» HÀM

Name: Nguyễn Chí Hiếu
Date: 2020

» NỘI DUNG

» CÁC KHÁI NIỆM CƠ BẢN

Hàm *(function)*

* Là một đoạn chương trình con độc lập. Hàm sẽ được thực hiện khi có lời gọi hàm.
* Mỗi hàm có tên hàm, các tham số hình thức *(nếu có)* và giá trị trả về.
* Do ngôn ngữ lập trình cung cấp sẵn hay do người dùng tự định nghĩa.
* Hàm được sử dụng nhằm mục đích để chương trình được gọn hơn và tái sử dụng.

» CÁC KHÁI NIỆM CƠ BẢN

Khai báo và định nghĩa hàm

Cú pháp:

```
1 [Kieu_du_lieu] Ten_ham([Danh_sach_cac_tham_so_hinh_thuc
     ])
2 {
3       Lenh;
4       [return [bieu_thuc];]
5 }
```

Ví dụ

Khai báo và định nghĩa hàm in dòng chữ xin chào với tham số là tên người.

```
1 static void XinChao(string name)
2 {
3       Console.WriteLine("Xin chao {0}", name);
4 }
```

Giải thích

* Kiểu dữ liệu: phải khai báo kiểu dữ liệu trả về của hàm.
* Tên hàm: đặt tên theo quy định và phải mô tả được ý nghĩa của hàm.
* Danh sách các tham số hình thức: nếu hàm có nhận giá trị đầu vào để tính toán thì phải khai báo các tham số cho hàm. Mỗi tham số cách nhau bởi dấu phẩy.
* Thân hàm đặt trong cặp dấu ngoặc nhọn { ... }
* Nếu hàm có trả về giá trị cần thêm lệnh return vào cuối hàm.

» CÁC KHÁI NIỆM CƠ BẢN

Gọi hàm

Cú pháp:

```
Ten_ham([Danh_sach_cac_tham_so_thuc])
```

Ví dụ

Khai báo và định nghĩa hàm in dòng chữ xin chào với tham số là tên người.

```
static void XinChao(string name)
{
    Console.WriteLine("Xin chao {0}", name);
}

static void Main(string[] args)
{
    XinChao("CNTT K20");
}
```

» CÁC KHÁI NIỆM CƠ BẢN

Các bước thực hiện

1. Khai báo *(declare)* và định nghĩa *(define)* thân hàm.

```
1 // 1
2 static void XinChao(string name)
3 {
4     Console.WriteLine("Xin chao {0}", name);
5 }
```

2. Gọi hàm.

```
1 static void Main(string[] args)
2 {
3     // 2
4     XinChao("CNTT K20");
5 }
```

Giá trị trả về của hàm

* Kiểu void
* Kiểu bool, int, float, ...
* Không khai báo kiểu trả về (sẽ được học trong lập trình hướng đối tượng).

» HÀM

Giá trị trả về của hàm

* Trả về một giá trị

```
1  static double Cong(double a, double b)
2  {
3      return a + b;
4  }
5
6  static double Tru(double a, double b)
7  {
8      return a - b;
9  }
```

Giá trị trả về của hàm

* Trả về nhiều giá trị cùng kiểu dữ liệu: dùng kiểu dữ liệu mảng

```
1  static double[] CongTruNhanChia(int a, int b)
2  {
3      double[] kq = new double[4];
4
5      kq[0] = a + b;
6      kq[1] = a - b;
7      kq[2] = a * b;
8      kq[3] = a / b;
9
10     return kq;
11 }
```

Giá trị trả về của hàm

* Trả về nhiều giá trị cùng kiểu dữ liệu: dùng kiểu dữ liệu mảng

```
1  static void Main(string[] args)
2  {
3      int a = 10;
4      int b = 5;
5      double[] kq = new double[4];
6      kq = CongTruNhanChia(a, b);
7      for (int i = 0; i < 4; i++)
8      {
9          Console.WriteLine(kq[i]);
10     }
11 }
```

» TRUYỀN THAM SỐ CHO HÀM

Tham số của hàm

* Tham số hình thức *(Formal Parameter)* là biến được liệt kê trong danh sách các tham số thường đặt ở phần đầu của hàm.
* Tham số thực *(Actual Parameter)* là biến hay biểu thức được truyền cho hàm khi gọi hàm.

