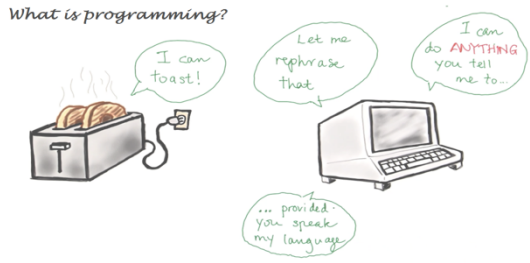


Tuần 4

Chương trình con

CT101 – Lập trình căn bản

Khoa CNTT&TT – ĐHCT

Mục đích

- Đến thời điểm này, chúng ta đã có thể tạo ra các chương trình mà trong đó chỉ có duy nhất một chương trình chính (hàm chính – main()).
- Trong phần này, chúng ta sẽ xem xét cách lập trình tạo ra chương trình con (hàm con – function) với mục tiêu là tạo ra cơ chế để chúng ta có thể viết các chương trình có tính môdun hoá cao, logic và rõ ràng.

(Slides được tham khảo từ bài giảng CS110: Introduction to Computer Programming, Chris Alvin, University of Wisconsin - Madison)

Yêu cầu

- Sau khi học xong phần này, người học:
  - Hiểu được như thế nào là hàm (function), cách thiết kế và gọi nó.
  - Thành thạo các kỹ thuật truyền tham số cho hàm.
  - Có thể tạo ra các hàm để hiện thực hoá một yêu cầu tính toán, khoa học nào đó.

Nội dung

- Định nghĩa hàm, cách thiết kế, cách gọi hàm.
- Các cơ chế truyền tham số cho hàm.

Hàm

- Một hàm là một đơn vị được đặt tên trong chương trình có nhiệm vụ thực thi một tác vụ (operation) cụ thể. Có thể chúng ta chưa để ý rằng, chúng ta đã sử dụng một hàm đặc biệt – hàm main().
- Nhìn chung, các hàm được thể hiện bởi cặp dấu ngoặc '()' đi kèm ngay sau tên hàm. Ví dụ, **erase()** là một hàm thực hiện chức năng "erase".
- Lập trình hàm là cách thức hữu hiệu để cải thiện chương trình, theo nghĩa là: chương trình vừa rõ ràng đối với người đọc và dễ cài cài đặt đối với lập trình viên.

Hình thức của hàm

- Mỗi hàm đều có hình thức như sau (với nhiều phần khác nhau được đánh số thứ tự trong hình). Các từ khoá được in đậm:

```
(1) ReturnType  (2) FunctionName( (3) arguments passed )
{
    (4) // declarations and initializations

    (5) // body

    (6) return ReturnExpression (no expression if void
return type);
}
```

Hình thức của hàm

- Trong đó:
  ① ***ReturnType***: là kiểu dữ liệu mà hàm sẽ trả về. Nếu hàm không trả về giá trị nào thì khai báo kiểu là **void**
  ② ***FunctionName***: là tên của hàm. Tên hàm phải mô tả được công việc mà hàm thực hiện và cũng nên đơn giản, dễ hiểu.
  ③ Các tham số được truyền cho hàm: Các tham số này được khai báo và gán giá trị bởi hàm khác muốn gọi hàm này. Nếu không có tham số truyền cho hàm thì bỏ trống phần này, giữ lại hai dấu ngoặc ().

Hình thức của hàm

- Trong đó:
  ④ Đoạn này dùng để khai báo và khởi tạo các biến cục bộ của hàm.
  ⑤ Phần thân của hàm: Bao gồm một dãy các mã lệnh sẽ được thực thi khi hàm được gọi.
  ⑥ Lệnh **return** được dùng để trả về kết quả mà hàm xử lý. Nội dung trả về có thể là nội dung của một biến, một hằng hoặc kết quả của một biểu thức. Kiểu trả về phải giống như ***ReturnType*** (1). Nếu hàm có kiểu trả về là **void**, chỉ gọi **return;**

Hàm chính (main)

- Xem đoạn mã lệnh sau:

```
int main()
{
    int i;

    for (i = 0; i < 5; i++)
        printf("i is %d\n", i);

    return 0;
}
```

- Đây là hàm chính (main). Hàm main là hàm luôn được gọi trước tiên khi chương trình được thực thi. Từ hàm main này những hàm khác có thể được gọi theo kiến trúc phân cấp hình cây.
- Anh chị hãy phân tích và chỉ ra các thành phần của hàm main trên.

