Phương pháp Lập trình Hướng đối tượng

Nhìn lại lập trình cơ bản Qua lăng kính của Lập trình Hướng đối tượng

GV: Lê Xuân Định

Hàm

– Đơn vị Xử lý Dữ liệu –

"Mọi tương tác đều qua Giao diện!"

Hàm – Đơn vị xử lý dữ liệu

- Mỗi thao tác xử lý dữ liệu được thực hiện bởi 1 **hàm**.
  - Tương đương với 1 **động từ** [1] trong ngôn ngữ tự nhiên.
- Với mỗi hàm, phải xác định những *dữ liệu được xử lý* (**đầu vào**) và những *kết quả xử lý* (**đầu ra**).
- Ví dụ:
  - Tính tổng tất cả các ước số của một số nguyên cho trước.
  - Cho một mảng các số thập phân, tìm số lớn nhất trong những phần tử mảng nhỏ hơn một số nguyên cho trước.
  - Kiểm tra xem tổng các số trong một mảng các số nguyên có phải là một số nguyên tố hay không.

---

1) Nếu hàm có *giá trị trả về*, ta thường đặt tên hàm là *danh/tính từ* tương ứng.

Hàm – Đơn vị xử lý dữ liệu

- Mỗi thao tác xử lý dữ liệu được thực hiện bởi 1 **hàm**.
  - Tương đương với 1 **động từ** [1] trong ngôn ngữ tự nhiên.
- Với mỗi hàm, phải xác định những *dữ liệu được xử lý* (**đầu vào**) và những *kết quả xử lý* (**đầu ra**).
- Ví dụ:
  - ***Tính* tổng** tất cả các ước số của một **số nguyên cho trước**.
  - Cho **một mảng các số thập phân**, ***tìm* số lớn nhất** trong những phần tử mảng nhỏ hơn một số nguyên cho trước.
  - ***Kiểm tra*** xem tổng các số trong **một mảng các số nguyên** có phải là một số nguyên tố **hay không**.

---

1) Nếu hàm có *giá trị trả về*, ta thường đặt tên hàm là *danh/tính từ* tương ứng.

Hàm – Đơn vị xử lý dữ liệu

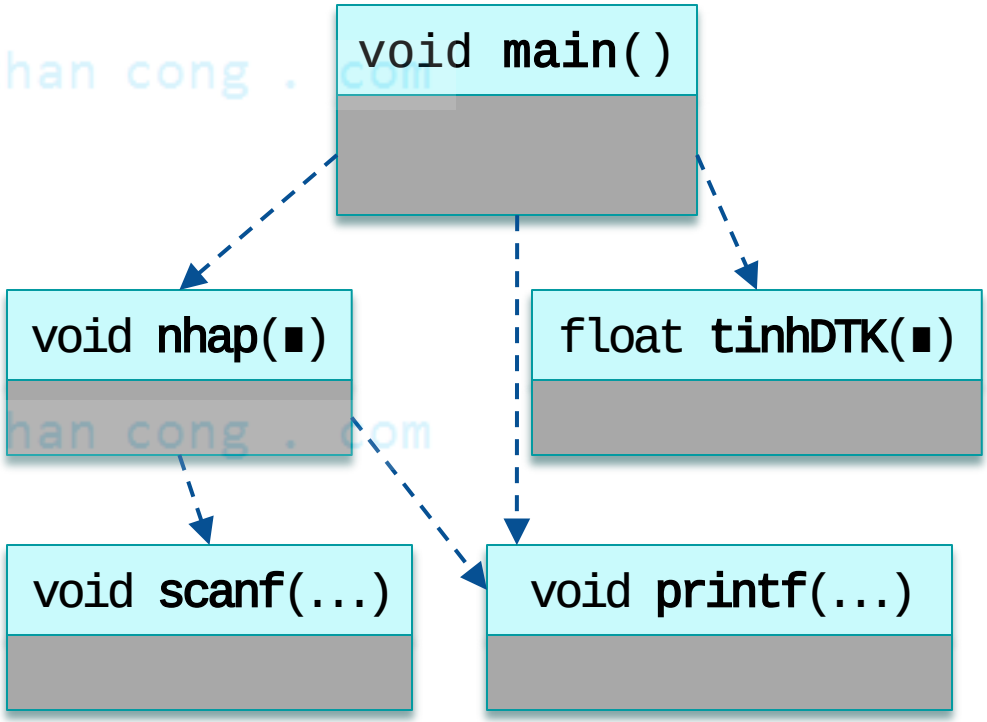
- Chương trình là một cuộc ***trò chuyện*** *giữa các hàm*.
  - Bắt đầu từ hàm main():
    - main() chia toàn chương trình thành các tác vụ, và
    - main() giao việc (tác vụ) cho các hàm con.
  - Mỗi hàm con lại nói chuyện với các hàm khác để hoàn thành công việc.
  - **Giao tiếp** giữa các hàm: *Gọi tên, truyền đối số, trả về kết quả*.

Hàm – Đơn vị xử lý dữ liệu

- Chương trình là một cuộc ***trò chuyện*** *giữa các hàm*. VD: *Tính điểm tổng kết của SV.*

Hàm *Nhập*:

- **Gọi hàm *In màn hình (printf)*** để xuất câu thông báo nhập điểm.
- **Gọi hàm *Quét bàn phím (scanf)*** để nhập điểm.

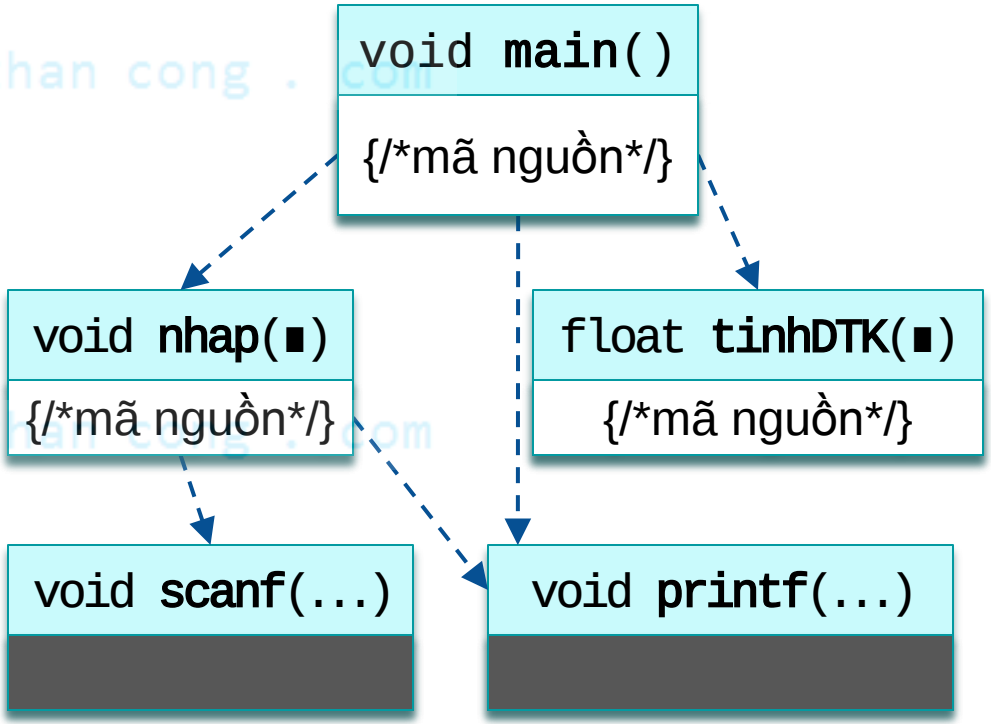


Hàm – Đơn vị xử lý dữ liệu

- Chương trình là một cuộc ***trò chuyện*** *giữa các hàm*. VD: *Tính điểm tổng kết của SV.*

Hàm *scanf*: mã đóng!
Hàm *printf*: mã đóng!