» TRUYỀN THAM SỐ CHO HÀM

Tham số của hàm

```
1  int min(int x, int y)
2  {
3      if (x < y)
4          return x;
5      return y;
6  }
```

Tham số của hàm

```
1  static void Main(string[] args)
2  {
3      int x, y;
4      x = 5;
5      y = 10;
6
7      Console.WriteLine("min = {0}", min(2, 7));
8      Console.WriteLine("min = {0}", min(x, y));
9      Console.WriteLine("min = {0}", min(x, y + 20));
10 }
```

* Dòng 7: gọi hàm min(2, 7) với 2 và 7 là hai tham số thực.
* Dòng 8: gọi hàm min(x, y) với x và y là hai tham số thực.
* Dòng 9: gọi hàm min(x, y + 20) với x và y + 20 là hai tham số thực.

» TRUYỀN THAM SỐ CHO HÀM

Truyền tham trị: giá trị các tham số không đổi sau khi hàm thực hiện.

```
1  static void HoanVi1(int a, int b)
2  {
3      int c = a;
4      a = b;
5      b = c;
6  }
7  static void Main(string[] args)
8  {
9      int a = 10;
10     int b = 5;
11     HoanVi1(a, b);
12     Console.WriteLine("a = {0}, b = {1}", a, b);
13 }
```

» TRUYỀN THAM SỐ CHO HÀM

Truyền tham biến/tham chiếu

* Giá trị các tham số thay đổi sau khi hàm thực hiện.
* Phương pháp này có thể được dùng để trả về nhiều giá trị cho hàm.

```
1  static void HoanVi2(ref int a, ref int b)
2  {
3      int c = a;
4      a = b;
5      b = c;
6  }
7  static void Main(string[] args)
8  {
9      //
10     HoanVi2(ref a, ref b); // ref
11 }
```

» TRUYỀN THAM SỐ CHO HÀM

Tham số mặc định

* Trong một số trường hợp đặc biệt, tham số của hàm được gán một giá trị mặc định. Khi hàm được gọi, nếu tham số không được truyền thì tham số sẽ nhận giá trị mặc định.

```
1  public static void Main()
2  {
3      Console.WriteLine(LuyThua(2)); // 4
4  }
5  static int LuyThua(int x, int y = 2)
6  {
7      int kq = x;
8      for (int i = 2; i <= y; i++)
9          kq *= x;
10     return kq;
11 }
```

» NẠP CHỒNG HÀM (OVERLOAD)

Hàm được khai báo *trùng tên* với nhau nhưng thỏa một trong hai điều kiện:

* Kiểu dữ liệu của các tham số *khác nhau*
* Thứ tự của các tham số *khác nhau*

```
1  static int Cong(int a, int b)
2  {
3      return a + b;
4  }
5  static int Cong(int n)
6  {
7      int kq = 0;
8      for (int i = 1; i <= n; i++)
9          kq += i;
10     return kq;
11 }
```

» HÀM ĐỆ QUY

Khái niệm

Một hàm được gọi là đệ quy *(recursion)* nếu bên trong thân hàm đó có gọi lại chính nó một cách trực tiếp hay gián tiếp.

Hàm đệ quy gồm hai thành phần:

* Phần cơ sở: điều kiện dừng của quá trình gọi đệ quy
* Phần quy nạp: thân hàm chứa lời gọi đệ quy

» HÀM ĐỆ QUY

Phân loại đệ quy

* Theo số lần gọi hàm
  * Đệ quy tuyến tính
  * Đệ quy nhị phân
  * Đệ quy phi tuyến
  * Đệ quy tương hỗ *(gián tiếp)*
* Theo thứ tự gọi hàm
  * Đệ quy đầu: hàm được gọi đệ quy trước khi thực hiện các xử lý
  * Đệ quy đuôi: hàm được gọi đệ quy sau khi thực hiện các xử lý

» Phân loại đệ quy

ĐỆ QUY TUYẾN TÍNH

* Thân hàm có duy nhất một lời gọi hàm gọi lại chính nó một cách tường minh.

```
1  Kieu_du_lieu Ten_ham(Tham_so)
2  {
3      if (Dieu_kien_dung)
4      {
5          khoi_lenh;
6          return Gia_tri;
7      }
8      ...;
9      Ten_ham(Tham_so);
10 }
```

