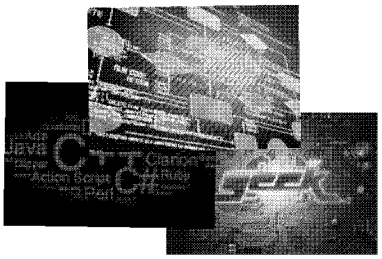
ĐẠI HỌC QUỐC GIA TPHCM
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

NHẬP MÔN LẬP TRÌNH

CHƯƠNG III (tt)

KIỂU DỮ LIỆU CƠ BẢN



Nguyễn Trọng Chỉnh
chinhnt@uit.edu.vn

KIỂU DỮ LIỆU CƠ BẢN (tt)

- CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN
- KÝ TỰ VÀ CHUỖI
- CÁC HÀM THÔNG DỤNG

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖KIỂU SỐ NGUYÊN

Số nguyên được hiểu là số không có phần thập phân. Trong ngôn ngữ C/C++, có các kiểu dữ liệu cho phép lưu trữ và tính toán các giá trị số nguyên sau:

•**char**: kích thước lưu trữ 1 byte. Phạm vi tùy theo cách sử dụng:

-**signed**: có phạm vi trong đoạn [-127, 127]

-**unsigned**: có phạm vi trong đoạn [0, 255]

-Khai báo: có thể có hoặc không có giá trị khởi tạo

```
char a; // từ khóa signed có thể không chỉ ra
unsigned char b, c = 90;
```

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖KIỂU SỐ NGUYÊN

•**short**: kích thước lưu trữ 2 byte. Phạm vi tùy theo cách sử dụng:

-**signed**: có phạm vi trong đoạn [-32767, 32767]

-**unsigned**: có phạm vi trong đoạn [0, 65535]

-Khai báo: có thể có hoặc không có giá trị khởi tạo

```
short a;
short int A = 100;
unsigned short b,c = 20;
unsigned short int B,C;
```

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖KIỂU SỐ NGUYÊN

•**long**: kích thước lưu trữ 4 byte. Phạm vi tùy theo cách sử dụng:

-**signed**: phạm vi [-2147383648, 2147383648]

-**unsigned**: phạm vi [0, 4294967295]

-Khai báo: có thể có hoặc không có giá trị khởi tạo

```
long a = 92;
long int A;
unsigned long b,c;
unsigned long int B,C = 770;
```

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖KIỂU SỐ NGUYÊN

•**int**: kích thước lưu trữ phụ thuộc vào kích thước thanh ghi. Với các máy tính có kiến trúc thanh ghi 16, 32, 64 bit, kích thước int lần lượt là 2, 4, 8 byte. Mục đích của kiểu int là để thao tác trên bộ nhớ nhanh hơn.

-**signed**: phạm vi tùy theo máy tính

-**unsigned**: phạm vi tùy theo máy tính.

-Khai báo: có thể có hoặc không có giá trị khởi tạo

```
int a;
unsigned int b=10,c;
```

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖KIỂU SỐ NGUYÊN

•**Các phép toán**:

-**Phép gán**: gán một hằng số nguyên hoặc một giá trị số nguyên cho biến. Ký hiệu **=**. Hằng số nguyên có thể là số trong hệ bát phân hoặc hệ thập lục phân được bắt đầu tương ứng bằng cách chỉ định với ký hiệu 0, 0x. Nếu không chỉ định, hằng số nguyên là số trong hệ thập phân.

Ví dụ:

```
int a, b, c, d;
a = 10; b = 079; c = 0xFF;
d = c / a;
```

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖KIỂU SỐ NGUYÊN

•**Các phép toán**:

-**Phép gán**: Trong trường hợp muốn chỉ định hằng số có kiểu long int, thêm ký hiệu L ngay sau hằng số.

Ví dụ:

```
long int a, b;
a = 200362985L;
b = 0xFFABL;
```

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖KIỂU SỐ NGUYÊN

•**Các phép toán**:

-**Phép đảo dấu**: áp dụng cho loại có dấu, ký hiệu -. Kết quả của phép đảo dấu của một số nguyên là số đối của nó.