Tự viết hàm

- Trước khi tự viết ra các hàm cho mình, lập trình viên phải trả lời các câu hỏi sau:
  - Mục tiêu của mình có đòi hỏi một hành động cụ thể nào đó mà nó lặp đi lặp lại hoặc được gọi từ chính hàm này hoặc những hàm khác hay không?
  - Có hành động nào đòi hỏi mình phải viết rất nhiều hàng lệnh mà cuối cùng tạo ra sự hỗn độn trong chương trình? Việc chia nhỏ mã lệnh trong chương trình chính có tạo ra sự ngăn nắp, rõ ràng hay không?
- Khi câu trả lời là “có” cho tất cả các câu hỏi trên thì rất có thể là lập trình viên phải tự viết hàm cho mình.

Làm sao để viết hàm

- Thiết kế hàm theo khung được giới thiệu ở trên tương ứng với với 6 điểm:
  1) Hàm có trả về giá trị nào không? Kiểu dữ liệu trả về là gì?
  2) Mục tiêu của hàm là gì? Đặt tên hàm tương ứng với mục tiêu này.
  3) Quyết định xem cần phải chuyển thông tin đầu vào cho hàm xử lý hay không? Nếu có thì cần phải chỉ ra kiểu dữ liệu và tên biến chứa thông tin này. Ví dụ 1: `int main(int argc, const char * argv[]);` Ví dụ 2: `int max(int A, int B)`
  4) , 5) Viết mã lệnh: Khai báo biến và thân của hàm
  6) Trả về một biểu thức thích hợp.

Làm sao để viết hàm

- Bài tập: Viết hàm tìm giá trị lớn nhất của 2 số nguyên.

Làm sao để gọi hàm

- Giả sử hàm `int maximum(int A, int B)` được thiết kế để trả về giá trị lớn nhất giữa 2 số nguyên A, B. Để sử dụng nó trong hàm main, chúng ta có thể viết chương trình sau:

```
#include <stdio.h>

int maximum(int A, int B){
    if(A>=B) return A;
    else return B;
}
int main()
{
    int x, y, max;

    scanf("%d%d", &x, &y);
    max = maximum(x, y);
    printf("The maximum of the integers you entered was %d",max);

    return 0;
}
```


Các phương pháp truyền tham số

- Truyền giá trị:
  - Là phương pháp truyền tham số mà sau đó hàm được truyền có được một phiên bản được lưu trữ riêng biệt giá trị của các tham số đó.
  - Khi truyền giá trị, thì giá trị gốc (được truyền) sẽ không bị thay đổi cho dù hàm được truyền có thay đổi các giá trị này đi nữa.
- Truyền tham chiếu:
  - Là phương pháp truyền tham số mà nó cho phép hàm được truyền tham khảo đến vùng nhớ lưu trữ giá trị gốc của tham số.
  - Nếu ta truyền tham chiếu thì giá trị gốc của tham số có thể được thay đổi bởi các mã lệnh bên trong hàm.

Truyền giá trị

- Xét hàm **swap** dùng để đổi nội dung 2 biến **x, y**:

```
#include <stdio.h>
void swap(int x, int y) {
    int temp = x;
    x = y;
    y = temp;
}
int main(){
    int A, B;

    scanf("%d%d", &A, &B);

    printf("A=%d, B=%d\n",A,B);
    swap(A, B);
    printf("A=%d, B=%d\n",A,B);

    return 0;
}
```

Kết quả chạy:
Nhập:
1 3
Trước khi swap:
A=1, B=3
Sau khi swap:
A=1, B=3

Truyền giá trị

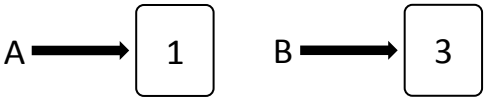
Kết quả chạy:
Nhập:
1 3
Trước khi swap:
A=1, B=3
Sau khi swap:
A=1, B=3

- Hãy nhìn vào kết quả: Đó có phải là kết quả mà ta mong muốn hay không?
- Có lẽ là không! Sau khi swap, A nên là 3 và B nên là 1.
- Để hiểu rõ hiện tượng này, chúng ta sẽ xem những gì diễn ra trong bộ nhớ.

