# TIN HỌC ĐẠI CƯƠNG

## BÀI 2: THUẬT TOÁN & CÁC KHÁI NIỆM CƠ BẢN TRONG C++

Phạm Xuân Cường
Khoa Công nghệ thông tin
cuongpx@tlu.edu.vn

# Nội dung bài giảng

1. Thuật toán
    - Biểu diễn bằng mã giả
    - Biểu diễn bằng sơ đồ khối
2. Các khái niệm cơ bản trong C++
    - Chú thích
    - Câu lệnh và khối lệnh
    - Định danh
    - Các kiểu dữ liệu
    - Biến & Hằng
    - Toán tử
    - Các hàm toán học

# Thuật toán

# Thuật toán

- Dãy hữu hạn bước giải quyết một vấn đề
- Ví dụ: Tính tổng $\mathbf{S = a + b + c}$
  - Bước 1: Cung cấp giá trị cho **a, b, c**
  - Bước 2: Tính $\mathbf{t = a + b}$
  - Bước 3: Tính $\mathbf{S = t + c}$
  - Bước 4: Thông báo giá trị của tổng **S**

# Thuật toán

- Có thể có nhiều thuật toán giải quyết cùng một vấn đề
- Ví dụ: Tính giá trị biểu thức **bt = a * (b + c)**

**Thuật toán 1**

1. Nhập giá trị của **a, b, c**
2. Tính **t = b + c**
3. Tính **bt = a * t**
4. Thông báo giá trị của **bt**

**Thuật toán 2**

1. Nhập giá trị của **a, b, c**
2. Tính **t1 = a * b**
3. Tính **t2 = a * c**
4. Tính **bt = t1 + t2**
5. Thông báo giá trị của **bt**

# Biểu diễn thuật toán

- Dùng mã giả:
  - Ngôn ngữ linh hoạt, tùy người viết
  - Không dài dòng như ngôn ngữ tự nhiên
  - Không khắt khe như ngôn ngữ lập trình
- Dùng sơ đồ khối:
  - Mỗi khối có một ý nghĩa xác định
  - Mũi tên nối các khối thể hiện trình tự các bước

