Giới thiệu về các cấu trúc điều khiển

Nhập môn lập trình

Trình bày: …; Email: …@fit.hcmus.edu.vn

Nội dung

- Khối lệnh trong lập trình
- Dùng cấu trúc rẽ nhánh trong lập trình
- Xử lý lặp trong lập trình
- Các vấn đề tìm hiểu mở rộng kiến thức nghề nghiệp
- Thuật ngữ và bài đọc thêm tiếng Anh

Khối lệnh trong lập trình

Định nghĩa & ví dụ

- Một dãy các câu lệnh được bao bởi dấu {} gọi là một khối lệnh.
- Ví dụ:

```
{
    a = 2;
    b = 3;
    printf("\n%d%d", a, b);
}
```

Khái niệm về namespace

- Một namespace là giới hạn phạm vi ý nghĩa của một cái tên, nghĩa là tên chỉ có ý nghĩa trong phạm vi được định nghĩa bởi namespace.
- Namespace giúp tránh đụng độ tên biến, tên hàm…

Ví dụ về namespace

```
// namespaces
#include <iostream>
using namespace std;
namespace first {
	int var = 5;
}
namespace second {
	double var = 3.1416;
}
void main () {
	cout << first::var << endl;
	cout << second::var << endl;
}
```

Phạm vi sử dụng của biến

- Khi lập trình, cần phải nắm rõ phạm vi của biến. Nếu khai báo và sử dụng không đúng, không rõ ràng sẽ dẫn đến sai sót khó kiểm soát được, vì vậy bạn cần phải xác định đúng vị trí, phạm vi sử dụng biến trước khi sử dụng biến.
- Có 2 loại biến:
  - Biến toàn cục (Global variable)
  - Biến cục bộ (Local variable)

Biến toàn cục & nguyên tắc sử dụng

- **Biến toàn cục (Global variables):** vị trí biến đặt bên ngoài tất cả các hàm, cấu trúc...Các biến này có ảnh hưởng đến toàn bộ chương trình. Chu trình sống của nó là bắt đầu chạychương trình đến lúc kết thúc chương trình.
- **Nguyên tắc sử dụng:** có thể được sử dụng ở bất kỳ đâu trong chương trình, ngay sau khi nó được khai báo.

Biến cục bộ & nguyên tắc sử dụng

- **Biến cục bộ (Local variables):** Vị trí biến đặt bên trong hàm, cấu trúc…. Chỉ ảnh hưởng nội bộ bên trong hàm, cấu trúc đó…. Chu trình sống của nó bắt đầu từ lúc hàm, cấu trúc được gọi thực hiện đến lúc thực hiện xong.
- **Nguyên tắc sử dụng:** bị giới hạn trong phần mã mà nó được khai báo. Nếu chúng được khai báo ở đầu một hàm (như hàm main), tầm hoạt động sẽ là toàn bộ hàm main. Điều đó có nghĩa là các biến được khai báo trong hàm main() chỉ có thể được dùng trong hàm đó, không được dùng ở bất kỳ đâu khác.

Dùng cấu trúc rẽ nhánh trong lập trình

Định nghĩa cấu trúc điều khiển

- Các cấu trúc điều khiển cho phép chúng ta thay đổi thứ tự thực hiện các câu lệnh. Việc sử dụng các cấu trúc điều khiển trong chương trình giúp chúng ta thực hiện các câu lệnh trong chương trình theo ý của mình chứ không cứng nhắc là từ trên xuống dưới.