- Với người sử dụng hàm, phần cài đặt hàm là mô ***hộp đen*** (không thấy mã nguồn).
- Để ***sử dụng*** hàm
  - ***Chỉ cần thấy giao diện!!!*** *(nguyên mẫu hàm)*



Giao tiếp giữa các hàm

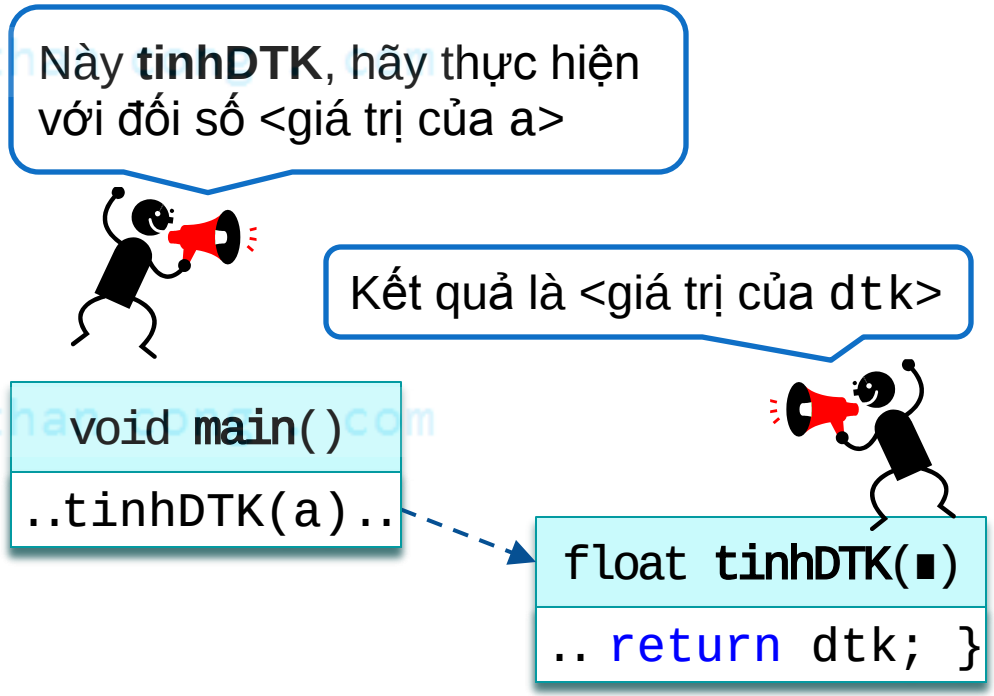
- Chương trình là một cuộc ***trò chuyện*** giữa các hàm. Gồm nhiều cuộc **đối thoại**.

main() - tinhDTK()

- main() ***gọi hàm*** tinhDTK():
  - main() ***truyền tham số*** cho tinhDTK().
  - tinhDTK() ***trả kết quả*** về cho main().

Còn nói

- main() ***sử dụng*** tinhDTK() (trong phần cài đặt).

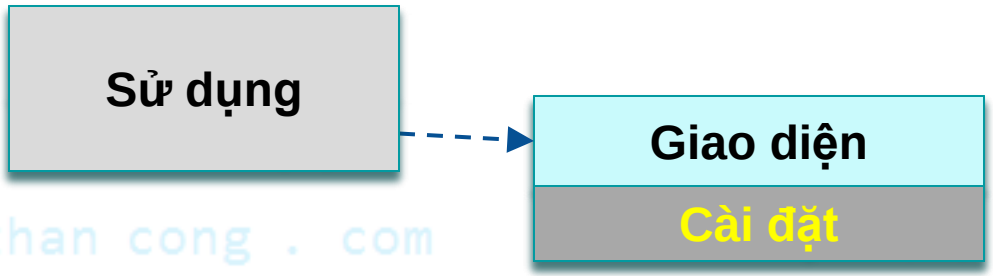


Các thành phần của hàm

- Chương trình là một cuộc ***trò chuyện*** giữa các hàm. Gồm nhiều cuộc **đối thoại**.

Đối với hàm tinhDTK()

- **Bên ngoài** có những hàm ***sử dụng*** tinhDTK().
- **Bên trong** là phần ***cài đặt*** xử lý của tinhDTK().
- **Trung gian ở giữa** là phần ***giao diện*** lập trình của tinhDTK() quy định cách giao tiếp giữa hai bên.



