Ｃ＃の基礎について③

ラムダ式

発表の目的

学ばせてもらったことをどれぐらい理解できているか。
理解した内容はあっているか。
確認と理解を深めることを目的としています。

今日はラムダ式についてを発表します。

ラムダ式の概要

- ラムダ式とは

関数内で定義された即席の関数のことを言います。

- 何をもってラムダ式と呼ぶのか？

ラムダ式はその名の通り、式の表記になります。
ラムダ式自体を調べると、無名関数や匿名関数などの名前を目にしますが、
ラムダ式はラムダ演算子を使用した式をさします。
ラムダ演算子は「**=>**」この記載です。
ラムダ演算子を使用して処理を記載することにより、無名関数や匿名関数と呼ばれるものになります。

ラムダ式の概要

- ラムダ式の導入目的

アルゴリズムと高階関数をより簡単に扱うことを
主目的として導入されたもの。

「ラムダ式によるデザインパターン in C#」より一部抜粋。

- ラムダ式のメリット

・ラムダ式を渡すことによって
　外部から処理をカスタマイズできること
・簡単な処理なら関数定義せずにその場で無名関数を作成

ラムダ式の書き方

```
() => Console.WriteLine("Hello");
```

パラメータなしの関数を定義するためのラムダ式

```
x => x * x;
```

パラメータ１つの関数を定義するためのラムダ式

```
(x,y) => x * y;
```

複数パラメータの関数を定義するためのラムダ式

```
(x,y) => { int c = x * y;
           return c; }
```

複数行の関数を定義するためのラムダ式

ラムダ式を変数にもつ

```
Func<int,int,int>  f  =  (x,y)  =>  x  *  y;
```

- int型のパラメータを２つ
  受取り、int型を返す
  ラムダ式をもつ変数 f
- ラムダ式
  第一引数と
  第二引数の積

Funcの最後のパラメータが**戻り値の型**、
それ以外が**受け取る変数の型**となりま
す。

ラムダ式を引数で受け渡す

```
Foo((x,y) => x * y);
```
ラムダ式を渡す

```
Foo(Func<int,int,int> f)
{
  Console.WriteLine(f(5,5));
}
```
ラムダ式を受け取る

関数に**ラムダ式を引数として渡す**場合、
受け取るラムダ式の形(引数、戻り値の型)を指定する。

```
// 実行結果
25
```

ラムダ式

```
Func<parameter１,parameter２,…result>
```

```
Func<int,int,string>
```

int型のパラメータを２つ受取り、string型を返す

```
Action<parameter１,parameter２,…parameterN>
```

```
Action<int,int,string>
```

int型のパラメータを2つ、string型のパラメータを１つ受取る

Ｃ＃の基礎について③

本日の発表は以上です。