Phân loại cấu trúc điều khiển

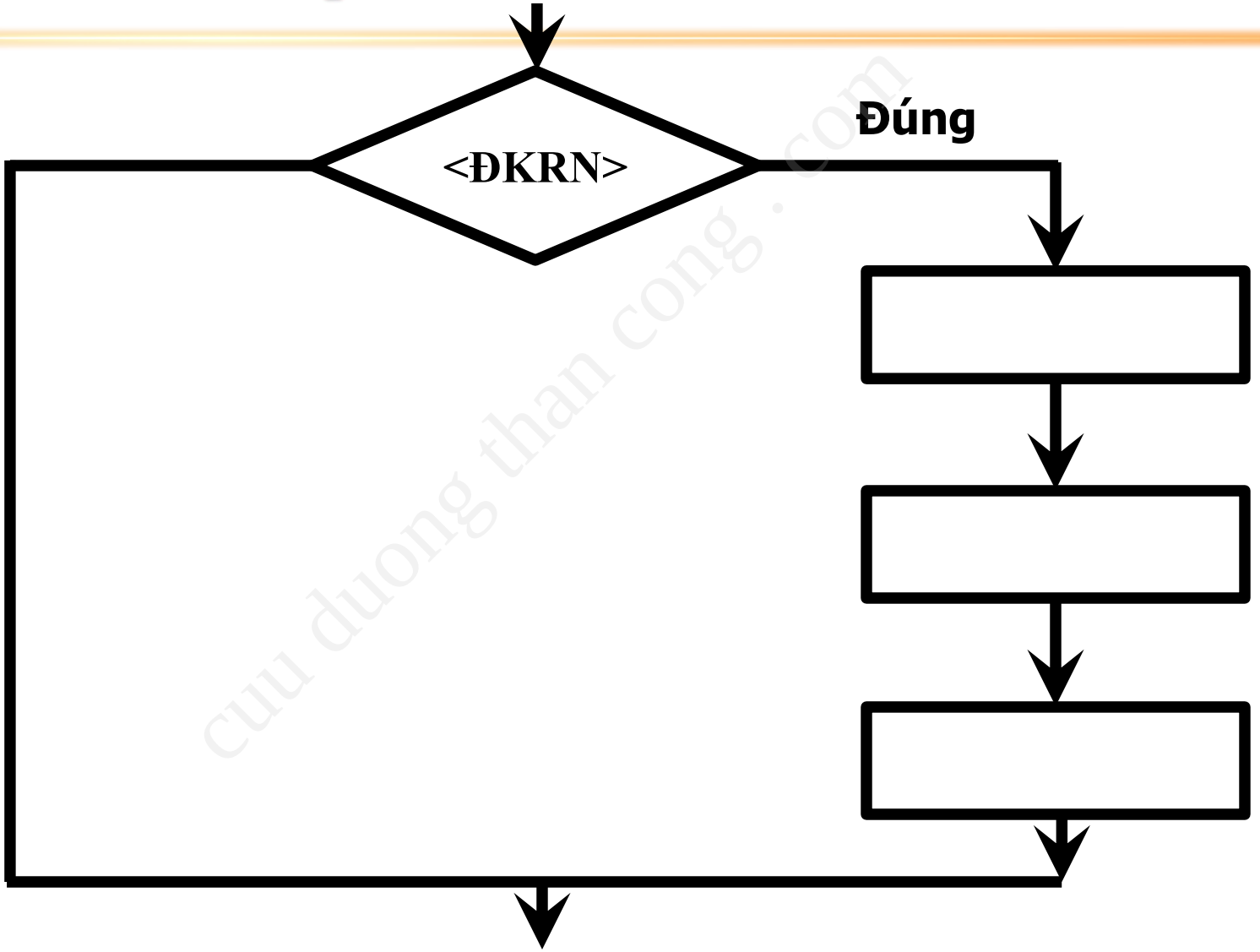
- **Cấu trúc điều khiển có 2 loại:**
  - Cấu trúc điều khiển rẽ nhánh:
    - if else
    - switch
  - Cấu trúc điều khiển vòng lặp:
    - for
    - while
    - do while

Cấu trúc điều khiển rẽ nhánh if

- Cấu trúc rẽ nhánh if cho phép lựa chọn thực hiện một lệnh hay khối lệnh đi sau cấu trúc điều khiển if hay không, việc lựa chọn này tùy thuộc vào giá trị trả về của biểu thức điều kiện.

```
if (biểu_thức_điều_kiện)
{
    Lệnh 1;
    Lệnh 2;
    ...
    Lệnh n;
}
```

Lưu đồ thuật toán của cấu trúc if



Ví dụ cấu trúc if

- Tìm số lớn nhất trong 3 số thực a, b, c

```
#include <stdio.h>
void main() {
	float a, b, c, max;
	printf("Nhap 3 so thuc: ");
	scanf("%f%f%f", &a, &b, &c);
	max = a;
	if (b > max)
		max = b;
	if (c > max)
		max = c;
	printf("So lon nhat la: %.2f\n", max);
}
```

