C# Basic.

A beginner guide for C#

Mine Archive

Outline

C 言語と C# の違い

型システムの違い

C の型	C# の型	値の制限
int	int	−2147483648 〜 2147483647
float	float	±1.5e-45 〜 ±3.4e+38
double	double	±5.0e-324 〜 ±1.7e+308
long	long	−9,223,372,036,854,775,808 〜 9,223,372,036,854,775,807
char	char	0 〜 65,535
該当なし	bool	true or false
char*	string	文字列

加えて、unsigned 型も存在します。

メモリ管理

- C 言語では malloc, realloc を使ってメモリを手動で確保していた。
  - メモリ管理の責任がプログラマにある。
  - メモリリークが発生する可能性がある。
  - 不適切な free()呼び出しでダブルフリーエラーが発生することもある。
- C# ではガベージコレクション(GC)によって自動管理される。
  - 今後使われないと考えられるメモリは自動的に free()される感じ。
  - プログラマはメモリ管理の細かい部分を気にする必要がない。
  - メモリリークのリスクが大幅に低減される。
- 新しく配列を取るときにいちいち長さとか考える必要がない。
  - 配列のサイズを動的に確保できる。
  - List などのコレクション型を使えば、サイズ変更も容易。

名前空間と using ディレクティブ

名前空間(Namespace)は、コードを論理的に整理・管理するための仕組みです。 同じ名前の型が複数存在する場合の名前衝突を避けることができます。

using ディレクティブを使うことで、名前空間内の型を直接参照できます。

C 言語	C#
#include <stdio.h>	using System;
ヘッダファイルをインクルード	名前空間をインポート
標準ライブラリの宣言を取り込む	名前空間内の型を直接参照可能にする

名前空間と using ディレクティブ (ii)

例:

```
using System;

namespace MyApplication
{
    public class Program
    {
        public void Print()
        {
            //System.Console.WriteLine("Hello!");
            Console.WriteLine("Hello!");
        }
    }
}
```

名前空間と using ディレクティブ (iii)

```
public class Main
{
    public static void Main()
    {
      // 上でMyApplicationの中にProgramを指定しているから
      // MyApplication.Programとしていける
      var program = new MyApplication.Program();
      program.Print();
    }
}
```

- `using System;` により、System 名前空間の型(Console など)を直接使用可能
- `namespace MyApplication { }` で、自分のコードを名前空間内に配置

オブジェクト指向プログラミングとは

クラス

- C 言語の構造体が関数を持つようになった感じ
- クラスはデータ(フィールド)とそれを操作する処理(メソッド)を 1 つの単位にまとめたもの
- オブジェクト指向プログラミングの中心的な概念
- アクセス修飾子(public, private など)でカプセル化が可能
  - public: 外部からアクセス可能
  - private: クラス内からのみアクセス可能
- コンストラクタによってオブジェクトの初期化が可能
- インスタンスを生成することで、クラスの具体的な実体が作られる

継承

文字列とコレクションとジェネリック型

コード文法

継承 (iv)

```c
int main() {
    printf("Hello, World!\n");

    return 0;
}
```

```c#
using System;

public class Hello
{
    public static void Main()
    {
        Console.WriteLine("Hello World!");
    }
}
```