

Ngôn ngữ lập trình C


Bài 3. Kiểu số & Biểu thức

Soạn bởi: TS. Nguyễn Bá Ngọc

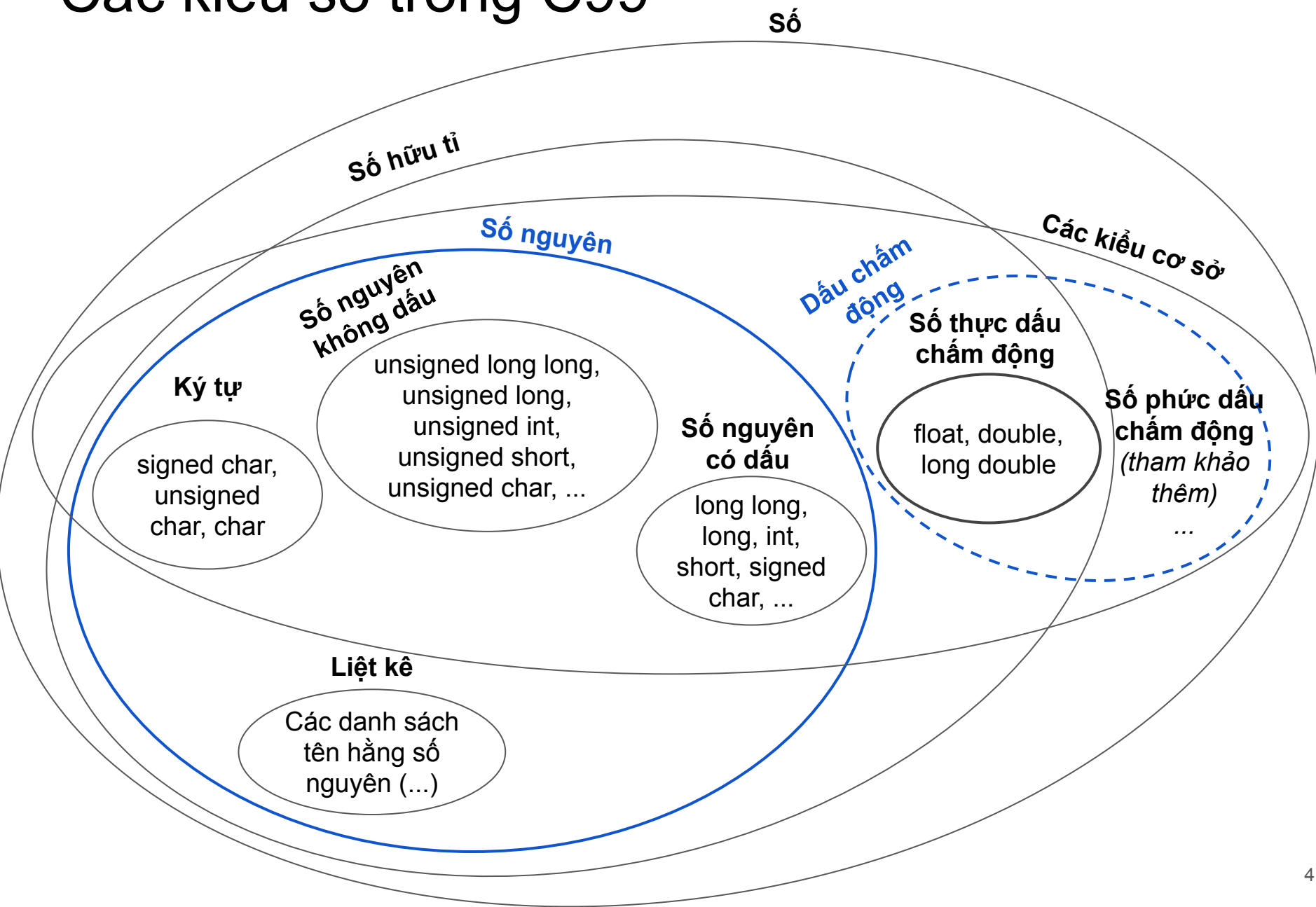
Nội dung

- Kiến thức đã học
- Hằng giá trị
- Kiểu số và các toán tử
- Kiểu số và ép kiểu

Nội dung

- 
- Kiến thức đã học
 - Hằng giá trị
 - Kiểu số và các toán tử
 - Kiểu số và ép kiểu

