Kỹ thuật lập trình

Programming Techniques

Ts. Nguyễn Đức Thuần
BM Hệ thống Thông Tin

Giới thiệu môn học

Nội dung môn hoc

- Chương 1: Kỹ thuật tổ chức chương trình
- Chương 2: Lập trình có cấu trúc – Hàm nâng cao
- Chương 3:
- Chương 4:
- Chương 5:

LẬP TRÌNH CÓ CẤU TRÚC

Chương 2: LẬP TRÌNH CẤU TRÚC – HÀM NÂNG CAO

I. Lập trình cấu trúc (Structured Programming)

- Trong SP, điều khiển trình tự của chương trình được giới hạn trong ba cấu trúc cơ bản:
  - Tuần tự (sequence)
  - Sự lựa chọn (Choice)
  - Vòng lặp (Loop)

(hoặc một cấu trúc có thể sinh ra từ sự kết hợp của ba cơ bản)

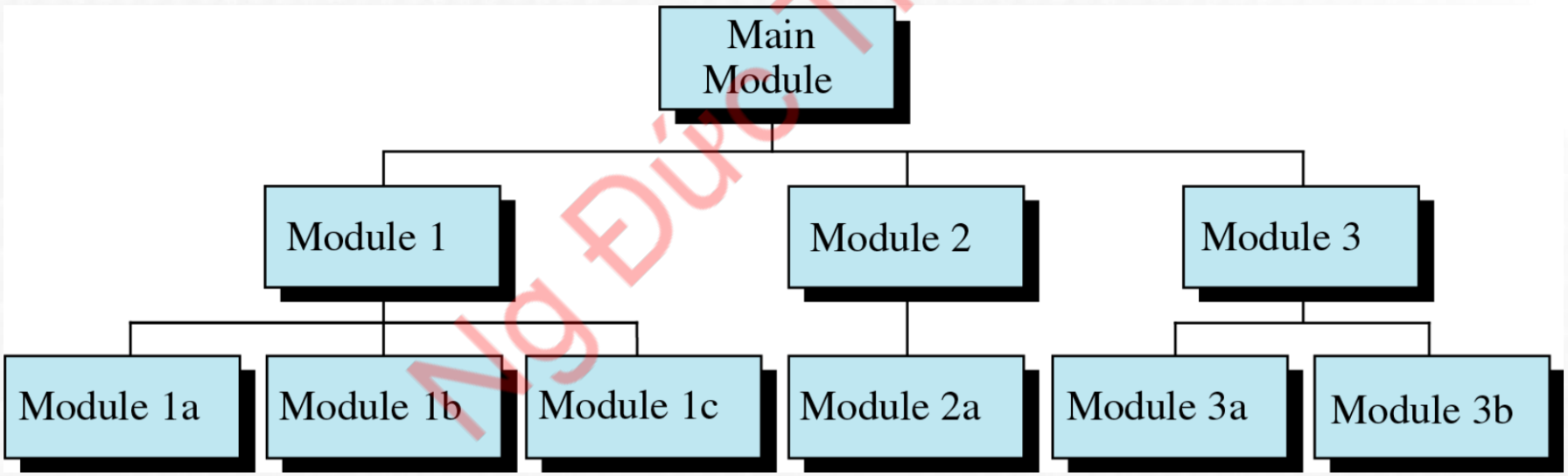
☞ Không cần phải sử dụng GOTO

☞ Một chương trình được xây dựng từ các mô-đun rất độc lập với nhau.

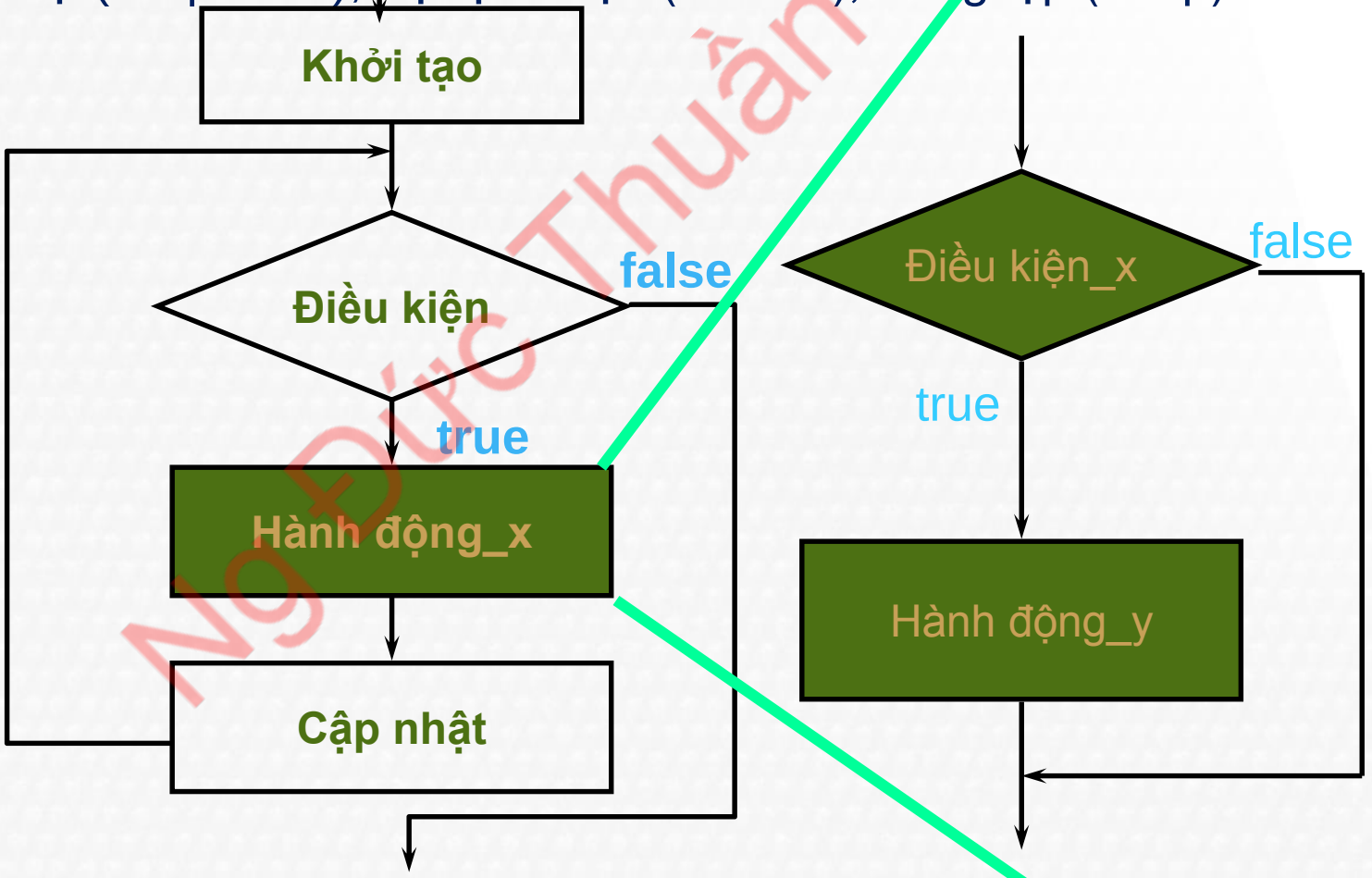
☞ Lập trình mã chứa ít lỗi logic hơn và sẽ dễ dàng hơn để gỡ lỗi và thay đổi trong tương lai. SP có thể kém hiệu quả hơn so với đối tác không có cấu trúc. Một số ngôn ngữ không hỗ trợ nó

Lập trình cấu trúc thiết kế theo kiểu Top-down:

- Một chương trình được chia thành một module chính và các module liên quan. Mỗi module lần lượt được chia thành các module cho đến khi các module kết quả được hiểu mà không cần phân chia thêm.