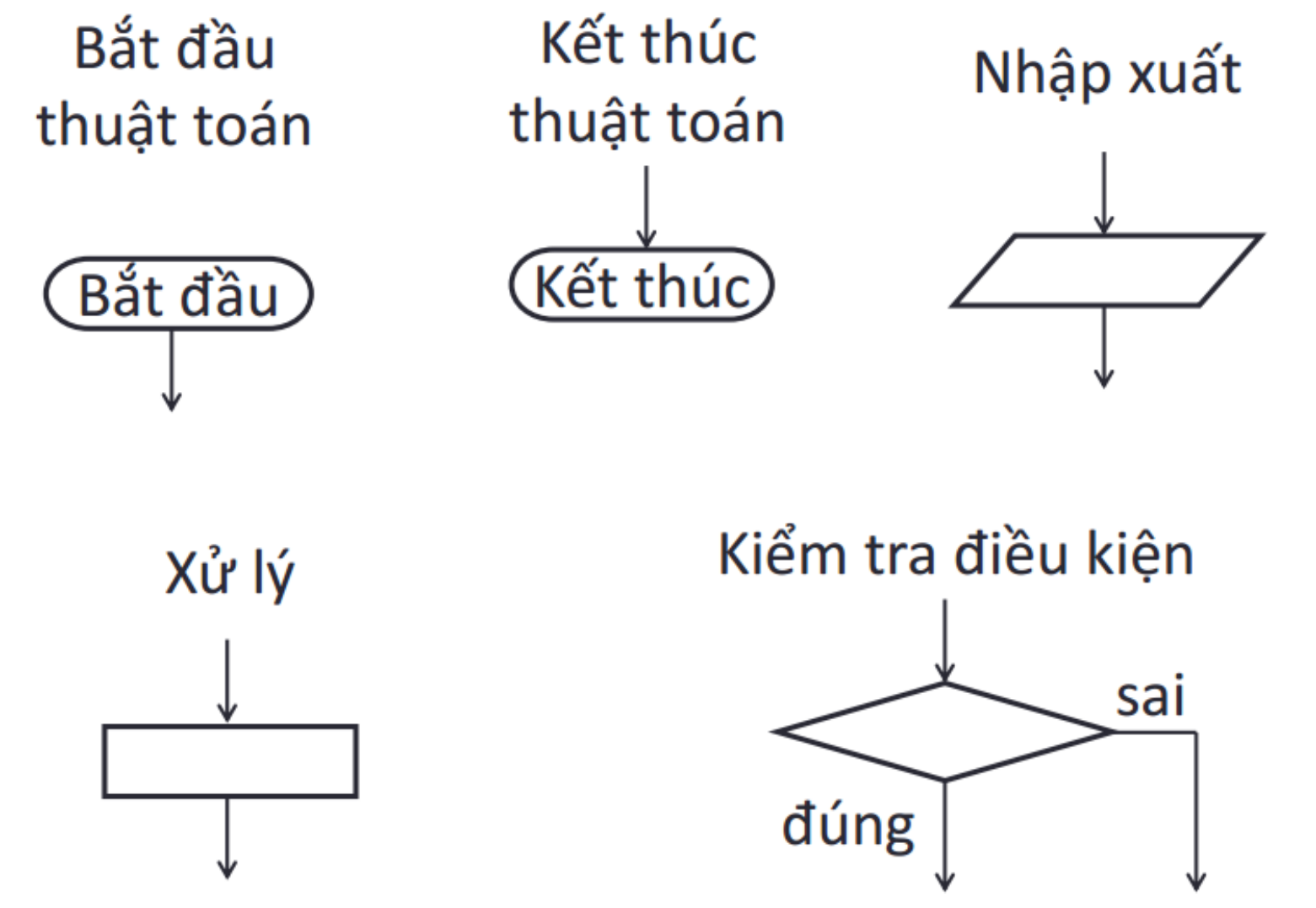
# Ví dụ về mã giả

**Bài toán:** Tính điện trở tương đương Rtd của hai điện trở R1 và R2 mắc song song (Công thức đã biết: $1/Rtd = 1/R1 + 1/R2$)

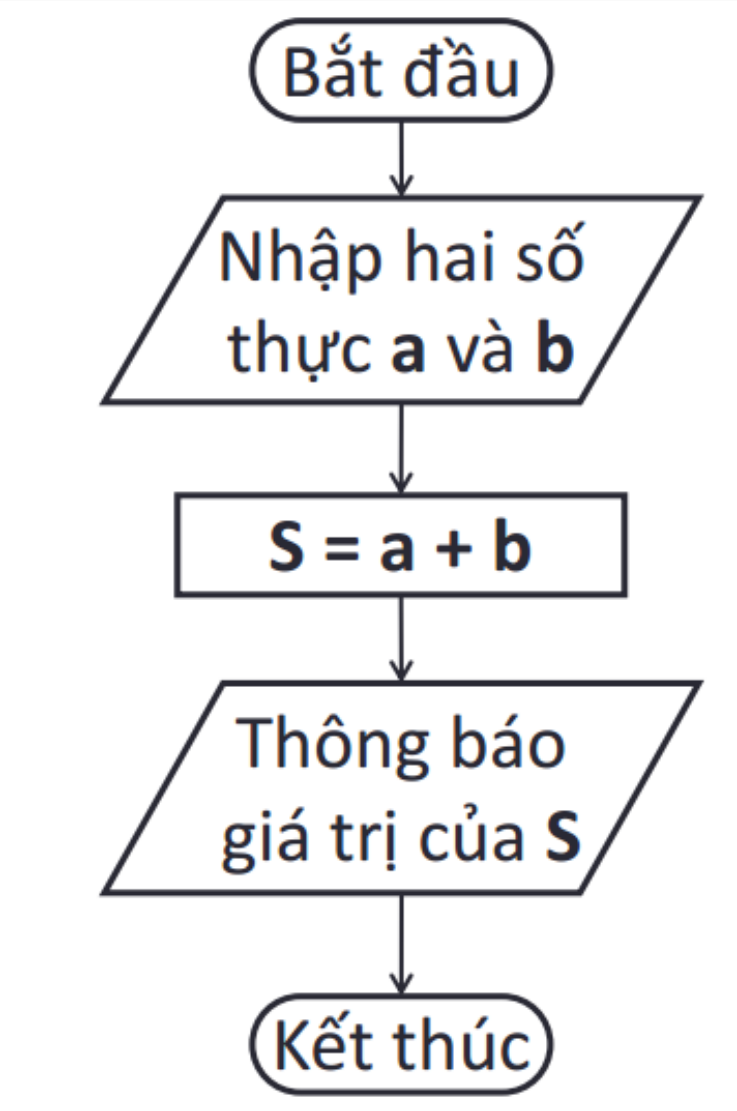
**Thuật toán:** Tính điện trở tương đương
**Đầu vào:** R1 và R2
**Đầu ra:** Rtd
**Bắt đầu**
1. Nhập giá trị của R1 và R2
2. if R1 <= 0 or R2 <= 0 then
3. Báo lỗi và kết thúc
4. else
5. Tính tg = 1/R1 + 1/R2
6. Tính Rtd = 1/tg
7. end if
8. return Rtd

