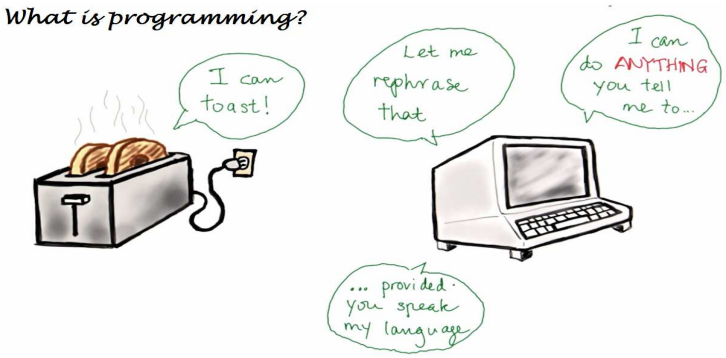



Tuần 2

# Lệnh đơn và cấu trúc rẽ nhánh

CT101 – Lập trình căn bản

Khoa CNTT&TT – ĐHCT

## Mục đích

- Cung cấp các khái niệm về biến, hằng, biểu thức
- Mô tả các kiểu dữ liệu cơ bản và cách sử dụng
- Giới thiệu lệnh gán, các lệnh nhập xuất cơ bản
- Giới thiệu cấu trúc lệnh rẽ nhánh và cách sử dụng

# Yêu cầu

- Hiểu các khái niệm về biến, hằng, biểu thức
- Biết cách khai báo biến, hằng, viết một biểu thức
- Hiểu các lệnh nhập xuất cơ bản
- Ứng dụng các lệnh đơn, kiểu dữ liệu để viết một chương trình cơ bản
- Hiểu lệnh rẽ nhánh
- Viết được những chương trình đơn giản có lưu trữ dữ liệu, tính toán, hiển thị kết quả tùy theo điều kiện của vấn đề đặt ra

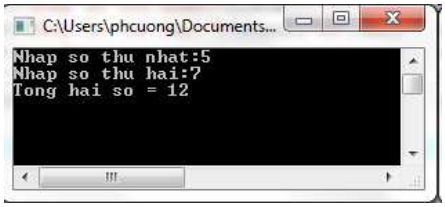


# Nội dung

- Biến và các vấn đề liên quan
- Các kiểu dữ liệu cơ bản
  - Kiểu int
  - Kiểu char
  - Kiểu float
- Biểu thức
- Lệnh nhập xuất cơ bản
- Cấu trúc lệnh rẽ nhánh

# Biến(Variables)

```
#include <stdio.h>
#include <conio.h>
int main(){
    int a, b, tong;
    printf("Nhap so thu nhat:");scanf("%d",&a);
    printf("Nhap so thu hai:");scanf("%d",&b);
    tong = a + b;
    printf("Tong hai so = %d", tong);
    getch();
}
```



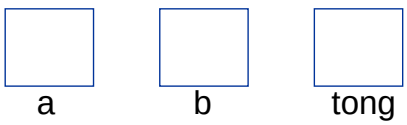

Chương trình cần có nơi lưu trữ dữ liệu tạm thời trong lúc thực hiện



# Biến(Variables)

- Biến: vùng nhớ được đặt tên để lưu trữ dữ liệu
- Trong ngôn ngữ C, biến phải được *khai báo* trước khi sử dụng. Việc khai báo bao gồm đặt tên cho biến và quy định kiểu dữ liệu của biến
- Dùng phép toán gán trị để thay đổi giá trị của biến

```
#include <stdio.h>
#include <conio.h>
int main(){
    int a, b, tong;
    printf("Nhap so thu nhat:");
    scanf("%d",&a);   5
    printf("Nhap so thu hai:");
    scanf("%d",&b);   10
    tong = a + b;
    printf("Tong hai so = %d", tong);   15
    getch();
}
```



