# PHIẾU HỌC TẬP CHỦ ĐỘNG (PHT) - C# CĂN BẢN (TIẾP THEO)

**Môn học:** FIT4016: Thiết kế, Lập trình Back-End  
**Học phần:** C# Căn Bản (Chương 1 - Tổng Quan Ngôn Ngữ C#) - Phần 2

## PHIẾU HỌC TẬP C# [05] - XỬ LÝ NGOẠI LỆ (EXCEPTION HANDLING)

**Họ và tên:** Đoàn Quốc Bảo  
**Lớp:** CNTT 18-01  
**Video đã xem (Link):** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### **1. Kiến thức cốt lõi**

#### **1.1 Khái niệm Exception**

**Exception (Ngoại lệ):** Là những sự cố, lỗi xảy ra trong quá trình chương trình chạy.

| **Loại Exception** | **Mô tả** | **Ví dụ** |
| --- | --- | --- |
| **DivideByZeroException** | Chia cho 0 | int x = 10 / 0; |
| **IndexOutOfRangeException** | Chỉ số mảng ngoài phạm vi | int[] arr = { 1, 2 }; arr[5]; |
| **NullReferenceException** | Truy cập object null | string s = null; s.Length; |
| **FormatException** | Chuyển đổi định dạng sai | int x = int.Parse("abc"); |
| **ArgumentException** | Tham số không hợp lệ | int.Parse(null); |
| **IOException** | Lỗi nhập/xuất file | Đọc file không tồn tại |

#### **1.2 Cấu trúc try...catch...finally**

try

{

// Đoạn code có thể gặp lỗi

int x = int.Parse("abc");

}

catch (FormatException ex)

{

// Bắt lỗi cụ thể: định dạng sai

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

}

catch (Exception ex)

{

// Bắt lỗi chung (phải ở cuối cùng)

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

}

finally

{

// Luôn chạy, dù có lỗi hay không

Console.WriteLine("Kết thúc xử lý");

}

#### **1.3 Từ khóa throw**

public int Divide(int a, int b)

{

if (b == 0)

{

throw new ArgumentException("Mẫu số không được là 0!");

}

return a / b;

}

#### **1.4 Bảng So Sánh Các Cách Xử Lý**

| **Cách** | **Cú pháp** | **Khi nào dùng** |
| --- | --- | --- |
| **try...catch** | try { } catch { } | Bắt lỗi cụ thể |
| **try...catch...finally** | try { } catch { } finally { } | Bắt + dọn dẹp |
| **try...finally** | try { } finally { } | Chỉ dọn dẹp, không bắt |
| **throw** | throw new Exception() | Ném lỗi tự định nghĩa |
| **throw (re-throw)** | catch { throw; } | Ném lỗi lên cao hơn |

### **2. Ví dụ thực hành (Bắt buộc)**

**Kịch bản:** Tạo chương trình tính toán an toàn với xử lý lỗi.

#### **2.1 Code Khởi đầu:**

using System;

class Program

{

// TODO 1: Viết hàm ChuyenDoiSo

// - Tham số: string (chuỗi cần chuyển)

// - Trả về: int (số nguyên)

// - Mục đích: chuyển chuỗi sang số, xử lý lỗi

// - Yêu cầu: Sử dụng try...catch bắt FormatException

// (Viết code của bạn tại đây)

// TODO 2: Viết hàm ChiaHaiSo

// - Tham số: int a, int b

// - Trả về: int (kết quả chia)

// - Mục đích: chia hai số an toàn

// - Yêu cầu: Sử dụng try...catch bắt DivideByZeroException

// (Viết code của bạn tại đây)

// TODO 3: Viết hàm TimSoTrongMang

// - Tham số: int[] mang, int chIso

// - Trả về: int (phần tử ở vị trí)

// - Mục đích: truy cập mảng an toàn

// - Yêu cầu: Bắt IndexOutOfRangeException

// (Viết code của bạn tại đây)

static void Main()

{

Console.WriteLine("=== Xử Lý Ngoại Lệ An Toàn ===\n");

// TODO 4: Gọi hàm ChuyenDoiSo với các chuỗi khác nhau

// - "123" (thành công)

// - "abc" (lỗi)

// - "45.67" (có dấu phẩy)

// (Viết code gọi hàm của bạn tại đây)

// TODO 5: Gọi hàm ChiaHaiSo

// - 10 / 2 (thành công)

// - 10 / 0 (lỗi)

// - -5 / 3 (thành công)

// (Viết code gọi hàm của bạn tại đây)

// TODO 6: Gọi hàm TimSoTrongMang

// - Tạo mảng, truy cập chỉ số hợp lệ

// - Truy cập chỉ số ngoài phạm vi

int[] arr = { 10, 20, 30, 40, 50 };

// (Viết code gọi hàm của bạn tại đây)

// TODO 7: (Nâng cao) Viết try...catch...finally

// - Mở file (hoặc giả lập)

// - Đọc dữ liệu

// - finally: đóng file

try

{

Console.WriteLine("\nMở file...");

// string content = File.ReadAllText("file.txt");

throw new System.IO.FileNotFoundException("File không tìm thấy!");

}

catch (System.IO.FileNotFoundException ex)

{

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

}

finally

{

Console.WriteLine("Đóng file và dọn dẹp tài nguyên.");

}

}

}

#### **2.2 Ví dụ hoàn chỉnh từng hàm**

**Ví dụ TODO 1: Hàm ChuyenDoiSo**

public static int ChuyenDoiSo(string chuoi)

{

try

{

int ketqua = int.Parse(chuoi);

Console.WriteLine($"✓ Chuyển đổi thành công: '{chuoi}' → {ketqua}");

return ketqua;

}

catch (FormatException)

{

Console.WriteLine($"✗ Lỗi: '{chuoi}' không phải là số nguyên hợp lệ!");

return 0; // Giá trị mặc định

}

catch (OverflowException)

{

Console.WriteLine($"✗ Lỗi: '{chuoi}' quá lớn!");

return 0;

}

}

**Ví dụ TODO 2: Hàm ChiaHaiSo**

public static int ChiaHaiSo(int a, int b)

{

try

{

int ketqua = a / b;

Console.WriteLine($"✓ {a} / {b} = {ketqua}");

return ketqua;

}

catch (DivideByZeroException)

{

Console.WriteLine($"✗ Lỗi: Không được chia cho 0!");

return 0;

}

}

**Ví dụ TODO 3: Hàm TimSoTrongMang**

public static int TimSoTrongMang(int[] mang, int chiSo)

