2. Chương trình CSharp đầu tiên với cấu trúc Tuần tự

GitHub: k1enn Codeforces: dinhtrungkien LinkedIn: Trung Kiên

Note: Các kiến thức cơ bản về C# các bạn có thể coi slide cho chính xác.

IDE là chương trình để viết chương trình máy tính. Bao gồm:

- Code editor: hỗ trợ viết code
- Compiler: biên dịch
- Debugger: kiểm tra và báo lỗi

Cấu trúc chương trình C#

`using Directives` : Khai báo cho compiler biết chương trình sử dụng các câu lệnh thuộc thư viện nào trong BCL

`namespace` , `class` , `method` : Các cấu trúc tổ chức code khác nhau của một chương trình, lớn nhất là namespace, nhỏ nhất là method.

`{…}` : Các cặp dấu `{…}` dùng để xác định điểm bắt đầu và điểm kết thúc của một cấu trúc (namespace/class/method) hoặc một khối lệnh (code block)

`Main()` : Đây là một Hàm đặc biệt, xác định điểm bắt đầu khi thực thi một chương trình -> Mỗi chương trình chỉ có 1 hàm `Main()` duy nhất.

Biến trong C#

Biến (variable) là tên gọi chỉ nơi trong bộ nhớ để lưu trữ dữ liệu của chương trình.

Khai báo biến (variable declaration) để sử dụng. Như Python hoặc Javascript sẽ không cần khai báo kiểu cụ thể nhưng C# có các kiểu cơ bản sau:

- `string` : Chuỗi, đây là biến không thể thay đổi (immutable)
- `int` : Số nguyễn
- `float` / `double` : Số thực, khác biệt là float có 32-bit, còn double 64-bit.
- `char` : Kí tự
- `bool` : True/False Nhiều kiểu biến khác nhau nữa tùy theo yêu cầu của bộ nhớ, nhưng chủ yếu là các loại này

Phép gán =

Phép gán = (Assignment) có thể hiểu là đưa giá trị cụ thể của biến. Hay lí thuyết hơn là ""

Khi thực thi chương trình, tại từng thời điểm, biến có một giá trị cụ thể.

Ví dụ

```
int a = 5; // Lúc này a = 5
int b = 2; // Lúc này b = 5
```

Vậy dòng trên sẽ là a = 5 và b = 2 nhưng

```
b = a // Lúc này b = 5
a = -3 // Lúc này a = -3
```

Vậy sau khi đi qua 4 dòng trên theo cấu trúc Tuần tự, kết quả a = -3 và b = 5

Các phép toán số học

Toán tử một ngôi (unary operators):

- Tăng một đơn vị: ++
- Giảm một đơn vị: -
- Hỗ trợ 2 cách viết: `--i` hoặc `i--` và `++i` hoặc `i++`
- Đổi dấu: `-i` Nếu i là số âm sẽ thành dương và ngược lại.

Toán tử hai ngôi (binary operators):

- Cộng: +
- Trừ: -
- Nhân: *
- Chia: /
- Chia lấy dư (Số dư): %

Phép toán chuyển kiểu:

- **Implicit Casting** (automatically) - chuyển từ kiểu có kích thước nhỏ hơn sang lớn hơn, không mất mát dữ liệu `char -> int -> long -> float -> double`
- **Explicit Casting** (manually) - chuyển từ kiểu có kích thước lớn hơn sang nhỏ hơn, gây mất mát dữ liệu `double -> float -> long -> int -> char`

Một số hàm toán học: Lớp `Math` :

```
Math.Max(5, 10); // Lớn nhất
Math.Min(5, 10); // Nhỏ nhất
Math.Sqrt(64); // Căn bậc 2
Math.Abs(-4.7); // Giá trị tuyệt đối
Math.Round(9.99); // Làm tròn
```

Nhập dữ liệu

`Console.ReadLine()` – đọc toàn bộ một dòng dữ liệu do người dùng nhập trong cửa sổ cho đến khi nhấn Enter

Dữ liệu đọc bằng `Console.ReadLine()` có kiểu String

Phép chuyển kiểu:

1. `Covert.To…()`
2. `type.Parse()`

Xuất dữ liệu

- `Console.WriteLine(…)` – xuất một dòng dữ liệu ra cửa sổ Console và kết thúc bằng ký tự xuống dòng \n
- `Console.Write()` - Không xuống dòng

Cú pháp định dạng cho indexed placeholder: `{index[,alignment][:formatString]}`

- `index` : bắt buộc, xác định vị trí item được in ra trong dãy `Object[]` (đánh số bắt đầu từ 0)
- `[,alignment]` : không bắt buộc, một số nguyên xác định độ rộng (số ký tự) và cách canh lề khi in:
  Nếu alignment < chiều dài cần để in item -> tham số này sẽ được bỏ qua.
  Ngược lại, alignment là số dương -> canh lề phải , alignment là số âm -> canh lề trái
- `[:formatString]` : không bắt buộc, chuỗi định dạng phù hợp với kiểu dữ liệu của item

Kiểu này cũ và không còn được sử dụng nhiều hiện nay.

Escape Characters: sử dụng dấu backslash `\` . Nếu ký tự theo sau `\` là:
• Ký tự đặc biệt -> dịch thành ký tự đặc biệt tương ứng. Ví dụ: `\t` (ký tự tab), \n (ký tự xuống dòng),…
• Các ký tự khác, in ra như một ký tự bình thường.
Ví dụ: `\" (in ký tự ")` , `\\ (in ký tự \)` , `\{ (in ký tự {)` ,…

Verbatim String Literal: đặt ký tự '@' trước chuỗi à các ký tự đặc biệt trong chuỗi được xem như là một ký tự bình thường. Ngoại trừ các trường hợp đặc biệt sau:
• Dùng `""` -> `"`
• Dùng `{{` -> `{`
• Dùng `}}` -> `}`

Ghi chú

Ghi chú một dòng `// ...` Ghi chú nhiều dòng `/* ... */`

Tạo ghi chú hiệu quả:

1. Viết ghi chú cho đoạn mã trước hoặc ngay khi hoàn thành xong đoạn mã
2. Ghi chú phải cung cấp thêm thông tin cho đoạn mã

- Đặt mình vào hoàn cảnh đây là đoạn mã do người khác viết và giờ mình cần phải đọc hiểu đoạn mã này thì cần ghi chú những gì
- Mô tả thuật toán của đoạn mã
- Những kỹ thuật đặc biệt đã áp dụng

3. Đừng ghi chú mọi dòng code hay không ghi chú gì cả!