Truyền giá trị

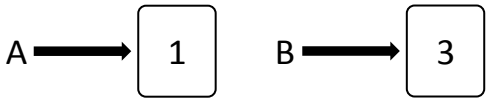
- Chúng ta khai báo 2 biến **A** và **B**. Do đó trong bộ nhớ sẽ có không gian chứa giá trị 2 biến này:

- Sau đó chúng ta nhập giá trị cho **x, y** từ bàn phím:



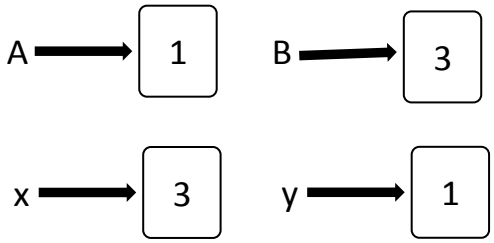
Truyền giá trị

- Khi **A** và **B** được truyền cho **swap()**, một phiên bản của **A** và **B** được tạo ra cho **x** và **y**:



Truyền giá trị

- Khi ta chạy các lệnh trong hàm **swap()** chỉ **x, y** được hoán đổi giá trị, còn giá trị của **A, B** được giữ nguyên:



Truyền tham chiếu

- Xét hàm **swap** dùng để đổi nội dung 2 biến **x, y** được truyền tham chiếu với dấu * được đặt trước 2 biến **x, y**:

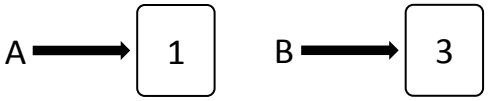
```c
#include <stdio.h>

void swap(int *x, int *y) {
    int temp = *x;
    *x = *y;
    *y = temp;
}

int main()
{
    int A, B;

    scanf("%d%d", &A, &B);

    printf("A=%d, B=%d\n",A,B);
    swap(&A, &B);
    printf("A=%d, B=%d\n",A,B);

    return 0;
}
```

Kết quả chạy:
Nhập:
1 3
Trước khi swap:
A=1, B=3
Sau khi swap:
A=3, B=1

Truyền tham chiếu

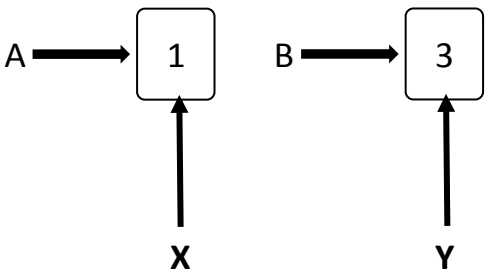
- Chúng ta khai báo 2 biến **A** và **B**. Do đó trong bộ nhớ sẽ có không gian chứa giá trị 2 biến này:

- Sau đó chúng ta nhập giá trị cho **x, y** từ bàn phím:



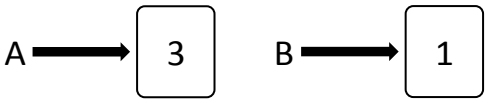
Truyền tham chiếu

- Khi **A** và **B** được truyền cho **swap()**, **x** và **y** sẽ trỏ tới cùng vùng nhớ của **A** và **B**:



Truyền tham chiếu

- Khi ta trở về chương trình chính từ hàm **swap**(), giá trị sau cùng của A và B là:



- Kết quả là các giá trị được hoán đổi đúng.