Các kiểu số trong C99



Kiểu số nguyên

Triển khai thường gặp

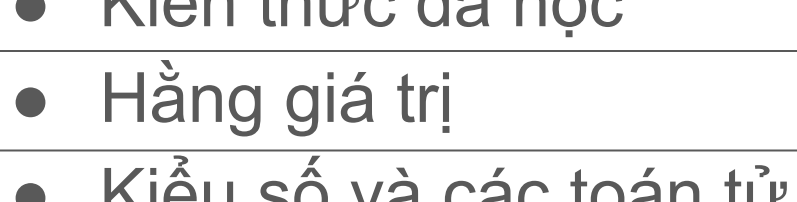
Kiểu	#bits	Min	Max
Có dấu (thường là mã bù-2)			
signed char	8	-128	127
short	16	-32768	32767
int	32	-2147483648	2147483647
long	64	-9223372036854775808	9223372036854775807
long long	64		
Không dấu (thuần nhị phân)			
unsigned char	8	0	255
unsigned short	16		65535
unsigned	32		4294967295
unsigned long	64		18446744073709551615
unsigned long long	64		

Kiểu số thực

Triển khai thường gặp

Kiểu	#bits	Bố cục(s-w-t)	#chữ số (~)
float	32	1-8-23	6
double	64	1-11-52	15
long double	80	1-15-64	18

Nội dung

- Kiến thức đã học
 - Hằng giá trị
 - Kiểu số và các toán tử
 - Kiểu số và ép kiểu
- 

Sơ lược về hằng số nguyên

- Hằng số nguyên bắt đầu bằng chữ số và có thể được viết ở HCS 10, 8, hoặc 16:
 - Hằng HCS 10 bắt đầu với chữ số $\neq 0$
 - Hằng HCS 8 bắt đầu với chữ số 0
 - Hằng HCS 16 bắt đầu với 0x hoặc 0X
- Hằng số nguyên có thể có hậu tố để mô tả kiểu:
 - u hoặc U \Rightarrow có kiểu thuộc nhóm unsigned/không dấu
 - l hoặc L \Rightarrow có kiểu thuộc nhóm long
 - Có thể kết hợp với u hoặc U
 - ll hoặc LL \Rightarrow có kiểu thuộc nhóm long long
 - Có thể kết hợp với u hoặc U
- Kiểu của hằng số nguyên được xác định là kiểu đầu tiên trong danh sách tương ứng với hậu tố và HCS, và có thể biểu diễn được giá trị của nó.

Kiểu của hằng số nguyên

Hậu tố	HCS 10	HCS 8 hoặc 16
không	int long long long	int unsigned int long int unsigned long long long unsigned long long
l hoặc L	long long long	long unsigned long long long unsigned long long
ll hoặc LL	long long	long long unsigned long long
u hoặc U	unsigned int unsigned long unsigned long long	
Kết hợp u hoặc U với l hoặc L	unsigned long unsigned long long	
Kết hợp u hoặc U với ll hoặc LL	unsigned long long	

Ví dụ 3.1. Hằng số nguyên

Hằng số nguyên	Kiểu của hằng
123 hoặc 0173 hoặc 0x7B	int
2147483648 (giả sử int - 32 bits)	long
0x80000000 (int - 32 bits)	unsigned int
123u hoặc 0173u hoặc 0x7Bu	unsigned int
123l hoặc 0173l hoặc 0x7Bl	long
123lu hoặc 0173ul hoặc 0x7BuL	unsigned long
123ll hoặc 0173ll hoặc 0x7BLL	long long
123llu hoặc 0173ull hoặc 0x7BuLL	unsigned long long