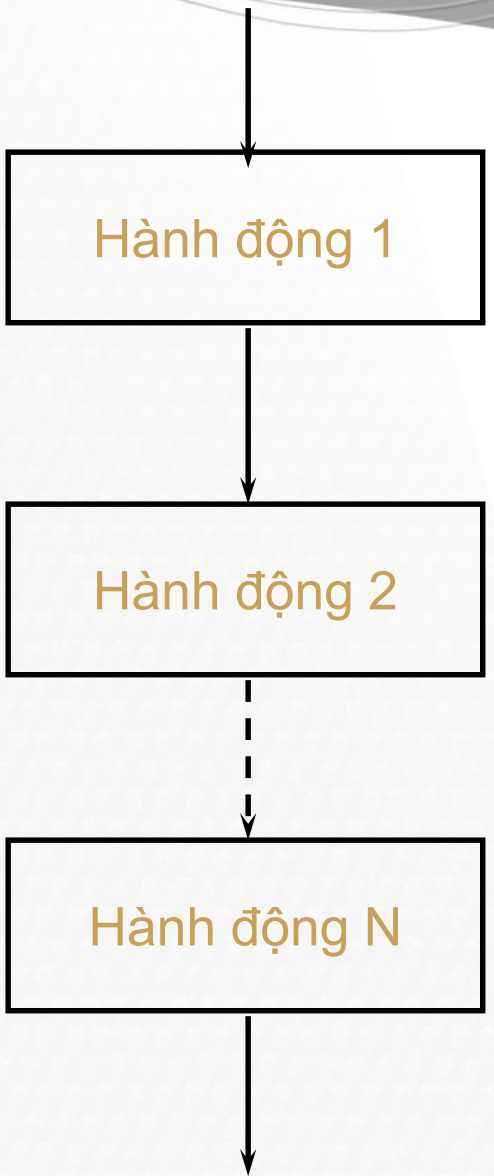


- Điều khiển trình tự của chương trình được giới hạn trong ba cấu trúc cơ bản: Trình tự (Sequence), Sự lựa chọn (Choice), Vòng lặp (Loop)



❖ Tuần tự

Rất tự nhiên khi viết 1 chương trình bằng một dãy tự 3 cấu trúc cơ bản: Tuần tự, Sự lựa chọn, Vòng lặp



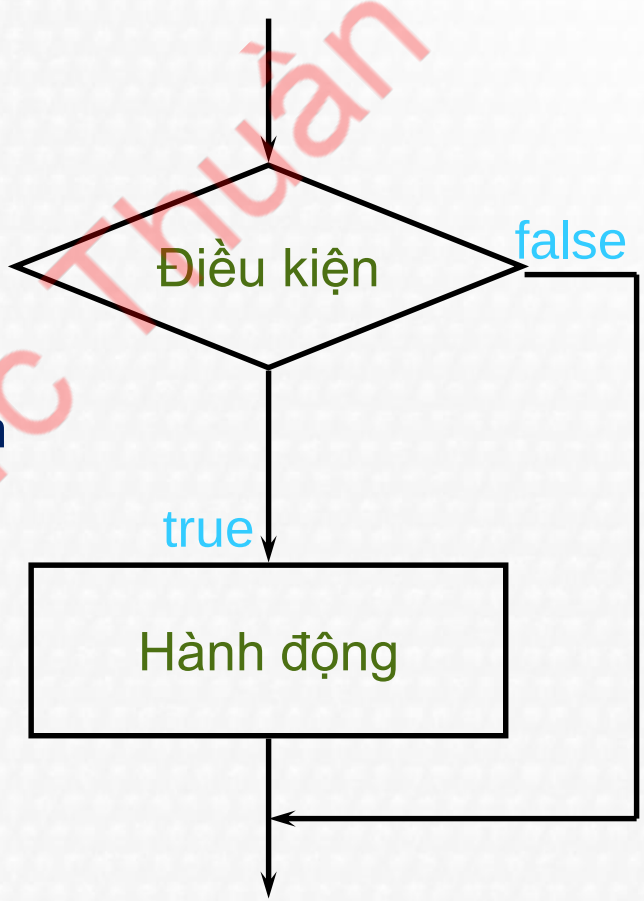
❖ Sự lựa chọn

- Cú pháp Syntax

```
if(condition)
    action
```

- Nếu điều kiện đúng thì thực hiện hành động

✓Hành động có thể là câu lệnh đơn hay câu lệnh kép

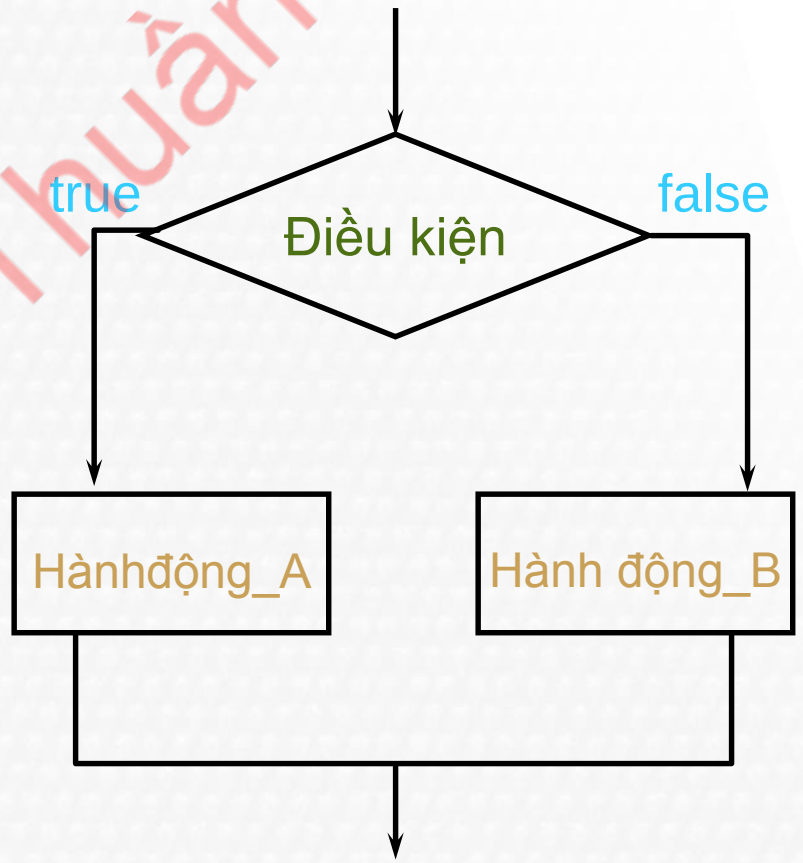


❖ Một câu lệnh lựa chọn khác

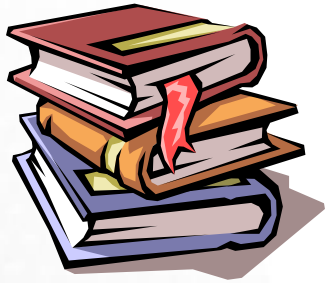
- Cú pháp

```
if (điều kiện)
   Hành động_A
else
   Hành động_B
```

- Nếu điều kiện là đúng thì thực hiện Hành động_A ngược lại thực hiện Hành động_B.