**Kết thúc**

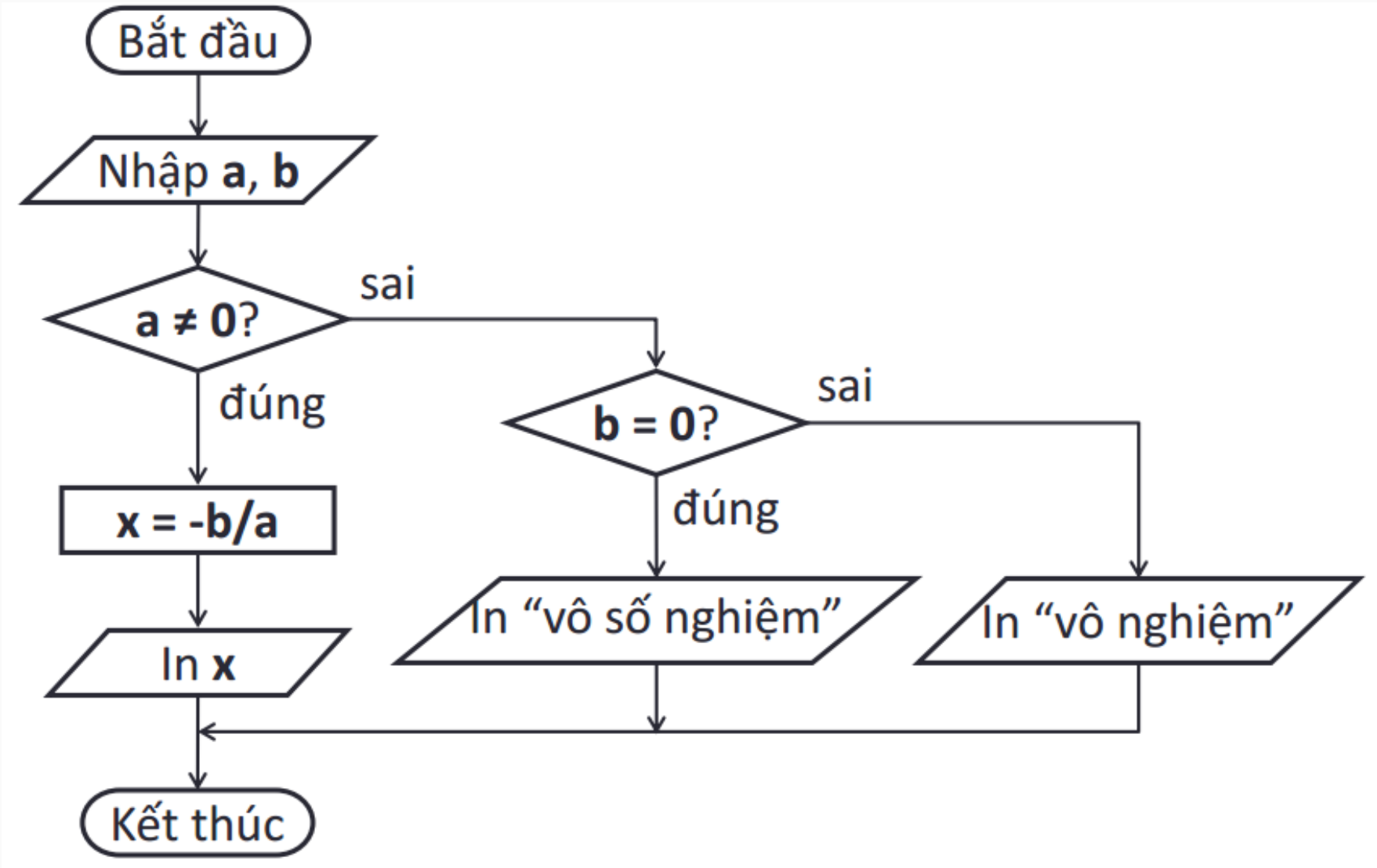
# Các khối cơ bản trong sơ đồ khối



# Sơ đồ khối tính tổng hai số thực



# Sơ đồ khối giải phương trình bậc nhất $ax + b = 0$

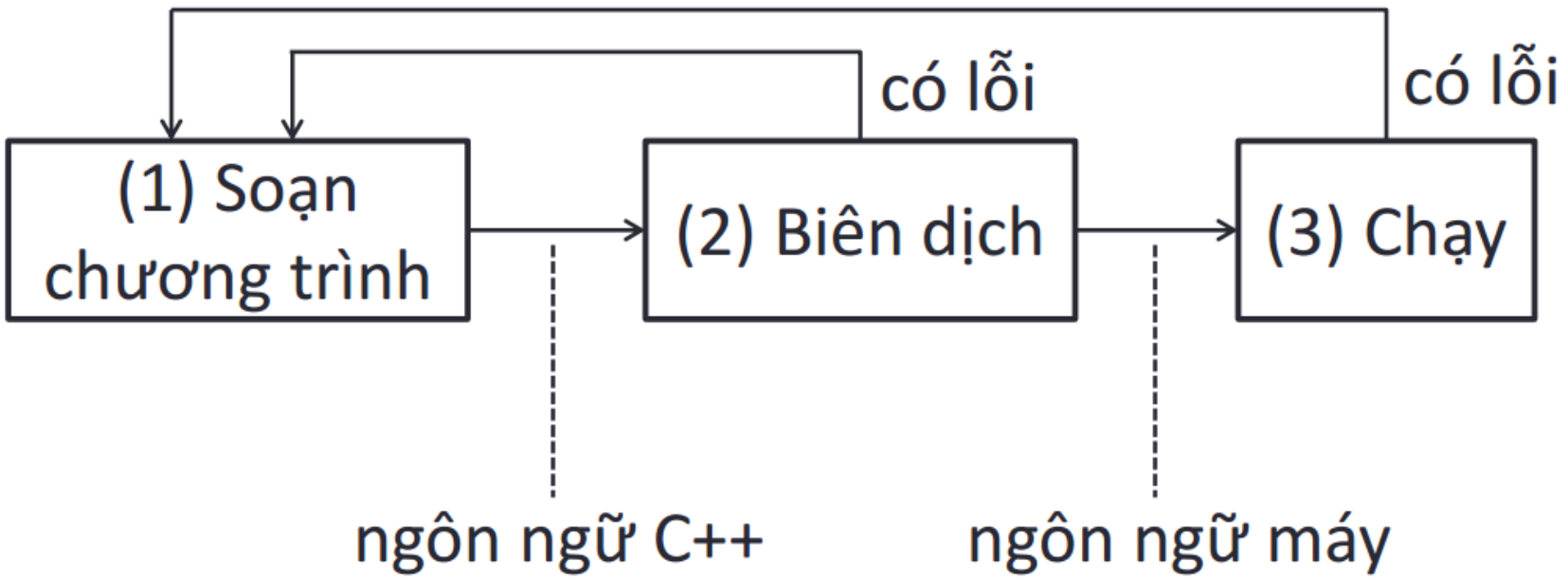


# Các khái niệm cơ bản trong C++

# Giải quyết vấn đề bằng lập trình

1. Xác định và phân tích vấn đề
2. Xây dựng thuật toán giải quyết vấn đề
3. Viết chương trình theo thuật toán ở bước 2
4. Chạy thử, kiểm tra và sửa các lỗi

# Các bước lập trình




*Phần mềm Dev-C++ hỗ trợ các bước lập trình bên trên.*

# Phần mềm Dev–C++

- Cho phép viết, biên dịch (ấn phím F9), chạy (ấn phím F10) và gỡ lỗi các chương trình C++
- Bao gồm:
  - Trình biên tập chương trình nguồn C++
  - Trình biên dịch C++
  - Trình gỡ lỗi C++
- Tải về tại đây: Dev–C++

# Viết chương trình C++ đầu tiên

**Bước 1:** Mở phần mềm Dev–C++

**Bước 2:** Tạo tệp nguồn C++ (ấn tổ hợp phím Ctrl + N)

**Bước 3:** Gõ vào chương trình C++ sau đây

```
// Day la chuong trinh C++ dau tien
#include <iostream>
using namespace std;
int main()
{
    cout << "Xin chao cac ban";
    return 0;
}
```

