Hàm &
Kỹ thuật tổ chức chương trình
Phần a: Các vấn đề mở rộng nghề nghiệp

Nhập môn lập trình

Trình bày: Nguyễn Sơn Hoàng Quốc
Email: nshquoc@fit.hcmus.edu.vn

Nội dung

- Các vấn đề tìm hiểu mở rộng kiến thức nghề nghiệp
- Thuật ngữ và bài đọc thêm tiếng Anh

CÁC VẤN ĐỀ TÌM HIỂU MỞ RỘNG KIẾN THỨC NGHỀ NGHIỆP

Hàm trùng tên

- Nhu cầu
  - Thực hiện một công việc với nhiều cách khác nhau. Nếu các hàm khác tên sẽ khó nhớ.
- Ví dụ:
  - Các hàm tính trị tuyệt đối trong C (math.h)
    - `int abs(int n);`
    - `long labs(long n);`
    - `double fabs(double n);`
  - Các hàm tính căn bậc 2: sqrt(), sqrtf()

Hàm trùng tên

- Khái niệm
  - Là các hàm cùng tên nhưng có tham số đầu vào hoặc kiểu trả về khác nhau nhằm cho phép người dùng chọn cách thuận lợi nhất để thực hiện công việc.
  - Nguyên mẫu hàm khi bỏ tên tham số phải khác nhau.
  - Việc sử dụng các hàm trùng tên được gọi là nạp chồng hay quá tải (overload) hàm.

Ví dụ hàm trùng tên

```
1. // prints integers from 1 to n
2. void PrintIntegers(int n);

3. // prints integers from x to y
4. void PrintIntegers(int x, int y);

5. // prints integers from x to y
6. // with an arithmetic progression a
7. void PrintIntegers(int x, int y, int a);
```

Chú ý về hàm trùng tên

- Các hàm sau đây là như nhau do cùng nguyên mẫu hàm: int Sum(int, int);

```
1. // calculates a + b
2. int Sum(int a, int b);

3. // calculates b + a
4. int Sum(int b, int a);

5. // calculates x + y
6. int Sum(int x, int y);
```

Sự nhập nhằng, mơ hồ

```
1. float f(float x)   { return x/2; }
2. double f(double x) { return x/2; }

3. void main() {
4.    float x = 29.12;
5.    double y = 17.06;
6.    printf("%.2f\n", f(x));          // float
7.    printf("%.2lf\n", f(y));         // double
8.    printf("%.2f\n", f(10));         // ???
9.    printf("%.2f\n", f((float)10));  // float
10.}
```

Sự nhập nhằng, mơ hồ

```
1. void f(unsigned char c) { printf("%d", c); }
2. void f(char c) { printf("%c", c); }

3. void main()
4. {
5.    f('A');                    // char
6.    f(65);                     // ???
7.    f((char)65);               // char
8.    f((unsigned char)65); // unsigned char
9. }
```

Sự nhập nhằng, mơ hồ

```
1. int f(int a, int b) { return a + b; }
2. int f(int a, int &b) { return a + b; }

3. void main()
4. {
5.     int x = 1, y = 2;
6.     printf("%d", f(x, 2));      // b = 2
7.     printf("%d", f(x, y));      // ???
8. }
```

Sự nhập nhằng, mơ hồ

```
1. int f(int a) { return a*a; }
2. int f(int a, int b = 1) { return a*b; }

3. void main()
4. {
5.    printf("%d\n", f(2912, 1706));
6.    printf("%d\n", f(2912));          // ???
7. }
```

Hàm có đối số mặc định

- Khái niệm
  - Là hàm có một hay nhiều tham số hình thức được gán sẵn giá trị mặc định. Các tham số này nhận giá trị mặc định đó nếu không có đối số tương ứng được truyền vào.
  - Các tham số mặc định phải được dồn về tận cùng bên phải.
- Ví dụ

```
void PrintFraction(int num, int denom = 1);
```

Hàm có đối số mặc định

- Lưu ý:
  - Muốn truyền đối số khác thay cho đối số mặc định, phải truyền đối số thay cho các đối số mặc định trước nó.
- Ví dụ:

```
void SolveEq2(int a, int b = 0, int c = 0);
```

- Giải phương trình: $2x^2 + 0x + 3 = 0$
  - Sai: `SolveEq2(2, 3);` `// a=2, b=3, c=0`
  - Đúng: `SolveEq2(2, 0, 3);` `// a=2, b=0, c=3`

Hàm có đối số mặc định

- Lưu ý:
  - Nếu $x = a$ thường xuyên xảy ra thì nên chuyển x thành tham số có đối số mặc định là a.

    Ví dụ, $Gender = 0$ (Nam), $Age = 18$.
  - Nếu $x = a$ và $y = b$ thường xuyên xảy ra nhưng $y = b$ thường xuyên hơn thì nên đặt tham số mặc định y sau x.

    Ví dụ, trong cùng lớp học $Age = 18$ xảy ra nhiều hơn $Gender = 0$ do đó nên đặt Age sau $Gender$.