❖ Một câu lệnh lặp



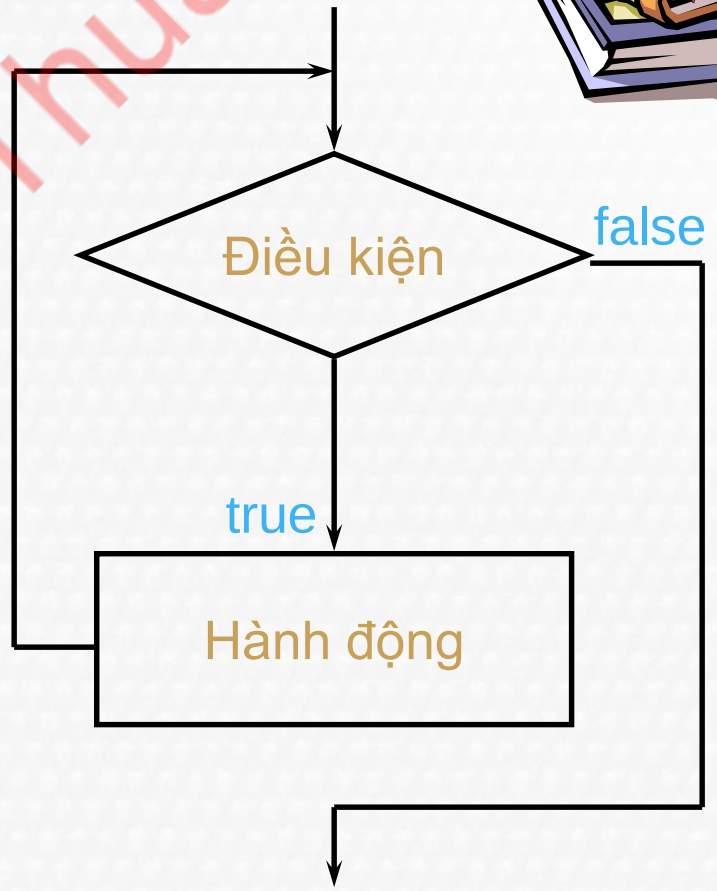
- Cú pháp

  while (điều kiện)

  Hành động

- Thực hiện:
  - Nếu điều kiện là đúng thực hiện hành động
  - Lặp quá trình trên cho đến khi điều kiện là sai

Hành động có thể là câu lệnh đơn hoặc kép.



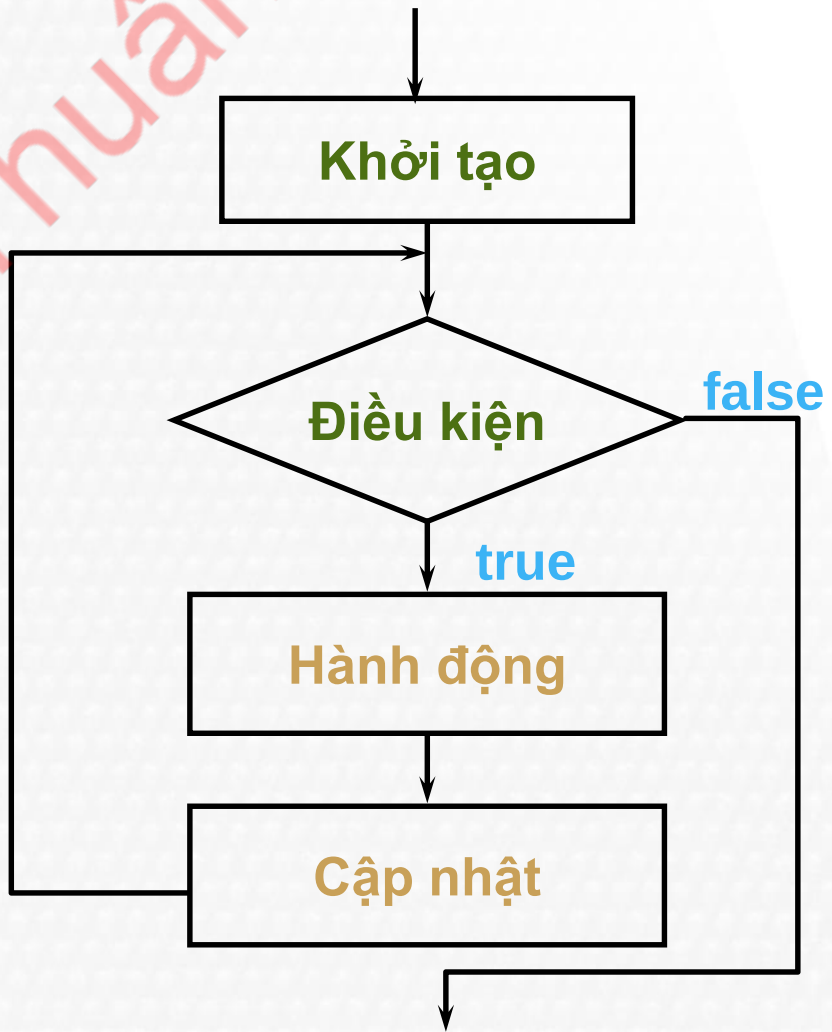
❖ Một câu lệnh lặp khác

- Cú pháp

  for (initialization; condition; update)

  action

- Thực hiện:
  - Thực hiện khởi tạo câu lệnh
  - Trong khi điều kiện đúng
    - Thực hiện hành động
    - Thực hiện cập nhật



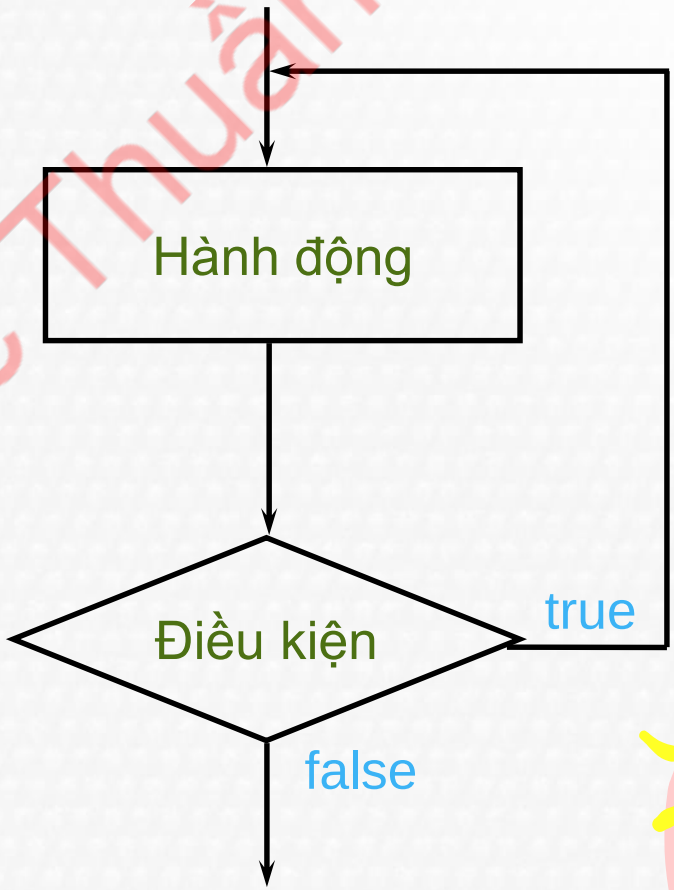
❖ Một câu lệnh lặp khác

Cú pháp

do hành động

while (điều kiện)

- Thực hiện:
  - Thực hiện hành động
  - Nếu điều kiện đúng thì thực hiện lại hành động
  - Lặp quá trình trên cho đến khi điều kiện mang giá trị sai
- Hành động có thể là câu lệnh đơn hoặc kép.