Cấu trúc điều khiển rẽ nhánh if else

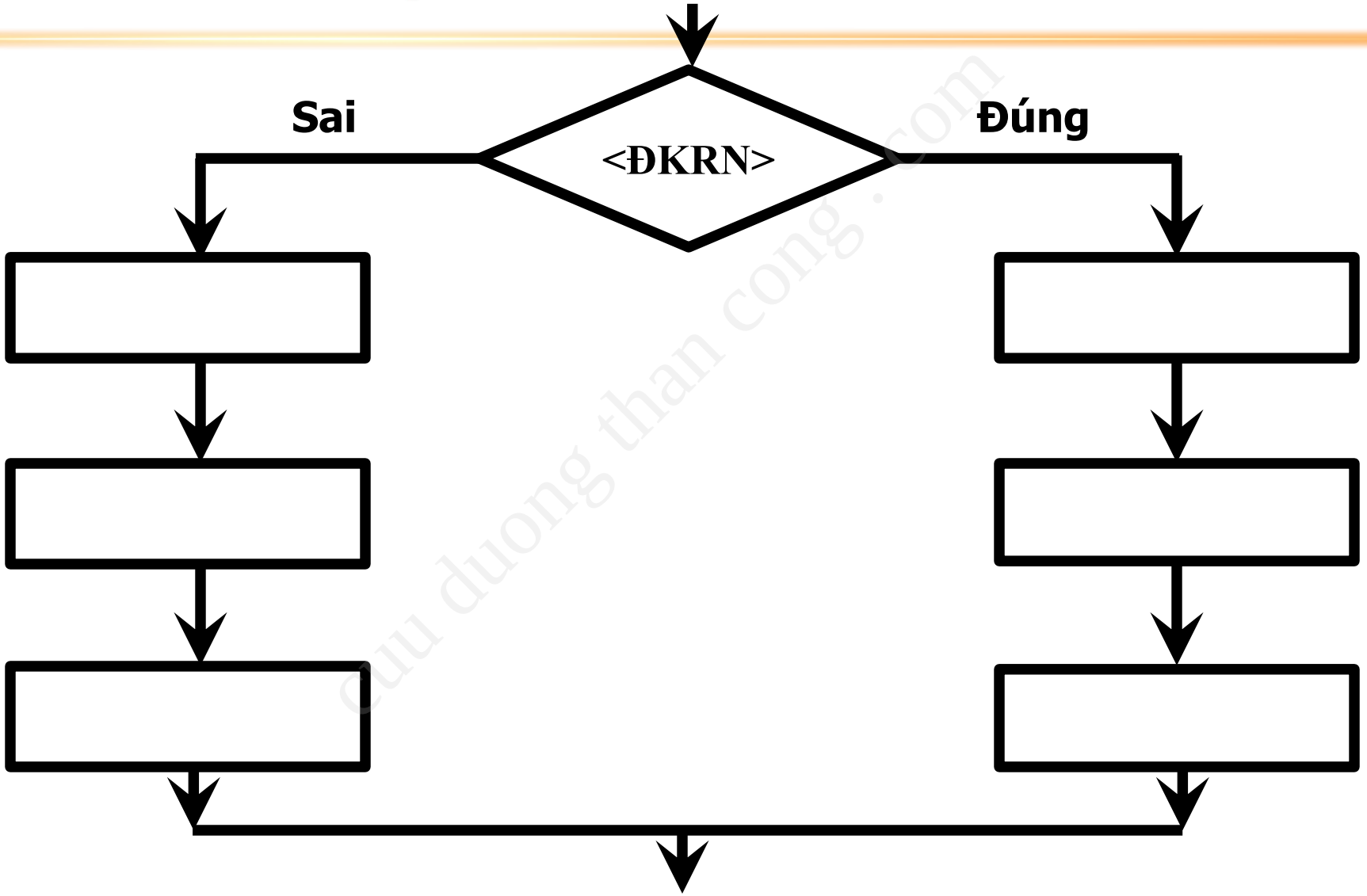
- Cấu trúc điều khiển rẽ nhánh if else cho phép lựa chọn một trong hai nhánh lệnh của chương trình và việc lựa chọn này tùy thuộc giá trị trả về của biểu thức điều kiện.

Cấu trúc điều khiển rẽ nhánh if

- Cấu trúc điều khiển rẽ nhánh if else cho phép lựa chọn một trong hai nhánh lệnh của chương trình và việc lựa chọn này tùy thuộc giá trị trả về của biểu thức điều kiện.

```
if (biểu_thức_điều_kiện)
{
    Lệnh 1;
    Lệnh 2;
    …
    Lệnh n;
}
else
{
    Lệnh 1;
    Lệnh 2;
    …
    Lệnh n;
}
```

Lưu đồ thuật toán của cấu trúc if else



Ví dụ cấu trúc if else

- Kiểm tra 2 số thực cho trước có cùng dấu hay không?

```
#include <stdio.h>
void main() {
	float a, b;
	printf("Nhap 2 so thuc: ");
	scanf("%f%f", &a, &b);
	if (a * b > 0)
		printf("%.2f va %.2f cung dau!\n", a, b);
	else
		printf("%.2f va %.2f trai dau!\n", a, b);
}
```

Ví dụ cấu trúc if else

- Giải phương trình bậc nhất ax + b = 0

```
#include <stdio.h>
void main() {
	float a, b;
	printf("Nhap 2 so thuc: ");
	scanf("%f%f", &a, &b);
	if (a == 0)
		if (b == 0)
			printf("Phuong trinh vo so nghiem!\n");
		else
			printf("Phuong trinh vo nghiem!\n");
	else
		 printf("Phuong trinh co nghiem x = %.2f\n", -b / a);
}
```