# Biến(Variables)

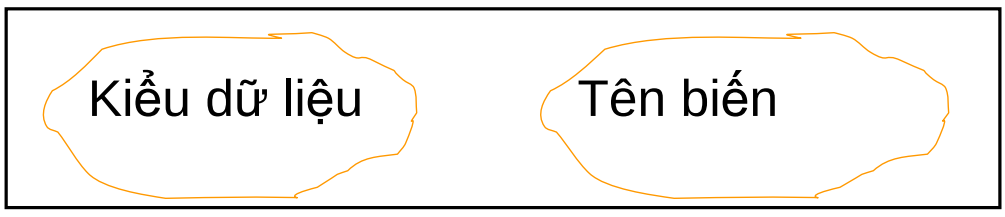
```
int a;          // khai báo một biến kiểu số nguyên
int sum, a1,a2; // khai báo 3 biến cùng kiểu
int x=7; // khai báo và khởi tạo biến
a=5;  // gán giá trị 5 vào biến a
a1=a; // gán giá trị của a vào biến a1
```

***L-value*** ***R-value***

```
a1=a1+1;  // gán giá trị a1+1 vào biến a1
          // (tăng giá trị a1 lên 1)
```



# Biến(Variables)




Có những kiểu dữ liệu gì?

Đặt tên như thế nào ?

```
int a;                    double luong;
          float diemTB;
char ch;                      int i,j,k;
```

# Biến(Variables)

- Tên có thể gồm chữ cái, chữ số và dấu gạch dưới.
- Tên phải bắt đầu bằng chữ cái hoặc dấu gạch dưới ( `_` ) .
- Tên không được trùng với từ khóa (các từ dành riêng của C)
- Ví dụ các tên đúng quy tắc: `Sum`, `pieceFlag`, `I`, `J5x7`, `Number_of_moves`, `_sysflag`
- Các tên không đúng quy tắc: `sum$value`, `3Spencer`, `int`.
- C phân biệt chữ hoa và chữ thường. Ví dụ: `sum`, `Sum`, `SUM` là 3 tên khác nhau



# Kiểu dữ liệu cơ bản

- Các kiểu dữ liệu cơ bản trong C: `int`, `float`, `char`
- Mỗi kiểu dữ liệu có miền giá trị khác nhau
- Kiểu dữ liệu `int`: dùng để lưu trữ số nguyên
- Kiểu dữ liệu `float`: dùng để lưu trữ số thực dấu chấm động
- Kiểu dữ liệu `char`: dùng để lưu trữ một ký tự (chữ cái, chữ số, ký tự đặc biệt)

# Kiểu int

- Hằng số nguyên bao gồm các ký số viết liền nhau (có thể có dấu - ở đầu để biểu diễn số âm), ví dụ: 158, –10, 0
- Dùng định dạng %d trong lệnh `printf` để hiển thị số nguyên ra màn hình.
- Bình thường hằng số nguyên được viết ở cơ số 10, nhưng nếu cần có thể biểu diễn ở cơ số 8 hoặc cơ số 16

```
#include <stdio.h>
int main (){
    int integerVar = 100;
    printf ("integerVar = %d\n", integerVar);
}
```



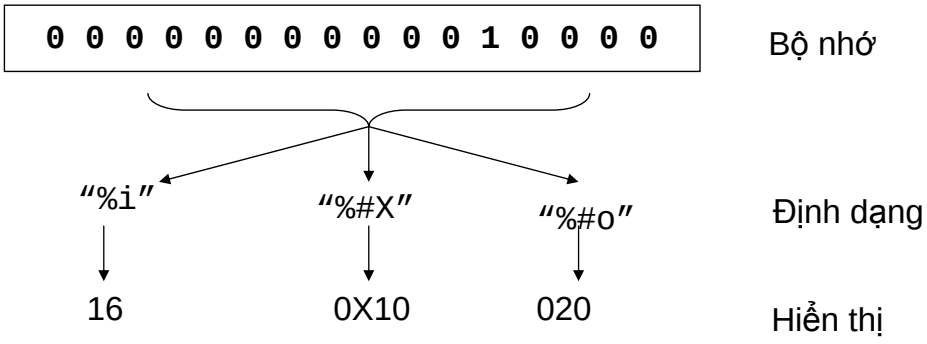
# Kiểu int

- **Viết hằng số nguyên ở cơ số 8**: nếu ký số đầu tiên của hằng số nguyên là 0 thì C sẽ hiểu đây là số nguyên biểu diễn theo cơ số 8. Trong trường hợp này, các ký số tiếp theo phải là các ký hiệu hợp lệ trong cơ số 8 (từ 0 đến 7)
- Ví dụ: hằng `0177` biểu diễn giá trị `127` trong hệ thập phân ($1 \times 8^2 + 7 \times 8 + 7$).
- **Viết hằng số nguyên ở cơ số 16**: nếu phần đầu của một hằng số nguyên là 0x(hoặc 0X) thì C hiểu đây là số nguyên biểu diễn theo cơ số 16. Các ký hiệu tiếp theo phải là các ký hiệu hợp lệ trong hệ cơ số 16(0–9 và a–f (hoặc A–F)
- Ví dụ: hằng `0xA3F` biểu diễn giá trị `2623` trong hệ thập phân ($10 \times 16^2 + 3 \times 16 + 15$)

