply và in kết quả

double product = Multiply(2.5, 4);

Console.WriteLine("Tích = " + product);

}

}

}

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

using System;

namespace VoidMethods

{

class Program

{

// Phương thức in hộp trang trí

static void PrintBox(string text)

{

int length = text.Length;

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

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

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

}

static void Main(string[] args)

{

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

PrintBox("Hello");

PrintBox("C#");

PrintBox("Programming");

}

}

}

**TODO 4.3: [PHƯƠNG THỨC VỚI MẢNG]**

using System;

namespace ArrayMethods

{

class Program

{

// Phương thức tính tổng các phần tử trong mảng

static int SumArray(int[] numbers)

{

int sum = 0;

foreach (int n in numbers)

{

sum += n;

}

return sum;

}

// Phương thức tìm số lớn nhất trong mảng

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 };

// Gọi SumArray và in kết quả

int total = SumArray(scores);

Console.WriteLine("Tổng các phần tử = " + total);

// Gọi FindMax và in kết quả

int maxScore = FindMax(scores);

Console.WriteLine("Số lớn nhất = " + maxScore);

}

}

}

**TODO 4.4: [NẠP CHỒNG PHƯƠNG THỨC]**

using System;

namespace MethodOverloading

{

class Program

{

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

static void Print(int x)

{

Console.WriteLine("Số nguyên: " + x);

}

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

static void Print(string text)

{

Console.WriteLine("Chuỗi: " + text);

}

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

static int Add(int a, int b)

{

return a + b;

}

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

static double Add(double a, double b)

{

return a + b;

}

static void Main(string[] args)

{

// Gọi Print với int

Print(10);

// Gọi Print với string

Print("Hello C#");

// Gọi Add(int, int)

int sumInt = Add(5, 7);

Console.WriteLine("Tổng số nguyên = " + sumInt);

// Gọi Add(double, double)

double sumDouble = Add(2.5, 3.5);

Console.WriteLine("Tổng số thực = " + sumDouble);

}

}

}

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

using System;

namespace Recursion

{

class Program

{

// Phương thức tính giai thừa (đệ quy)

static long Factorial(int n)

{

// Điều kiện dừng

if (n == 0)

{

return 1;

}

// Gọi đệ quy

return n \* Factorial(n - 1);

}

static void Main(string[] args)

{

// Tính 5!

long result1 = Factorial(5);

Console.WriteLine("5! = " + result1);

// Tính 10!

long result2 = Factorial(10);

Console.WriteLine("10! = " + result2);