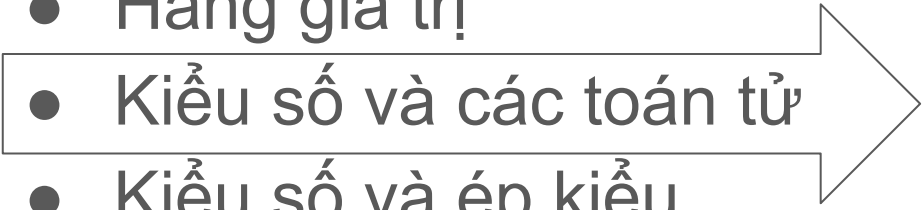
Sơ lược về hằng số thực

- Hằng số thực bao gồm:
 - Phần định trị
 - Bao gồm phần nguyên và/hoặc phần thập phân được ngăn cách bởi dấu chấm (.).
 - Có thể được biểu diễn ở HCS 10 hoặc HCS 16.
 - Phần lũy thừa
 - e hoặc E theo sau là giá trị phần mũ cho cơ số 10 (HCS 10).
 - p hoặc P theo sau là giá trị phần mũ cho cơ số 2 (HCS 16).
 - Hằng số thực ở HCS 10 cần phải có dấu (.) hoặc phần lũy thừa (phân biệt với hằng số nguyên).
 - Hằng số thực ở HCS 16 bắt buộc phải có phần lũy thừa (ký tự p, giúp phân biệt hậu tố F với chữ số F).
 - Ký tự hậu tố (không bắt buộc) để xác định kiểu dữ liệu
 - f hoặc F cho kiểu float
 - l hoặc L cho kiểu long double
 - Nếu không có ký tự hậu tố thì kiểu là double

Ví dụ 3.2. Hằng số thực

Hằng số thực	Kiểu của hằng
3.1415	double
.123	double
3.	double
3.1415f	float
1e-9L	long double
0x1.1p-3f	float
0x1.1fp-3f	float
1e10	double
0x0.0Fp0f	float

Nội dung

- Kiến thức đã học
 - Hằng giá trị
 - Kiểu số và các toán tử
 - Kiểu số và ép kiểu
- 

Tổng quan về biểu thức trong C

- Trong C định danh đối tượng, hằng giá trị là những biểu thức cơ bản.
- Các biểu thức có thể được kết hợp bằng các toán tử tạo thành các biểu thức lớn hơn.
- Biểu thức thu được bằng cách đặt 1 biểu thức trong cặp dấu () cũng là 1 biểu thức cơ bản
 - Biểu thức dạng (E) có ý nghĩa giống như biểu thức E.
 - Có thể được sử dụng như toán hạng của các toán tử.
- Khác với khái niệm toán tử trong toán học, toán tử trong C có thể có hiệu ứng phụ (ví dụ làm thay đổi giá trị của biến).
 - Chỉ thay đổi giá trị của 1 biến tối đa 1 lần trong 1 biểu thức.
 - *(Tham khảo thêm về đơn vị tuần tự/sequence point).*
 - Trong chương này chúng ta sẽ học một số toán tử thông dụng cho kiểu số. Các trường hợp khác sẽ được khám phá dần theo tiến trình học.

Các toán tử cộng, trừ

- Định dạng:
 - $a + b$
 - $a - b$
- Điều kiện:
 - Các toán hạng có kiểu số
 - *(trường hợp 1 toán hạng có kiểu con trỏ sẽ được học sau)*
- Ý nghĩa:
 - $a + b$ cho kết quả là tổng của a và b ;
 - $a - b$ cho kết quả là hiệu a trừ b
 - Quy tắc ép kiểu số được áp dụng
- Ví dụ:
 - $1 + 2.0$ có kết quả $= 3.0$ và có kiểu `double`
 - $3 - 1$ có kết quả $= 2$ và có kiểu `int`

Các toán tử dấu

- Định dạng:
 - $+a$
 - $-a$
- Điều kiện:
 - Toán hạng có kiểu số
- Ý nghĩa:
 - $+a$ cho kết quả là giá trị của toán hạng
 - $-a$ cho kết quả là số đối của toán hạng
 - Trong trường hợp toán hạng là số nguyên có thể là số bù-2
 - Quy tắc nâng hạng số nguyên được áp dụng
- Ví dụ:
 - `unsigned char ch = 10;`
 - $+ch$ cho kết quả = 10 và có kiểu `int`
 - $-ch$ cho kết quả = -10 và có kiểu `int`

Các toán tử dấu là các toán tử 1 ngôi dạng tiền tố.

Toán tử nhân

- Định dạng:
 - $a * b$
- Điều kiện:
 - Các toán hạng có kiểu số
- Ý nghĩa:
 - $a * b$ cho kết quả là tích của a và b
 - Quy tắc ép kiểu số được áp dụng.
- Ví dụ:
 - $1.5 * 2$ cho kết quả $= 3.0$ và có kiểu `double`

Toán tử chia

- Định dạng:
 - a / b
- Điều kiện:
 - Các toán hạng có kiểu số
- Ý nghĩa:
 - a / b :
 - thương chia a cho b nếu có ít nhất 1 toán hạng là số thực;
 - phần nguyên trong phép chia nếu ngược lại (a, b là các số nguyên);
 - hành vi bất định nếu $b == 0$
 - Quy tắc ép kiểu số được áp dụng.
- Ví dụ:
 - $3.0 / 2$ cho kết quả $= 1.5$ và có kiểu `double`
 - $3 / 2$ cho kết quả $= 1$ và có kiểu `int`
 - $3 / 0$ là hành vi bất định