# Kiểu int

- Các định dạng hiển thị không làm thay đổi cách dữ liệu được biểu diễn trong bộ nhớ

```
int x =16;
printf("%i %#X %#o\n", x,x,x);
```






# Kiểu float

- Hằng số thực bao gồm các ký số viết liền nhau, phần nguyên và phần lẻ được phân cách bằng dấu chấm. Ví dụ: 3., 125.8, –.0001
- Dùng định dạng %f trong lệnh `printf` để hiển thị số thực ra màn hình
- Hằng số thực còn có thể viết dưới dạng ký pháp khoa học.Ví dụ hằng `1.7e4` biểu diễn giá trị $1.7\times10^4$

```
#include <stdio.h>
int main (){
    float floatingVar = 331.79;
    printf ("floatingVar = %f\n", floatingVar);
}
```

# Kiểu char

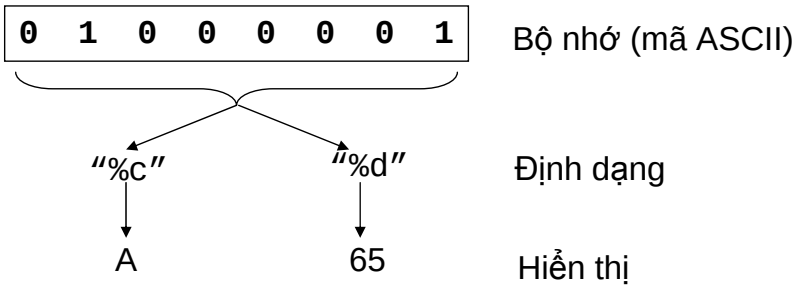
- Biến kiểu `char` dùng để lưu trữ một ký tự
- Hằng ký tự được đặt trong cặp dấu nháy đơn. Ví dụ: 'a', ';', '0'
- Dùng định dạng %c trong lệnh `printf` để hiển thị ký tự ra màn hình
- Một số ký tự theo sau dấu \ sẽ có ý nghĩa đặc biệt. Ví dụ: \n, \t
- Mỗi ký tự có mã số, thông thường là mã ASCII

```
#include <stdio.h>
int main (){
    char charVar = 'W';
    printf ("charVar = %c\n", charVar);
}
```

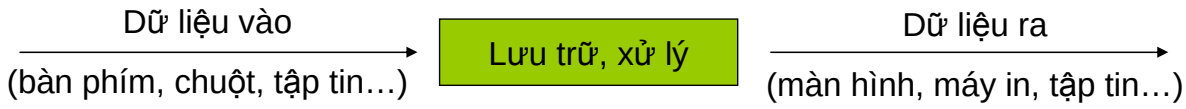


# Kiểu char

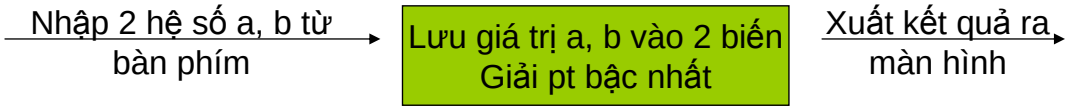
```
/* displays ASCII code for a character */

#include <stdio.h>
int main(){
    char ch;
    ch='A';
    printf("The code for %c is %d.\n", ch, ch);
    return 0;
}
```



# Nhập, xuất cơ bản




Ví dụ: giải phương trình bậc nhất ax+b = 0




scanf(chuỗi định dạng, danh sách biến)

printf(chuỗi định dạng, danh sách tham số)