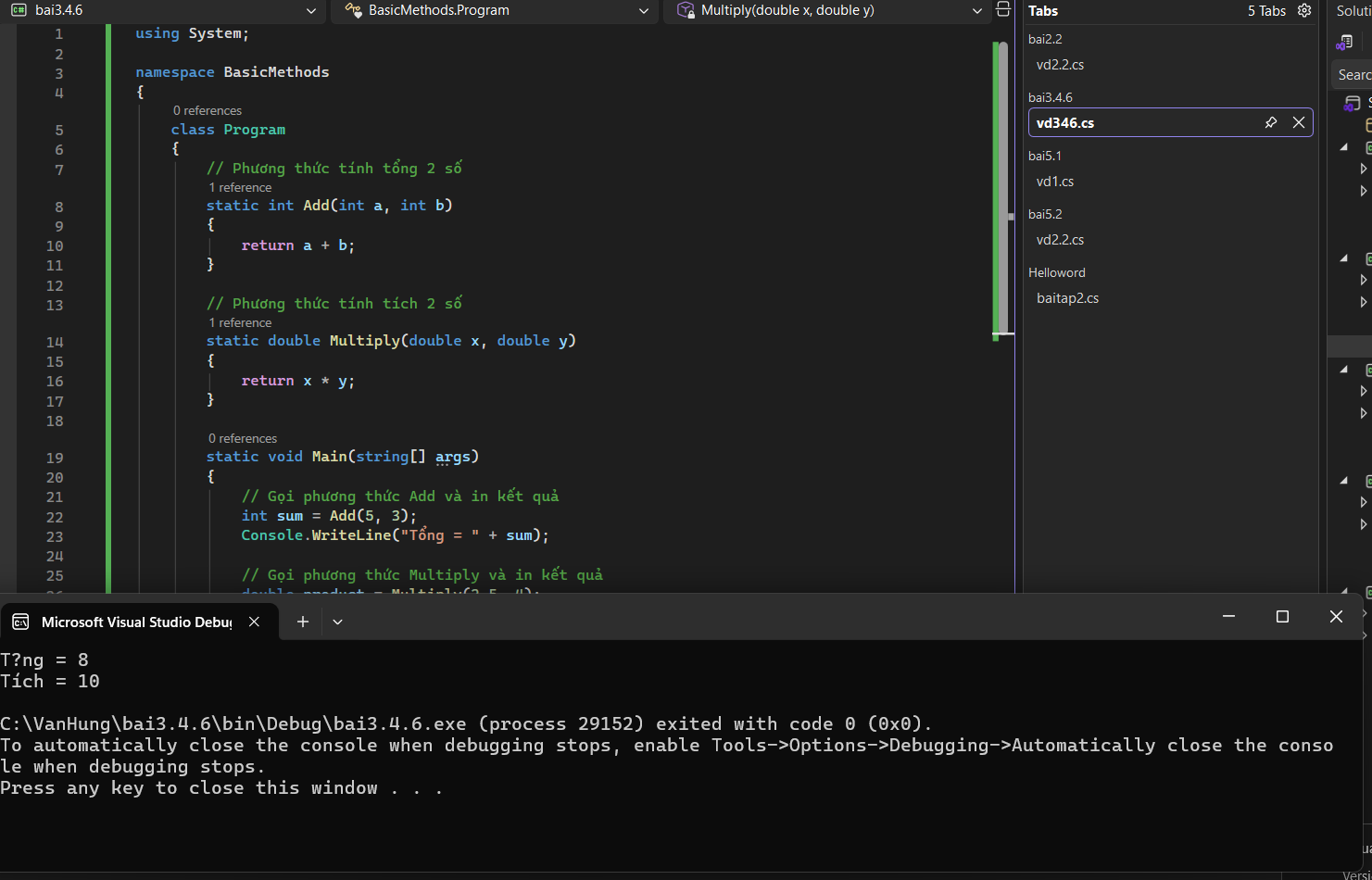
}

}

}

**📸 CHỨNG THỰC [4.1 -> 4.5]:**

**4.1:**



**Khái niệm**

* Phương thức (method) là **một khối lệnh dùng để thực hiện một công việc cụ thể**.
* Có thể **nhận tham số** và **trả về kết quả**.

**Đặc điểm**

* Có **kiểu trả về** (int, double, string,…).
* Dùng return để trả kết quả.
* Giúp **tái sử dụng code**, tránh lặp lại.

4.2

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

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

**Khái niệm**

* Phương thức void **không trả về giá trị**.
* Thường dùng để **in ra màn hình**, thông báo, xử lý hiển thị.

**Đặc điểm**

* Không dùng return (hoặc chỉ return;).
* Thực hiện hành động, **không sinh kết quả**.

**Khi nào dùng void?**

* Khi chỉ cần **hiển thị**, **in thông tin**, **thực hiện thao tác**.

4.3 :

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

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

* Phương thức có thể nhận **mảng làm tham số**.
* Dùng để xử lý nhiều dữ liệu cùng lúc.

**Ví dụ tác vụ**

* Tính tổng mảng
* Tìm giá trị lớn nhất / nhỏ nhất
* Đếm phần tử

**Đặc điểm**

* Tham số dạng int[], double[],…
* Thường dùng vòng lặp for hoặc foreach.

**Lợi ích**

* Code gọn
* Dễ quản lý và mở rộng
* Tách xử lý logic ra khỏi Main

4.4 :

**Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, màn hình

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

**Khái niệm**

* **Nạp chồng là việc:  
   Nhiều phương thức cùng tên  
   Nhưng khác tham số**

**Điều kiện để nạp chồng**

**✔ Khác số lượng tham số  
✔ Hoặc khác kiểu dữ liệu tham số**

**❌ Không dựa vào kiểu trả về để phân biệt**

**Lợi ích**

* **Dễ nhớ tên hàm**
* **Code rõ ràng**
* **Tăng tính linh hoạt**

**4.5:**

**Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

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

**Khái niệm**

* Đệ quy là phương thức **tự gọi lại chính nó**.
* Dùng để giải bài toán có tính lặp tự nhiên.

**Thành phần bắt buộc**

1. **Điều kiện dừng**  
   → để kết thúc đệ quy
2. **Bước đệ quy**  
   → gọi lại chính nó với bài toán nhỏ hơn

**Ví dụ giai thừa**

* Công thức:  
  n! = n × (n−1)!
* Điều kiện dừng:  
  0! = 1

**Ưu & nhược điểm**

**Ưu điểm**

* Code ngắn gọn
* Dễ hiểu với bài toán toán học

**Nhược điểm**

* Tốn bộ nhớ
* Dễ gây tràn stack nếu thiếu điều kiện dừng

### 🤔 CÂU HỎI PHẢN BIỆN

1. **Tại sao cần nạp chồng phương thức? Không thể dùng tên khác nhau cho từng trường hợp không?**

Không thể dùng tên khác nhau cho từng trường hợp không?\*\*

**✔ Có thể dùng tên khác nhau, nhưng không nên**

**🔹 Lý do cần nạp chồng**

* Các phương thức **thực hiện cùng một chức năng** nhưng **khác kiểu dữ liệu hoặc số tham số**.
* Giúp **giữ tên phương thức nhất quán**, dễ hiểu ý nghĩa.