**Bước 4:** Biên dịch và chạy (ấn phím F11)

# Cấu trúc chương trình C++

- Phần định hướng bộ tiền xử lý:

  #include <iostream>

- Phần khai báo sử dụng không gian tên:

  using namespace std;

- Phần thân chương trình

  int main() { ... }

  - Khi chạy chương trình, các câu lệnh trong hàm main được thực thi tuần tự

# Chú thích

- Chú thích dùng để giải thích ý nghĩa của các câu lệnh
- Bắt đầu từ hai dấu gạch chéo ( // ) cho đến hết dòng
- Chú thích một dòng:

```
// Day la chu thich tren mot dong
```

- Chú thích nhiều dòng:

```
// Day la chu thich
// tren hai dong
```

- Có thể viết chú thích ngay sau câu lệnh:

```
cout << "Xin chao";     // hien thi dong "Xin chao"
```

# Câu lệnh và khối lệnh

- Câu lệnh:
  - Phải kết thúc bằng dấu chấm phẩy ( **;** )
  - Thực hiện một thao tác cụ thể:
    - Hiển thị thông điệp: cout << "Xin chao";
    - Gán giá trị cho biến: x = 10;
    - v.v...
- Khối lệnh: gồm nhiều câu lệnh đặt trong cặp dấu móc ( {} )

```
if (x > 0) {
    y = 1/x;
    cout << y;
}
```

# Định danh (tên)

- Có nhiều thực thể trong chương trình C++: biến, hằng, hàm, v.v...
- Mỗi thực thể có một định danh (tên)

```
int n1; // bien nguyen co ten la n1
```

- Các quy định khi đặt tên:
  - Chỉ dùng chữ cái (a...z, A...Z), chữ số (0...9) và dấu gạch dưới (_)
  - Bắt đầu bằng chữ cái hoặc dấu gạch dưới
  - Không trùng với các từ khóa C++ (như int, return, if, for, while, v.v...)

# Định danh (tên)

- Định danh có phân biệt chữ hoa chữ thường
- Những định danh sau đây là khác nhau:
  HoTen, hoten, Hoten, hoTen, HOTEN
- Một vài quy ước (không bắt buộc) khi đặt tên:
  - Tên biến và hàm dùng chữ thường
    Ví dụ: x1, x2, hoten, tinh_tong
  - Tên hằng dùng toàn chữ hoa:
    Ví dụ: SO_PI, DIEM_CHUAN, MIN, MAX

# Các kiểu dữ liệu

| Tên kiểu | Ý nghĩa | Kích thước (Byte) | Miền giá trị |
|---|---|---|---|
| **char** | Ký tự | 1 | các ký tự (a, b, c, +, −, . . . ) |
| **short** | Số nguyên | 2 | $-32,768$ đến $32,767$ |
| **int** | Số nguyên | 4 | $-2^{32}/2$ đến $2^{32}/2-1$ |
| **float** | Số thực | 4 | $\approx -3.4 \times 10^{38}$ đến $3.4 \times 10^{38}$ |
| **double** | Số thực | 8 | $\approx -1.7 \times 10^{308}$ đến $1.7 \times 10^{308}$ |
| **bool** | Kiểu lôgic | 1 | **true** (đúng), **false** (sai) |

**Chú ý:** Ở đây, ta **dùng dấu chấm làm dấu thập phân** (giống như trong C++), dấu phẩy làm dấu phân tách từng cụm ba chữ số trong số nguyên cho dễ đọc.

# Cách viết số và ký tự trong C++

- Viết số nguyên và số thực như trong toán (nhưng phải dùng dấu chấm làm dấu thập phân)
  - Ví dụ: 25, $-38$, 1.48, $-12.9$
- Viết số thực khoa học: 1.2e3 $= 1.2 \times 10^3$
- Phải đặt các ký tự giữa hai dấu nháy đơn (')
  - Ví dụ: 'a', 'D', '+', '&'
- Các ký tự đặc biệt:

  '\n' ký tự xuống dòng

  '\t' dấu tab

  '\'' dấu nháy đơn

  '\"' dấu nháy kép

  '\\' dấu gạch chéo ngược

# Phân biệt ký tự và xâu ký tự

- Ký tự được đặt giữa hai dấu nháy đơn (’):
  ’a’, ’D’, ’+’, ’&’
- Xâu ký tự được đặt giữa hai dấu nháy kép ("):
  "Xin chao cac ban"