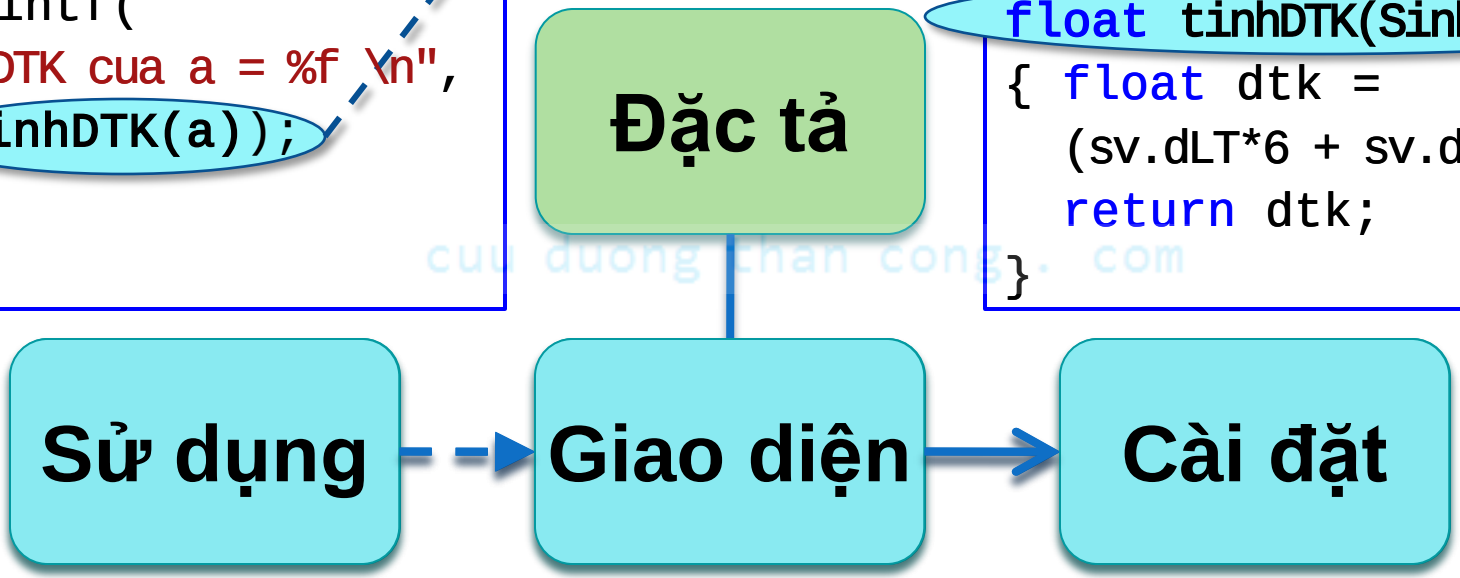
Các thành phần của hàm

- Với mỗi hàm, ta chia thế giới ra làm 4 phần.

```
void main()
{
  SinhVien a, b;
  nhap(a); nhap(b);
  printf(
   "DTK cua a = %f \n",
  tinhDTK(a));
...
}
```

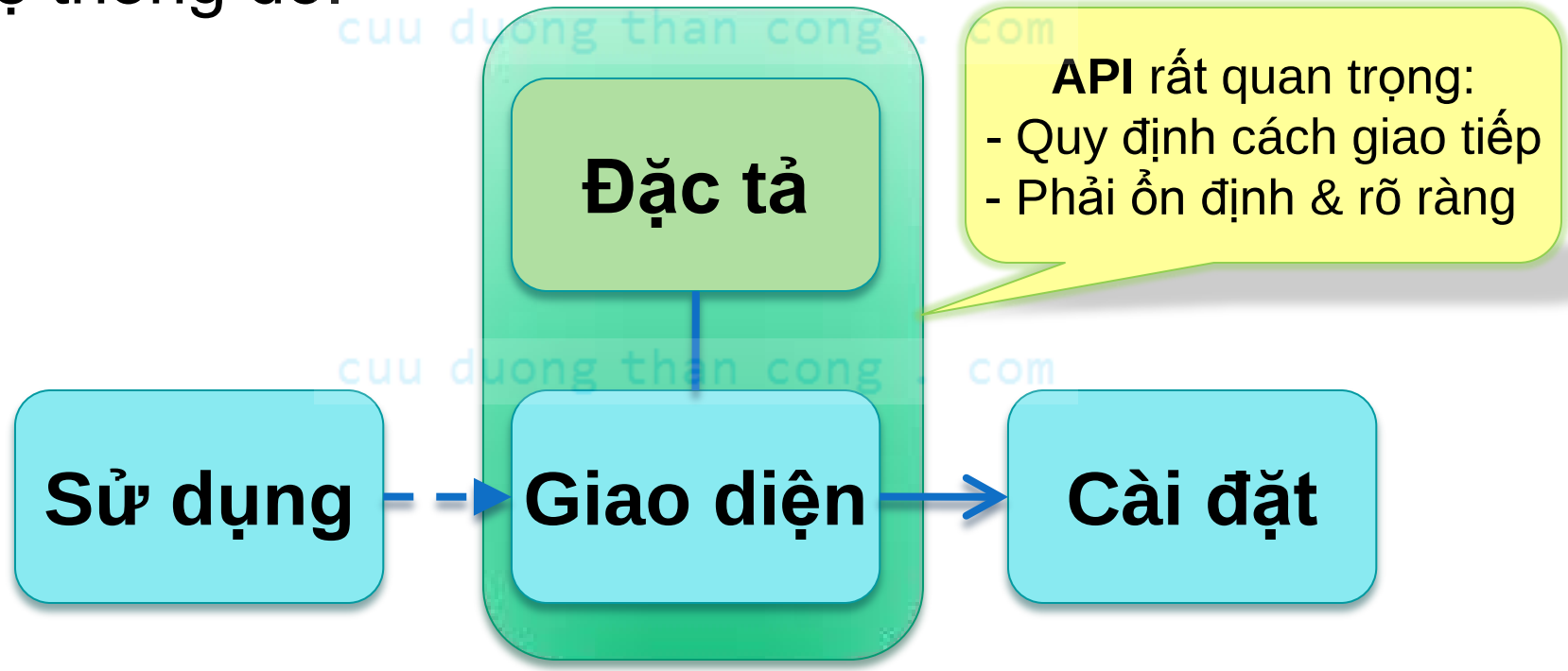
```
/// Hàm tính điểm tổng kết theo công thức:
///    điểm TK = (điểm LT*6 + điểm TH*4) / 10
/// Input: struct chứa điểm LT và TH; Output: điểm TK
float tinhDTK(SinhVien sv);
```

```
float tinhDTK(SinhVien sv)
{ float dtk =
  (sv.dLT*6 + sv.dTH*4)/10;
  return dtk;
}
```