{

try

{

int so = mang[chiSo];

Console.WriteLine($"✓ Phần tử tại vị trí {chiSo} = {so}");

return so;

}

catch (IndexOutOfRangeException)

{

Console.WriteLine($"✗ Lỗi: Chỉ số {chiSo} ngoài phạm vi mảng!");

return -1;

}

}

#### **2.3 Hướng dẫn chi tiết từng TODO**

| **TODO** | **Gợi ý** | **Cách làm** |
| --- | --- | --- |
| **TODO 1** | Bắt FormatException | try { int.Parse() } catch (FormatException) |
| **TODO 2** | Bắt DivideByZeroException | try { a / b } catch (DivideByZeroException) |
| **TODO 3** | Bắt IndexOutOfRangeException | try { mang[i] } catch (IndexOutOfRangeException) |
| **TODO 4** | Gọi hàm 3 lần | ChuyenDoiSo("123"); ChuyenDoiSo("abc"); |
| **TODO 5** | Gọi hàm 3 lần | ChiaHaiSo(10, 2); ChiaHaiSo(10, 0); |
| **TODO 6** | Gọi hàm 2 lần | TimSoTrongMang(arr, 1); TimSoTrongMang(arr, 99); |
| **TODO 7** | try...catch...finally | Mở → đọc → finally đóng |

### **3. Yêu cầu Bằng chứng (Proof of Work)**

#### **A. Code đã hoàn thiện**

#### **B. Ảnh chụp màn hình Kết quả**

Chạy chương trình và chụp ảnh. Kết quả phải hiển thị:

using System;

class Program

{

    // TODO 1: Viết hàm ChuyenDoiSo

    static int ChuyenDoiSo(string chuoi)

    {

        try

        {

            int so = int.Parse(chuoi);

            Console.WriteLine($"Chuyển đổi thành công: {so}");

            return so;

        }

        catch (FormatException)

        {

            Console.WriteLine($"Lỗi định dạng! Không thể chuyển \"{chuoi}\" sang số nguyên.");

            return 0;

        }

}

    // TODO 2: Viết hàm ChiaHaiSo

    static int ChiaHaiSo(int a, int b)

    {

        try

        {

            int kq = a / b;

            Console.WriteLine($"{a} / {b} = {kq}");

            return kq;

        }

        catch (DivideByZeroException)

        {

            Console.WriteLine("Lỗi: Không thể chia cho 0!");

            return 0;

        }

}

    // TODO 3: Viết hàm TimSoTrongMang

    static int TimSoTrongMang(int[] mang, int chiSo)

    {

        try

        {

            int giaTri = mang[chiSo];

            Console.WriteLine($"Phần tử tại chỉ số {chiSo} là {giaTri}");

            return giaTri;

        }

        catch (IndexOutOfRangeException)

        {

            Console.WriteLine("Lỗi: Chỉ số vượt quá phạm vi mảng!");

            return 0;

        }

    }

    static void Main()

    {

        Console.WriteLine("=== Xử Lý Ngoại Lệ An Toàn ===\n");

        // TODO 4: Gọi hàm ChuyenDoiSo

        ChuyenDoiSo("123");

        ChuyenDoiSo("abc");

        ChuyenDoiSo("45.67");

        Console.WriteLine();

        // TODO 5: Gọi hàm ChiaHaiSo

        ChiaHaiSo(10, 2);

        ChiaHaiSo(10, 0);

        ChiaHaiSo(-5, 5);

        Console.WriteLine();

        // TODO 6: Gọi hàm TimSoTrongMang

        int[] arr = { 10, 20, 30, 40, 50 };

        TimSoTrongMang(arr, 2);   // hợp lệ

        TimSoTrongMang(arr, 10);  // lỗi

        // TODO 7: try...catch...finally

        try

        {

            Console.WriteLine("\nMở file...");

            throw new System.IO.FileNotFoundException("File không tìm thấy!");

        }

        catch (System.IO.FileNotFoundException ex)

        {

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

        }

        finally

        {

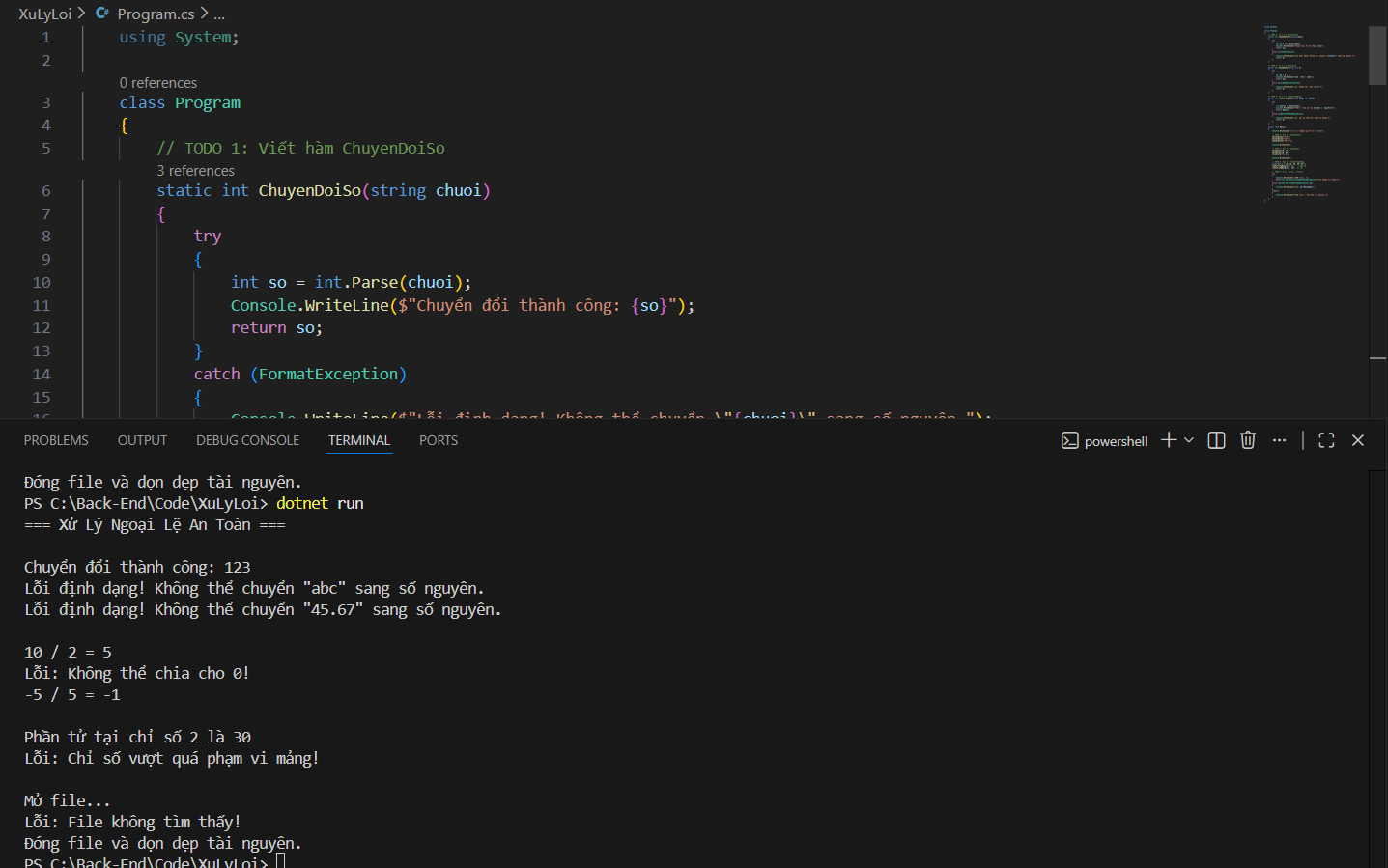
            Console.WriteLine("Đóng file và dọn dẹp tài nguyên.");