Hàm có tham số là hàm

- Khái niệm
  - Hàm có thể truyền vào hàm khác dưới dạng đối số đầu vào.
  - Việc khai báo tham số là hàm tương tự như khai báo nguyên mẫu hàm (không cần tên các tham số hình thức)
  - Chỉ được phép truyền các hàm có nguyên mẫu hàm (sau khi bỏ đi tên các tham số hình thức) giống với nguyên mẫu hàm của tham số hình thức hàm được khai báo.

Ví dụ hàm có tham số là hàm

```
1. int FindBestNumber(int a, int b, int
   Better(int, int)) {
2.     int numBest = a;
3.     if (Better(b, a)) {
4.         numBest = b;
5.     }
6.     return numBest;
7. }
8. int MaxNumber(int x, int y) { return x > y; }
9. int MinNumber(int x, int y) { return x < y; }
```

Ví dụ hàm có tham số là hàm

```
1. int FindBestNumber(int a[], int n,
                      int Better(int, int)) {
2.     int i, idBest = 0;
3.     for (i = 1; i < n; i++) {
4.         if (Better(a[i], a[idBest]))
5.             idBest = id;
6.     }
7.     return a[idBest];
8. }
9. int MaxNumber(int x, int y) { return x > y; }
10.int MinNumber(int x, int y) { return x < y; }
```

Hàm đệ qui

- Khái niệm
  - Đệ qui chỉ một tình huống mà trong đó hàm gọi chính nó theo cách trực tiếp hay gián tiếp.
- Ví dụ
  - Tính giai thừa: $n! = n * (n-1) * \cdots * 2 * 1$
  - Do $(n-1) * \cdots * 2 * 1 = (n-1)! \Rightarrow n! = n * (n-1)$
  - Tương tự $(n-1)! = (n-)! * (n-1)$
  - Tiếp tục cho đến khi tính 1! ta có ngay kết quả là 1, thế ngược lại tính được $n!$

Ví dụ hàm đệ qui

- Khai báo hàm:

```
unsigned int factorial(unsigned int n);
```

- Định nghĩa hàm:

```
1. unsigned int factorial(unsigned int n)
2. {
3.     if (n == 1)
4.         return 1;
5.     else
6.         return n * factorial(n – 1);
7. }
```

So sánh với các NNLT khác

Tiêu chí so sánh/Ngôn ngữ	C	C++	C#	Java
Khai báo hàm độc lập với các thành phần khác	✓	✓		
Khai báo hàm (phương thức) trong lớp đối tượng (class)		✓	✓	✓
Truyền bằng giá trị (tham trị)	✓	✓	✓	✓ (kiểu cơ sở)
Truyền bằng địa chỉ	✓	✓		
Truyền bằng biến (tham biến/tham chiếu)		✓ (sử dụng &)	✓ (sử dụng từ khóa ref, out)	✓ (đối tượng và mảng)
Tham số có giá trị mặc định		✓	✓	✓
Hàm trùng tên (nạp chồng hàm)		✓	✓	✓

THUẬT NGỮ VÀ BÀI ĐỌC THÊM TIẾNG ANH

Thuật ngữ tiếng Anh

- ***function***: hàm (chương trình con)
- ***structured programming***: lập trình cấu trúc
- ***modular programming***: lập trình đơn thể
- ***parameter***: tham số
- ***argument*** : đối số
- ***formal parameter***: tham số hình thức, tương đương với ***parameter***
- ***actual parameter***: tham số thực, tương đương với ***argument***
- ***function prototype***: nguyên mẫu hàm
- ***function header***: tiêu đề hàm
- ***function declaration***: khai báo hàm
- ***function definition***: định nghĩa hàm

Thuật ngữ tiếng Anh

- ***local variable***: biến cục bộ
- ***extern varialbe***: biến ngoài
- ***global variable***: biến toàn cục, tương tự ***extern variable***
- ***call by value***: truyền đối số bằng giá trị (tham trị)
- ***call by reference***: truyền đối số bằng tham biến (tham chiếu)
- ***scope***: tầm vực, phạm vi hiệu quả
- ***recursion***: sự đệ qui
- ***overload***: nạp chồng, quá tải
- ***ambiguity***: nhập nhằng, mơ hồ

Bài đọc thêm tiếng Anh

- **Bradley L. Jones and Peter Aitken**, *Teach Yourself C in 21 days*, 6th Edition, SAMS, 2003.
  - Day 5. Packaging Code in Functions, pp. 97-122.
  - Day 12. Understanding Variable Scope, pp. 285-303.
  - Day 21. Advanced Compiler Use – Programming with Multiple Source-Code Files, pp. 593-600.
- **Bjarne Stroustrup**, *The C++ Programming Language*, 3rd Edition, AT&T, 1997.
  - Chapter 7. Functions, pp. 143-164.
  - Chapter 9. Source Files and Programs, pp. 197-220.