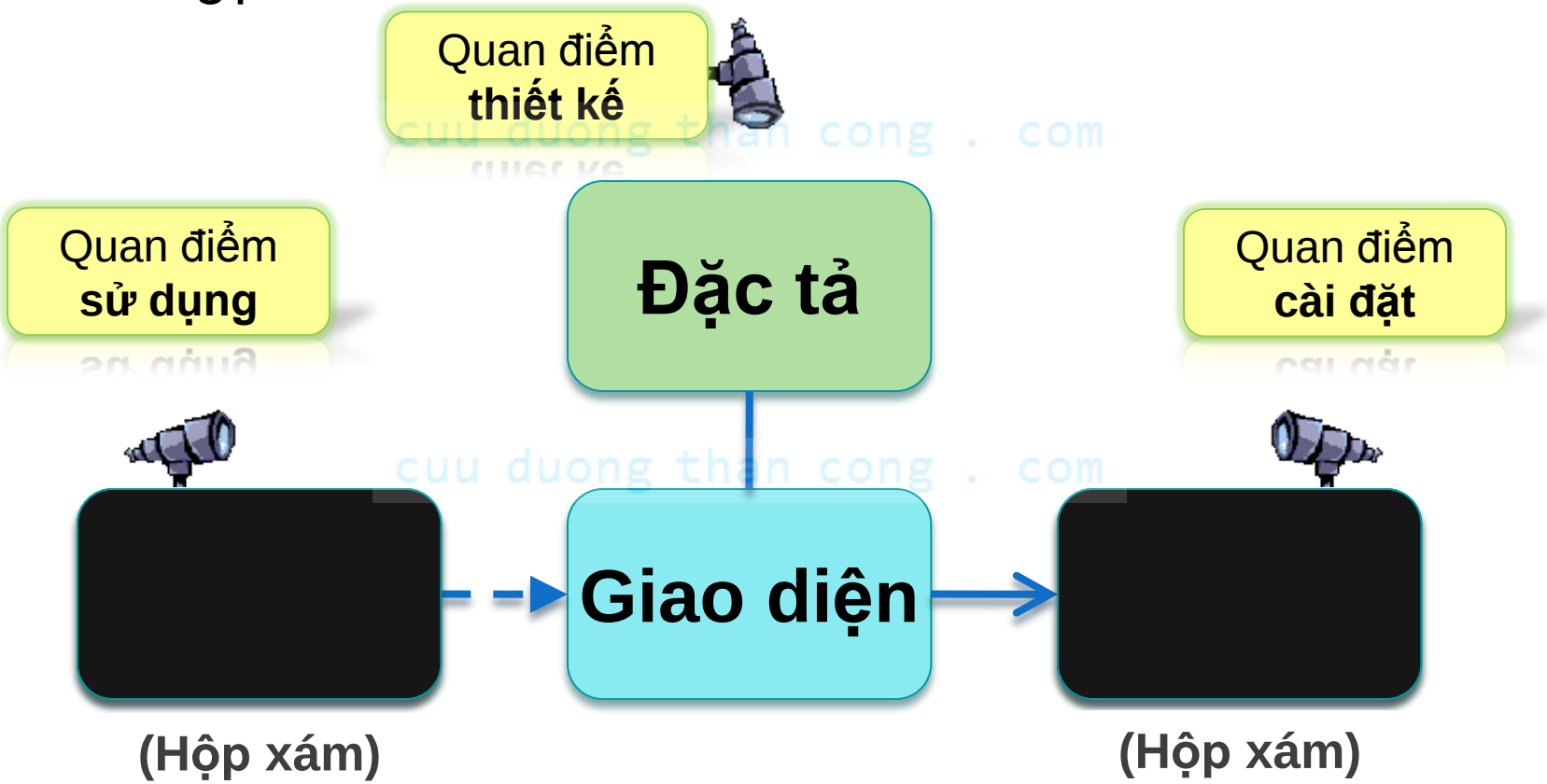
Các thành phần của hàm

- Tập hợp các giao diện và đặc tả của một hệ thống tạo nên bộ ***Giao diện Lập trình*** (**API**) của hệ thống đó.



Các thành phần của hàm

- Tuỳ góc nhìn, ta có thể thấy hoặc không thấy nội dung của từng phần.

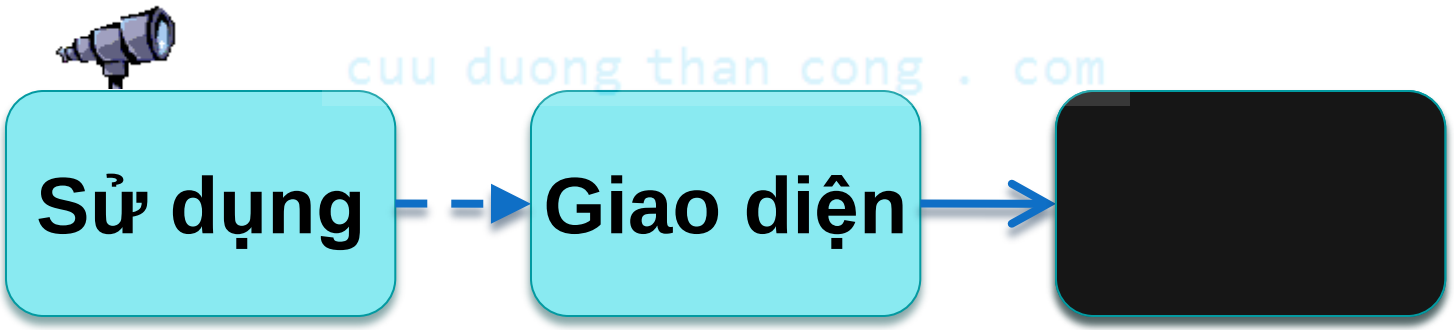


Quan điểm Sử dụng Hàm

- Ta đứng trên quan điểm sử dụng hàm *f*() khi
  - *f*() là một hàm *có sẵn*, VD: printf() trong thư viện `stdio`; Hoặc
  - Trách nhiệm thiết kế & cài đặt hàm là *của người khác*,
    VD: nhà phát triển thư viện, thành viên khác trong nhóm.

Khi đó...

- *Bên **sử dụng** là **bên ngoài**!*
  - Coi phần bên trong hàm tức *phần **cài đặt** là **hộp đen***.



Nguyên mẫu hàm:

Quan điểm Sử dụng Hàm

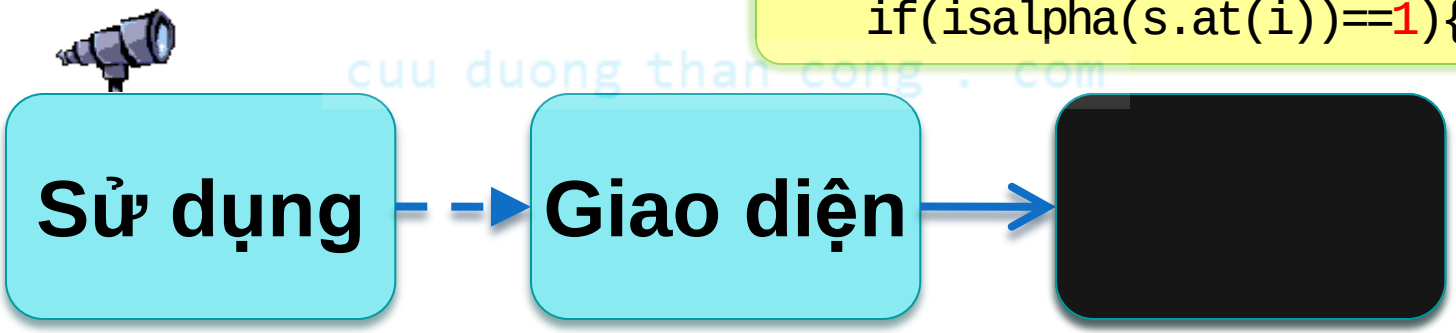
- *Bên **sử dụng** là **bên ngoài**!*
  - Coi phần bên trong hàm tức *phần **cài đặt** là **hộp đen***.
    - Chỉ cần biết hàm này ***làm cái gì** (qua đặc tả)*, không nên quan tâm nó ***làm thế nào** ở bên trong*.
    - Tuân thủ quy tắc "hộp đen":

      Dù thấy được cũng coi như không thấy!!!

      (Chớ nên phụ thuộc vào nó!)

An toàn!!!