        }

    }

}

Kết quả:



* Chuyển đổi chuỗi thành công
* Lỗi chuyển đổi (không phải số)
* Phép chia thành công
* Lỗi chia cho 0
* Truy cập mảng thành công
* Lỗi truy cập mảng ngoài phạm vi
* Xử lý finally

### **4. Câu hỏi Phản biện (Bắt buộc)**

**Một điều tôi chưa hiểu rõ:**

Nếu trong các hàm trên chúng ta không sử dụng try...catch mà để chương trình tự phát sinh lỗi, thì điều gì sẽ xảy ra đối với trải nghiệm người dùng và độ ổn định của chương trình? Việc sử dụng try...catch có thể gây ra nhược điểm gì nếu lạm dụng?

## PHIẾU HỌC TẬP C# [06] - COLLECTIONS VÀ GENERIC COLLECTIONS

**Họ và tên:** Đoàn Quốc Bảo  
**Lớp:** CNTT 18-01  
**Video đã xem (Link):** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### **1. Kiến thức cốt lõi**

#### **1.1 Các loại Collection**

| **Collection** | **Cú pháp** | **Đặc điểm** | **Ví dụ** |
| --- | --- | --- | --- |
| \*\*List\*\* | List<int> list = new(); | Danh sách có thứ tự, có thể thay đổi kích thước | list.Add(1); list[0]; |
| **Dictionary<K,V>** | Dictionary<string, int> dict = new(); | Lưu cặp key-value | dict["key"] = 10; |
| \*\*HashSet\*\* | HashSet<int> set = new(); | Không có phần tử trùng lặp | set.Add(1); |
| \*\*Queue\*\* | Queue<string> queue = new(); | FIFO (First In First Out) | queue.Enqueue("A"); |
| \*\*Stack\*\* | Stack<int> stack = new(); | LIFO (Last In First Out) | stack.Push(1); |
| **Array** | int[] arr = new int[5]; | Kích thước cố định | arr[0] = 10; |

#### \*\*1.2 Phương thức thường dùng của List\*\*

List<int> numbers = new List<int>();

// Thêm phần tử

numbers.Add(10); // Thêm vào cuối

numbers.Insert(0, 5); // Thêm vào vị trí 0

// Xoá phần tử

numbers.Remove(10); // Xoá phần tử = 10

numbers.RemoveAt(0); // Xoá tại vị trí 0

numbers.Clear(); // Xoá hết

// Tìm kiếm

bool contains = numbers.Contains(10); // Kiểm tra có tồn tại

int index = numbers.IndexOf(10); // Tìm vị trí

// Duyệt

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

{

Console.WriteLine(numbers[i]);

}

foreach (int n in numbers)

{

Console.WriteLine(n);

}

// LINQ

var evenNumbers = numbers.Where(n => n % 2 == 0).ToList();

#### **1.3 Phương thức thường dùng của Dictionary<K,V>**

Dictionary<string, int> ages = new();

// Thêm

ages["Alice"] = 25; // Cách 1

ages.Add("Bob", 30); // Cách 2

// Lấy giá trị

int aliceAge = ages["Alice"]; // Nếu không tồn tại sẽ lỗi

if (ages.TryGetValue("Charlie", out int charlieAge))

{

Console.WriteLine(charlieAge);

}

// Xoá

ages.Remove("Alice");

ages.Clear();

// Duyệt

foreach (var pair in ages)

{

Console.WriteLine($"{pair.Key}: {pair.Value}");

}

foreach (string key in ages.Keys)

{

Console.WriteLine(key);

}

#### **1.4 Queue vs Stack**

// Queue: FIFO (Hàng chờ)

Queue<string> queue = new();

queue.Enqueue("A"); // A

queue.Enqueue("B"); // A, B

queue.Enqueue("C"); // A, B, C

string first = queue.Dequeue(); // Lấy A, còn B, C

// Stack: LIFO (Xếp chồng)

Stack<string> stack = new();

stack.Push("A"); // A

stack.Push("B"); // A, B

stack.Push("C"); // A, B, C

string last = stack.Pop(); // Lấy C, còn A, B

### **2. Ví dụ thực hành (Bắt buộc)**

**Kịch bản:** Quản lý danh sách sinh viên, điểm, và lịch sử thao tác.

#### **2.1 Code Khởi đầu:**

using System;

using System.Collections.Generic;

using System.Linq;

class Program

{

// TODO 1: Khai báo List<string> để lưu tên sinh viên

// List<string> danhSachTen = new();

// TODO 2: Khai báo Dictionary<string, double> để lưu điểm

// Key: Tên sinh viên, Value: Điểm trung bình

// Dictionary<string, double> bangDiem = new();

// TODO 3: Khai báo Stack<string> để lưu lịch sử thao tác

// Stack<string> lichSu = new();

// TODO 4: Viết hàm ThemSinhVien

// - Tham số: tên, điểm

// - Thêm vào List và Dictionary

// - Ghi lịch sử: "Thêm Nguyễn Văn A"