# Nhập, xuất cơ bản

```
#include <stdio.h>
#include <conio.h>
int main(){
    float a,b,x;
    printf("Chuong trinh giai phuong trinh bac nhat ax+b=0\n");
    printf("Nhap he so a = ");
    scanf("%f",&a);
    printf("Nhap he so b = ");
    scanf("%f",&b);
    if (a==0)
       if (b==0) printf("Phuong trinh vo so nghiem");
       else printf("Phuong trinh vo nghiem");
    else {
         x = -b/a;
         printf("Phuong trinh co nghiem x = %.2f", x);
    }
    getch();
}
```

# Biểu thức toán học

- Biểu thức toán học gồm các toán hạng và các toán tử. Trong biểu thức, các toán tử có thể là các phép toán cơ bản như +, -, *, / hoặc các hàm lời gọi hàm, các toán hạng là các hằng, biến, kết quả của các hàm
- Trong biểu thức toán học, các toán tử có độ ưu tiên khác nhau



# Biểu thức toán học

```
#include <stdio.h>
int main (){
    int a = 100;
    int b = 2;
    int c = 25;
    int d = 4;
    int result;
    result = a - b; // subtraction
    printf ("a - b = %d\n", result);
    result = b * c; // multiplication
    printf ("b * c = %d\n", result);
    result = a / c; // division
    printf ("a / c = %d\n", result);
    result = a + b * c; // precedence
    printf ("a + b * c = %d\n", result);
    printf ("a * b + c * d = %d\n", a * b + c * d);
    return 0;
}
```

# Biểu thức toán học

```
#include <stdio.h>
int main (){
    int a = 25;
    int b = 2;
    float c = 25.0;
    float d = 2.0;
    printf ("6 + a / 5 * b = %i\n", 6 + a / 5 * b);
    printf ("a / b * b = %i\n", a / b * b);
    printf ("c / d * d = %f\n", c / d * d);
    printf ("-a = %i\n", -a);
    return 0;
}
```

Toán tử -
(một ngôi)



# Biểu thức toán học

```
// The modulus operator
#include <stdio.h>
int main (){
    int a = 25, b = 7;
    printf ("a %% b = %d\n", a % b);
    return 0;
}
```

Phép toán chia
lấy phần dư

# Chuyển đổi kiểu

- Khi gán một số nguyên vào một biến số thực, giá trị được tự động chuyển đổi và lưu thành số thực
- Ngược lại, khi gán một số thực vào biến số nguyên, phần lẻ thập phân sẽ bị cắt
- Trường phép toán chia (/):
  - Chia 2 số nguyên => kết quả là phép chia lấy phần nguyên
  - Chia một số thực và một số nguyên => kết quả là phép chia số thực



# Chuyển đổi kiểu

```
// Basic conversions in C
#include <stdio.h>
int main (){
float f1 = 123.125, f2;
int i1, i2 = -150;
char c = 'a';
i1 = f1; // floating to integer conversion
printf ("%f assigned to an int produces %d\n", f1, i1);
f1 = i2; // integer to floating conversion
printf ("%d assigned to a float produces %f\n", i2, f1);
f1 = i2 / 100; // integer divided by integer
printf ("%d divided by 100 produces %f\n", i2, f1);
f2 = i2 / 100.0; // integer divided by a float
printf ("%d divided by 100.0 produces %f\n", i2, f2);
f2 = (float) i2 / 100; // type cast operator
printf ("(float) %d divided by 100 produces %f\n", i2, f2);
return 0;
}
```

# Chuyển đổi kiểu

- `f2 = (float) i2 / 100;//type cast operator`
- Phép toán `(float)`chuyển đổi kiểu của giá trị biến i2 thành số thực trong lúc định trị biểu thức.
- Phép toán này không tác động đến giá trị lưu trữ trong biến i2
- Ví dụ:

  `(int) 29.55 + (int) 21.99` kết quả là `50`

  `(float) 6 / (float) 4` kết quả là `1.5`

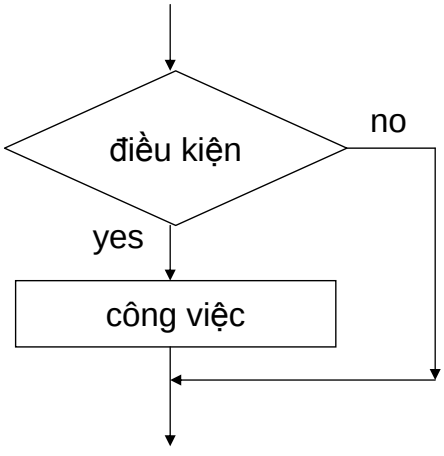
  `(float) 6 / 4` kết quả là `1.5`



# Lệnh rẽ nhánh

if ( điều kiện )
*công việc*

Nếu điều kiện đúng (khác 0), thực hiện công việc. Ngược lại bỏ qua công việc.