VD sai: `if(s.compare(t)==-1){...}`
`if(isalpha(s.at(i))==1){...}`



Nguyên mẫu hàm:

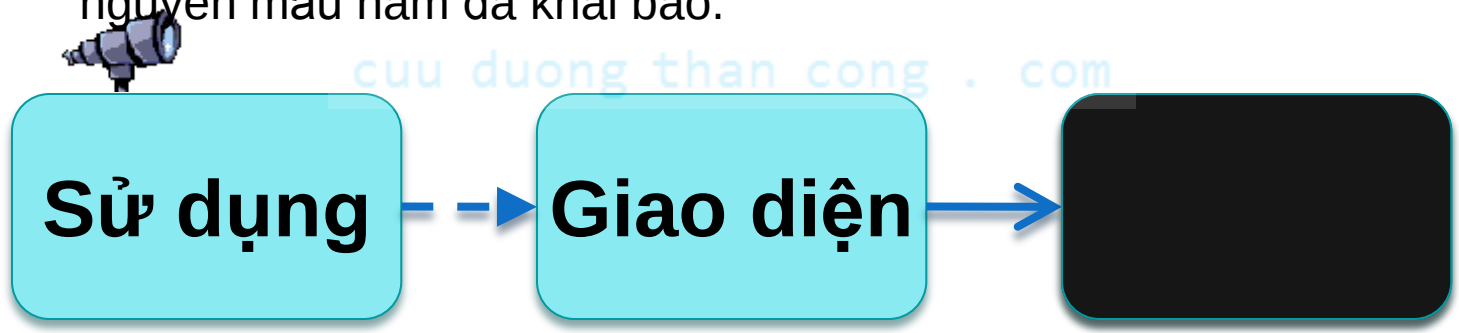
Quan điểm Sử dụng Hàm

- *Bên **sử dụng** là **bên ngoài**!*
  - Coi phần bên trong hàm tức *phần **cài đặt** là **hộp đen***.
    - Chỉ cần biết hàm này ***làm cái gì** (qua đặc tả),* không nên quan tâm nó ***làm thế nào** ở bên trong*.
  - Phải ***gọi hàm** **đúng theo** **giao diện***, tức nguyên mẫu hàm.

Tương ứng 1-1

Cũng như dùng biến:

- Muốn dùng hàm (gọi hàm) thì phải *khai báo **nguyên mẫu hàm***;
- Rồi gọi đúng tên hàm và truyền tham số có kiểu tương thích với nguyên mẫu hàm đã khai báo.



Nguyên mẫu hàm:

Quan điểm Sử dụng Hàm

- *Bên **sử dụng** là **bên ngoài**!*
  - Coi phần bên trong hàm tức *phần **cài đặt** là **hộp đen***.
    - Chỉ cần biết hàm này ***làm cái gì*** *(qua đặc tả),* không nên quan tâm nó ***làm thế nào*** *ở bên trong*.
  - Phải ***gọi hàm đúng theo giao diện***, tức nguyên mẫu hàm.

    Cũng như dùng biến,
    - Muốn dùng hàm (gọi hàm) thì phải *khai báo **nguyên mẫu hàm***;
    - Rồi gọi đúng tên hàm và truyền tham số có kiểu tương thích với nguyên mẫu hàm đã khai báo.
  - Về việc "khai báo nguyên mẫu hàm":
    - Các thư viện thường gom sẵn các nguyên mẫu hàm trong file header

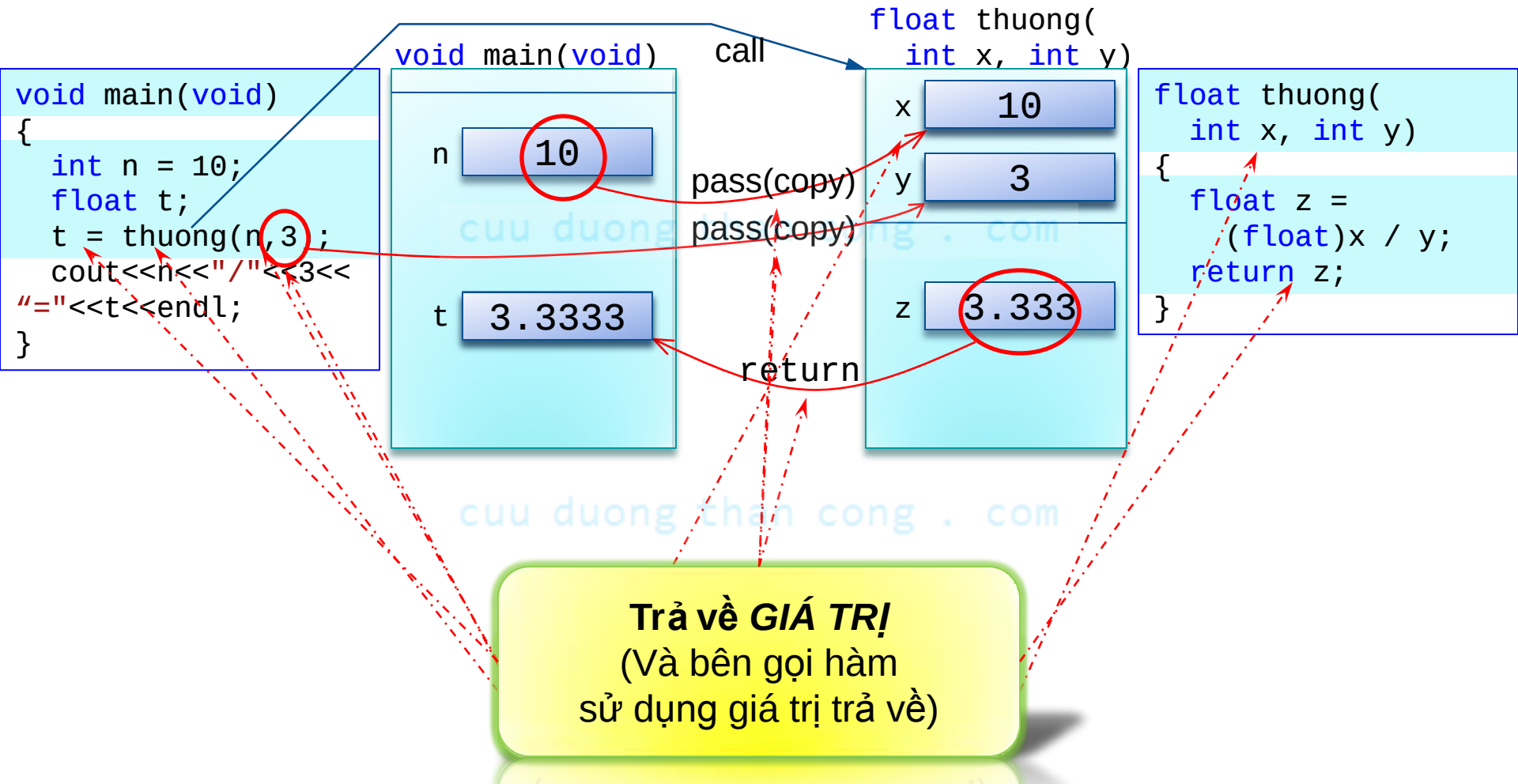
      `#include "myLib.h"` ⇔ ***khai báo tất cả*** các hàm trong `"myLib.h"`.
    - Nếu phần cài đặt hàm đặt trước lời gọi hàm thì không cần khai báo lại vì ngay trên phần cài đặt hàm đã có nguyên mẫu hàm.