// (Viết code của bạn tại đây)

// TODO 5: Viết hàm XoaSinhVien

// - Tham số: tên

// - Xoá từ List và Dictionary

// - Ghi lịch sử: "Xoá Nguyễn Văn A"

// (Viết code của bạn tại đây)

// TODO 6: Viết hàm InDanhSach

// - Duyệt List và in tên, điểm

// - Sử dụng Dictionary để lấy điểm

// (Viết code của bạn tại đây)

// TODO 7: Viết hàm TimSinhVienTheoTen

// - Tham số: tên cần tìm

// - Trả về: điểm (hoặc -1 nếu không tìm thấy)

// - Sử dụng Dictionary.TryGetValue()

// (Viết code của bạn tại đây)

// TODO 8: Viết hàm HienThiLichSu

// - In tất cả các thao tác từ Stack

// - Stack.Pop() để lấy từ mới nhất

// (Viết code của bạn tại đây)

static void Main()

{

Console.WriteLine("=== Quản Lý Sinh Viên với Collections ===\n");

List<string> danhSachTen = new();

Dictionary<string, double> bangDiem = new();

Stack<string> lichSu = new();

// TODO 9: Thêm 3 sinh viên

// (Viết code gọi hàm của bạn tại đây)

// TODO 10: In danh sách

// (Viết code gọi hàm của bạn tại đây)

// TODO 11: Tìm sinh viên

// (Viết code gọi hàm của bạn tại đây)

// TODO 12: Xoá 1 sinh viên

// (Viết code gọi hàm của bạn tại đây)

// TODO 13: In danh sách lại

// (Viết code gọi hàm của bạn tại đây)

// TODO 14: Hiển thị lịch sử

// (Viết code gọi hàm của bạn tại đây)

}

}

#### **2.2 Ví dụ hoàn chỉnh**

**Hàm TODO 4: ThemSinhVien**

public static void ThemSinhVien(List<string> ten, Dictionary<string, double> diem,

Stack<string> lichSu, string hoTen, double diemTB)

{

if (!ten.Contains(hoTen))

{

ten.Add(hoTen);

diem[hoTen] = diemTB;

lichSu.Push($"✓ Thêm: {hoTen} (Điểm: {diemTB:F2})");

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

}

else

{

Console.WriteLine($"✗ Sinh viên '{hoTen}' đã tồn tại!");

}

}

**Hàm TODO 6: InDanhSach**

public static void InDanhSach(List<string> ten, Dictionary<string, double> diem)

{

Console.WriteLine("\n╔════════════════════════════════════╗");

Console.WriteLine("║ DANH SÁCH SINH VIÊN ║");

Console.WriteLine("╠════════════════════════════════════╣");

Console.WriteLine("║ TT │ Tên │ Điểm ║");

Console.WriteLine("╠════════════════════════════════════╣");

for (int i = 0; i < ten.Count; i++)

{

string tenSV = ten[i];

double diemSV = diem[tenSV];

Console.WriteLine($"║ {i+1} │ {tenSV,-15} │ {diemSV:F2} ║");

}

if (ten.Count == 0)

Console.WriteLine("║ Danh sách trống! ║");

Console.WriteLine("╚════════════════════════════════════╝");

}

**Hàm TODO 7: TimSinhVienTheoTen**

public static double TimSinhVienTheoTen(Dictionary<string, double> diem, string tenCanTim)

{

if (diem.TryGetValue(tenCanTim, out double diemTim))

{

Console.WriteLine($"✓ Tìm thấy: {tenCanTim} - Điểm: {diemTim:F2}");

return diemTim;

}

else

{

Console.WriteLine($"✗ Không tìm thấy sinh viên '{tenCanTim}'!");

return -1;

}

}

### **3. Yêu cầu Bằng chứng (Proof of Work)**

#### **A. Code đã hoàn thiện**

using System;

using System.Collections.Generic;

using System.Linq;

class Program

{

    // TODO 1: Khai báo List<string> để lưu tên sinh viên

static List<string> danhSachTen = new();

    // TODO 2: Khai báo Dictionary<string, double> để lưu điểm

    // Key: Tên sinh viên, Value: Điểm trung bình

static Dictionary<string, double> bangDiem = new();

    // TODO 3: Khai báo Stack<string> để lưu lịch sử thao tác

static Stack<string> lichSu = new();

    // TODO 4: Viết hàm ThemSinhVien

    static void ThemSinhVien(string ten, double diem)

    {

        danhSachTen.Add(ten);

        bangDiem[ten] = diem;

        lichSu.Push($"Thêm {ten}");

}

    // TODO 5: Viết hàm XoaSinhVien

    static void XoaSinhVien(string ten)

    {

        if (danhSachTen.Remove(ten))

        {

            bangDiem.Remove(ten);

            lichSu.Push($"Xoá {ten}");

        }

        else

        {

            Console.WriteLine($"Không tìm thấy sinh viên: {ten}");

        }

}

    // TODO 6: Viết hàm InDanhSach

    static void InDanhSach()

    {

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

        foreach (string ten in danhSachTen)

        {

            Console.WriteLine($"- {ten}: {bangDiem[ten]} điểm");

        }

        Console.WriteLine();

}

    // TODO 7: Viết hàm TimSinhVienTheoTen

    static double TimSinhVienTheoTen(string ten)

    {

        if (bangDiem.TryGetValue(ten, out double diem))

        {

            return diem;

        }

        return -1;

}

    // TODO 8: Viết hàm HienThiLichSu

    static void HienThiLichSu()

    {

        Console.WriteLine("Lịch sử thao tác:");

        while (lichSu.Count > 0)

        {

            Console.WriteLine(lichSu.Pop());

        }

    }

    static void Main()

    {

        Console.WriteLine("=== Quản Lý Sinh Viên với Collections ===\n");

        // TODO 9: Thêm 3 sinh viên

        ThemSinhVien("Nguyễn Văn A", 8.5);

        ThemSinhVien("Trần Thị B", 7.8);

        ThemSinhVien("Lê Văn C", 9.0);

        // TODO 10: In danh sách

        InDanhSach();

        // TODO 11: Tìm sinh viên

        string tenCanTim = "Trần Thị B";

        double ketQua = TimSinhVienTheoTen(tenCanTim);

        if (ketQua != -1)

        {

            Console.WriteLine($"Tìm thấy {tenCanTim}: {ketQua} điểm\n");

        }

        else

        {

            Console.WriteLine($"Không tìm thấy {tenCanTim}\n");