» ĐỆ QUY TUYẾN TÍNH

Ví dụ

Viết hàm đệ quy tính n giai thừa.

```
1 static int GiaiThua(int n)
2 {
3     if (n == 0)
4         return 1;
5     return n * GiaiThua(n - 1);
6 }
```

» Phân loại đệ quy

ĐỆ QUY NHỊ PHÂN

* Thân hàm có hai lời gọi hàm gọi lại chính nó một cách tường minh.

```
1  Kieu_du_lieu Ten_ham(Tham_so)
2  {
3      if (Dieu_kien_dung)
4      {
5          ...;
6          return Gia_tri;
7      }
8      ...;
9      Ten_ham(Tham_so);
10     ...;
11     Ten_ham(Tham_so);
12 }
```

» ĐỆ QUY NHỊ PHÂN

Ví dụ

Dãy Fibonacci được định nghĩa như sau

$$f(n) = \begin{cases} 1 & , n = 0, 1 \\ f(n-1) + f(n-2) & , n > 1. \end{cases}$$

Viết hàm đệ quy tính dãy Finonacci.

```
1  static int Fibonaci(int n)
2  {
3      if (n <= 1)
4          return 1;
5      return Fibonaci(n - 1) + Fibonaci(n - 2);
6  }
```

» Phân loại đệ quy

ĐỆ QUY PHI TUYẾN

* Thân hàm có lời gọi hàm lại chính nó trong vòng lặp.

```
1  Kieu_du_lieu Ten_ham(Tham_so)
2  {
3      if (Dieu_kien_dung)
4      {
5          ...;
6          return Gia_tri;
7      }
8      loop (Dieu_kieu_lap)
9      {
10         ....;
11         Ten_ham(Tham_so);
12     }
13 }
```

» ĐỆ QUY PHI TUYẾN

Ví dụ

Viết hàm đệ quy tính giá trị của $f(n)$ được định nghĩa như sau:

$$f(n) = \begin{cases} n & ,n \leq 4 \\ f(n-1) + f(n-2) + f(n-3) + f(n-4) & ,n > 4 \end{cases}$$

```
1  static int F(int n)
2  {
3      int i, result = 0;
4      if (n <= 4)
5          result n;
6      for (i = 1; i <= 4; i ++ )
7          result +=  F(n - i);
8      return result;
9  }
```

» Phân loại đệ quy

ĐỆ QUY TƯƠNG HỖ

* Thân hàm 1 có lời gọi hàm tới hàm 2 và bên trong thân hàm 2 có lời gọi hàm đến hàm 1.

```
1  Kieu_du_lieu Ten_ham_1(Tham_so)
2  {
3      if (Dieu_kien_dung)
4          return Gia_tri;
5      Ten_ham_2(Tham_so);
6  }
7  Kieu_du_lieu Ten_ham_2(Tham_so)
8      {
9      if (Dieu_kien_dung)
10         return Gia_tri;
11     Ten_ham_1(Tham_so);
12 }
```

» ĐỆ QUY TƯƠNG HỖ

Ví dụ

Cho số nguyên không âm n. Viết hàm đệ quy xác định n là số chẵn hay lẻ.

$$\begin{cases} n & , n = 0, 1 \\ \texttt{SoLe}\,(n-1) & , n \neq 0 \\ \texttt{SoChan}\,(n-1) & , n \neq 1 \end{cases}$$

trong đó, kết quả trả về 1 là số lẻ và 0 là số chẵn.

```
1 static bool SoLe(int n)
2 {
3     if (n == 1)
4         return true;
5     return SoChan(n - 1);
6 }
```

```
1 static bool SoChan(int n)
2 {
3     if (n == 0)
4         return false;
5     return SoLe(n - 1);
6 }
```

» BÀI TẬP

1. Nhập vào một số nguyên dương s tương ứng với tổng thời gian theo giây. In ra màn hình thời gian theo hh:mm:ss.
2. Tính $S(n) = \sqrt{2 + \sqrt{2 + \sqrt{2 + \cdots \sqrt{2 + \sqrt{2}}}}}$ với n là số lần lấy căn bậc hai.
3. Tìm ước số chung lớn nhất của hai số nguyên dương a và b.
4. Viết hàm đệ quy tính dãy Fibonnaci:

$$f(n) = \begin{cases} 1 & , n = 0, 1 \\ f(n-1) + f(n-2) & , n > 1 \end{cases}$$

5. Viết hàm đệ quy giải bài toán tháp Hà Nội.