Ghi chú quan trọng

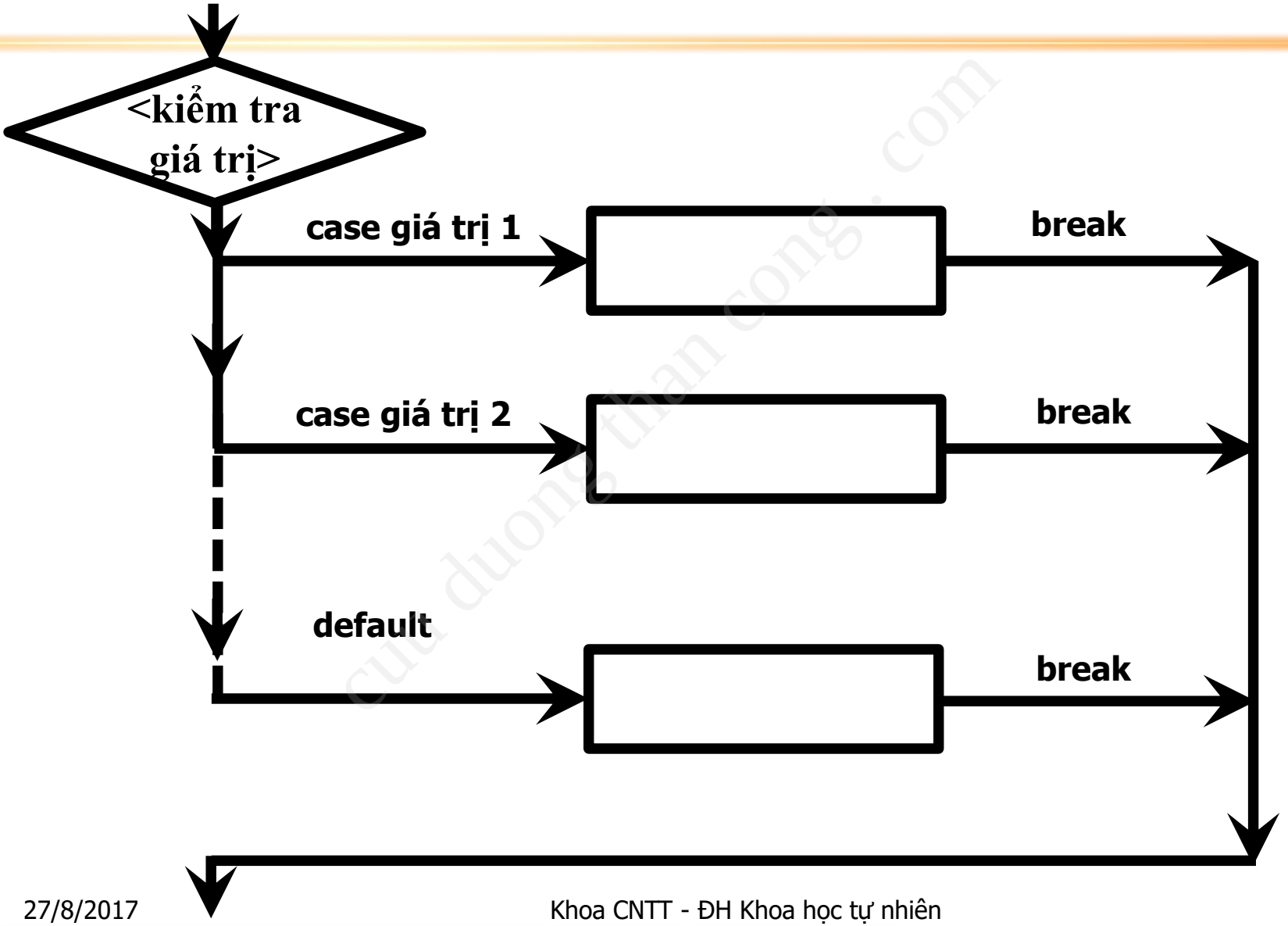
- Nếu sau if hoặc else chỉ có một khối lệnh thì không cần phải để lệnh ấy trong khối lệnh "{}". Ngoài ra NNLT C cũng cho phép chúng ta sử dụng cấu trúc chọn if, if else lồng nhau, nhưng phải xác định khối lệnh một cách rõ ràng.

Cấu trúc điều khiển rẽ nhánh switch

- Cấu trúc điều khiển rẽ nhánh switch cho phép căn cứ vào giá trị của biểu thức nguyên để cho một trong nhiều cách nhảy.

```
switch (biểu_thức_chọn)
{
        case Giá_Trị_1:
                Lệnh 1;
                ..
                Lệnh n;
                break;
        case Giá_Trị_2:
                Lệnh 1;
                ..
                Lệnh n;
                break;
        default:
                Lệnh 1;
                ..
                Lệnh n;
                break;
}
```

Lưu đồ thuật toán của cấu trúc switch case



Ví dụ cấu trúc switch

- Cho biết tháng cho trước thuộc quý mấy?

```
#include <stdio.h>
void main() {
	int thang;
	printf("Nhap thang: ");
	scanf("%d", &thang);
	switch (thang)
	{
		case 1: case 2: case 3: printf("Quy I\n"); break;
		case 4: case 5: case 6: printf("Quy II\n"); break;
		case 7: case 8: case 9: printf("Quy III\n"); break;
		case 10: case 11: case 12: printf("Quy IV\n"); break;
	}
}
```

Ghi chú quan trọng

- Biểu thức chọn trong cấu trúc điều khiển rẻ nhánh switch sẽ được tính toán, ước lượng và so sánh với các giá trị trong tương ứng với các mệnh đề case.
- Nếu giá trị của biểu thức bằng Giá_Trị_i thì khối lệnh của mệnh đề case i được thực hiện.

Ghi chú quan trọng

- Nếu giá trị của biểu thức không bằng với bất kỳ Giá_Trị_i nào trong các mệnh đề case thì khối lệnh tương ứng với khóa default được thực hiện.
- Mỗi khối lệnh của mỗi mệnh đề case thường được kết thúc bởi một câu lệnh break.