Toán tử phần dư

- Định dạng:
 - $a \% b$
- Điều kiện:
 - a và b có kiểu số nguyên
- Ý nghĩa:
 - $a \% b$:
 - phần dư trong phép chia a cho b nếu $b \neq 0$;
 - hành vi bất định nếu ngược lại ($b == 0$).
 - Quy tắc ép kiểu số được áp dụng
- Ví dụ:
 - $7 \% 5$ cho kết quả $= 2$ và có kiểu `int`
 - $7 \% 0$ là hành vi bất định
 - $3.5 \% 2$ là biểu thức không hợp lệ

Phần nguyên và phần dư trong phép chia

Với a, b là các số nguyên trong đó $b \neq 0$ chúng ta có:

$$a = (a / b) * b + a \% b$$

- Trong C89 kết quả phép chia với 2 toán hạng số nguyên trong đó có số âm phụ thuộc vào triển khai (có thể được làm tròn lên hoặc làm tròn xuống).
 - $7 / (-3)$ có thể cho kết quả -2 hoặc -3 tùy theo triển khai.
 - *(Phản ánh đúng hiện trạng triển khai của phần cứng.)*
- Trong C99 kết quả phép chia được quy ước làm tròn về phía 0 (là hành vi xác định), ví dụ:
 - $7/(-3) \Rightarrow -2$ kiểu int, và $7 \% (-3) \Rightarrow 1$ kiểu int
 - $(-7)/3 \Rightarrow -2$ kiểu int, và $(-7) \% 3 \Rightarrow -1$ kiểu int.

Toán tử dịch trái

- Định dạng:
 - $a \ll b$
- Điều kiện:
 - Các toán hạng có kiểu số nguyên.
- Ý nghĩa:
 - Đặt n là số lượng bits trong biểu diễn giá trị của a
 - Nếu $0 \leq b$ và $b < n$ thì $a \ll b$ cho kết quả:
 - $(a * 2^b) \% 2^n$ nếu a có kiểu không dấu;
 - $a * 2^b$ nếu a có kiểu có dấu và $a \geq 0$ và $a * 2^b < 2^{n-1}$;
 - hành vi bất định nếu ngược lại.
 - Nếu ngược lại ($b < 0$ hoặc $b \geq n$) thì hành vi là bất định.
 - Quy tắc nâng hạng số nguyên được áp dụng
- Ví dụ:
 - $1 \ll 3$ cho kết quả = 8 và có kiểu int
 - $1 \ll -1$ là hành vi bất định

Toán tử dịch phải

- Định dạng:
 - $a \gg b$
- Điều kiện:
 - Các toán hạng có kiểu số nguyên.
- Ý nghĩa:
 - Đặt n là số lượng bits trong biểu diễn giá trị của a
 - Nếu $0 \leq b$ và $b < n$ thì $a \gg b$ cho kết quả:
 - $a / 2^b$ nếu a có kiểu không dấu;
 - $a / 2^b$ nếu a có kiểu có dấu và $a \geq 0$;
 - hành vi phụ thuộc triển khai nếu ngược lại.
 - Nếu ngược lại ($b < 0$ hoặc $b \geq n$) thì hành vi là bất định.
 - Quy tắc nâng hạng số nguyên được áp dụng
- Ví dụ:
 - $9 \gg 2$ cho kết quả $= 2$ và có kiểu `int`
 - $-9 \gg 1$ cho kết quả phụ thuộc vào triển khai

Các toán tử AND, OR, và XOR trên dãy bits

- Định dạng:
 - $a \& b$
 - $a | b$
 - $a ^ b$
- Điều kiện:
 - Các toán hạng có kiểu số nguyên
- Ý nghĩa:
 - $a \& b$ cho kết quả là giá trị mà mỗi bit trong biểu diễn = 1 nếu các bits ở vị trí tương ứng của các toán hạng = 1
 - $a | b$ - mỗi bit = 1 nếu có ít nhất 1 bit ở vị trí tương ứng = 1
 - $a ^ b$ - mỗi bit = 1 nếu các bits tương ứng khác nhau
 - Quy tắc ép kiểu số được áp dụng
- Ví dụ:
 - *(biểu diễn số nguyên trong chương 2)*

Toán tử đảo dãy bits (\sim)

- Định dạng:
 - $\sim a$
- Điều kiện:
 - a có kiểu số nguyên
- Ý nghĩa:
 - $\sim a$ cho kết quả là giá trị với mỗi bit = nghịch đảo của bit tương ứng trong a
 - $\sim a$ cho kết quả $= 2^n - 1 - a$ nếu kiểu của a sau khi nâng hạng là kiểu số nguyên không dấu.
 - Quy tắc nâng hạng số nguyên được áp dụng
- Ví dụ:
 - *(biểu diễn số nguyên trong chương 2)*

Toán tử đảo dãy bits là toán tử 1 ngôi dạng tiền tố.