❖ Mẫu Kim cương (Diamond Pattern)

- In ra mẫu kim cương sau:

```
        *
      * * *
    * * * * *
  * * * * * * *
* * * * * * * * *
  * * * * * * *
    * * * * *
      * * *
        *
```

Mẫu kim Cương (Diamond Pattern)

- Bài toán con:
  - In nửa trên
  - In nửa dưới
- In nửa trên:
  - Dòng 1: in 4 k.trắng, 1 sao; *
  - Dòng 2: in 3 k.trắng, 3 sao; * * *
  - Dòng 3: in 2 , 5 ; * * * * *
  - 4: 1 ,7 ; * * * * * * *
  - 5: 0, ,9 ; * * * * * * * * *
- In nửa dưới:
  - Dòng 4: in 1 k.trắng, 7 sao; * * * * * * *
  - 3: 2 , 5 ; * * * * *
  - 2: 3 , 3 ; * * *
  - 1: 4 , 1 ; *

Diamond Pattern

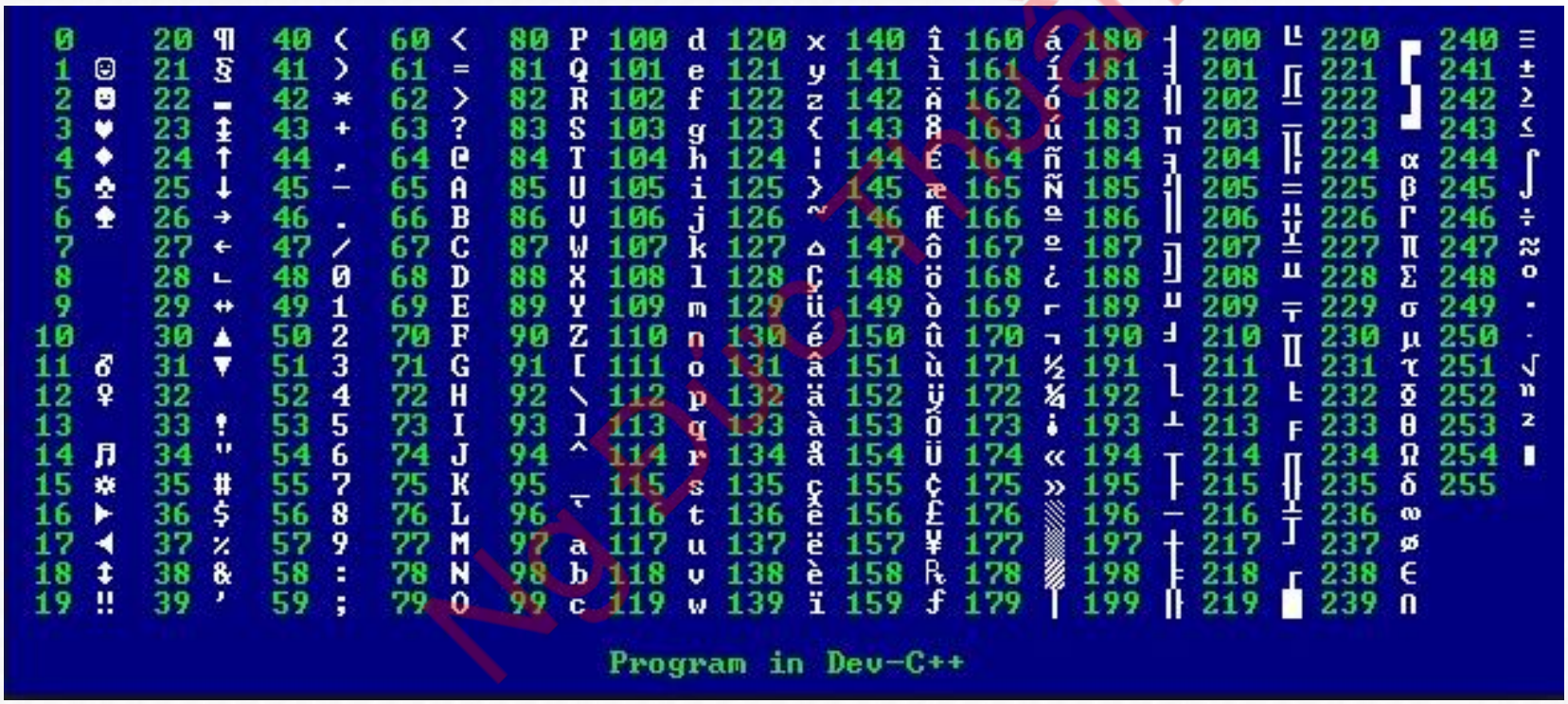
- Thuật toán cho nửa trên:
  - Dòng 1: in (5-row) k.trắng, (2*row - 1) sao; *
  - 2: (5-row) , (2*row - 1) ; * * *
  - 3: (5-row) , (2*row - 1) ; * * * * *
  - 4: (5-row) , (2*row - 1) ; * * * * * * *
  - 5: (5-row) , (2*row - 1) ; * * * * * * * * *

- Thuật toán cho nửa dưới:
  - Dòng 4: in (5-row) k.trắng, (2*row - 1) sao; * * * * * * *
  - 3: (5-row) , (2*row - 1) ; * * * * *
  - 2: (5-row) , (2*row - 1) ; * * *
  - 1: (5-row) , (2*row - 1) ; *

Diamond Pattern

```
int row, space, star;
for(row=1; row<=5; row++){                    //top half
      for(space=1; space<=5-row; space++)
            cout << " ";
            for(star=1; star<=2*row-1; star++)
            cout << "*";
      cout << endl ;
}
for(row=4; row>=1; row--){                    //bottom half
      for(space=1; space<=5-row; space++)
            cout << " ";
      for(star=1; star<=2*row-1; star++)
            cout << "*";
      cout << endl ;       }
```

HÀM NÂNG CAO



HÀM NÂNG CAO

Tham trị/Tham chiếu

(pass by value/pass by reference)

Phân biệt tham trị/tham chiếu

❖ Giống nhau:

- Đều là tham số hình thức.

❖ Khác nhau:

- Hình thức:

Tham trị không đi sau ký hiệu &: int f(int x){}

Tham chiếu đi sau ký hiệu &: int f(int &x){}

- Bản chất:

- Khi truyền giá trị cho tham trị, ngôn ngữ lập trình sẽ tạo ra vùng nhớ phụ để lưu trữ giá trị giá trị được truyền. (Còn tham biến thì không, chương trình con sẽ dùng chung vùng nhớ với giá trị được truyền)

Phân biệt tham trị/tham chiếu

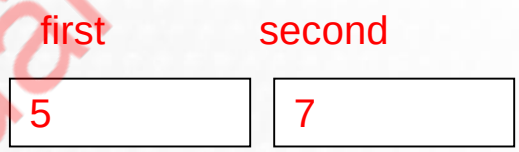
❖ Ví dụ:

```
#include <stdio.h>
 void Swap(int &a, int &b){
 int tmp = a;
   a = b;
   b = tmp; }
   int main() {
          int first =5;
          int second =7;
             printf("\nfirst = %d, second = %d", first,
second);
            Swap(first, second);
            printf("\nfirst = %d, second = %d", first, second);
 }
```

Phân biệt tham trị/tham chiếu

❖ Ví dụ:

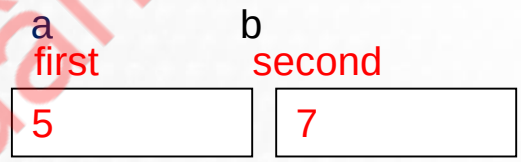
```
#include <stdio.h>
 void Swap(int &a, int &b){
 int tmp = a;
   a = b;
   b = tmp; }
   int main() {
          int first =5;
          int second =7;


}
```



Phân biệt tham trị/tham chiếu

❖ Ví dụ:

```
#include <stdio.h>
 void Swap(int &a, int &b){
 int tmp = a;
   a = b;
   b = tmp; }
   int main() {
          int first =5;
          int second =7;
             printf("\nfirst = %d, second = %d", first,
second);
           Swap(first, second);
}
```

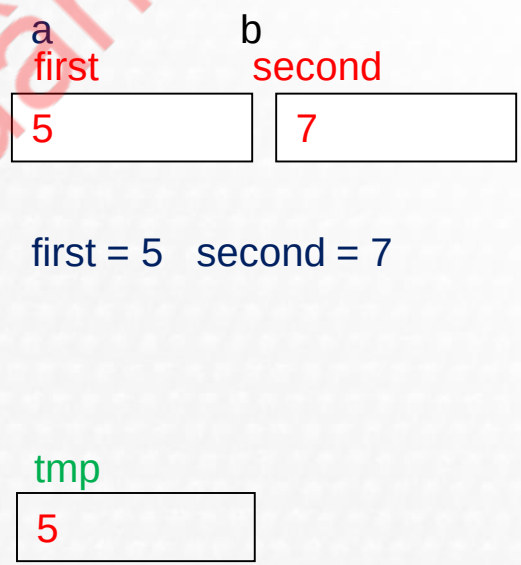


first = 5 second = 7

Phân biệt tham trị/tham chiếu

❖ Ví dụ:

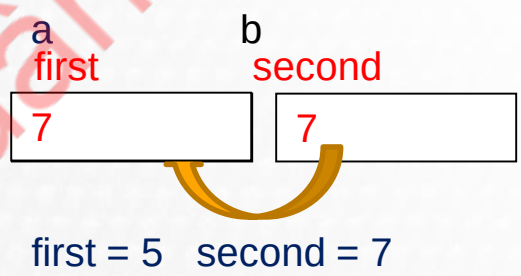
```
#include <stdio.h>
 void Swap(int &a, int &b){
 int tmp = a;

}
   int main() {
          int first =5;
          int second =7;
              printf("\nfirst = %d, second = %d", first,
second);
             Swap(first, second);

 }
```



Phân biệt tham trị/tham chiếu

❖ Ví dụ:

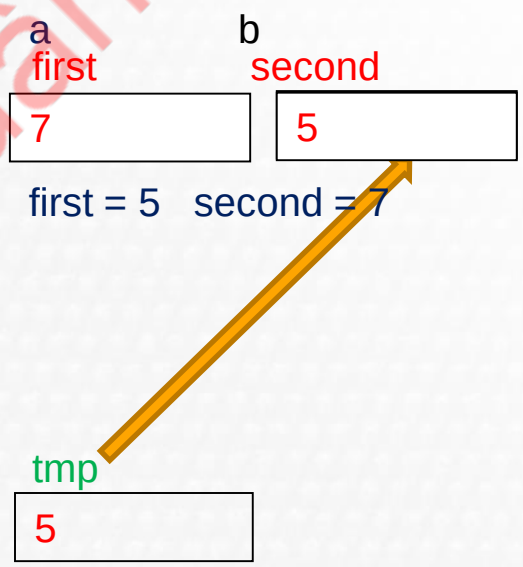
```
#include <stdio.h>
 void Swap(int &a, int &b){
int tmp = a;
a = b;

}
   int main() {
          int first =5;
          int second =7;
              printf("\nfirst = %d, second = %d", first,
second);
             Swap(first, second);

 }
```



Phân biệt tham trị/tham chiếu

❖ Ví dụ:

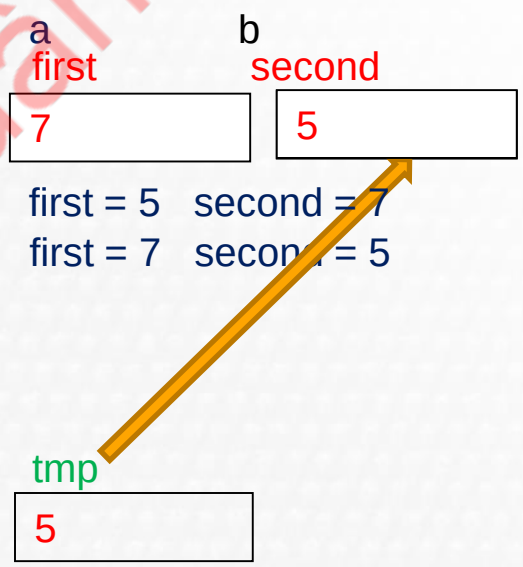
```
#include <stdio.h>
 void Swap(int &a, int &b){
int tmp = a;
a = b;
b = tmp;
}
   int main() {
          int first =5;
          int second =7;
             printf("\nfirst = %d, second = %d", first,
second);
            Swap(first, second);
             printf("\nfirst = %d, second = %d", first, second);
 }
```



Phân biệt tham trị/tham chiếu

❖ Ví dụ:

```
#include <stdio.h>
 void Swap(int &a, int &b){
int tmp = a;
a = b;
b = tmp;
}
   int main() {
          int first =5;
          int second =7;
              printf("\nfirst = %d, second = %d", first,
second);
             Swap(first, second);
              printf("\nfirst = %d, second = %d", first, second);
 }
```



Phân biệt tham trị/tham chiếu

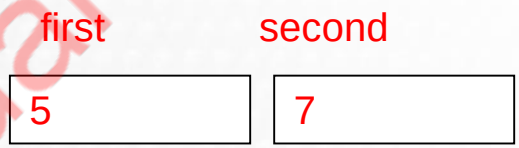
❖ Ví dụ:

```
#include <stdio.h>
 void Swap(int &a, int b){
 int tmp = a;
   a = b;
   b = tmp; }
   int main() {
          int first =5;
          int second =7;
             printf("\nfirst = %d, second = %d", first,
second);
            Swap(first, second);
            printf("\nfirst = %d, second = %d", first, second);
 }
```

Phân biệt tham trị/tham chiếu

❖ Ví dụ:

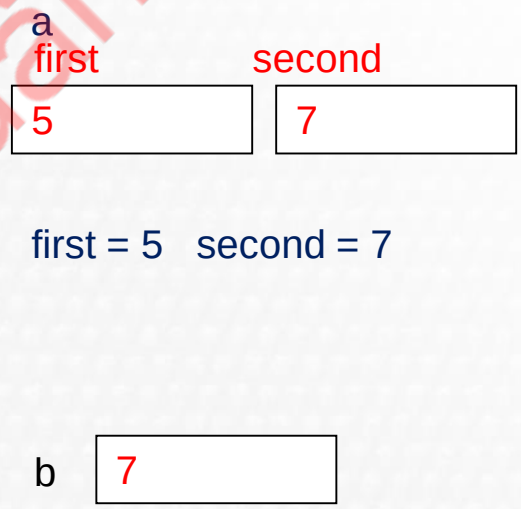
```
#include <stdio.h>
 void Swap(int &a, int b){
 int tmp = a;
   a = b;
   b = tmp; }
   int main() {
          int first =5;
          int second =7;


}
```



Phân biệt tham trị/tham chiếu

❖ Ví dụ:

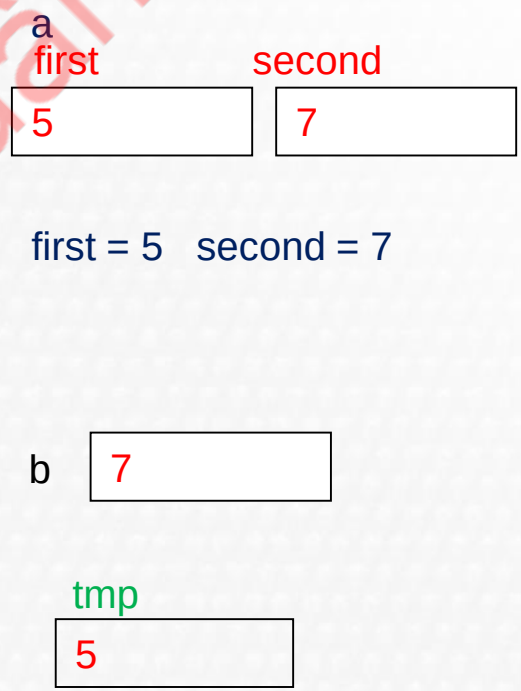
```
#include <stdio.h>
 void Swap(int &a, int b){
 int tmp = a;
   a = b;
   b = tmp; }
   int main() {
          int first =5;
          int second =7;
             printf("\nfirst = %d, second = %d", first,
second);
           Swap(first, second);
}
```



Phân biệt tham trị/tham chiếu

❖ Ví dụ:

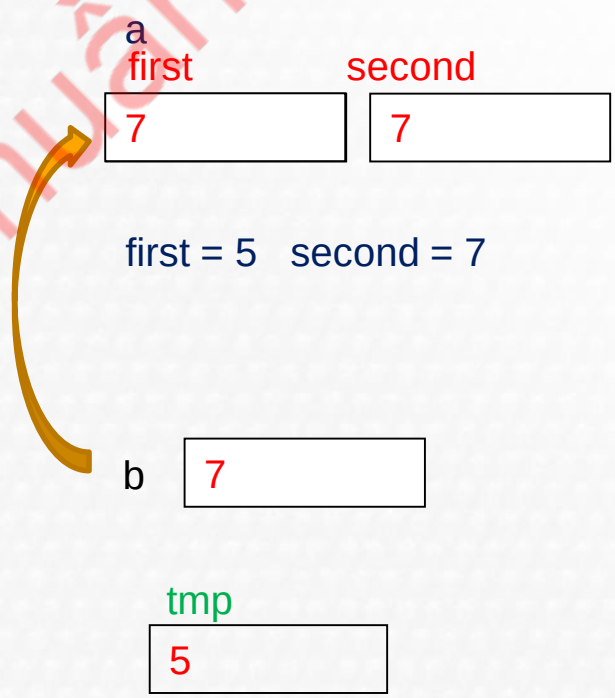
```
#include <stdio.h>
 void Swap(int &a, int b){
 int tmp = a;

}
   int main() {
          int first =5;
          int second =7;
              printf("\nfirst = %d, second = %d", first,
second);
             Swap(first, second);

 }
```



Phân biệt tham trị/tham chiếu

❖ Ví dụ:

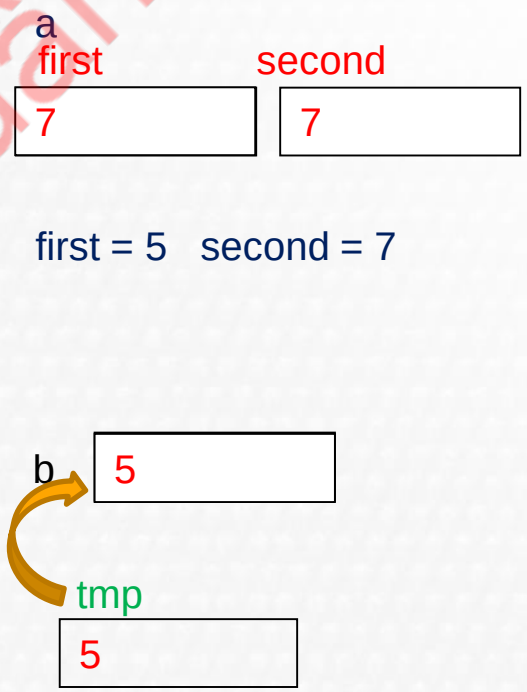
```
#include <stdio.h>
 void Swap(int &a, int b){
 int tmp = a;
 a = b;

}
   int main() {
          int first =5;
          int second =7;
             printf("\nfirst = %d, second = %d", first,
second);
            Swap(first, second);

 }
```



Phân biệt tham trị/tham chiếu

❖ Ví dụ:

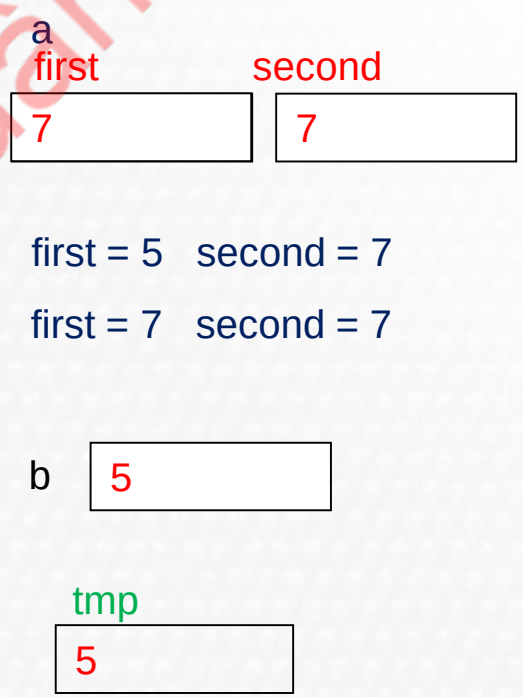
```
#include <stdio.h>
 void Swap(int &a, int b){
 int tmp = a;
 a = b;
b = tmp;
}
   int main() {
          int first =5;
          int second =7;
             printf("\nfirst = %d, second = %d", first,
second);
            Swap(first, second);

 }
```



Phân biệt tham trị/tham chiếu

❖ Ví dụ:

```
#include <stdio.h>
 void Swap(int &a, int b){
 int tmp = a;
 a = b;
b = tmp;
}
   int main() {
          int first =5;
          int second =7;
             printf("\nfirst = %d, second = %d", first,
second);
            Swap(first, second);
             printf("\nfirst = %d, second = %d", first,
second);
 }
```



Hệ quả

- Giá trị truyền cho tham chiếu là một biến; giá trị truyền cho tham trị có thể là 1 biểu thức
- Những thay đổi trong chương trình con có liên quan đến tham biến giữ lại, những thay đổi trong chương trình con liên quan đến tham trị không ảnh hưởng đến đối tượng truyền.