Truyền tham chiếu

- Xét hàm **swap** dùng để đổi nội dung 2 biến **x, y** được truyền tham chiếu:

```c
#include <stdio.h>

void swap(int *x, int *y) {
    int temp = *x;
    *x = *y;
    *y = temp;
}

int main()
{
    int A, B;

    scanf("%d%d", &A, &B);

    printf("A=%d, B=%d\n",A,B);
    swap(&A, &B);
    printf("A=%d, B=%d\n",A,B);

    return 0;
}
```

Kết quả chạy:
Nhập:
1 3
Trước khi swap:
A=1, B=3
Sau khi swap:
A=3, B=1

Tổng kết

- Chúng ta đã tìm hiểu như thế nào là hàm, cách thiết kế và gọi nó.
- Các cơ chế truyền tham số cho hàm cũng được thảo luận.
- Phần sau sẽ là các bài tập về lý thuyết cũng như lập trình cho học viên có cơ hội ôn tập lại lý thuyết cũng như kỹ năng lập trình hàm.

Kiểm tra kiến thức - Quiz

Câu hỏi 1 đến 4 sử dụng chương trình sau:

```
const int LIMIT = 50;          // Line 1
int AddEm(int x, int y);       // Line 2
int main() {                   // Line 3
    int x = 42,                // Line 4
    y = 35;                    // Line 5
    int Sum;                   // Line 6
    Sum = AddEm(x, y);         // Line 7
    return 0;                  // Line 8
}                              // Line 9
int AddEm(int x, int y) {      // Line 10
    int Total;                 // Line 11
    Total = x + y;             // Line 12
    if (Total > LIMIT)         // Line 13
        Total = 0;             // Line 14
    return (Total);            // Line 15
}                              // Line 16
```

Kiểm tra kiến thức - Quiz

1. Phạm vi của biến *Sum* được khai báo ở dòng 6:
   a) Dòng 1 đến dòng 16 b) 6 đến 16
   c) 6 d) 6 đến 7
   e) 6 đến 9 f) Không phải các trường hợp trên.
2. Phạm vi của biến *x* được khai báo ở dòng 10:
   a) Dòng 1 đến dòng 16 b) 4 đến 16
   c) 10 d)10 đến 12
   e) 10 đến 16 f) Không phải các trường hợp trên.
3. Phạm vi của biến *LIMIT* được khai báo ở dòng 1:
   a) Dòng 1 đến dòng 16 b) 1 đến 3
   c) 1 d) 10 đến 13
   e) 10 đến 16 f) Không phải các trường hợp trên.

Kiểm tra kiến thức - Quiz

4. Câu nào sau đây là đúng:
   a) *LIMIT* là cục bộ trong hàm *main*()
   b) *Total* là cục bộ đối với hàm *AddEm*()
   c) *Sum* là cục bộ trong hàm *main*()
   d) *LIMIT* là toàn cục
   e) *x* là toàn cục
   f) Tất cả các câu trên đều đúng.
   g) Tất cả các câu trên đều đúng trừ câu a.
   h) Chỉ câu b và c là đúng
   i) Chỉ câu b, c và d là đúng
   j) Không có câu nào ở trên là đúng.

Kiểm tra kiến thức - Quiz

5. Tham số hình thức được liệt kê trong ___ của hàm và tham số thực tế được liệt kê trong _____ của hàm.
   a) Lời gọi, cài đặt
   b) Cài đặt, lời gọi
   c) Tiêu đề, thân
   d) Thân, tiêu đề
   e) Không phải các câu trên.
6. Một tham số có kiểu đơn giản (ví dụ như *int* hoặc *double*) nên được truyền giá trị nếu luồng dữ liệu của tham số đó là:
   a) Một chiều, đi vào trong hàm
   b) Một chiều, đi ra khỏi hàm
   c) Hai chiều, vào và ra khỏi hàm
   d) Không phải các câu trên.

Kiểm tra kiến thức - Quiz

7. Câu khẳng định nào sau đây là đúng khi tham số được truyền theo giá trị:
   a) Tham số thật không bao giờ được thay đổi bởi sự thực thi của hàm.
   b) Tham số hình thức không bao giờ được thay đổi bởi sự thực thi của hàm.
   c) Tham số thật phải là một biến.
   d) Câu a, b và c đúng.
   e) Câu a, b và c sai.
   f) Chỉ có câu a và b là đúng.
   g) Chỉ có câu a và c là đúng.
   h) Chỉ có câu b và c là đúng.
   i) Không có câu nào là đúng.