Ví dụ:

```
int a, b, c;
a = 97; b = -5;
c = a +-b;
---------------------
```

Kết quả c = 102

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖KIỂU SỐ NGUYÊN

•**Các phép toán**:

-**Phép chia nguyên**: ký hiệu **/**. Kết quả của phép chia của hai số nguyên với nhau là phần nguyên của phép chia

Ví dụ:

```
int a, b, c;
a = 051; b = 2;
c = a / b;
---------------------
```

Kết quả c = 20

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖KIỂU SỐ NGUYÊN

•**Các phép toán**:

-**Phép chia lấy phần dư**: ký hiệu **%**. Kết quả của phép chia lấy phần dư của hai số nguyên với nhau là phần dư trong phép chia nguyên.

Ví dụ:

```
int a, b, c;
a = 051; b = 2;
c = a % b;
```

Kết quả c = 1

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖KIỂU SỐ NGUYÊN

•**Các phép toán**:

-**Phép cộng, trừ, nhân**: ký hiệu lần lượt là +, -, *.

Ví dụ:

```
int a, b;
a = 12*2 + 4;
b = a / 2;
c = b - 7
```

---

a = 27, b = 13, c = 6.

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖KIỂU SỐ NGUYÊN

•**Các phép toán**:

-**Phép phủ định bit**: ký hiệu ~. Kết quả phép phủ định bit của một số nguyên là một số nguyên có được bằng cách thực hiện phép phủ định trên từng bit của số nguyên đó theo quy tắc 0 tương ứng với sai, 1 tương ứng với đúng.

Ví dụ:

```
char a, b;
a = 97;      // biểu diễn nhị phân     01100001
b = ~a;      // biểu diễn nhị phân     10011110
```

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖KIỂU SỐ NGUYÊN

•**Các phép toán**:

-**Phép and bit**: ký hiệu &. Kết quả phép and bit của hai số nguyên là một số nguyên có được bằng cách thực hiện phép hội trên từng bit của hai số nguyên theo quy tắc 0 tương ứng với sai, 1 tương ứng với đúng.

Ví dụ:

```
char a, b, c;
a = 10;      // biểu diễn nhị phân     00001010
b = 5;       // biểu diễn nhị phân     00000101
c = a & b;   // biểu diễn nhị phân     00000000
```

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖KIỂU SỐ NGUYÊN

•**Các phép toán**:

-**Phép or bit**: ký hiệu |. Kết quả phép or bit của hai số nguyên là một số nguyên có được bằng cách thực hiện phép tuyển trên từng bit của hai số nguyên theo quy tắc 0 tương ứng với sai, 1 tương ứng với đúng.

Ví dụ:

```
char a, b, c;
a = 11;      // biểu diễn nhị phân     00001011
b = 5;       // biểu diễn nhị phân     00000101
c = a | b;   // biểu diễn nhị phân     00001111
```

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖KIỂU SỐ NGUYÊN

•**Các phép toán**:

-**Phép xor bit**: ký hiệu ^. Kết quả phép xor bit của hai số nguyên a và b là ~a&b | a&~b.

Ví dụ:

```
char a, b, c;
a = 97;      // biểu diễn nhị phân     01100001
b = 5;       // biểu diễn nhị phân     00000101
c = a ^ b;   // biểu diễn nhị phân     01100100
```

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖KIỂU SỐ NGUYÊN

•**Các phép toán**:

-**Phép dịch trái bit**: ký hiệu <<. Kết quả phép dịch trái bit của số nguyên a với số nguyên dương b là một số nguyên có được bằng cách dịch chuyển toàn bộ bit của a đi b vị trí bên trái.

Ví dụ:

```
char a, b, c;
a = 97;      // biểu diễn nhị phân     01100001
b = 5;       // biểu diễn nhị phân
c = a << b; // biểu diễn nhị phân     00100000
```

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖KIỂU SỐ NGUYÊN

•**Các phép toán**:

-**Phép dịch phải bit**: ký hiệu >>. Kết quả phép dịch phải bit của số nguyên a với số nguyên dương b là một số nguyên có được bằng cách dịch chuyển toàn bộ bit của a đi b vị trí bên phải.

Ví dụ:

```
char a, b, c;
a = 97;      // biểu diễn nhị phân     01100001
b = 5;       // biểu diễn nhị phân
c = a << b; // biểu diễn nhị phân     00000011
```

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖KIỂU SỐ NGUYÊN

- **Bài tập**:

1) Nhập vào một số nguyên không âm có giá trị không quá 250 và in ra biểu diễn nhị phân của nó.
2) Nhập vào một số là tháng trong năm và in ra tên quý chứa tháng đó. Trong đó, Quý I gồm tháng 1,2,3; Quý II gồm tháng 4,5,6; Quý III gồm tháng 7,8,9; Quý IV gồm tháng 10,11,12.