Bài tập 1

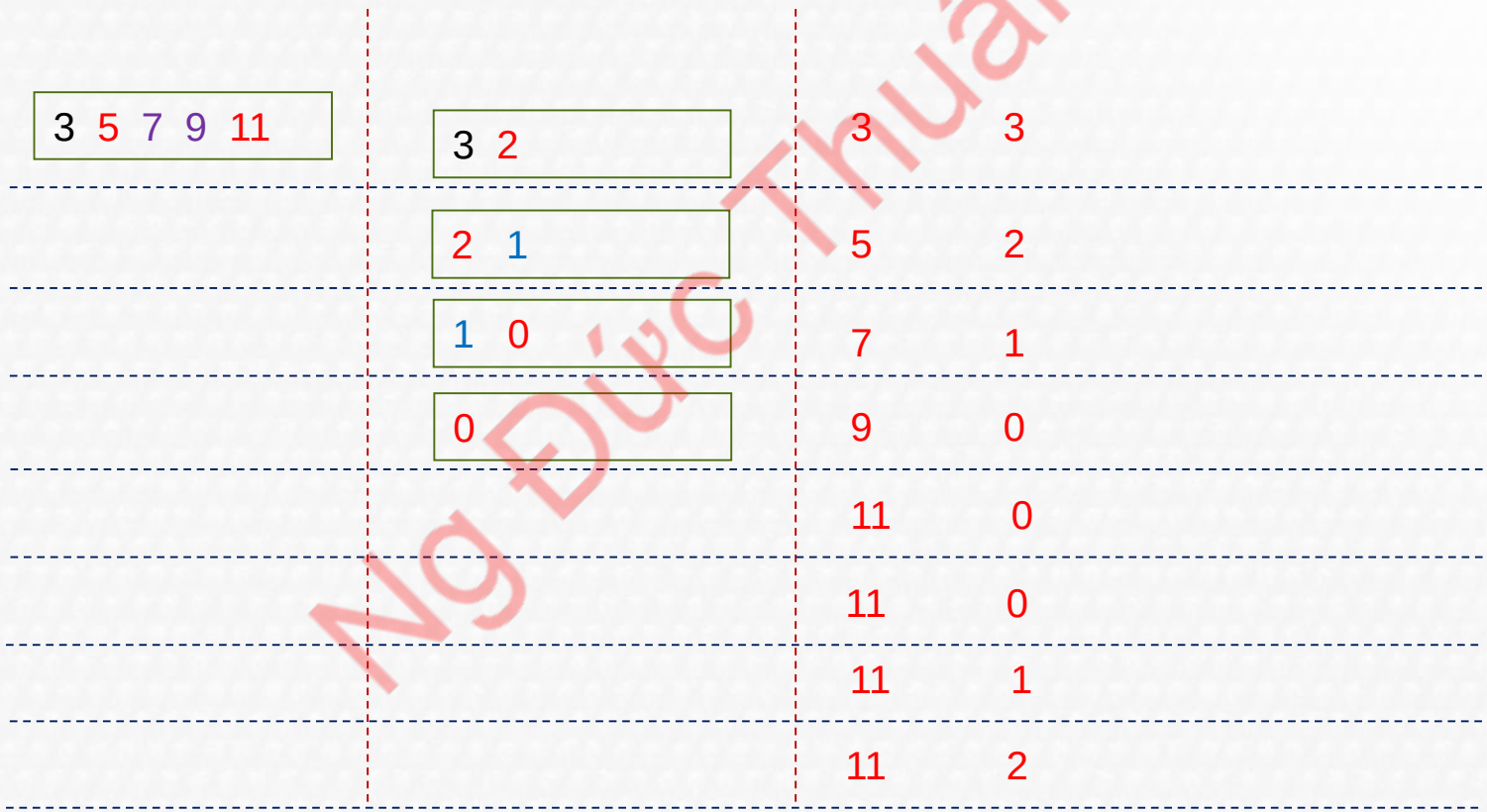
```
#include <stdio.h>
#include <conio.h>
int x;
void them2bot1(int &x,int y)
{               printf("\n %4d %4d",x,y);
                        x=x+2;
                                if (y>0) {y=y-1; them2bot1(x,y);}
                 printf("\n %4d %4d",x,y);}
int main()
{               x=3;
                them2bot1(x,x);
                getch();
                return 0;}
```

Đáp án

- 3 3
- 5 2
- 7 1
- 9 0
- 11 0
- 11 0
- 11 1
- 11 2

Đáp án

- x y



Bài tập 2

- *Cho biết kết quả hiển thị lên màn hình khi thực hiện đoạn chương trình:*

```
#include <stdio.h>
   int a;
int  f(int &x)
    { if (x%2==0) x=x/2;
      else x=3*x+1;
                if (x!=1) return 1;
                  else return 0;}
int main()
                { a=10;
                  while (f(a)==1)
                    printf("%3d",a);
return 0;  }
```

Đáp án

5 16 8 4 2

Tham trị/Tham chiếu

THAM SỐ CỦA HÀM
TRUYỀN ĐỐI SỐ

Hàm main()

❖ Tham số của hàm main()

- Các đối số của chương trình
  - Hàm main là hàm nên cũng có tham số.
  - Chương trình tự động thực hiện hàm main mà không cần lời gọi hàm.

  ➔ Làm sao truyền đối số?

  ➔ Khi thực thi tập tin chương trình (.exe), ta truyền kèm đối số. Tất nhiên, hàm main cũng phải định nghĩa các tham số để có thể nhận các đối số này.

Các tham số của hàm main

- Các tham số của hàm main

```
int main(int argc, char *argv[])
{
        ...
}
```

  - Trong đó
    - argc là số lượng đối số (tính luôn tên tập tin chương trình)
    - argv là mảng chứa các đối số (dạng chuỗi)

Các tham số của hàm main

- Ví dụ
  - Viết chương trình có tên Cong, nhận 2 đối số x và y và xuất ra giá trị x + y.

```
argv = {"Cong.EXE", "2912", "1706"};
```

```
argc = 3
```

Các tham số của hàm main

- Ví dụ
  - Viết chương trình có tên Cong, nhận 2 đối số x và y và xuất ra giá trị x + y.

```
#include <stdio.h>
#include <stdlib.h>      // atoi
void main(int argc, char *argv[]) {
        if (argc == 3) {
                int x = atoi(argv[1]);
                int y = atoi(argv[2]);
                printf("%d + %d = %d", x, y, x+y);
        }
        else
                printf("Sai! VD: Cong 2912 1706");
}
```

Hàm **int atoi(const char *str)** trong Thư viện C chuẩn chuyển đổi một chuỗi được trỏ tới bởi tham số **str** thành một số nguyên (kiểu int)

Các tham số của hàm main

- Ví dụ
  - Viết chương trình có tên test nhận dữ liệu từ tập tin input.txt, xử lý và xuất kết quả ra tập tin output.txt.

```
argv = {"test", "input.txt", "output.txt"};
```

```
argc = 3
```

Các tham số của hàm main

- Ví dụ
  - Viết chương trình có tên test nhận dữ liệu từ tập tin input.txt, xử lý và xuất kết quả ra tập tin output.txt.

```
#include <stdio.h>
void main(int argc, char *argv[]) {
        if (argc == 3) {
                // Nhập dữ liệu từ tập tin argv[1]
                // Xử lý
                // Xuất kết quả ra tập tin argv[2]
        }
        else
                printf("Sai! VD: test in.txt out.txt");
}
```

HÀM CÓ ĐỐI SỐ MẶC ĐỊNH

Hàm có đối số mặc định

- Ví dụ
  - Viết hàm Tong để tính tổng 4 số x, y, z, t

```
int Tong(int x, int y, int z, int t)
{
        return x + y + z + t;
}
```

  - Tính tổng 4 số 2912, 1706, 1506, 1904

```
Tong(2912, 1706, 1506, 1904);
```

  - Nếu chỉ muốn tính tổng 2 số 2912, 1706

```
Tong(2912, 1706, 0, 0);     // z = 0, t = 0
```

MACRO & TIỀN XỬ LÝ

❖ MACRO

❑ Macro là gì?