Tham số trong Giao diện Hàm

Trong C/C++ có 2 loại tham số (và 2 loại đối số tương ứng)

- Trị (tham trị & đối trị): Truyền và nhận **giá trị** dữ liệu
  - Bên ngoài truyền giá trị vào qua *đối trị*
    VD1: `s=cong(12,21);` //cộng 12 với 21, lưu vào *s*.
    VD2: `s=cong(s,33);` //cộng ***giá trị*** *của s* với 33, lưu vào *s*.
  - Giá trị truyền vào được **copy** & lưu vào biến tham số tương ứng bên trong.
    VD:
    ```
    int cong(int x, int y)
       {  /* biến x, y lưu giá trị đối số */   }
    ```
- Biến (tham biến & đối biến): Truyền và nhận **biến** DL

Cơ chế Truyền Tham số (trị)

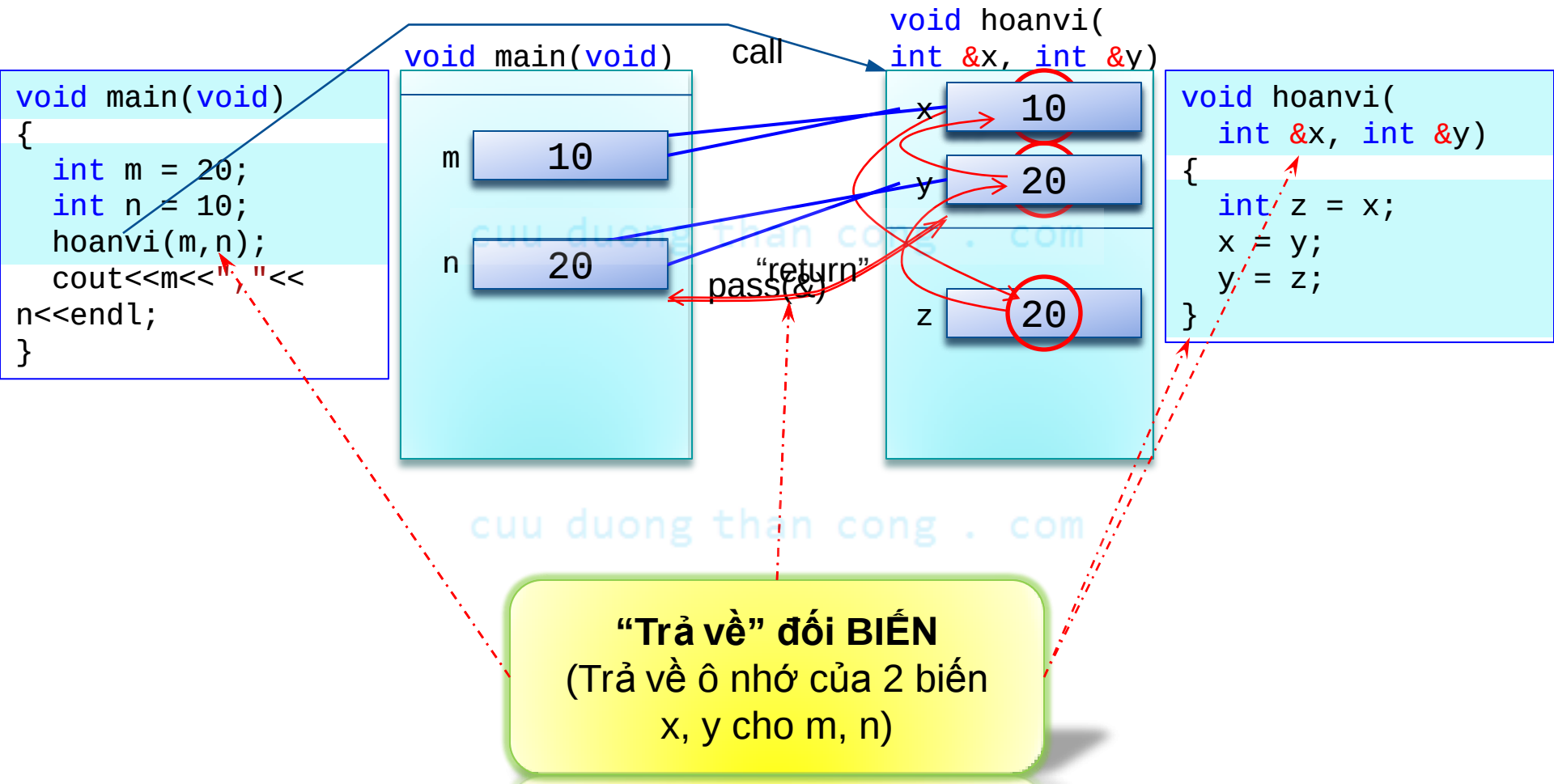


Tham số trong Giao diện Hàm

Trong C/C++ có 2 loại tham số (và 2 loại đối số tương ứng)

- Trị (tham trị & đối trị): Truyền và nhận **giá trị** dữ liệu
- Biến (tham biến & đối biến): Truyền và nhận **biến** DL
  - Bên ngoài **truyền biến** (chứa giá trị cần truyền) vào qua *đối biến*, và cũng **nhận lại giá trị** qua biến đó.
    VD1: `hoanvi(a,b);` //hoán vị giá trị lưu trong a và b.
    VD2: `hoanvi(a,10);` //lỗi cú pháp! (10 ko phải biến)
  - Với bên trong, **biến** truyền vào được **sử dụng trực tiếp** thông qua tên tham biến tương ứng.
    VD:
    ```
    void hoanvi(int &x, int &y)
      {/* biến x,y chính là 2 biến bên ngoài với tên khác */}
    ```
  - Đặc biệt: Tham số **mảng** mặc định là *tham biến*.
    → Không thêm "&" vào trước tham số mảng.

Cơ chế Truyền Tham số (biến)



Dữ liệu Đầu ra của Hàm

- Trị trả về
  - Trong thân hàm: **giá trị** của kết quả tính toán được trả về thông qua câu lệnh `return`.

    VD0:
    ```
    int cong(int x, int y)
       { return 0; } //cài đặt rỗng: trả về giá trị mặc định.
    ```
    VD1:
    ```
    int cong(int x, int y)
       { int s = x + y;
         return s; } //trả về giá trị lưu trong biến s.
    ```
  - Người gọi hàm nhận giá trị trả về và sử dụng.