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖KIỂU SỐ NGUYÊN

1) Chương trình:

```
int a, b0, b1, b2, b3, b4, b5, b6, b7;
cin >> a;
b0 = a & 1; b1 = (a & 2) >> 1; b2 = (a & 4) >> 2;
b3 = (a & 8) >> 3; b4 = (a & 16) >> 4;
b5 = (a & 32) >> 5; b6 = (a & 64) >> 6;
b7 = (a & 128) >> 7;
cout << b7 << b6 << b5 << b4 << b3 << b2 << b1
     << b0 << endl;
```

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖KIỂU SỐ CÓ PHẦN THẬP PHÂN

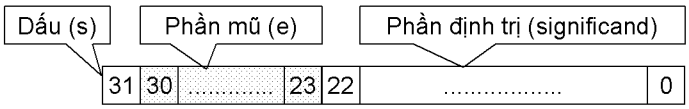
Trong ngôn ngữ C/C++, có các kiểu dữ liệu cho phép lưu trữ và tính toán các giá trị số có phần thập phân:

•**float**: kích thước lưu trữ 4 byte.

-Khai báo: có thể có hoặc không có giá trị khởi tạo

float a=1.0F, b;

-Biểu diễn kiểu float theo hệ nhị phân:



CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖KIỂU SỐ CÓ PHẦN THẬP PHÂN

Công thức tính giá trị số biểu diễn bởi kiểu float:

$$f = (-1)^s x(1+mx2^{-23})x2^{e-127}$$

trong đó m là giá trị của 23 bit định trị

Ví dụ:

cho số float f được lưu trữ với dãy bit:

0-01111101-10100000000000000000000

s = 0,

$m = 1x2^{22} + 0x2^{21} + 1x2^{20} + 0x2^{19} + ... + 0x2^0$

$1 + m.2^{-23} = 1 + 0.5 + 0.125 = 1.625$

$e = 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^0 = 125$

$f = (-1)^0x1.625x2^{-2}=1.625x0.25 = 0.40625$

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖KIỂU SỐ CÓ PHẦN THẬP PHÂN

Để biểu diễn một số có phần thập phân f dưới dạng nhị phân theo kiểu float:

-Đổi phần nguyên a sang nhị phân $a_1a_2...a_n$.

-Đổi phần thập phân b thành dạng nhị phân theo cách:

1) Lần thứ i, i=1..k, nhân b với 2, được một số có phần nguyên là b_i và phần thập phân r_i.

2) Nếu b bằng 0 hoặc n + k = 24 thì dừng. Ngược lại, thực hiện bước 1) với $b = r_i$

3) Dạng nhị phân của b là $b_1b_2..b_k$

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖KIỂU SỐ CÓ PHẦN THẬP PHÂN

-Từ biểu diễn nhị phân $a_1a_2--a_n.b_1b_2--b_k$ chuyển thành dạng $a_1.a_2a_3--a_nb_1b_2--b_kx2^{n-1}$

-Phần định trị là m là $(a_2a_3--a_nb_1b_2--b_k0_{n+k-1}--0_{23})_2$

-Phần mũ là e = 127 + n-1 được biểu diễn trong hệ nhị phân.

-Nếu f < 0 thì s = 1, ngược lại s = 0.

-Kết quả biểu diễn f là: s-e-m.

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖KIỂU SỐ CÓ PHẦN THẬP PHÂN

Ví dụ: biểu diễn số f = 3.25 theo kiểu float

-a = 3, biểu diễn nhị phân là 11, n = 2.

-b = 0.25.

+ lần 1, b = 0.25, b x 2 = 0.5, b_1=0, r_1 =0.5

+ lần 2, b = 0.5, b x 2 = 1.0, b_2 = 1, r_2 = 0, dừng.