        }

        // TODO 12: Xoá 1 sinh viên

        XoaSinhVien("Nguyễn Văn A");

        // TODO 13: In danh sách lại

        InDanhSach();

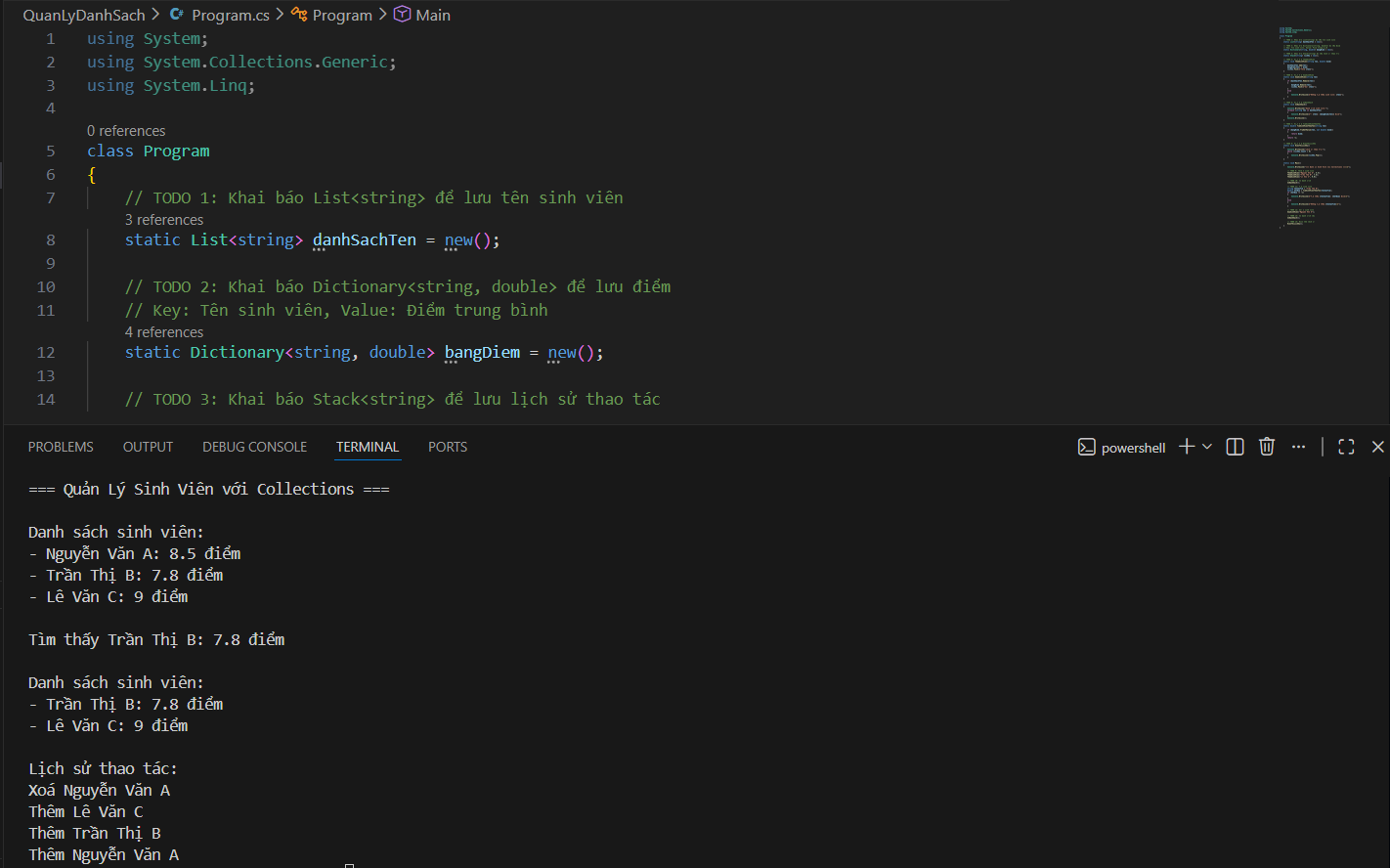
        // TODO 14: Hiển thị lịch sử

        HienThiLichSu();

    }

}

#### **Ảnh chụp màn hình Kết quả**



* Thêm 3 sinh viên thành công
* In danh sách
* Tìm kiếm sinh viên
* Xoá sinh viên
* Danh sách sau khi xoá
* Lịch sử thao tác

### **4. Câu hỏi Phản biện (Bắt buộc)**

**Một điều tôi chưa hiểu rõ:**

- Nếu danh sách sinh viên có hai người trùng tên, việc sử dụng Dictionary<string, double> với tên làm khóa sẽ gây ra vấn đề gì?

- Theo bạn, cần thay đổi cấu trúc dữ liệu như thế nào để chương trình vẫn quản lý chính xác từng sinh viên?

## PHIẾU HỌC TẬP C# [07] - DELEGATE, ANONYMOUS METHOD VÀ LAMBDA EXPRESSION

**Họ và tên:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**Lớp:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**Video đã xem (Link):** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### **1. Kiến thức cốt lõi**

#### **1.1 Delegate là gì?**

**Delegate:** Là một kiểu dữ liệu đại diện cho một hàm.

// Khai báo delegate

public delegate int MathOperation(int a, int b);

// Tạo hàm phù hợp

public static int Add(int a, int b)

{

return a + b;

}

public static int Multiply(int a, int b)

{

return a \* b;

}

// Sử dụng delegate

MathOperation operation = Add;

int result = operation(5, 3); // Gọi Add(5, 3) = 8

operation = Multiply;

result = operation(5, 3); // Gọi Multiply(5, 3) = 15

#### **1.2 Anonymous Method**

// Khai báo delegate

delegate int Calculate(int x, int y);

// Sử dụng anonymous method

Calculate calc = delegate(int x, int y)

{

return x + y;

};

int result = calc(10, 20); // 30

#### **1.3 Lambda Expression**

**Lambda:** Cách viết ngắn gọn hơn của anonymous method.

// Cú pháp: (tham số) => biểu thức

// Ví dụ 1: Một tham số

delegate int Double(int x);

Double doubleFunc = x => x \* 2;

Console.WriteLine(doubleFunc(5)); // 10

// Ví dụ 2: Hai tham số

delegate int Add(int a, int b);

Add addFunc = (a, b) => a + b;

Console.WriteLine(addFunc(5, 3)); // 8

// Ví dụ 3: Có body

Func<int, int, int> multiply = (a, b) =>

{

int result = a \* b;

return result;

};

Console.WriteLine(multiply(5, 3)); // 15

// Ví dụ 4: Không tham số

Action greet = () => Console.WriteLine("Xin chào!");

greet(); // In: Xin chào!