# Lệnh rẽ nhánh

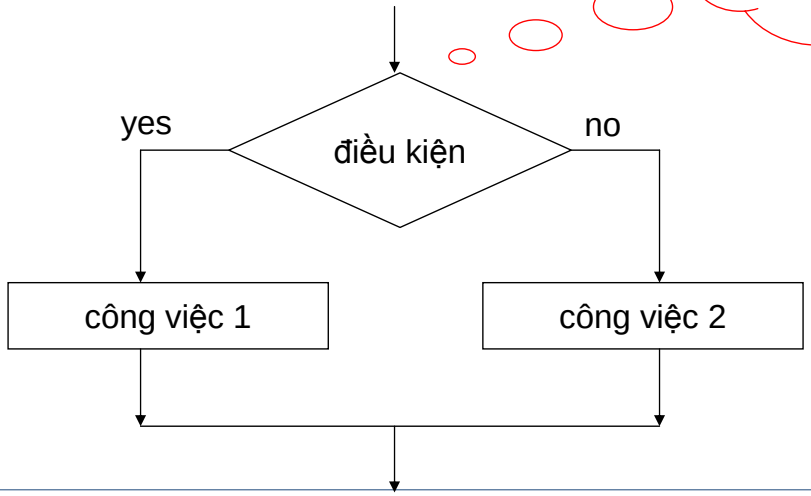
```
// Program to calculate the absolute value of an integer
int main ()
{
    int number;
    printf ("Type in your number: ");
    scanf ("%i", &number);
    if ( number < 0 )
        number = -number;
    printf ("The absolute value is %i\n", number);
    return 0;
}
```



# Lệnh rẽ nhánh

if ( *điều kiện* )
*công việc 1*
*else*
*công việc 2*

Câu lệnh if-else cho phép chọn lựa giữa 2 công việc



# Lệnh rẽ nhánh

```
// Program to determine if a number is even or odd
#include <stdio.h>
int main ()
{
    int number_to_test, remainder;
    printf ("Enter your number to be tested: ");
    scanf ("%i", &number_to_test);
    remainder = number_to_test % 2;
    if ( remainder == 0 )
        printf ("The number is even.\n");
    else
        printf ("The number is odd.\n");
    return 0;
}
```



# Lệnh rẽ nhánh

if ( *điều kiện* )
    *công việc 1*
*else*
    *công việc 2*

Trong C, dấu **;** là phần kết thúc của câu lệnh, được đặt trước ngay cả `else` !

```
if ( remainder == 0 )
    printf ("The number is even.\n");
else
    printf ("The number is odd.\n");
```

```
if ( x == 0 );
    printf ("The number is zero.\n");
```

Không sai về cú pháp nhưng có thể sai về ngữ nghĩa !

# Lệnh rẽ nhánh

```
// Program to determine if a year is a leap year
#include <stdio.h>
int main (void)
{
    int year, rem_4, rem_100, rem_400;
    printf ("Enter the year to be tested: ");
    scanf ("%i", &year);
    rem_4 = year % 4;
    rem_100 = year % 100;
    rem_400 = year % 400;
    if ( (rem_4 == 0 && rem_100 != 0) || rem_400 == 0 )
        printf ("It's a leap year.\n");
    else
        printf (“It's not a leap year.\n");
    return 0;
}
```



# Lệnh rẽ nhánh

| Toán tử | Ký hiệu | Ý nghĩa |
|---|---|---|
| AND | && | x && y đúng khi cả x và y cùng đúng |
| OR | \|\| | x \|\| y đúng khi có ít nhất x đúng hoặc y đúng |
| NOT | ! | !x đúng nếu x sai |

# Lệnh rẽ nhánh

Độ ưu tiên

!, ++, --, (type)