-Biểu diễn nhị phân của b là 01

-Biểu diễn nhị phân của 3.25 là 11.01

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖KIỂU SỐ CÓ PHẦN THẬP PHÂN

-Biểu diễn nhị phân của 3.25 là 11.01 = $(1.101)x2^1$

-Phần định trị: m = 101 0000 0000 0000 0000 0000

-Phần mũ e = 127+1 = 128 = 1000 0000

-Phần dấu: f>0, s=0

Vậy biểu diễn kiểu float của 3.25 là

0-1000 0000-101 0000 0000 0000 0000 0000

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖KIỂU SỐ CÓ PHẦN THẬP PHÂN

•**double**: kích thước lưu trữ 8 byte, được lưu trữ theo cấu trúc tương tự kiểu float với 1 bit phần dấu, 11 bit phần mũ và 52 bit phần định trị.
- Khai báo: có thể có hoặc không có giá trị khởi tạo.
double a, b = 20.14;

•**long double**: kích thước lưu trữ 10 byte, được lưu trữ theo cấu trúc tương tự kiểu float với 1 bit phần dấu, 15 bit phần mũ, 64 bit phần định trị.
-Khai báo: có thể có hoặc không có giá trị khởi tạo.
long double a, b = 20.14L;

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖KIỂU SỐ CÓ PHẦN THẬP PHÂN

•**Các phép toán**:
Phép gán: gán một hằng số nguyên, hằng số có phần thập phân, giá trị số nguyên hoặc giá trị số có phần thập phân cho biến, Ký hiệu **=**. Hằng số có phần thập phân phải chứa dấu **.** trong giá trị. Kiểu float hoặc long double phải thêm lần lượt ký hiệu F hoặc L ngay sau giá trị, nếu không có các ký hiệu này, hằng số là kiểu double.

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖KIỂU SỐ CÓ PHẦN THẬP PHÂN

•**Các phép toán**:

-**Phép gán**:

Ví dụ:

```
float f, r;
double d;
int a, b;
a = 5; b = 2;
f = 1.0F;
r = a / b; // kết quả r?
d = 2.0;
```

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖KIỂU SỐ CÓ PHẦN THẬP PHÂN

•**Các phép toán**:

-**Phép đảo dấu**: ký hiệu -. Kết quả của phép đảo dấu là số đối của nó.

-**Phép cộng, trừ, nhân, chia**: ký hiệu lần lượt là +, -, *, /

Ví dụ:

```
float a, b, c;
a = 18.23F; b = -3.14F;
c = a * 13.1 +-b;
----------------------
Kết quả c = 241.953
```

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖PHẠM VI CÁC KIỂU DỮ LIỆU

Trong file **limits.h**

Kiểu		**Nhỏ nhất**	**Lớn nhất**
char	%c	CHAR_MIN	CHAR_MAX
unsigned char	%c	0	UCHAR_MAX
short	%hi	SHRT_MIN	SHRT_MAX
unsigned short	%hu	0	USHRT_MAX
long	%li	LONG_MIN	LONG_MAX
unsigned long	%lu	0	ULONG_MAX
int	%i	INT_MIN	INT_MAX
unsigned int	%u	0	UINT_MAX

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖PHẠM VI CÁC KIỂU DỮ LIỆU

Trong file **float.h**

Kiểu		**Nhỏ nhất**	**Lớn nhất**
float	%f	FLT_MIN	FLT_MAX
double	%lf	DBL_MIN	DBL_MAX
long double	%Lf	LDBL_MIN	LDBL_MAX

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖KIỂU LUẬN LÝ

Trong C không định nghĩa kiểu luận lý, thay vào đó sử dụng số khác 0 làm chân trị đúng và số 0 làm chân trị sai. Trong C++ có định nghĩa kiểu **bool** với các chân trị là **true** và **false** được định nghĩa là những số nguyên 1 và 0.

•**Các phép toán**:

-**Phép phủ định, hội, tuyển**: ký hiệu lần lượt là !, &&, ||.

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖CÁC PHÉP TOÁN KHÁC

•**Phép so sánh bằng, không bằng, nhỏ hơn, nhỏ hơn hoặc bằng, lớn hơn và lớn hơn hoặc bằng**: ký hiệu lần lượt là ==, !=, <, <=, >, >=. Các phép so sánh được thực hiện trên các kiểu dữ liệu cơ sở và trả kết quả là một số nguyên.

Ví dụ:

```
int a, b;
a = 2; b = 3;
a = a > b + a && b – 2;
```

---

kết quả a = 0

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖CÁC PHÉP TOÁN KHÁC

•**Phép gán phức**: sử dụng cho hai số a và b thuộc các kiểu dữ liệu có các phép +, -, *, /, %, >>, <<, &, | ký hiệu lần lượt là +=, -=, *=, /=, %=, >>=, <<=, &=, |=. Kết quả của phép gán phức tương đương với việc gán kết quả của phép toán tương ứng thực hiện trên a và b cho a.