**🔹 Lợi ích**

* Code **dễ đọc**, dễ nhớ
* Dễ sử dụng, không cần học nhiều tên hàm
* Trình biên dịch tự chọn phương thức phù hợp

**🔹 Nếu dùng tên khác nhau**

* Code dài
* Khó quản lý
* Giảm tính trực quan

**Kết luận:**  
Có thể dùng tên khác, nhưng **nạp chồng giúp code gọn, rõ ràng và chuyên nghiệp hơn**.

1. **Đệ quy có ưu và nhược điểm gì?**

**Ưu điểm**

* Code **ngắn gọn**
* Dễ hiểu với bài toán toán học (giai thừa, Fibonacci)
* Phù hợp với cấu trúc phân cấp (cây, thư mục)

**🔹 Nhược điểm**

* Tốn bộ nhớ (mỗi lần gọi tạo stack mới)
* Dễ gây **tràn stack** nếu không có điều kiện dừng
* Hiệu năng thường kém hơn vòng lặp

**Khi nào nên dùng đệ quy?**

* Bài toán có cấu trúc **lặp lại tự nhiên**
* Có **điều kiện dừng rõ ràng**
* Độ sâu đệ quy không quá lớn

📌 Ví dụ: giai thừa, duyệt cây

**Khi nào nên dùng vòng lặp?**

* Bài toán lặp đơn giản
* Cần **hiệu năng cao**
* Số lần lặp lớn

📌 Ví dụ: duyệt mảng, tính tổng

1. **Phương thức nên ngắn hay dài? Tại sao?**

**Nên ngắn**

**🔹 Lý do**

* Dễ đọc, dễ hiểu
* Dễ kiểm tra và sửa lỗi
* Dễ tái sử dụng
* Mỗi phương thức chỉ nên làm **một nhiệm vụ**

**🔹 Phương thức quá dài gây**

* Khó bảo trì
* Khó hiểu logic
* Dễ sinh lỗi

**Nguyên tắc quan trọng:**  
**“Một phương thức – một nhiệm vụ” (Single Responsibility)**

### 💫 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# [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)

{

// Validation

if (string.IsNullOrWhiteSpace(id))

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

if (string.IsNullOrWhiteSpace(name))

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

if (score < 0 || score > 10)

throw new Exception("Điểm phải nằm trong khoảng 0 - 10");

StudentId = id;

Name = name;

Score = score;

}

// In thông tin sinh viên

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;

// Thêm sinh viên

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

{

if (count >= 50)

throw new Exception("Danh sách sinh viên đã đầy");

// Kiểm tra trùng ID

if (FindStudentById(id) != null)

throw new Exception("StudentId đã tồn tại");

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

Console.WriteLine("Thêm sinh viên thành công!");

}

// Xóa sinh viên theo ID

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];

students[count - 1] = null;

count--;

Console.WriteLine("Xóa sinh viên thành công!");

return;

}

}

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

}

// Cập nhật điểm

public void UpdateScore(string id, double newScore)

{

if (newScore < 0 || newScore > 10)

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

Student s = FindStudentById(id);

if (s == null)

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

s.Score = newScore;

Console.WriteLine("Cập nhật điểm thành công!");

}

// Tính điểm trung bình

public double GetAverageScore()

{

if (count == 0)

throw new Exception("Danh sách rỗng");

double sum = 0;

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

sum += students[i].Score;

return sum / count;

}

// Tìm điểm cao nhất

public double GetMaxScore()

{

if (count == 0)

throw new Exception("Danh sách rỗng");

double max = students[0].Score;

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

if (students[i].Score > max)

max = students[i].Score;

return max;

}

// Tìm sinh viên theo ID

public Student FindStudentById(string id)

{

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

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

return students[i];

return null;

}

// In danh sách sinh viên

public void DisplayAllStudents()

{

if (count == 0)

{

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

return;

}

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

students[i].Display();

}

}

// =============================

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

namespace StudentManagementSystem

{

class Program

{

static void Main(string[] args)

{

StudentManager manager = new StudentManager();

bool running = true;

while (running)

{

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 theo ID");

Console.WriteLine("0. Thoát");

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

Console.Write("Chọn chức năng: ");

try

{

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

switch (choice)

{

case 1:

Console.Write("Nhập ID: ");

string id = Console.ReadLine();

Console.Write("Nhập tên: ");

string name = Console.ReadLine();

Console.Write("Nhập điểm: ");

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

manager.AddStudent(id, name, score);

break;

case 2:

Console.Write("Nhập ID cần xóa: ");

manager.RemoveStudent(Console.ReadLine());

break;

case 3:

Console.Write("Nhập ID: ");

string uid = Console.ReadLine();

Console.Write("Nhập điểm mới: ");

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

manager.UpdateScore(uid, newScore);

break;

case 4:

manager.DisplayAllStudents();

break;

case 5:

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

break;

case 6:

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

break;

case 7:

Console.Write("Nhập ID cần tìm: ");

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

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

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

break;

case 0:

running = false;

break;

default:

Console.WriteLine("Lựa chọn không hợp lệ");

break;

}

}

catch (Exception ex)

{

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

}

}

}

}

}

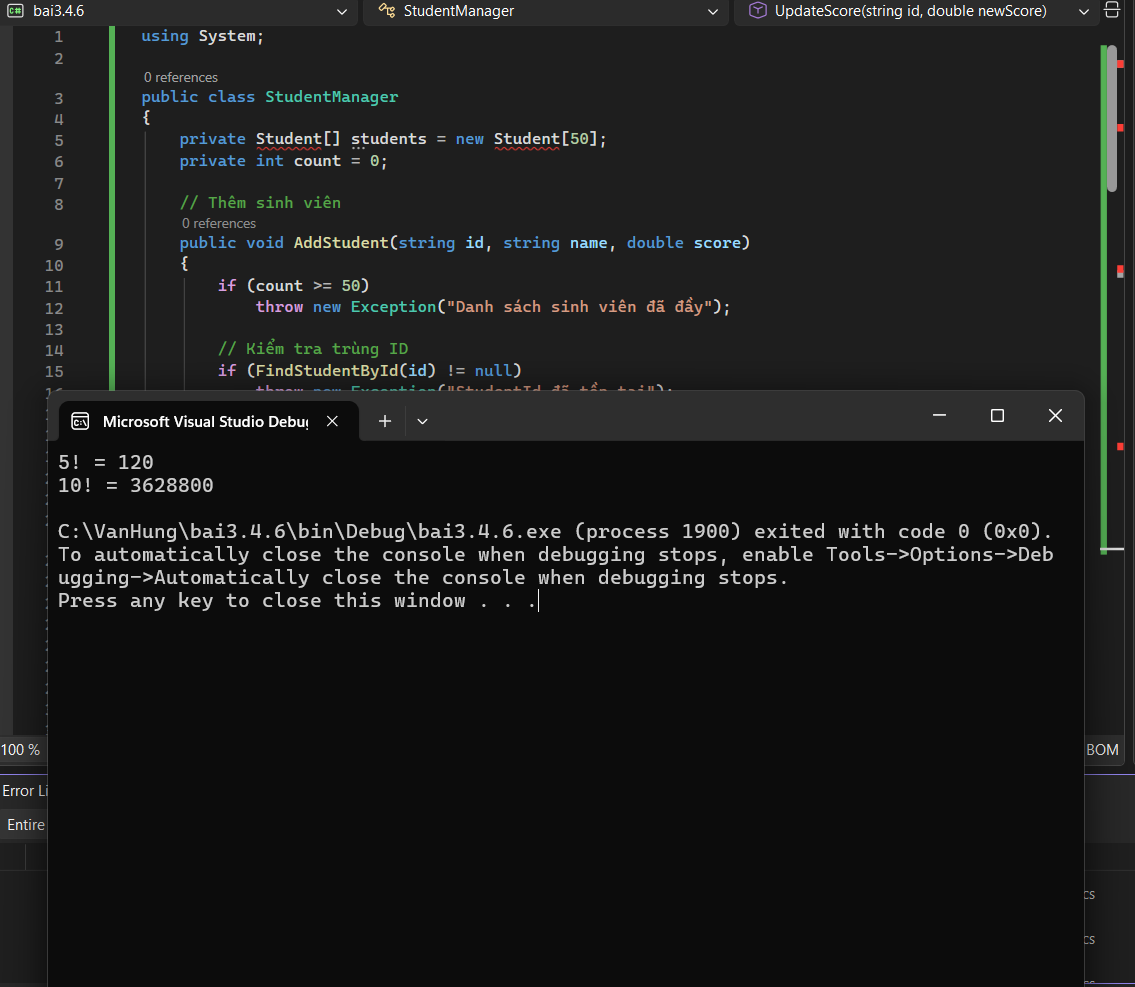
**📸 CHỨNG THỰC DỰ ÁN:**

**6.1 :**

Ảnh có chứa văn bản, đồ điện tử, ảnh chụp màn hình, phần mềm

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

6.2 :



6.3 :

### 🤔 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?**
2. \*\*Có thể dùng Listthay vì array không? Ưu điểm gì?\*\*
3. **Nếu muốn lưu dữ liệu xuống file sau khi chương trình thoát, làm sao?**

### 💫 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% |