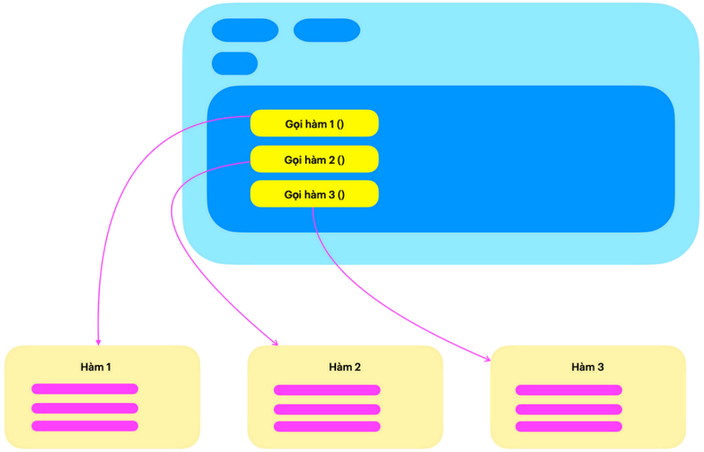
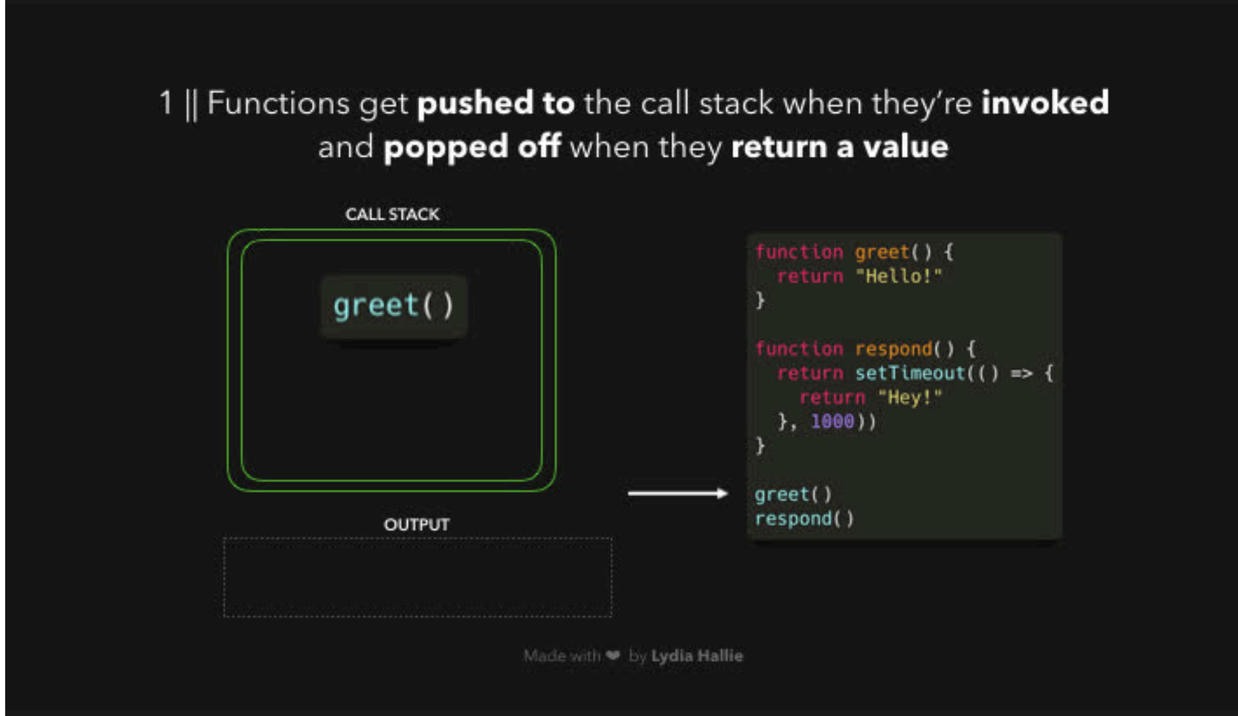
NỘI DUNG

- Khái niệm về hàm
- Cú pháp khai báo hàm - gọi hàm
- Cách xây dựng hàm tách hàm
- Bài tập tổng hợp