Ví dụ:

```
int a = 3, b = 1; float f = 1.0, r = 1.2;
a += b;   // tương đương với a = a + b;
a <<= b; // tương đương với a = a << b;
f >>= r;   // sai vì trong kiểu float không có phép >>
```

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖CÁC PHÉP TOÁN KHÁC

•**Phép tăng, giảm 1 đơn vị**: áp dụng cho một biến kiểu cơ sở. Kết quả của phép tăng, giảm (ký hiệu lần lượt là **++**, **--**) là giá trị của biến đó tăng hoặc giảm 1 đơn vị thuộc kiểu của biến đó. Có hai kiểu tăng, giảm:

-Tăng, giảm trước: được đặt trước biến, giá trị của biến sẽ thay đổi trước khi thực hiện các phép toán khác trong biểu thức.

-Tăng, giảm sau: được đặt sau biến, giá trị của biến sẽ thay đổi sau khi thực hiện các phép toán khác trong biểu thức.

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖CÁC PHÉP TOÁN KHÁC

Ví dụ:

```
int a = 5, b = 7, c;
c = a++ + b;
----------------------------
```

kết quả: c = 12, a = 6

```
int a = 5, b = 7, c;
c = ++a + b
----------------------------
```

kết quả c = 13, a = 6

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖CÁC PHÉP TOÁN KHÁC

•**Phép nhóm các số hạng**: ký hiệu **(** biểuthức **)**.
Phép nhóm các số hạng để ưu tiên thực hiện biểu thức trong phép nhóm trước.

Ví dụ:

```
int a = 4, b = 3;
a = a > b + a && b + 2; // a = 0
------------------------------
int a = 4, b = 3;
a = (a > b) + (a && b) + 2; // a = 4
```

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖CÁC PHÉP TOÁN KHÁC

•**Phép chuyển kiểu**: ký hiệu **(*kiểu*) *biến*** dùng để chuyển kiểu của biến thành kiểu khác trong phạm vi một biểu thức.

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖CÁC PHÉP TOÁN KHÁC

Ví dụ:

```
int a = 5, b = 2;
float f;
f = a / b;       // f = 2
----------------------------------------
int a = 5, b = 2;
float f;
f = (float) a / b; // f = 2.5
```

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖CÁC PHÉP TOÁN KHÁC

•**Phép chọn lựa**: ký hiệu **(*chântrị*) ? *bt1* : *bt2***.
Phép chọn lựa sẽ thực hiện biểu thức ***bt1*** nếu ***chântrị*** đúng và thực hiện ***bt2*** nếu ***chântrị*** là sai.
Ví dụ:

```
int a = 4, b = 3;
(a + b) ? a = a * 2 : a = a + 1; // a = 8
---------------------------------------
int a = 4, b = -4;
(a + b) ? a = a * 2 : a = a + 1 // a = 5
```

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖CÁC PHÉP TOÁN KHÁC

•**Phép phân tách biểu thức**: ký hiệu **,** dùng để phân tách các biểu thức con trong một biểu thức. Giá trị của biểu thức là giá trị của biểu thức con cuối cùng.
Ví dụ:

```
int a = 4, b = 3;
a = a * 2 + b, b = a + 1, a = b + a;
-------------------------------------------
kết quả a = 23, b = 12.
```

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖CÁC PHÉP TOÁN KHÁC

Ví dụ:

```
int a;
cin >> a;
(a) ? cout << a << " binh phuong la ", a = a * a,
        cout << a << endl : cout << "nhap so khac 0";
```

```
float a, b, c, d;
cin >> a >> b >> c;
(d = b*b, a = 4 * a * c, (d = d - a) >= 0) ?
        cout << "co nghiem" : cout << "vo nghiem";
```

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖CÁC PHÉP TOÁN KHÁC

Bài tập:

Viết chương trình giải phương trình bậc 2 chỉ bằng một câu lệnh C/C++ duy nhất, không tính câu lệnh khai báo biến.

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖CÁC PHÉP TOÁN KHÁC

•**Phép lấy kích thước của đối tượng**: ký hiệu **sizeof(ĐT)**, dùng để xác định kích thước tính theo byte của đối tượng **ĐT**. Đối tượng **ĐT** có thể là biến hoặc kiểu dữ liệu.