*, /, %

+, -

<, <=, >, >=, ==, !=

&&

||

=

Ví dụ về độ ưu tiên của các toán tử:

```
a > b && b > c || b > d
```

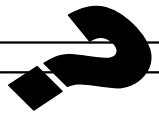
Tương đương với:

```
((a > b) && (b > c)) || (b > d)
```



# Lệnh rẽ nhánh

```
if (x >= 5 && x <= 10)
        printf("in range");
```



```
if (5 <= x <= 10)
        printf("in range");
```

# Lệnh rẽ nhánh

YES

```
if (x >= 5 && x <= 10)
        printf("in range");
```

NO!

```
if (5 <= x <= 10)
        printf("in range");
```

Đúng cú pháp nhưng sai ý nghĩa !!!

Bởi vì trong biểu thức này 2 toán tử <= có độ ưu tiên như nhau, quy tắc kết hợp từ trái sang phải sẽ được áp dụng như sau:
`(5<= x) <= 10`
Biểu thức `5 <= x` có giá trị 1 (nếu đúng) hoặc 0 (nếu sai). Trong cả hai trường hợp thì đều nhỏ hơn 10, vì thế toàn bộ biểu thức luôn luôn đúng, không phụ thuộc giá trị của x !



# Lệnh rẽ nhánh

```
if (number > 5)
    if (number < 10)
        printf("1111\n");
    else printf("2222\n");
```

Quy tắc: else sẽ được kết hợp với if gần nhất nếu không có dấu ngoặc móc

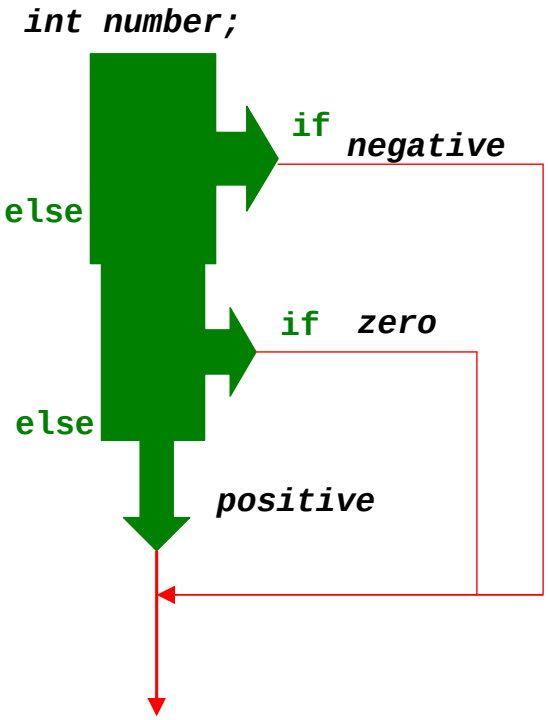
```
if (number > 5) {
    if (number < 10)
        printf("1111\n");
}
else printf("2222\n");
```

# Lệnh rẽ nhánh

```
// Program to implement the sign function
#include <stdio.h>
int main (void)
{
    int number, sign;
    printf ("Please type in a number: ");
    scanf ("%i", &number);
    if ( number < 0 )
        sign = -1;
    else if ( number == 0 )
        sign = 0;
    else // Must be positive
        sign = 1;
    printf ("Sign = %i\n", sign);
    return 0;
}
```



# Lệnh rẽ nhánh




```
if ( expression 1)
    program statement 1
else if ( expression 2)
    program statement 2
else
    program statement 3
```

Cấu trúc if-else liên tục với nhiều nhánh rẽ gây khó khăn khi đọc hiểu chương trình

# Lệnh rẽ nhánh

```
/* Program to evaluate simple expressions of the form
number operator number */
#include <stdio.h>
int main (void) {
    float value1, value2;
    char operator;
    printf ("Type in your expression.\n");
    scanf ("%f %c %f", &value1, &operator, &value2);
    if ( operator == '+' )
        printf ("%.2f\n", value1 + value2);
    else if ( operator == '-' )
        printf ("%.2f\n", value1 - value2);
    else if ( operator == '*' )
        printf ("%.2f\n", value1 * value2);
    else if ( operator == '/' )
        printf ("%.2f\n", value1 / value2);
    else printf ("Unknown operator.\n");
    return 0;
}
```



# Lệnh rẽ nhánh

```
switch ( expression )
{
    case value1:
        program statement
        program statement
        ...
        break;
    case value2:
        program statement
        program statement
        ...
        break;
        ...
    case valuen:
        program statement
        program statement
        ...
        break;
    default:
        program statement
        program statement
        ...
        break;
}
```

Giá trị biểu thức *expression* được so sánh với các giá trị *value1*, *value2*, ..., *valuen*. Khi một giá trị *valuei* bằng với *expression*, nhóm lệnh trong *case* tương ứng sẽ được thực hiện

Giá trị của *expression* phải là một **số nguyên** (bao gồm kiểu char).