// Ví dụ 5: Với LINQ

List<int> numbers = new() { 1, 2, 3, 4, 5 };

var evens = numbers.Where(n => n % 2 == 0).ToList(); // [2, 4]

var squares = numbers.Select(n => n \* n).ToList(); // [1, 4, 9, 16, 25]

#### **1.4 Delegate vs Action vs Func**

| **Loại** | **Khai báo** | **Trả về** | **Ví dụ** |
| --- | --- | --- | --- |
| **Delegate tự định nghĩa** | delegate int MyDelegate(int); | Tùy ý | MyDelegate f = x => x \* 2; |
| \*\*Action\*\* | Action<int> action; | void | Action<int> print = x => Console.WriteLine(x); |
| **Func<T, TResult>** | Func<int, int> func; | TResult | Func<int, int> square = x => x \* x; |
| \*\*Predicate\*\* | Predicate<int> check; | bool | Predicate<int> isEven = n => n % 2 == 0; |

### **2. Ví dụ thực hành (Bắt buộc)**

**Kịch bản:** Xử lý danh sách sinh viên với các hàm lambda.

#### **2.1 Code Khởi đầu:**

using System;

using System.Collections.Generic;

using System.Linq;

class Program

{

// TODO 1: Khai báo delegate kiểm tra điều kiện

// delegate bool KiemTra(double diem);

// TODO 2: Khai báo delegate xử lý dữ liệu

// delegate void XuLy(string ten, double diem);

// TODO 3: Viết hàm LocSinhVien

// - Tham số: List<(string, double)>, Predicate

// - Lọc sinh viên theo điều kiện

// - Trả về: List<(string, double)> kết quả

// (Viết code của bạn tại đây)

// TODO 4: Viết hàm DuyetDanhSach

// - Tham số: List<(string, double)>, Action

// - Duyệt từng sinh viên, gọi action

// - Mục đích: in, tính toán, v.v.

// (Viết code của bạn tại đây)

// TODO 5: Viết hàm TinhDiemTrungBinh

// - Tham số: List<double> (danh sách điểm)

// - Sử dụng lambda để cộng các điểm

// - Trả về: double (điểm TB)

// (Viết code của bạn tại đây)

static void Main()

{

Console.WriteLine("=== Delegate, Lambda, và LINQ ===\n");

// Dữ liệu: (Tên, Điểm)

List<(string, double)> danhSach = new()

{

("Nguyễn Văn A", 8.5),

("Trần Thị B", 7.2),

("Lê Văn C", 5.8),

("Phạm Thị D", 9.0),

("Hoàng Văn E", 6.5)

};

Console.WriteLine("=== Danh sách gốc ===");

// TODO 6: In toàn bộ danh sách

// Dùng foreach hoặc hàm DuyetDanhSach

foreach (var (ten, diem) in danhSach)

{

Console.WriteLine($"{ten,-20} │ {diem:F2}");

}

// TODO 7: Lọc sinh viên điểm >= 7.0

// Sử dụng lambda: n => n.Item2 >= 7.0

var gioi = /\* Gọi hàm LocSinhVien \*/;

Console.WriteLine("\n=== Sinh viên loại Giỏi (>= 7.0) ===");

foreach (var (ten, diem) in gioi)

{

Console.WriteLine($"{ten,-20} │ {diem:F2}");

}

// TODO 8: Lọc sinh viên điểm < 6.0

// Sử dụng lambda: n => n.Item2 < 6.0

var yeu = /\* Gọi hàm LocSinhVien \*/;

Console.WriteLine("\n=== Sinh viên loại Yếu (< 6.0) ===");

foreach (var (ten, diem) in yeu)

{

Console.WriteLine($"{ten,-20} │ {diem:F2}");

}

// TODO 9: Sắp xếp theo điểm giảm dần

// Sử dụng LINQ: .OrderByDescending(n => n.Item2)

var sapXep = /\* Thực hiện sắp xếp \*/;

Console.WriteLine("\n=== Xếp theo điểm giảm dần ===");

foreach (var (ten, diem) in sapXep)

{

Console.WriteLine($"{ten,-20} │ {diem:F2}");

}

// TODO 10: Lấy top 2 sinh viên

// Sử dụng LINQ: .Take(2)

var top2 = /\* Thực hiện \*/;

Console.WriteLine("\n=== Top 2 Sinh Viên ===");

foreach (var (ten, diem) in top2)

{

Console.WriteLine($"{ten,-20} │ {diem:F2}");

}

// TODO 11: Tính điểm trung bình

// Sử dụng lambda và Average()

double diemTB = /\* Gọi hàm hoặc LINQ \*/;

Console.WriteLine($"\n=== Điểm Trung Bình: {diemTB:F2} ===");

// TODO 12: Tìm sinh viên có điểm cao nhất

// Sử dụng LINQ: .MaxBy(n => n.Item2)

var topDiem = /\* Thực hiện \*/;

Console.WriteLine($"\n=== Sinh viên có điểm cao nhất: {topDiem.Item1} ({topDiem.Item2:F2}) ===");

}

}

#### **2.2 Ví dụ hoàn chỉnh**

**Hàm TODO 3: LocSinhVien**

public static List<(string, double)> LocSinhVien(

List<(string, double)> danhSach,

Predicate<(string, double)> condition)

{

List<(string, double)> ketqua = new();

foreach (var item in danhSach)

{

if (condition(item))

ketqua.Add(item);

}

return ketqua;

}

// Cách sử dụng:

var gioi = LocSinhVien(danhSach, sv => sv.Item2 >= 7.0);

**Hàm TODO 5: Tính TB với Lambda**

Func<List<double>, double> tinhTrungBinh = (diem) =>

{

double tong = 0;

foreach (double d in diem)

tong += d;

return tong / diem.Count;

};

// Hoặc dùng LINQ:

double diemTB = danhSach.Select(sv => sv.Item2).Average();

**Các LINQ Query thường dùng:**

// Lọc

var filtered = danhSach.Where(sv => sv.Item2 >= 7.0).ToList();

// Sắp xếp

var sorted = danhSach.OrderByDescending(sv => sv.Item2).ToList();

// Lấy n phần tử

var topN = danhSach.Take(2).ToList();