Ví dụ:

```
char a = 2;
printf("%d\n", sizeof(a));
printf("%d\n", sizeof(short));
```

1
2

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖THỨ TỰ ƯU TIÊN CÁC PHÉP TOÁN

Toán tử	thứ tự cùng cấp
! ++ -- - + (kiểu) * & sizeof	←
* / %	→
+ -	→
<< >>	→
< <= >= >	→
== !=	→
&	→
|	→
^	→
&&	→
||	→
?:	←
= += -= *= /= %=	←

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖THỨ TỰ ƯU TIÊN CÁC PHÉP TOÁN

Ví dụ

```
int a = 1, b = 2, c = 3;
a += b *= c -= 1;
----------------------------
int a = 1, b = 2, c = 3;
a = b < c >= a;
----------------------------
```

KÝ TỰ VÀ CHUỖI

❖KÝ TỰ

Ký tự là kiểu dùng để lưu trữ và hiển thị các ký hiệu được con người sử dụng làm phương tiện ngôn ngữ. Thực chất, ký tự là một số nguyên có kích thước tùy theo bộ mã ký tự tương ứng. Với bộ mã ASCII gồm 256 ký hiệu, một ký tự có kích thước 1 byte.

Trong C/C++, biến ký tự được lưu trữ theo kiểu char, giá trị của biến kiểu char là mã ASCII của ký tự được lưu.

KÝ TỰ VÀ CHUỖI

❖KÝ TỰ

Ví dụ

```
char x = 65;
cout << "x la ky tu " << x << endl;
x = x + 3;
cout << "sau x 3 ky tu la " << x << endl;
```

---

màn hình kết quả:

x la ky tu A

sau x 3 ky tu la D

KÝ TỰ VÀ CHUỖI

❖KÝ TỰ

Hằng ký tự: được chỉ định bằng một số nguyên thập phân, một số nguyên bát phân bắt đầu bằng dấu \ được đặt trong cặp dấu ' hoặc một ký hiệu được đặt trong cặp dấu '

Ví dụ:

```
char a, b, c;
a = 65; b = 'A'; c = '\101';
cout << (a==b) << " " << (b == c) << endl;
```

---

màn hình kết quả: 1 1

KÝ TỰ VÀ CHUỖI

❖KÝ TỰ

Một số ký tự đặc biệt:

-'\t': di chuyển con trỏ màn hình qua phải một cột

-'\n': di chuyển con trỏ xuống dòng

-'\r': đưa con trỏ về đầu dòng

-'\'': ký tự '

-'\"': ký tự "

-'\\': ký tự \

KÝ TỰ VÀ CHUỖI

❖KÝ TỰ

Lưu ý: Không thể nhập trực tiếp số nguyên cho kiểu char vì C/C++ đã xử lý xem các ký số là từng ký tự nếu đối số của hàm scanf hoặc toán tử >> là kiểu char.

KÝ TỰ VÀ CHUỖI

❖CHUỖI

- Chuỗi là một loạt ký tự dùng để biểu diễn ngôn ngữ. Trong C/C++, chuỗi là một mảng các ký tự được đánh dấu kết thúc bằng ký tự có mã là 0.
- Hằng chuỗi được chỉ định bằng tập các ký tự được đặt trong cặp dấu "".

KÝ TỰ VÀ CHUỖI

❖CHUỖI

-Chuỗi được xử lý theo từng ký tự của nó. Để truy xuất đến ký tự thứ i của chuỗi s, dùng biểu thức s[i].

Ví dụ:

```
char hotenA[30] = "Nguyen Van A";
char hotenB[30] = {'N','g','u','y','e','n',' ','V','a','n',' ','B','\0'};
cout << hotenB[4]; // màn hình xuất hiện chữ e
```

KÝ TỰ VÀ CHUỖI

❖CHUỖI

Ví dụ:

printf("Cau 1\nCau 2\nCau 3");

---

màn hình kết quả

Cau 1

Cau 2

Cau 3

KÝ TỰ VÀ CHUỖI

❖CHUỖI

Lưu ý:

-Chuỗi là một mảng các phần tử kiểu char, nên không có phép gán trực tiếp nội dung của 2 chuỗi mà chỉ có thể khởi tạo giá trị ban đầu.

Ví dụ:

char hoten[30];

hoten = "Nguyen Van A";

câu lệnh gán trên không có nghĩa gán nội dung "Nguyen Van A" cho nội dung biến hoten.