Ghi chú quan trọng

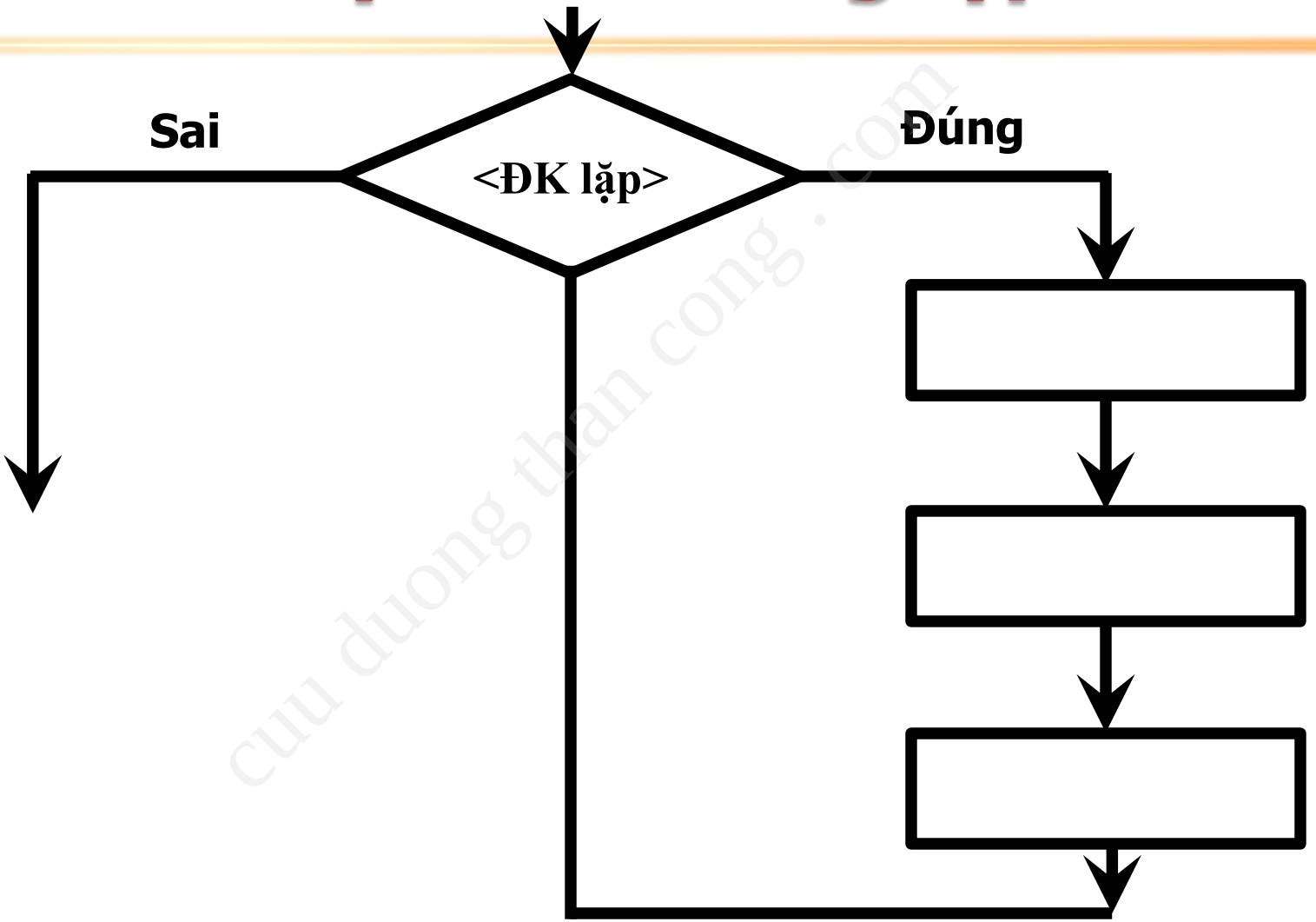
- Việc thực hiện khối lệnh sau khi so sánh giá trị của biểu thức bằng Giá_Trị_i như sau: thực hiện tất cả những lệnh ngay sau mệnh đề case của Giá_trị_i trên cho đến khi gặp từ khóa break.

Xử lý lặp trong lập trình

Cấu trúc điều khiển lặp while

```
while (điều_kiện_lặp)
{
    Lệnh 1;
    Lệnh 2;
    …
    Lệnh n;
}
```

Lưu đồ thuật toán vòng lặp while



Ví dụ 1 của vòng lặp while

- Tính S = $1^3 + 2^3 + \ldots + n^3$

```
#include <stdio.h>
void main() {
	int n, i, s;
	printf("Nhap n: ");
	scanf("%d", &n);
	i = 1;
	s = 0;
	while (i <= n) {
		s = s + i*i*i;
	}
	printf("Tong s la %d\n", s);
}
```

Ví dụ 2 của vòng lặp while

- Tìm số nguyên dương n nhỏ nhất sao cho 1 + 2 + … + n > 10000.

```
#include <stdio.h>
void main() {
	int s = 0, n = 0;
	while (s <= 10000) {
		n++;
		s = s + n;
	}
	printf("So n la %d\n", n);
}
```