Sau *case* phải là một hằng hay biểu thức hằng kiểu số nguyên, **không được sử dụng biến ở đây**

# Lệnh rẽ nhánh

```
/* Program to evaluate simple expressions of the form
value operator value */
#include <stdio.h>
int main (void) {
   float value1, value2;
   char operator;
   printf ("Type in your expression.\n");
   scanf ("%f %c %f", &value1, &operator, &value2);
   switch (operator)  {
      case '+':  printf ("%.2f\n", value1 + value2); break;
      case '-': printf ("%.2f\n", value1 - value2); break;
      case '*': printf ("%.2f\n", value1 * value2); break;
      case '/':
            if ( value2 == 0 ) printf ("Division by zero.\n");
            else   printf ("%.2f\n", value1 / value2);
            break;
      default:  printf ("Unknown operator.\n");  break;
   }
   return 0;
}
```



# Lệnh rẽ nhánh


```
switch (operator)
{
   ...
   case '*':
   case 'x':
      printf ("%.2f\n", value1 * value2);
      break;
   ...
}
```

# Tổng kết

- Sử dụng biến để lưu trữ dữ liệu
- Trong C có các kiểu dữ liệu cơ bản như `int`, `float`, `char`
- Nhập, xuất cơ bản bằng lệnh scanf, printf
- Sử dụng cấu trúc rẽ nhánh để điều khiển hoạt động của chương trình tùy theo giá trị của biểu thức điều kiện



# Kiểm tra kiến thức

1. Trong các tên biến sau, tên nào đặt sai quy cách

a. tong b. a15 c. _var d. 12A1

2. Trong các kiểu dữ liệu sau, kiểu nào có miền giá trị là các số thực?

a. int b. char c. float d. long int

3. Nếu công việc trong cấu trúc if-else có từ 2 câu lệnh trở lên ta phải làm gì?

a. Viết lệnh bình thường b. Viết xuống dòng

c. Dùng cặp ngoặc { } để bao các câu lệnh lại d. Tất cả đều sai

4. Khi gán a = 5.2 (với a là biến kiểu int), giá trị lưu trữ vào a là gì ?

a. 5.2 b. 0 c. 5 d. tất cả đều sai

5. Gán a = (1/2)*b với a, b là các biến kiểu int và giá trị b đang là 10. Giá trị của a là ?

a. 5 b. 0 c. 10 d. tất cả đều sai

# Bài tập tổng kết

**1.** Viết chương trình tính chu vi, diện tích của tam giác với yêu cầu sau khi nhập 3 số a, b, c phải kiểm tra lại xem a, b, c có tạo thành một tam giác không? Nếu có thì tính chu vi và diện tích. Nếu không thì in ra câu " Không tạo thành tam giác".

**2.** Viết chương trình giải phương trình bậc nhất ax+b=0 với a, b nhập từ bàn phím.

**3.** Viết chương trình giải phương trình bậc hai $ax^2+bx+c=0$ với a, b, c  nhập từ bàn phím.

**4.** Viết chương trình nhập vào ngày tháng năm của ngày hôm nay, in ra ngày tháng năm của ngày mai.



# Bài tập tổng kết

**5.** Viết  chương trình nhập từ bàn phím 2 số a, b và một ký tự ch.

Nếu:  ch là  "+"  thì thực hiện phép tính a + b và in kết quả lên màn hình.

ch là  "–"  thì thực hiện phép tính a - b và in kết quả lên màn hình.

ch là  "*"  thì thực hiện phép tính a * b và in kết quả lên màn hình.

ch là  "/"  thì thực hiện phép tính a / b và in kết quả lên màn hình.

CT101 – Lập trình căn bản

Khoa CNTT&TT – ĐHCT