    VD1: `int s=cong(12,21); //lưu kết quả vào biến s.`

    VD2: `xuat(cong(12,21),"Tong");//truyền kết quả vào hàm xuat().`

- Tham biến
  - Nếu có nhiều giá trị cần phải trả về thì chúng ta phải lưu chúng vào trong các tham biến.

Hoạt cảnh Sử dụng Hàm

- Demo truyền tham trị/biến & trả về (void)
  - VD1: Cộng 2 số nguyên, trả về 1 số nguyên

```
/// trả về a + b
int cong (int a, int b);
```

  - VD2: Cộng dồn một số nguyên vào một số nguyên khác

```
/// b = a + b
void congDon(int a, int &b);
```

Hoạt cảnh Sử dụng Hàm

- Hãy sử dụng các hàm sau:
  - `int Thương(int a, int b);` /// trả về phần nguyên [b/a]
  - `void Dư(int &a, int &b, int &c);` /// c = a - [a/b]*b
  - `int Nhập();` /// trả về số nguyên được nhập
  - `void XuấtUSCLN(int a, int b, int c);`
    /// xuất thông báo "Ước số chung lớn nhất của <a> và <b> là <c>"
  - `void XuấtTuầnThứ(int thu, int tuan);`
    /// hiển thị ngày này là Thứ mấy trong Tuần thứ mấy trong tháng

- Viết CTrình tính USCLN của 2 số
  - Nhập từ Khán giả/Random, xuất ra Bảng/Projector
  - Tính bằng Tay, sau nâng cấp lên bằng Máy

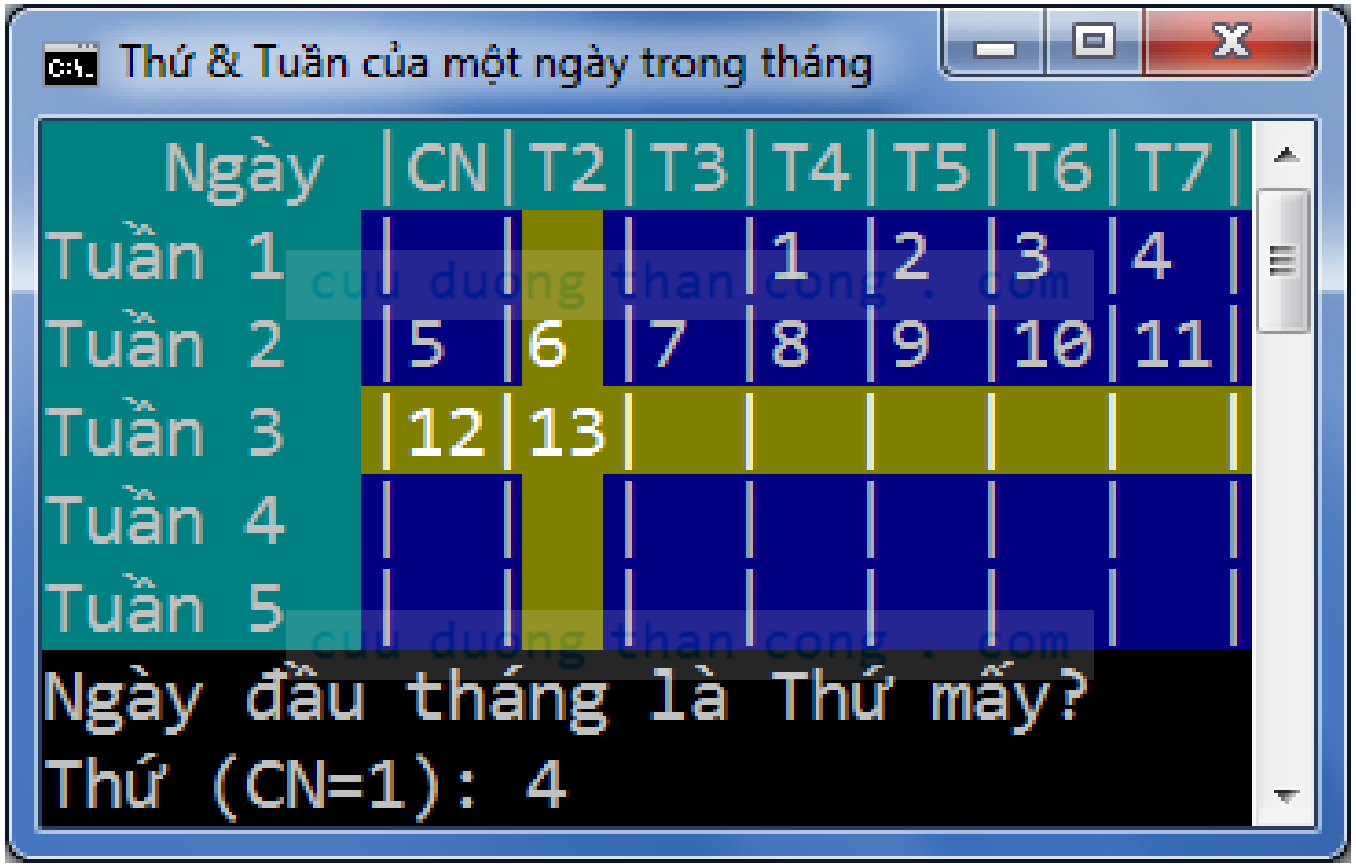
Ước số chung lớn nhất của 1334 và 383

là 1

Hoạt cảnh Sử dụng Hàm

- Hãy sử dụng các hàm sau:
  - `int Thương(int a, int b);` /// trả về phần nguyên [b/a]
  - `void Dư(int &a, int &b, int &c);` /// c = a - [a/b]*b
  - `int Nhập();` /// trả về số nguyên được nhập
  - `void XuấtUSCLN(int a, int b, int c);`
    /// xuất thông báo "Ước số chung lớn nhất của <a> và <b> là <c>"
  - `void XuấtTuầnThứ(int thu, int tuan);`
    /// hiển thị ngày này là Thứ mấy trong Tuần thứ mấy trong tháng

- Viết CTrình cho nhập một ngày trong tháng và nhập tháng này bắt đầu từ Thứ mấy, xong hiển thị cho biết ngày này là Thứ mấy trong Tuần thứ mấy trong tháng.