# Biến

- Chứa dữ liệu thuộc một kiểu cụ thể
- Chiếm một vùng trong bộ nhớ máy tính, có kích thước bằng kích thước kiểu dữ liệu của nó
- Cách khai báo:

  <tên kiểu> <tên biến>;
- Ví dụ:

```
int n; // bien nguyen ten la n
double x; // bien thuc ten la x
```

- Khai báo nhiều biến cùng kiểu:

```
int n1;
int n2;        ⇔        int n1, n2;
```

# Phép gán

- Dùng để gán giá trị cho biến
- Cú pháp:

  <tên biến> = <biểu thức>;

- Ví dụ:

```
double x; // x khong xac dinh
x = 1.6; // x co gia tri 1.6
x = 1.1 + 2; // x co gia tri moi la 3.1
// Ket hop khai bao va khoi tao gia tri
int n = 8; // n co gia tri 8
```

# Làm việc với biến

```
#include <iostream>
using namespace std;
int main()
{
    int n1, n2;
    int tong;
    cin >> n1; // nhap gia tri cho n1
    cin >> n2; // nhap gia tri cho n2
    tong = n1 + n2; // tinh tong hai so
    cout << tong; // hien thi tong
    return 0;
}
```

# Hằng

- Có giá trị không thay đổi được
- Cung cấp tên gọi cho một giá trị khó nhớ
- Cách khai báo: const <tên kiểu> <tên hằng> = <giá trị>;
- Ví dụ:

```
const int MAX = 100;
const float DIEM_CHUAN = 18.5;
```

# Làm việc với hằng

```
#include <iostream>
using namespace std;
int main()
{
    const float PI = 3.14;
    const int N = 10;
    N = 20; // error: loi bien dich
    float r = 2.2;
    float s = PI * r * r;
    cout << s;
    return 0;
}
```

# Toán tử

- Toán tử số học
- Toán tử so sánh
- Toán tử lôgic
- Toán tử điều kiện

# Toán tử số học

- Cộng ( + ), trừ ( - ), nhân ( * ), chia ( / ), chia lấy phần dư ( % )
- Ví dụ:

  $1.2 + 3.4 \rightarrow 4.6$

  $20 - 15 \rightarrow 5$

  $3 * 2.2 \rightarrow 6.6$

  $5 / 2 \rightarrow 2$ (chia lấy phần nguyên khi áp dụng vào hai số nguyên)

  $5 / 2.0 \rightarrow 2.5$ (chia như thông thường khi áp dụng vào hai số thực)

  $5 \% 2 \rightarrow 1$ (số dư là 1)

# Phép gán phức hợp

- Toán tử gán phức hợp gồm một toán tử và dấu bằng
- Ví dụ: +=, -=, *=, /=, %=

  double x = 1;

  x += 1; (lấy x cộng 1 rồi gán lại cho x, do đó x sẽ bằng 2 sau phép gán phức hợp này)

  x *= 2.4; (lấy x nhân 2.4 rồi gán lại cho x, do đó x sẽ bằng 4.8 sau phép gán phức hợp này)
- Toán tử gán phức hợp cho phép viết mã ngắn gọn hơn

# Toán tử tăng/giảm

- Toán tử tăng ( $++$ ): Tăng giá trị của biến một đơn vị, có thể viết trước hoặc sau tên biến

```
int n = 2;
++n; // n se bang 3 sau cau lenh nay
n++; // n se bang 4 sau cau lenh nay
```

- Toán tử giảm ( $--$ ): Giảm giá trị của biến một đơn vị, có thể viết trước hoặc sau tên biến

```
int n = 2;
--n; // n se bang 1 sau cau lenh nay
n--;// n se bang 0 sau cau lenh nay
```

# Toán tử so sánh

- So sánh giá trị của hai biểu thức
- Trả về giá trị lôgic (true/false)

| Toán tử | Ý nghĩa | Ví dụ |
|---|---|---|
| $>$ | Lớn hơn | $6 > 3$ trả về true, $6 > 8$ trả về false |
| $<$ | Nhỏ hơn | $2 < 5$ trả về true, $2 < 0$ trả về false |
| $>=$ | Lớn hơn hoặc bằng | $8 >= 7$ trả về true, $8 >= 9$ trả về false |
| $<=$ | Nhỏ hơn hoặc bằng | $8 <= 8$ trả về true, $6 <= 2$ trả về false |
| $==$ | Bằng | $1 == 1$ trả về true, $2 == 3$ trả về false |
| $!=$ | Khác | $4 != 5$ trả về true, $6 != 6$ trả về false |