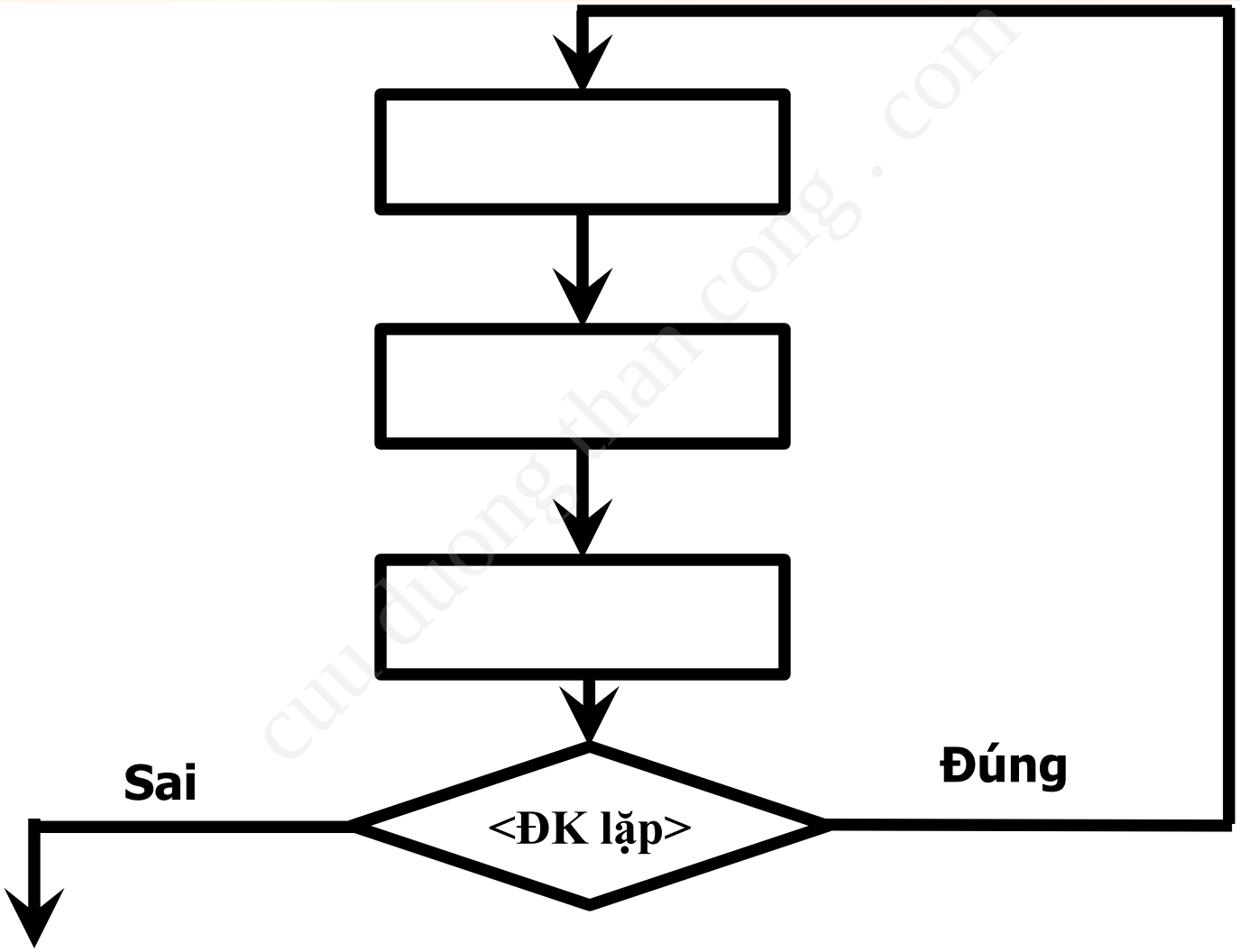
Nhận xét

- Các lệnh trong khối lệnh của vòng lặp while sẽ được thực hiện ít nhất một lần.
- Điều kiện lặp của vòng lặp while thường được cập nhật sau mỗi lần thực hiện khối lệnh hay có một biến cố nào thuận lợi xảy ra.

Cấu trúc điều khiển lặp do while

```
do
{
    Lệnh 1;
    Lệnh 2;
    ...
    Lệnh n;
}
while (điều_kiện_lặp);
```

Lưu đồ thuật toán vòng lặp do while



Ví dụ của vòng lặp do while

- Tìm số nguyên dương n lớn nhất sao cho 1 + 2 + … + n < 10000.

```
#include <stdio.h>
void main() {
	int n = 0, s = 0;
	do {
		n++;
		s = s + n;
	} while (s + n + 1 < 10000);
	printf("So n la %d\n", n);
}
```

Phân tích hoạt động của cấu trúc lặp do/while

- **Bước 1:** Thực hiện các câu lệnh trong khối lệnh lặp do while.
- **Bước 2:** Khi gặp đến cuối khối lệnh lặp do while, chương trình sẽ xác định giá trị của điều kiện lặp sau từ khóa while.
- **Bước 3:** Chương trình sẽ thực thi một trong 2 nhánh sau tùy theo giá trị của biểu thức vừa nhận được.