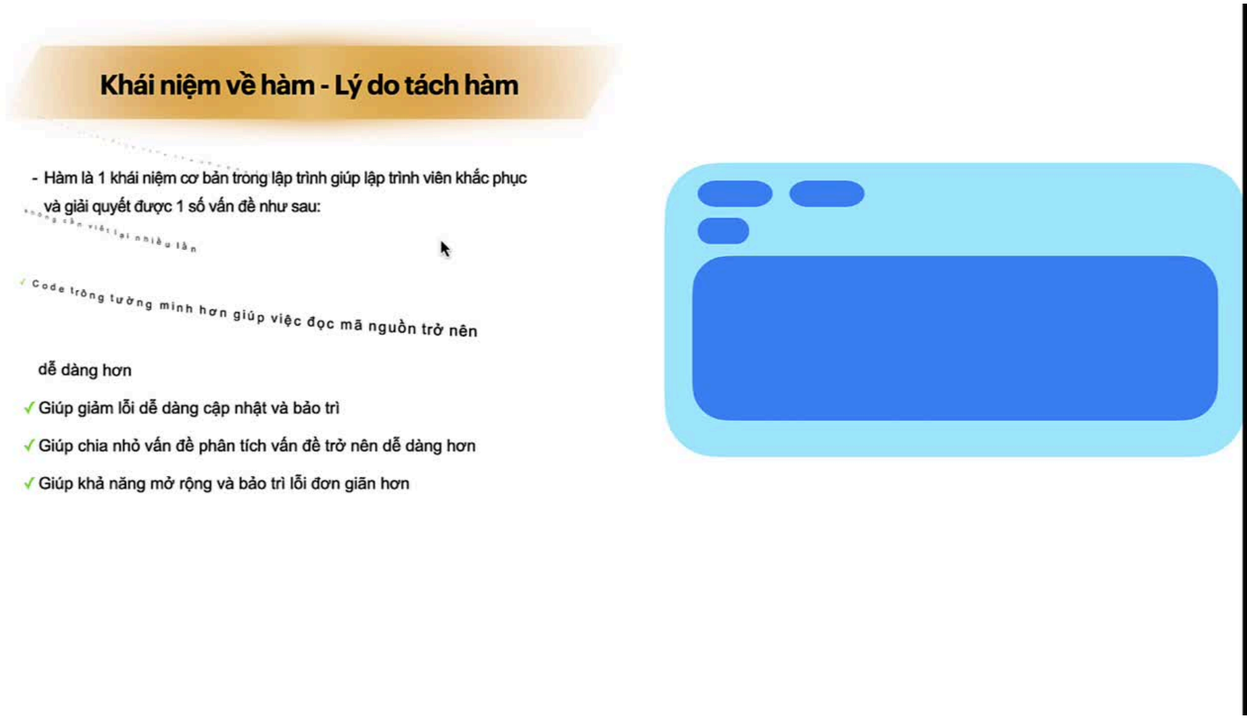


KHÁI NIỆM VỀ HÀM

HÀM (FUNCTION)

Hàm là 1 khái niệm cơ bản trong lập trình giúp lập trình viên khắc phục và giải quyết được 1 số vấn đề như sau:

- Giúp tái sử dụng lại các đoạn code của chương trình mà không cần viết lại nhiều lần
- Code trông tường minh hơn giúp việc đọc mã nguồn trở nên dễ dàng hơn
- Giúp giảm lỗi dễ dàng cập nhật và bảo trì
- Giúp chia nhỏ vấn đề phân tích vấn đề trở nên dễ dàng hơn
- Giúp khả năng mở rộng và bảo trì lỗi đơn giãn hơn



CÚ PHÁP KHAI BÁO HÀM - GỌI HÀM

KHAI BÁO HÀM

Trong C#, hàm (hay còn gọi là phương thức - method) có thể được phân loại dựa trên nhiều tiêu chí khác nhau.
Dưới đây là cú pháp để khai báo 1 hàm bình thường (hàm có giá trị trả về, ngoài ra còn các khái niệm như hàm tĩnh hay hàm không có giá trị trả về, ...)

KHAI BÁO HÀM

```
<ouput_type> function_name(input) {
[redacted]
[redacted]
[redacted]
 return output;
}
```

GỌI HÀM

```
function_name();
```

CÚ PHÁP KHAI BÁO HÀM - GỌI HÀM

CÁC LOẠI HÀM

HÀM CÓ GIÁ TRỊ TRẢ VỀ

- **Một số nguyên tắc khai báo hàm**
  - **Tên Hàm** : Gợi nhớ, Động từ
  - **Tham số truyền vào** : Không có, có 1 hoặc nhiều, TÊN THAM SỐ ĐẶT TÊN GÌ CŨNG ĐƯỢC
  - **Giá trị trả về** : Có thể có hoặc không, nếu không có thì hàm sẽ trả về **void**

kiểu dữ liệu của giá trị return (output)

tên hàm

input(parameter là tham số truyền vào hàm ứng với giá trị khi gọi hàm)