// Bỏ n phần tử

var skip = danhSach.Skip(2).ToList();

// Chuyển đổi

var names = danhSach.Select(sv => sv.Item1).ToList();

// Nhóm

var grouped = danhSach.GroupBy(sv => Math.Floor(sv.Item2)).ToList();

### **3. Yêu cầu Bằng chứng (Proof of Work)**

#### **A. Code đã hoàn thiện**

// (Dán code đầy đủ của bạn)

#### **B. Ảnh chụp màn hình Kết quả**

* Danh sách gốc
* Lọc sinh viên giỏi
* Lọc sinh viên yếu
* Sắp xếp theo điểm
* Top 2 sinh viên
* Điểm trung bình
* Sinh viên có điểm cao nhất

(Dán ảnh chụp màn hình)

### **4. Câu hỏi Phản biện (Bắt buộc)**

**Một điều tôi chưa hiểu rõ:**

## PHIẾU HỌC TẬP C# [08] - ASYNCHRONOUS PROGRAMMING (ASYNC/AWAIT)

**Họ và tên:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**Lớp:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**Video đã xem (Link):** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### **1. Kiến thức cốt lõi**

#### **1.1 Khái niệm Async/Await**

**Async/Await:** Cho phép chương trình chạy các tác vụ nặng mà không block (chặn) giao diện.

// Hàm bình thường (chặn)

public static void CongTac1()

{

System.Threading.Thread.Sleep(5000); // Chờ 5 giây

Console.WriteLine("Xong!");

// Trong 5 giây này, chương trình bị chặn, không làm gì được

}

// Hàm async (không chặn)

public static async Task CongTacAsync()

{

await Task.Delay(5000); // Chờ 5 giây

Console.WriteLine("Xong!");

// Trong 5 giây này, chương trình vẫn có thể làm việc khác

}

#### \*\*1.2 Task và Task\*\*

// Task: Công việc không trả về giá trị

public async Task XuLyDuLieu()

{

await Task.Delay(1000);

Console.WriteLine("Xử lý xong");

}

// Task<T>: Công việc trả về giá trị

public async Task<int> TinhToan()

{

await Task.Delay(1000);

return 10 + 20; // Trả về 30

}

// Sử dụng

await XuLyDuLieu(); // Chờ xong

int result = await TinhToan(); // Chờ xong, lấy kết quả

#### **1.3 Cú pháp async/await**

// Hàm async phải:

// 1. Khai báo với từ khóa async

// 2. Trả về Task hoặc Task<T>

// 3. Có ít nhất 1 await

public async Task<string> DocFile(string path)

{

// await: chờ tác vụ hoàn thành

string content = await File.ReadAllTextAsync(path);

return content;

}

// Main cũng có thể async (C# 7.1+)

static async Task Main(string[] args)

{

string content = await DocFile("file.txt");

Console.WriteLine(content);

}

#### **1.4 Xử lý lỗi trong async**

public async Task<int> DivideAsync(int a, int b)

{

try

{

if (b == 0)

throw new DivideByZeroException();

await Task.Delay(1000);

return a / b;

}

catch (DivideByZeroException ex)

{

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

return 0;

}

finally

{

Console.WriteLine("Xử lý xong");

}

}

#### **1.5 Bảng so sánh Sync vs Async**

| **Khía cạnh** | **Synchronous** | **Asynchronous** |
| --- | --- | --- |
| **Khai báo** | public void Method() | public async Task Method() |
| **Trả về** | Giá trị trực tiếp | Task / Task |
| **Chờ kết quả** | Gọi thường | await |
| **Khi nào dùng** | Tác vụ nhanh | Tác vụ chậm (I/O, mạng) |
| **Ưu điểm** | Đơn giản | Không chặn giao diện |
| **Nhược điểm** | Chặn giao diện | Phức tạp hơn |

### **2. Ví dụ thực hành (Bắt buộc)**

**Kịch bản:** Tạo chương trình tải dữ liệu từ nhiều nguồn mà không chặn giao diện.

#### **2.1 Code Khởi đầu:**

using System;

using System.Threading.Tasks;

using System.Collections.Generic;

class Program

{

// TODO 1: Viết hàm SimulateDownload

// - Tham số: string url, int delayMs

// - Trả về: Task<string> (dữ liệu tải về)

// - Mục đích: Giả lập tải dữ liệu từ URL

// - Sử dụng: await Task.Delay(delayMs);

// - Trả về: $"Dữ liệu từ {url}"

// (Viết code của bạn tại đây)

// TODO 2: Viết hàm TaiTuSinhVien

// - Tham số: string maSV

// - Trả về: Task<string> (tên sinh viên)

// - Mục đích: Lấy dữ liệu sinh viên (delay 2 giây)

// (Viết code của bạn tại đây)

// TODO 3: Viết hàm TaiDiem

// - Tham số: string maSV

// - Trả về: Task<double> (điểm)

// - Mục đích: Lấy điểm (delay 1.5 giây)

// (Viết code của bạn tại đây)

// TODO 4: Viết hàm TaiDonVi

// - Tham số: string maSV

// - Trả về: Task<string> (đơn vị)

// - Mục đích: Lấy đơn vị (delay 1 giây)

// (Viết code của bạn tại đây)

// TODO 5: Viết hàm LayThongTinSinhVien

// - Tham số: string maSV

// - Trả về: Task<(string, double, string)>

// - Mục đích: Gọi 3 hàm async song song (parallel)

// - Sử dụng: Task.WhenAll() hoặc await từng cái

// (Viết code của bạn tại đây)

static async Task Main()

{

Console.WriteLine("=== Asynchronous Programming ===\n");

// TODO 6: Gọi hàm SimulateDownload 3 lần

// Sử dụng async/await

Console.WriteLine("--- Tải tuần tự ---");

var start1 = DateTime.Now;

// (Viết code gọi 3 lần sequential)

Console.WriteLine($"Thời gian tuần tự: {(DateTime.Now - start1).TotalSeconds:F2}s");

// TODO 7: Gọi hàm SimulateDownload 3 lần song song

// Sử dụng Task.WhenAll()

Console.WriteLine("\n--- Tải song song ---");

var start2 = DateTime.Now;

// (Viết code gọi 3 lần parallel)

Console.WriteLine($"Thời gian song song: {(DateTime.Now - start2).TotalSeconds:F2}s");

// TODO 8: Gọi LayThongTinSinhVien

Console.WriteLine("\n--- Lấy thông tin sinh viên ---");

var (ten, diem, donvi) = await /\* Gọi hàm \*/;