Thứ & Tuần của một ngày trong tháng
Ngày |CN|T2|T3|T4|T5|T6|T7|
Tuần 1 | | | | 1 | 2 | 3 | 4 |
Tuần 2 |5 |6 |7 |8 |9 |10|11|
Tuần 3 |12|13| | | | | |
Tuần 4 | | | | | | | |
Tuần 5 | | | | | | | |
Ngày đầu tháng là Thứ mấy?
Thứ (CN=1): 4

Khai báo & Sử dụng Hàm

- Giao diện (nguyên mẫu hàm)

kiểu_trả_về **tên_hàm** (danh_sách_**khai_báo**_tham_số);

```
float  thuong (int a, int b);
```


- Sử dụng (lời gọi hàm)

biến_trả_về**= tên_hàm** (danh_sách_**đối_số**);

  - ```
    int x = 1;
    float q = thuong(x,15);
    ```
    //→ Đọc máy móc: ***gọi hàm*** *thuong() với 2* ***đối số (giá trị của*** *x và 15),* *rồi gán* ***giá trị trả về*** *vào biến q.*

    //→ Đọc tự nhiên: *lưu kết quả* ***tính thương*** *của x và 15 vào biến q.*
    ```

Khai báo & Sử dụng Hàm

- Giao diện (nguyên mẫu hàm)

  kiểu_trả_về **tên_hàm** (danh_sách_**khai_báo**_tham_số);

  - VD: `int f(char ch, float a[])`
    //→ đọc máy móc: *Hàm **f** nhận 2 tham số (1 ký tự ch và 1 mảng a các số thập phân) trả về một số nguyên.*
  - VD: `float thuong(int a, int b)`
    //→ đọc tự nhiên: *Hàm **tính thương** của 2 số nguyên (trả về một số thập phân).*

- Sử dụng (lời gọi hàm)

  biến_trả_về**= tên_hàm** (danh_sách_**đối_số**);

Tương ứng giữa **Giao diện & Sử dụng**

- Giao diện (nguyên mẫu hàm)

```
kiểu_trả_về  tên_hàm ( danh_sách_khai_báo_tham_số );
VD: float    thuong  (int a, int b)
```

  - *Khai báo tham số* := ***khai báo biến***

- Sử dụng (lời gọi hàm)

```
biến_trả_về = tên_hàm ( danh_sách_đối_số );
VD: float g = thuong  (x,15);
```

  - *Truyền đối số* := *truyền **giá trị** cho biến*
  - Giá trị vào/ra *có kiểu tương ứng với khai báo* trong giao diện.

Tương ứng 1-1

Tương ứng giữa **Giao diện & Sử dụng**

- **Giao diện** thì phải có **kiểu**, sử dụng thì ***không*** *ghi kiểu*, nhưng giá-trị vào/ra phải *có kiểu tương ứng với khai báo* trong giao diện.
- Ví dụ:
  - Giao diện: `float thuong(int a, int b)`
    Sử dụng: `q = thuong(x,y);`
    → Các biến `q,x,y` phải có kiểu: **`float q; int x,y;`**
  - Giao diện: `float thuong(int a, int b)`
    Sử dụng: `q = thuong(x,1.2);`
    → Lỗi cú pháp: `1.2` không thuộc kiểu `int`.

Tương ứng giữa **Giao diện & Sử dụng**

- **Giao diện** thì phải có **kiểu**, sử dụng thì ***không*** *ghi kiểu*, nhưng giá-trị vào/ra phải *có kiểu tương ứng với khai báo* trong giao diện.
- Ví dụ:
  - Giao diện: `float thuong(a, b)`

  → Lỗi cú pháp: `a, b` thiếu kiểu trong nguyên mẫu hàm.
  - Sử dụng: `float q = thuong(int x=2, int y=3)`

  → Lỗi cú pháp: ghi kiểu trong lời gọi hàm.

Tương ứng giữa **Giao diện & Sử dụng**

- **Giao diện** thì phải có **kiểu**, sử dụng thì ***không*** *ghi kiểu*, nhưng giá-trị vào/ra phải *có kiểu tương ứng với khai báo* trong giao diện.
- Ví dụ:
  - Giao diện: `void thuong(int a,`
    Sử dụng: `printf("%f", thuong(`
    → Lỗi cú pháp: sử dụng giá trị trả về không được khai báo.
  - Giao diện: `float thuong(int a,`
    Sử dụng: `thuong(2,3);`
    → Lỗi ngữ nghĩa: ***bỏ mất*** giá trị trả về.

Không có giá trị trả về thì sử dụng **bắt đầu** bằng tên hàm.

Có giá trị trả về thì sử dụng ***không*** **bắt đầu** bằng tên hàm.

Tóm tắt: Giao diện & Sử dụng

- Các cấp phụ thuộc: Biến ← Hàm
  - Khai báo (giao diện) & sử dụng phải tương thích với nhau.

	Khai báo (giao diện)	Sử dụng
Biến	Kiểu, tên biến	**Đọc/ghi giá trị** có kiểu tương ứng: • Khi đọc phải có giá trị xác định. **Truyền tham số** cho hàm: • Gián tiếp đọc/ghi giá trị.
Hàm	Nguyên mẫu hàm: • Kiểu trả về, tên hàm • Danh sách tham số	**Gọi hàm**: • Giá trị vào/ra phải có kiểu tương ứng với nguyên mẫu hàm.

BT Ứng dụng – 1 (Về nhà)

- Hãy khai báo **nguyên mẫu hàm** cho các hàm sau (trước hàm main) và sử dụng chúng (1 lời **gọi hàm** tương ứng với mỗi hàm) trong hàm main.

1. Kiểm tra xem một số nguyên có phải là số chẵn hay không.
2. Kiểm tra xem một số nguyên này có chia hết cho một số nguyên kia hay không.
3. Tính thương của hai số nguyên (chỉ lấy phần nguyên).
4. Tìm số lớn nhất trong mảng các số thập phân.
5. Tìm vị trí số lớn nhất trong mảng các số thập phân.
6. Chèn một số nguyên vào mảng các số thập phân tại vị trí cho trước.

BT Ứng dụng – 2

- Hãy khai báo **nguyên mẫu hàm** (tiếp theo)

  Hãy viết các ***khái báo*** *hàm* trong file MyLib***.h*** để chương trình sau dịch được:

```
#include “MyLib.h”
void main(){
  float a[100]; int x;
  for(int i=0; i<10; i++){
    a[i] = nhapBienThu(i);
  }
  x = nhapSoNguyen();
  int vtri = tim(a,10, x);
  if(laSoNguyenTo(vtri)){
    x = 10; xoaTaiVitri(a,x, vtri);
} }
```

Bảng đối chiếu thuật ngữ

Tiếng Việt	Tiếng Anh	Chú thích
Hàm	Function	Còn gọi "thủ tục" (procedure), "chương trình con" (subprogram, subroutine)
Kiểu trả về, g.trị trả về	Return type, return value	
Tham số, đối số, tham trị, tham biến	Parameter, argument, pass-by-value parameter, pass-by-reference parameter	"Tham số" là cái được dùng bên trong hàm, "đối số" là cái bên ngoài truyền vô cho hàm.
Giao diện (lập trình)	(Programming) interface	Application programming interface (API) là 1 bộ giao diện.
Nguyên mẫu hàm	(Function) prototype	Còn gọi "function signature", "function type"
Gọi hàm	Function call	
Phần cài đặt	Implementation	
Đặc tả	Specification	
Hộp đen, hộp xám	Black box, gray box	