Kiểm tra kiến thức - Quiz

8. Câu khẳng định nào sau đây là đúng khi tham số được truyền theo tham chiếu:
   a) Tham số thật có thể được thay đổi bởi sự thực thi của hàm.
   b) Tham số hình thức có thể được thay đổi bởi sụ thực thi của hàm.
   c) Tham số thật không thể là một biến.
   d) Câu a, b và c đúng.
   e) Câu a, b và c sai.
   f) Chỉ có câu a và b là đúng.
   g) Chỉ có câu a và c là đúng.
   h) Chỉ có câu b và c là đúng.
   i) Không có câu nào là đúng.

Kiểm tra kiến thức - Quiz

9. Nếu dấu chỉ con trỏ '*' không được đặt kèm với kiểu của tham số hình thức thì tham số thật tương ứng có thể là :
   a) Một hằng số.
   b) Một tên biến
   c) Một biểu thức bất kỳ nào
   d) Câu a, b và c đúng.
   e) Chỉ có câu a và b là đúng.
   f) Chỉ có câu a và c là đúng.
   g) Chỉ có câu b và c là đúng.
   h) Không có câu nào là đúng.

Kiểm tra kiến thức - Quiz

10. Cho một hàm được định nghĩa như sau:

```
void func(float *gamma){
    *gamma = 5.5;
}
```

Câu nào sau đây mô tả chính xác nhất luồng dữ liệu của tham số *gamma*:

a) Một chiều, đi vào hàm.
b) Một chiều, đi ra khỏi hàm.
c) Hai chiều, đi vào và ra khỏi hàm.
d) Không câu nào đúng.

Kiểm tra kiến thức - Quiz

11. Cho một hàm được định nghĩa như sau:

```
void demo( int intVal, double *doubleVal ) {
    intVal = intVal * 2;
    *doubleVal = (double)intVal + 3.5;
}
```

Đoạn mã lệnh sau đây sẽ in ra những gì?

```
int myInt = 20;
double myDble = 4.8;
demo(myInt, &myDble);
printf("myInt=%d; myDble=%f", myInt, myDble);
```

a) myInt=20; myDble=43.5
b) myInt=40; myDble=4.8
c) myInt=20; myDble=4.8
d) myInt=40; myDble=43.5
e) myInt=20; myDble=23.5
f) myInt=40; myDble=23.5

Kiểm tra kiến thức - Quiz

12. Cho một hàm được định nghĩa như sau:

```
int Power(int* Base, int* Exponent ) {
    int Product = 1;
    while (*Exponent >= 1) {
        Product = Product * (*Base);
        *Exponent=*Exponent-1;
    }
    return Product;
}
```

Đoạn mã lệnh sau đây sẽ in ra những gì?

```
int N = 2;
int Pow = 3;
int Result = Power(&N, &Pow);
printf("%d ^ %d=%d", N, Pow, Result);
```

a) 2 ^ 3 = 8
b) 2 ^ 0 = 8
c) 0 ^ 0 = 0
d) 2 ^ 3 = 1
e) Không có câu nào đúng

Bài tập tổng kết

1. Viết hàm tính giai thừa của số n (nhập từ bàn phím).
2. Viết hàm tính tổ hợp chập k của n (k, n nhập từ bàn phím).
3. Viết hàm trả về số Fibonacci thứ n (n nhập từ bàn phím. Với:
   Fibonacci(0) = 0
   Fibonacci(1) = 1
   Fibonacci(n) = Fibonacci(n-1)+Fibonacci(n-2)

Bài tập tổng kết

4. Viết hàm chuyển từ số thập phân n thành số nhị phân.
5. Viết chương trình in ra tam giác của các số nguyên tố tăng dần. Ví dụ:

```
2
3 5
7 11 13
17 19 23 29
```

Số dòng cần in ra (n) nhập từ bàn phím.

CT101 – Lập trình căn bản

Khoa CNTT&TT – ĐHCT