Console.WriteLine($"Tên: {ten}");

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

Console.WriteLine($"Đơn vị: {donvi}");

}

}

#### **2.2 Ví dụ hoàn chỉnh**

**Hàm TODO 1: SimulateDownload**

public static async Task<string> SimulateDownload(string url, int delayMs)

{

Console.WriteLine($"🔄 Đang tải từ {url}...");

await Task.Delay(delayMs);

Console.WriteLine($"✓ Xong: {url}");

return $"Dữ liệu từ {url}";

}

**Hàm TODO 5: LayThongTinSinhVien (Tuần tự)**

public static async Task<(string, double, string)> LayThongTinSinhVien(string maSV)

{

string ten = await TaiTuSinhVien(maSV);

double diem = await TaiDiem(maSV);

string donvi = await TaiDonVi(maSV);

return (ten, diem, donvi);

}

**Hàm TODO 5: LayThongTinSinhVien (Song song)**

public static async Task<(string, double, string)> LayThongTinSinhVienParallel(string maSV)

{

// Khởi động 3 tác vụ cùng lúc

var tenTask = TaiTuSinhVien(maSV);

var diemTask = TaiDiem(maSV);

var donviTask = TaiDonVi(maSV);

// Chờ tất cả hoàn thành

await Task.WhenAll(tenTask, diemTask, donviTask);

// Lấy kết quả

string ten = await tenTask;

double diem = await diemTask;

string donvi = await donviTask;

return (ten, diem, donvi);

}

**TODO 6 & 7: Tải tuần tự vs Song song**

// Tuần tự (tổng ~4.5 giây)

Console.WriteLine("--- Tải tuần tự ---");

var start1 = DateTime.Now;

string data1 = await SimulateDownload("URL1", 1500);

string data2 = await SimulateDownload("URL2", 1500);

string data3 = await SimulateDownload("URL3", 1500);

Console.WriteLine($"Thời gian: {(DateTime.Now - start1).TotalSeconds:F2}s");

// Song song (tổng ~1.5 giây)

Console.WriteLine("\n--- Tải song song ---");

var start2 = DateTime.Now;

var tasks = new List<Task<string>>

{

SimulateDownload("URL1", 1500),

SimulateDownload("URL2", 1500),

SimulateDownload("URL3", 1500)

};

string[] results = await Task.WhenAll(tasks);

Console.WriteLine($"Thời gian: {(DateTime.Now - start2).TotalSeconds:F2}s");

### **3. Yêu cầu Bằng chứng (Proof of Work)**

#### **A. Code đã hoàn thiện**

// (Dán code đầy đủ của bạn)

#### **B. Ảnh chụp màn hình Kết quả**

* Tải dữ liệu tuần tự (thời gian)
* Tải dữ liệu song song (thời gian)
* Lấy thông tin sinh viên
* Hiển thị kết quả cuối cùng

(Dán ảnh chụp màn hình)

### **4. Câu hỏi Phản biện (Bắt buộc)**

**Một điều tôi chưa hiểu rõ:**

## 📋 TÓM TẮT CHƯƠNG 1 - C# CĂN BẢN

### **Những gì đã học (8 Phiếu)**

| **Phiếu** | **Nội dung** | **Kỹ năng** |
| --- | --- | --- |
| **[01]** | C# là gì, khái niệm cốt lõi | Hiểu ngôn ngữ, cài đặt môi trường |
| **[02]** | Cấu trúc điều khiển, vòng lặp | Kiểm soát luồng chương trình |
| **[03]** | Hàm, mảng | Tổ chức code, xử lý dữ liệu |
| **[04]** | OOP, Class, Object | Lập trình hướng đối tượng |
| **[05]** | Xử lý ngoại lệ | Bắt và xử lý lỗi |
| **[06]** | Collections (List, Dict, Stack) | Quản lý dữ liệu phức tạp |
| **[07]** | Delegate, Lambda, LINQ | Lập trình hàm, truy vấn dữ liệu |
| **[08]** | Async/Await | Lập trình bất đồng bộ |

### **Kiến thức kỳ vọng sau Chương 1**

✅ **Nền tảng C#:**

* Biến, kiểu dữ liệu, hằng
* Cấu trúc điều khiển (if, switch)
* Vòng lặp (for, while, foreach)
* Hàm (khai báo, gọi, tham số)
* Mảng (một chiều, nhiều chiều)

✅ **OOP trong C#:**

* Class, Object, Properties, Methods
* Constructor, Destructor
* Access Modifiers (public, private, protected)
* Encapsulation, Inheritance, Polymorphism

✅ **Collections & LINQ:**

* List, Dictionary, Stack, Queue
* Phương thức: Add, Remove, Find, Sort
* LINQ: Where, Select, OrderBy, Take

✅ **Xử lý lỗi:**

* try...catch...finally
* Exception handling
* Custom Exception

✅ **Delegate & Lambda:**

* Delegate, Anonymous Method
* Lambda Expression (=>)
* Action, Func, Predicate
* Ứng dụng trong LINQ

✅ **Async/Await:**

* Task, Task
* Async method, Await
* Parallel execution
* Error handling trong async

### **Kỹ năng thực hành**

✅ **Thành thạo:**

* Viết chương trình C# từ đơn giản đến phức tạp
* Sử dụng Visual Studio / VS Code
* Debug chương trình
* Đọc và hiểu code
* Giải quyết bài toán lập trình

✅ **Hiểu sâu:**

* Khi nào dùng cấu trúc nào
* Performance: Collections nào tốt nhất
* Async vs Sync
* Design patterns cơ bản

## 📚 TÀI LIỆU THAM KHẢO

**Bắt buộc đọc:**

* Freeman, Adam - "Pro ASP.NET Core 6" (Chương 1-3)
* Tài liệu Khoa cung cấp (Tiếng Việt)

**Tham khảo thêm:**

* Microsoft Docs: <https://docs.microsoft.com/en-us/dotnet/csharp/>
* C# Language Reference: <https://www.w3schools.com/cs/>
* LINQ Tutorial: <https://www.tutorialsteacher.com/linq>
* Async/Await: <https://docs.microsoft.com/en-us/dotnet/csharp/asynchronous-programming>

**Video học:**

* Microsoft Learn C#: <https://learn.microsoft.com/en-us/training/paths/csharp-first-steps/>
* YouTube: C# Tutorial for Beginners

**Chúc bạn hoàn thành xuất sắc Chương 1! 🚀**