# Làm việc với toán tử số học và toán tử so sánh

```
#include <iostream>
using namespace std;
int main()
{
    double x = -2.6;
    if (x < 0)
        x = x + 5.6;
    cout << x; // in 3 len man hinh
    return 0;
}
```

# Toán tử lôgic

- Phép và lôgic ( && ):
  - Trả về true nếu cả hai toán hạng là true
  - VD: biểu thức "$3 > 2$ && $1 < 5$" có giá trị true
- Phép hoặc lôgic ( || )
  - Trả về true nếu có ít nhất một toán hạng là true
  - VD: biểu thức "$9 == 7$ || $2 > 1$" có giá trị true
- Phép phủ định lôgic ( ! ): Đảo ngược giá trị lôgic
  - VD1: biểu thức "$!(2 > 3)$" có giá trị true
  - VD2: biểu thức "$!(4 == 4)$" có giá trị false

# Làm việc với toán tử lôgic

```
#include <iostream>
using namespace std;
int main()
{
    double x = -2.6;
    if (x > 0 || x < -1)
        x = x + 0.6;
    cout << x; // in -2 len man hinh
    return 0;
}
```

# Toán tử điều kiện

- Cú pháp:

  <điều kiện> ? <biểu thức 1> : <biểu thức 2>
- Ý nghĩa:
  - Điều kiện là một biểu thức có giá trị lôgic
  - Nếu điều kiện đúng, trả về giá trị của biểu thức 1
  - Nếu điều kiện sai, trả về giá trị của biểu thức 2
- Ví dụ:

  $3 > 8$ ? 10 : $20 + 30 \rightarrow$ trả về 50

  $5 < 6$ ? 12 : -100 $\rightarrow$ trả về 12

# Làm việc với toán tử điều kiện

```
#include <iostream>
using namespace std;
int main()
{
    int n = -4;
    double x;
    x = n > 0 ? 1.2 : -3.4; // x se bang -3.4
    cout << x;
    return 0;
}
```

# Độ ưu tiên của các toán tử

| Toán tử | Loại toán tử |
|---|---|
| () | Cặp ngoặc |
| + - ++ −− ! | Một ngôi |
| * / % | Nhân chia |
| + - | Cộng trừ |
| < <= > >= | So sánh hơn |
| == != | So sánh bằng |
| && | Phép và lôgic |
| \|\| | Phép hoặc lôgic |
| ?: | Toán tử điều kiện |
| = += -= *= /= %= | Phép gán |

# Các hàm toán học

Để sử dụng các hàm toán học, phải viết thêm dòng sau đây ở đầu chương trình C++:

```
#include <cmath>
```

| Hàm | Ý nghĩa |
|---|---|
| sqrt(x) | Tính căn bậc hai của x |
| pow(x,y) | Tính hàm mũ $x^y$ |
| fabs(x) | Tính giá trị tuyệt đối của x |
| exp(x) | Tính hàm mũ $e^x$ (e $\approx$ 2.71828) |
| log(x) | Tính lôgarit cơ số e của x |
| log10(x) | Tính lôgarit cơ số 10 của x |
| round(x) | Làm tròn x (VD: 2.2 $\rightarrow$ 2, 2.6 $\rightarrow$ 3) |

# Các hàm toán học

| Hàm | Ý nghĩa |
|---|---|
| floor(x) | Hàm sàn, trả về số nguyên lớn nhất nhưng không lớn hơn x. VD: floor(3.8) trả về 3 |
| ceil(x) | Hàm trần, trả về số nguyên nhỏ nhất nhưng không nhỏ hơn x. VD: ceil(3.8) trả về 4 |
| sin(x)<br>cos(x)<br>tan(x) | Tính sin, cos và tg của x, trong đó x đo bằng radian |
| asin(x)<br>acos(x)<br>atan(x) | Tính arcsin, arccos và arctg của x, giá trị trả về đo bằng radian |

# Làm việc với các hàm toán học

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    double x = 9.0;
    cout << sqrt(x); // in ra 3
    cout << round(4.6); // in ra 5
    return 0;
}
```

# Questions?