C# Main.cs

app > C# Main.cs

```
1   //Khai báo hàm
2   int tinhTong (int a, int b){
3       int tong = a + b;
4       return tong;
5   }
```

Kết quả được trả ra trong lệnh return trong hàm

```
//Gọi hàm
int ketQua = tinhTong(1,2);
Console.WriteLine($"Tổng = {ketQua}");
```

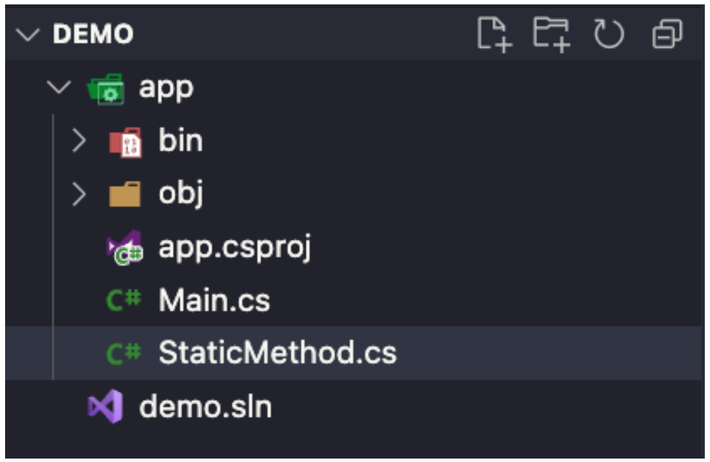
gọi hàm (giá trị truyền vào param)

CÚ PHÁP KHAI BÁO HÀM - GỌI HÀM

CÁC LOẠI HÀM

HÀM TĨNH :

Là hàm có thể gọi từ class này sang class khác, khi khác file (có thể hiểu đơn giãn ở thời điểm hiện tại như vậy). Sau này ta sẽ tìm hiểu chi tiết hơn.



class (Lớp đối tượng)

modifier (phạm vi truy xuất)

static (tĩnh) Gọi xuyên class

```cs
// StaticMethod.cs
1 reference
public class StaticMethod
{
    1 reference
    public static int tinhHieu(int a, int b)
    {
        int hieu =  a - b;
        return hieu;
    }
}
```

StaticMethod.cs

```cs
// Main.cs
int hieu = StaticMethod.tinhHieu(10,5);
Console.WriteLine($"Hiệu = {hieu}");
```

Sử dụng class.[ten_ham] để gọi hàm từ class khác

Main.cs

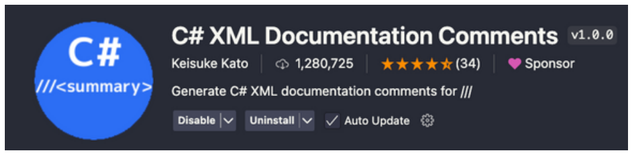
CÚ PHÁP KHAI BÁO HÀM - GỌI HÀM

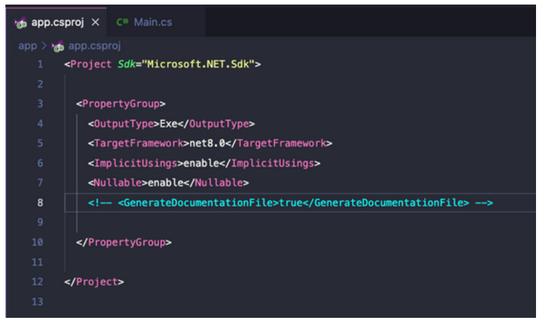
CÁC LOẠI HÀM

HÀM VOID (HÀM KHÔNG CÓ GIÁ TRỊ TRẢ VỀ)

Hàm không có giá trị trả về là hàm không có output mà chỉ để xử lý 1 số công việc cần chia nhỏ

Docstring: mô tả chức năng và các tham số của hàm để dev khác đọc





có thể thêm config này vào nếu không thấy

Main.cs

app > Main.cs

```
1   /// <summary>
2   /// Đây là hàm hiển thị thông tin bao gồm họ tên, email số, điện thoại
3   /// </summary>
4   /// <param name="hoTen">tham số thứ 1: Truyền vào họ tên kiểu string</param>
5   /// <param name="email">tham số thứ 2: Truyền vào email kiểu string</param>
6   /// <param name="sdt">tham số thứ 3: Truyền vào sdt kiểu string</param>
7   void hienThiThongTin(string hoTen, string email, string sdt)
8   {
9       Console.WriteLine($"Họ tên: {hoTen}");
10      Console.WriteLine($"Email: {email}");
11      Console.WriteLine($"Số điện thoại: {sdt}");
12  }
13
14  //Gọi  hàm
15  hienThiThongTin("Khải","khaitruong2112@gmail.com","0909999999");
```