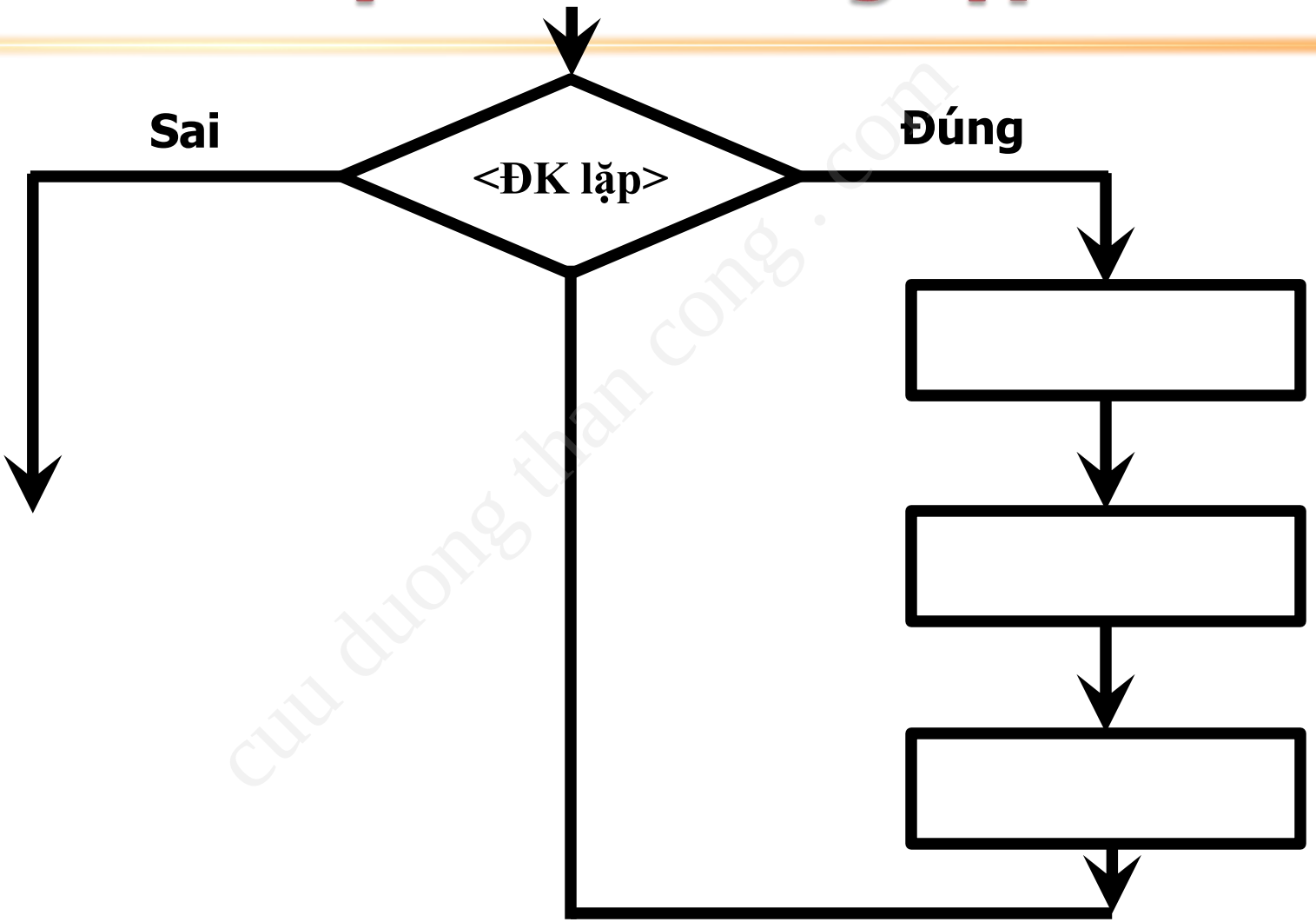
Phân tích hoạt động của cấu trúc lặp do/while

- **Bước 3.1:** Nếu biểu thức có giá trị đúng (khác 0), chương trình sẽ quay trở lại bước 1 để tiếp tục thực hiện vòng lặp mới.
- **Bước 3.2:** Nếu biểu thức có giá trị sau (bằng 0), chương trình sẽ ra khỏi chu trình và chuyển tới câu lệnh đúng sau dấu chấm phẩy đặt cuối từ khóa do while.

Cấu trúc điều khiển lặp for

```
for (biểu_thức_1;biểu_thức_2;biểu_thức_3)
{
    Lệnh 1;
    Lệnh 2;
    …
    Lệnh n;
}
```

Lưu đồ thuật toán vòng lặp for



Ví dụ 1 của vòng lặp for

- Xuất các ký tự từ 'A' đến 'Z'.

```
#include <stdio.h>
void main() {
	char kytu;
	for (kytu = 'A'; kytu <= 'Z'; kytu++) {
		printf("%c ", kytu);
	}
}
```

Ví dụ 2 của vòng lặp for

- Tính tổng các số dương lẻ của số nguyên dương n.

```
#include <stdio.h>
void main() {
	int n, i, s;
	printf("Nhap n: ");
	scanf("%d", &n);
	s = 0;
	for (i = 1; i < n; i = i + 2) {
		s = s + i;
	}
	printf("Tong cac so duong le nho hon %d la %d\n", n, s);
}
```

Ví dụ 2 của vòng lặp for (cách khác)