- Marco khối lệnh thực hiện một chức năng nào đó được gán1 tên (do lập trình viên đặt tên).
- Trong quá trình tiền xử lí (pre-processor), các macro được sử dụng trong chương trình được thay thế bởi các khối câu lệnh tương ứng.
- Định nghĩa macro bằng lệnh #define

```
#include <stdio.h>
#include <conio.h>

#define MAX(A, B) ((A) > (B) ? (A) : (B))

void main( void )
{
    int a = 5, b = 7;
    float c = 5.6, d = 4.5;
    printf("\nMAX(%d, %d) = %d", a, b, MAX(a, b));
    printf("\nMAX(%f, %f) = %f", c, d, MAX(c, d));
    getch();
}
```

1. Nếu sau # là define <*tên hằng*> <*giá trị*>

Khi biên dịch tất cả các *tên hằng* được thay bằng *giá trị*

```
#include<stdio.h>
#define max 100
int main()
{
    printf("max is %d", max);
    return 0;
}
// Output: max is 100
// Note that the max inside "" is not replaced
```

2. Macro có tác dụng như một hàm nhưng là một tác động lên tham số không quan tâm đến kiểu dữ liệu

```c
#include <stdio.h>
#define INCREMENT(x) ++x
int main()
{
    char *ptr = "GeeksQuiz";
    int x = 10;
    printf("%s  ", INCREMENT(ptr));
    printf("%d", INCREMENT(x));
    return 0;
}
// Output: eeksQuiz 11
```

- 3. Các tham số của Macro không được lượng giá trước khi thực hiện

```c
#include <stdio.h>
#define MULTIPLY(a, b) a*b
int main()
{
    // The macro is expended as 2 + 3 * 3 + 5, not as 5*8
    printf("%d", MULTIPLY(2+3, 3+5));
    return 0;
}
// Output: 16
```

4. Các token được chuyển đến macro có thể được nối bằng toán tử ## gọi là toán tử Token-Pasting.

```
#include <stdio.h>
#define merge(a, b) a##b
int main()
{
    printf("%d ", merge(12, 34));
}
// Output: 1234
```

5. Một mã thông báo được chuyển đến macro có thể được chuyển đổi thành một chuỗi ký tự bằng cách sử dụng # trước mã thông báo

```
#include <stdio.h>
#define get(a) #a
int main()
{
    // GeeksQuiz is changed to "GeeksQuiz"
    printf("%s", get(GeeksQuiz));
}
// Output: GeeksQuiz
```

6. Macro có thể được viết bằng nhiều dòng bằng cách sử dụng '\'. Dòng cuối cùng không cần phải có '\'.

```
#include <stdio.h>
#define PRINT(i, limit) while (i < limit) \
                        { \
                            printf("GeeksQuiz "); \
                            i++; \
                        }
int main()
{
    int i = 0;
    PRINT(i, 3);
    return 0;
}
// Output: GeeksQuiz  GeeksQuiz  GeeksQuiz
```

7. Macro nên tránh nhiều đối số vì đôi khi chúng gây lỗi. Trong trường hợp nhiều tham số nên được ưu tiên dùng chức năng Inline vì có kiểm tra đánh giá kiểu tham số

Ví dụ *Thường dễ dự báo kết quả là 1, nhưng kết quả 36*

```
#define square(x) x*x
int main()
{
  int x = 36/square(6); // Expended as 36/6*6
  printf("%d", x);
  return 0;
}
// Output: 36
```

Nếu chúng ta sử dụng các hàm inline, chúng ta sẽ có được kết quả mong đợi. Ngoài ra chương trình nêu ở điểm 4 ở trên có thể được sửa bằng các hàm nội tuyến.

```
inline int square(int x) { return x*x; }
int main()
{
  int x = 36/square(6);
  printf("%d", x);
  return 0;
}
// Output: 1
```

8. Các tiền xử lý cũng hỗ trợ các lệnh if-else

```
#include <stdio.h>
#include <conio.h>

#define vda 7;

int main()
{
  #ifdef vda >=2
  printf("Kho qua ta!");
  #endif
}
```

- ifdef có nghĩa là "nếu điều sau đây được định nghĩa"
- ifndef có nghĩa là "nếu điều sau đây không được định nghĩa".

```
# define one
# ifdef one printf ("một được định nghĩa");
#endif  #ifndef one printf ("một không được định nghĩa");
#endif.
```

Kết quả đoạn trên tương đương với: printf ("một được định nghĩa");

9. Có một số macro tiêu chuẩn có thể được sử dụng để in tệp chương trình (__FILE__), Ngày biên soạn (__DATE__), Thời gian biên soạn (__Time__) và Số Dòng trong mã C (__LINE__)

```c
#include <stdio.h>

int main()
{
    printf("Current File :%s\n", __FILE__ );
    printf("Current Date :%s\n", __DATE__ );
    printf("Current Time :%s\n", __TIME__ );
    printf("Line Number :%d\n", __LINE__ );
    return 0;
}

/* Output:
Current File :C:\Users\GfG\Downloads\deleteBST.c
Current Date :Feb 15 2014
Current Time :07:04:25
Line Number :8 */
```

10. Xóa macro đã định nghĩa

#undef MACRO_NAME

```c
#include <stdio.h>
#define LIMIT 100
int main()
{
   printf("%d",LIMIT);
   //removing defined macro LIMIT
   #undef LIMIT
   //Next line causes error as LIMIT is not defined
   printf("%d",LIMIT);
   return 0;
}
```

HÀM INLINE

Hàm nội tuyến: inline <hàm ….>

Hàm inline

- Khai báo hàm nội tuyến inline:

  Viết thêm từ khoá inline vào trước khai báo nguyên mẫu hàm

Dạng 1:

```
Inline float f(int n, float x);
        float f(int n, float x)
            { // Các câu lệnh trong thân hàm }
```

Dạng 2:

```
Inline float f(int n, float x)
            { // Các câu lệnh trong thân hàm }
```

Hàm inline

- **Cách biên dịch hàm trực tuyến**

- Chương trình dịch xử lý các hàm inline như các macro (được định nghĩa trong lệnh #define).

- Tốc độ chương trình tăng lên do không phải thực hiện các thao tác có tính thủ tục khi gọi hàm nhưng không gian lưu trữ chương trình dài hơn.

- **So sánh macro và hàm trực tuyến**

- Nguyên lý làm việc như nhau.

- Hàm nội tuyến hơn khai báo chặt chẽ tránh được các sai sót thường gặp khi dùng #define (như thiếu các dấu ngoặc, dấu chấm phẩy)

- **Khi nào thì nên dùng hàm trực tuyến**

- Dùng phương án hàm nội tuyến đối với các hàm nhỏ.

Hàm inline

- Chú ý
- Không phải khi gặp từ khoá inline là Trình biên dịch nhất thiết phải xử lý hàm theo kiểu nội tuyến.
- Từ khoá inline chỉ là một sự gợi ý cho Trình biên dịch chứ không phải là một mệnh lệnh bắt buộc.Có một số hàm mà các Trình biên dịch thường không xử lý theo cách inline như các hàm chứa biến static, hàm chứa các lệnh chu trình hoặc lệnh goto hoặc lệnh switch, hàm đệ quy. Trong trường hợp này từ khoá inline lẽ dĩ nhiên bị bỏ qua.
- Thậm chí từ khoá inline vẫn bị bỏ qua ngay cả đối với các hàm không có những hạn chế nêu trên nếu như Trình biên dịch thấy cần thiết (ví dụ đã có quá nhiều hàm inline làm cho bộ nhớ chương trình quá lớn)

Cám ơn đã theo dõi