Sử dụng từ khoá voi thay cho output_type

CÁCH XÂY DỰNG HÀM VÀ TÁCH HÀM

XÂY DỰNG HÀM VÀ CÁC VÍ DỤ

Mục đích của việc xây dựng hàm là để tái sử dụng, đóng gói 1 tác vụ (công việc nào đó) có khả năng sử dụng lại trong tương lai. Ví dụ như nhập liệu, tính toán, xử lý các vấn đề logic từ đơn giãn đến phức tạp.

Ví dụ : Viết hàm in ra ngày tháng năm

```
Console.WriteLine("Day: " + day);
    Console.WriteLine("Month: " + month);
    Console.WriteLine("Year: " + year);
}
```

Output

CÁCH XÂY DỰNG HÀM VÀ TÁCH HÀM

XÂY DỰNG HÀM VÀ CÁC VÍ DỤ

Mục đích của việc xây dựng hàm là để tái sử dụng, đóng gói 1 tác vụ (công việc nào đó) có khả năng sử dụng lại trong tương lai. Ví dụ như nhập liệu, tính toán, xử lý các vấn đề logic từ đơn giãn đến phức tạp.

Ví dụ : Viết hàm in ra ma trận sao dòng cột

```
static string InSao(int soSao) {
    string ketQua = "";
    int i = 1;
    while (i <= soSao) {
        ketQua += "*";
        i++;
    }
    return ketQua;
}
```

Output

Bài tập 1: Tính điểm trung bình và xếp loại

Đề bài: Viết chương trình nhập vào điểm toán lý hoá. Yêu cầu tính và in ra điểm trung bình và xếp loại

DTB = (toán + lý + hoá)/3

Loại yếu: DTB < 5

Loại TB: 5 <= DTB < 6.5

Loại Khá: 6.5 <= DTB < 8

Loại Giỏi: 8<= DTB <= 10

Không xếp loại: trường hợp còn lại

Ví dụ:

Input: toan = 10, ly = 8, hoa = 9

Output: 9.0 điểm - Xếp loại: Giỏi

Bài tập 2: In ra các số nguyên tố từ 2 đến <= n

Đề bài: Cho phép người dùng nhập vào 1 số n bất kỳ. In ra các số nguyên tố từ 2 đến nhỏ hơn bằng n (nếu n là số nguyên tố)

Ví dụ:

Input: 10

Output: Các số nguyên tố từ 2 đến 10 bao gồm: 2 3 5 7

Bài tập 3: In ra các số nguyên tố từ chuỗi số người dùng nhập

Đề bài: Viết chương trình cho phép người dùng nhập vào chuỗi số yêu cầu in ra các chữ số trong chuỗi là số nguyên tố, nếu không có số nguyên tố nào trong chuỗi thì in ra kết quả là không có số nguyên tố nào cả !

Ví dụ:

Input: "97482478273"

Output: Các số nguyên tố trong chuỗi gồm: 7 3

Bài tập 4: Length of Last Word (Easy)

Đề bài: Cho một chuỗi s chứa các từ cách nhau bằng một hoặc nhiều khoảng trắng, trả về độ dài của từ cuối cùng trong chuỗi. Nếu từ cuối cùng không tồn tại, trả về 0. Một từ được định nghĩa là một chuỗi ký tự liên tiếp không chứa khoảng trắng.

Ví dụ:

Input: s = "Hello World"

Output: 5

Bạn có thể tham khảo các bài toán này trên LeetCode để xem chi tiết và bắt đầu thực hiện giải quyết chúng.

Bài tập 5: Reverse Words in a String III (Easy)

Đề bài: Đảo ngược tất cả các từ trong một chuỗi.