- Tính tổng các số dương lẻ của số nguyên dương n.

```
#include <stdio.h>
void main() {
    int n, i, s;
    printf("Nhap n: "); scanf("%d", &n);
    s = 0; i = 1;
    for (;;) {
        s = s + i;
        i = i + 2;
        if (i >= n) break;
    }
    printf("Tong cac so duong le nho hon %d la %d\n", n, s);
}
```

Ý nghĩa của các biểu thức trong vòng lặp for

- **Biểu thức 1:** thường dùng để khởi tạo biến đếm của vòng lặp. Biểu thức này có thể có hoặc không có cũng được.
- **Biểu thức 2:** thường dùng để kiểm tra điều kiện của vòng lặp. Biểu thức này bắt buộc phải có (nếu bỏ qua biểu thức này ta phải dùng nó với từ khóa break)
- **Biểu thức 3:** thường dùng để điều khiển biến đếm của vòng lặp. Biểu thức này có thể có hoặc không có cũng được.

Phân tích hoạt động của cấu trúc lặp for

- **Bước 1:** Xác định biểu thức 1.
- **Bước 2:** Xác định biểu thức 2
- **Bước 3:** Tùy thuộc vào giá trị của biểu thức 2, chương tình thực thi một trong hai nhánh.
- **Bước 3.1:** Nếu biểu thức 2 có giá trị 0 (sai), chương trình sẽ thoát khỏi for và chuyển tới câu lệnh sau khối lệnh của for.
- **Bước 3.2:** Nếu biểu thức 2 có giá trị khác 0 (đúng) chương trình sẽ thực hiện các câu lệnh trong khối lệnh for.
- **Bước 4:** Trong biểu thức 3, sau đó quay lại bước 2 để bắt đầu một vòng lặp mới.

Điều kiện dừng của vòng lặp

- Điều kiện dừng của vòng lặp sẽ trả về **true** hoặc **false**, nếu true vòng lặp chạy tiếp và false sẽ thoát.

Các chỉ thị can thiệp vào vòng lặp

- Lệnh break
- Lệnh continue
- Lệnh return

Câu lệnh break

- Câu lệnh break cho phép ra khỏi cấu trúc điều khiển lặp (vòng for, while, do while) và cấu trúc chọn switch. Khi có nhiều vòng lặp lồng nhau, câu lệnh break sẽ thoát khỏi vòng lặp chứa nó bên trong khối lệnh lặp.
- Như vậy break cho ta khả năng ra khỏi một cấu trúc điều khiển lặp mà không dùng điều kiện kết thúc chương trình.

Câu lệnh continue

- Ngược lại với câu lệnh break, câu lệnh continue dùng để bắt đầu một vòng mới của cấu trúc điều khiển lặp chứa nó.
- Khi gặp câu lệnh continue bên trong thân của một toán tử for, chương trình sẽ thực hiện bước 4 trong phần "phân tích sự hoạt động của cấu trúc lặp for"

Câu lệnh continue

- Khi gặp câu lệnh continue bên trong thân của while hoặc do while, chương trình thực hiện bước 1 trong phần "phân tích sự hoạt động của cấu trúc lặp while"
- Ghi chú: Câu lệnh continue chỉ áp dụng cho các cấu trúc điều khiển lặp chứ không áp dụng cho cấu trúc điều khiển chọn switch.

Ví dụ lệnh continue

- In ra các số lẻ nhỏ hơn 100, trừ các số 5, 7, 93.

```
#include <stdio.h>
void main() {
	int i;
	for (i = 1; i < 100; i += 2) {
		if (i == 5 || i == 7 || i == 93)
			continue;
		printf("%5d", i);
	}
}
```

Câu lệnh return

- Trả về dòng điều khiển mà nơi nó gọi, khi lệnh return được theo sau bởi một biểu thức thì biểu thức đó sẽ được đánh giá và giá trị này sẽ được trả về cho nơi đã gọi hàm. Khi return được gọi mà không có biểu thức đi kèm thì giá trị trả về là không xác định.
- Câu lệnh return không chỉ thoát khỏi vòng lặp mà nó còn thoát luôn khỏi hàm mà đang chứa nó.

Các vấn đề tìm hiểu mở rộng kiến thức nghề nghiệp

Tìm hiểu thêm

- Tránh sự nhập nhằng và khó hiểu trong mã nguồn
- Các chỉ thị đặc biệt bao hàm cấu trúc điều khiển
- Cấu trúc điều khiển cấp cao trong các NNLT
- Sự khác biệt, tương đồng giữa các NNLT

Thuật ngữ và bài đọc thêm tiếng Anh

Thuật ngữ tiếng Anh

- ***block***: khối lệnh
- ***branching***: rẽ nhánh, phân nhánh.
- ***control structures***: các cấu trúc điều khiển.
- ***global variables***: biến toàn cục
- ***instruction***: lệnh.
- ***local variables***: biến cục bộ
- ***loop***: vòng lặp.
- ***program***: chương trình.
- ***variable***: biến.

Bài đọc thêm tiếng Anh

- **Thinking in C**, Bruce Eckel, E-book, 2006.
- **Theory and Problems of Fundamentals of Computing with C++**, John R.Hubbard, Schaum's Outlines Series, McGraw-Hill, 1998.