KÝ TỰ VÀ CHUỖI

❖CHUỖI

Lưu ý:

- Các hàm nhập scanf và >> trong C/C++ chỉ cho phép nhập giá trị các chuỗi không chứa khoảng trắng.

CÁC HÀM THÔNG DỤNG

❖HÀM CHUYỂN ĐỔI DỮ LIỆU

Được khai báo trong file stdlib.h

-double atof (const char *str): chuyển chuỗi ký tự thành số double. Ví dụ:

```
double d;
d = atof("10.8");
```

-int atoi (const char *str): chuyển chuỗi ký tự thành số int. Ví dụ:

```
int i;
i = atoi("2");
```

CÁC HÀM THÔNG DỤNG

❖HÀM CHUYỂN ĐỔI DỮ LIỆU

-long atol (const char *str): chuyển chuỗi ký tự thành số long int. Ví dụ:

long l;

l = atol("2000000");

CÁC HÀM THÔNG DỤNG

❖HÀM TOÁN HỌC

Được khai báo trong file stdlib.h

-int abs (int n): lấy giá trị tuyệt đối của số int n.

-long labs (long n): lấy giá trị tuyệt đối của số long n.

Được khai báo trong file math.h

- abs(n): lấy giá trị tuyệt đối của số n, n có thể thuộc kiểu float, double, long double.

-double cos(double x): tính cos(x), x tính theo RAD.

-double sin(double x): tính sin(x), x tính theo RAD.

-double tan(double x): tính tan(x), x tính theo RAD.

CÁC HÀM THÔNG DỤNG

❖HÀM TOÁN HỌC

-double acos(double x): tính arccos(x).

-double asin(double x): tính arcsin(x).

-double atan(double x): tính arctan(x).

-double ceil(double x): lấy số nguyên nhỏ nhất lớn hơn x.

-double floor(double x): lấy số nguyên lớn nhất nhỏ hơn x.

-double log(double x): lấy logarit cơ số e của x.

-double log10(double x): lấy logarit cơ số 10 của x.

CÁC HÀM THÔNG DỤNG

❖HÀM TOÁN HỌC

-double exp(double x): tính e mũ x.

-double pow(double a, double x): tính a mũ x.

-double sqrt(double x): tính căn bậc 2 của x.

-long int lround(double x): làm tròn x tới 0.5.

CÁC HÀM THÔNG DỤNG

❖HÀM XỬ LÝ CHUỖI

Được khai báo trong file string.h

-char* strcpy(char* dst, const char* src): copy nội dung chuỗi src vào chuỗi dst. Ví dụ:

char hoten[30];

strcpy(hoten, "Nguyen Van A");

-char* strncpy(char* dst, const char* src, size_t n): copy n ký tự đầu tiên của chuỗi src vào chuỗi dst.

CÁC HÀM THÔNG DỤNG

❖HÀM XỬ LÝ CHUỖI

-char* strcat(char* dst, const char* src): nối nội dung chuỗi src vào ngay sau chuỗi dst. Ví dụ:

char hoten[30] = "Nguyen";

strcat(hoten, " Van A");

-int strcmp(char* dst, const char* src): so sánh mã của từng cặp ký tự trong chuỗi dst và src theo thứ tự từ trái qua phải. Nếu kết quả là 0 thì dst = src, -1 nếu trong cặp ký tự đầu tiên khác, ký tự của dst nhỏ hơn src và 1 trong trường hợp ngược lại.

CÁC HÀM THÔNG DỤNG

❖HÀM NHẬP CHUỖI

Được khai báo trong stdlib.h

-char* gets(char* s): nhập một chuỗi chứa khoảng trắng từ bàn phím cho đến khi gặp ký tự xuống dòng, không đưa ký tự xuống dòng vào s.

CÁC KIỂU DỮ LIỆU CƠ SỞ VÀ PHÉP TOÁN

❖BÀI TẬP

1. Viết chương trình nhập vào tọa độ tâm và bán kính của 2 hình tròn, cho biết tình trạng giao nhau của chúng, gồm: tiếp xúc, cắt và tách rời.
2. Viết chương trình nhập vào tọa độ của một tứ giác và 1 điểm, cho biết vị trí tương quan giữa điểm đó và tứ giác, gồm: nằm trong, nằm ngoài, nằm trên cạnh.
3. Viết chương trình giải hệ phương trình bậc nhất 2 ẩn số.
4. Viết chương trình tính diện tích của tam giác khi biết hai cạnh và số đo góc xen giữa hai cạnh đó.