Ví dụ:

Input: "Let's take LeetCode contest"

Output: "s'teL ekat edoCteeL tsetnoc"

CÁCH XÂY DỰNG HÀM VÀ TÁCH HÀM

HÀM MỞ RỘNG LAMDA EXPRESSION

Trong C#, **Func<T1, T2, TResult>** giống như một kiểu dữ liệu dùng để lưu trữ công thức hay hàm mà bạn có thể truyền vào để thực hiện một phép tính hoặc hành động. Trong đó Func là 1 **delegete** (một kiểu dữ liệu đại diện cho phương thức)

- **T1, T2**: Là kiểu dữ liệu lần lượt của tham số đầu vào.
- **TResult**: Là kiểu dữ liệu kết quả trả về của hàm

C# Main.cs

app > C# Main.cs

```
1   //Lambda expression
2   Func<int, int, int> add = (x, y) => x + y;
3   int result = add(3, 4);  // Kết quả là 7
4   Console.WriteLine(result);
5
```

Ngoài Func còn có Action: Action dành cho các hàm không cần output trả về (ứng với void với hàm thường)

```
7   Action<string> greet = name =>
8   {
9       Console.WriteLine("Chuẩn bị chào hỏi...");
10      string message = $"Xin chào, {name}!";
11      Console.WriteLine(message);
12  };
```

CÁCH XÂY DỰNG HÀM VÀ TÁCH HÀM

NÂNG CAO: HÀM CALLBACK FUNCTION

11
12
13
14
15
16
17
18
19
20

Callback function là một khái niệm trong lập trình dùng để chỉ một hàm được truyền như một tham số vào một hàm khác, sau đó sẽ được gọi lại (callback) và thực thi khi hàm kia hoàn thành nhiệm vụ của mình.
Một callback là một hàm (function) được truyền như một đối số cho một hàm khác và sẽ được gọi tại một thời điểm cụ thể, thường là khi một nhiệm vụ hoàn thành hoặc một sự kiện xảy ra.
Callback rất phổ biến trong lập trình bất đồng bộ (asynchronous programming), nơi các hàm cần xử lý công việc sau khi các tác vụ khác hoàn tất mà không chặn dòng chương trình.

Main.cs

app > C# Main.cs > ...

```
1   // Định nghĩa callback là một delegate
    1 reference
2   delegate void NotifyCallback(string message);
3
4   // Hàm nhận callback như một tham số
5   void ProcessData(NotifyCallback callback)
6   {
7       Console.WriteLine("Đang xử lý dữ liệu...");
8       // Sau khi xử lý xong, gọi lại callback để thông báo
9       callback("Dữ liệu đã xử lý xong!");
10  }
11
12  // Sử dụng callback
13  void StartProcess()
14  {
15      //Nơi gọi hàm và truyền callback function
16      ProcessData((message) => Console.WriteLine(message));
17  }
18
19
20
21
```

CALLBACK FUNCTION

NƠI CALLBACK BẮT ĐẦU THỰC THI

NƠI TRUYỀN HÀM XỬ LÝ THỰC THI

Main.cs

app > C# Main.cs

```
1   // Định nghĩa hàm ProcessData nhận một callback
2   void ProcessData(Func<int, int, int> callback)
3   {
4       Console.WriteLine("Đang xử lý dữ liệu...");
5       int result = callback(3, 5);  // Truyền hai số 3 và 5 vào hàm callback
6       Console.WriteLine($"Kết quả của callback: {result}");
7   }
8
9   // Sử dụng lambda expression để tạo ra một hàm callback
10  void StartProcess()
11  {
12      // Tạo một lambda expression nhận hai tham số và trả về tổng
13      Func<int, int, int> myCallback = (x, y) => x + y;
14
15      // Truyền lambda expression này vào ProcessData
16      ProcessData(myCallback);
17  }
```

CÁCH XÂY DỰNG HÀM VÀ TÁCH HÀM

BÀI TẬP LUYỆN

- **6. Tìm từ dài nhất**
  - Đề bài: Viết một hàm nhận vào một chuỗi s, trả về từ dài nhất trong chuỗi đó. Nếu có nhiều từ có độ dài bằng nhau, trả về từ đầu tiên tìm thấy.
  - Ví dụ:
  - Input: "I love programming"
  - Output: "programming"
- **7. Loại bỏ ký tự đặc biệt**
  - Đề bài: Cho một chuỗi s chứa các từ và ký tự đặc biệt, hãy loại bỏ tất cả các ký tự đặc biệt và trả về chuỗi chỉ chứa các từ và khoảng trắng.
  - Ví dụ:
  - Input: "he@llo! worl#d"
  - Output: "hello world"
- **8. Tách từ và trả về từ dài nhất có chứa số**
  - Đề bài: Cho một chuỗi s chứa các từ cách nhau bởi khoảng trắng, trong đó có các từ chứa cả chữ cái và chữ số. Viết hàm trả về từ dài nhất có chứa ít nhất một số. Nếu không có từ nào chứa số, trả về chuỗi rỗng.
  - Ví dụ:
  - Input: "abc123 def45 ghi6789"
  - Output: "ghi6789"

TO BE CONTINUED