Lưu ý chung đối với các toán tử trên dãy bits

Áp dụng cho: \gg , \ll , $\&$, $|$, \wedge , \sim .

Chỉ nên áp dụng các toán tử trên dãy bits với toán hạng có kiểu không dấu.

Trong trường hợp phải sử dụng kiểu có dấu thì chỉ nên sử dụng các giá trị không âm. Không nên áp dụng các toán tử trên dãy bits với các toán hạng có giá trị âm do nhiều hành vi bất định và hành vi phụ thuộc triển khai liên quan.

Quy chuẩn C không quy ước phương pháp biểu diễn số nguyên có dấu (tuy hầu hết các trình biên dịch hiện đang sử dụng mã bù-2).

Toán tử gán

- Định dạng:
 - $a = b$
- Điều kiện:
 - a phải là biến có thể thay đổi giá trị hoặc đối tượng tương tự.
 - a là lvalue
 - Có thể ép kiểu b thành kiểu của a ;
- Ý nghĩa:
 - $a = b$ ép kiểu b thành kiểu của a , sau đó lưu kết quả ép kiểu b vào vùng nhớ của a , kết quả là giá trị của a sau cập nhật.
 - Nếu vùng nhớ của a và b chồng lấn thì hành vi là bất định.
- Ví dụ:
 - `int a;`
 - `a = 101;` // Giá trị của `a` == 101, kết quả biểu thức là 101
 - `3 = 5;` // Không hợp lệ

Các toán tử gán kết hợp

- Định dạng:
 - $a \text{ op} = b$ với $op \in \{*, /, \%, +, -, <<, >>, \&, ^, |\}$
- Điều kiện:
 - Các yêu cầu cho toán tử $=$ và toán tử op được đáp ứng.
- Ý nghĩa:
 - Tương đương $a = a \text{ op } b$ nhưng a chỉ được tính 1 lần.
- Ví dụ:
 - $a += 5; \text{ // } a = a + 5;$
 - $a -= 3; \text{ // } a = a - 3;$
 - $a *= 10; \text{ // } a = a * 10;$
 - $a /= 2.0; \text{ // } a = a / 2.0;$
 - $a \% = 10; \text{ // } a = a \% 10;$
 - $a << = 1; \text{ // } a = a << 3;$
 - ...

Các toán tử tăng và giảm 1 tiền tố

- Định dạng:
 - `++a`
 - `--a`
- Điều kiện:
 - Toán hạng là lvalue và có kiểu số hữu tỉ
- Ý nghĩa:
 - `++a`: giống như `(a += 1)`, cộng 1 vào a sau đó cho kết quả là giá trị của a.
 - `--a`: giống như `(a -= 1)`, trừ 1 từ a sau đó cho kết quả là giá trị của a.
- Ví dụ:
 - với `int a = 10;`
 - `++a` cho kết quả = 11 và sau khi thực hiện biểu thức giá trị `a == 11`
 - `--a` cho kết quả = 9 và sau khi thực hiện biểu thức giá trị `a == 9`

Các toán tử tăng và giảm 1 hậu tố

- Định dạng:
 - `a++`
 - `a--`
- Điều kiện:
 - Toán hạng là lvalue và có kiểu số hữu tỉ
- Ý nghĩa:
 - `a++`: giống như (`tmp = a, a += 1, tmp`), cho kết quả là giá trị của `a` sau đó cộng 1 vào `a`
 - `a--`: giống như (`tmp = a, x -= 1, tmp`), cho kết quả là giá trị của `a` sau đó trừ 1 từ `a`
- Ví dụ:
 - với `int a = 10;`
 - `a++` cho kết quả = 10 và sau khi thực hiện biểu thức giá trị `a == 11`
 - `a--` cho kết quả = 10 và sau khi thực hiện biểu thức giá trị `a == 9`

Các toán tử tăng/giảm 1 là các toán tử 1 ngôi.

Toán tử sizeof

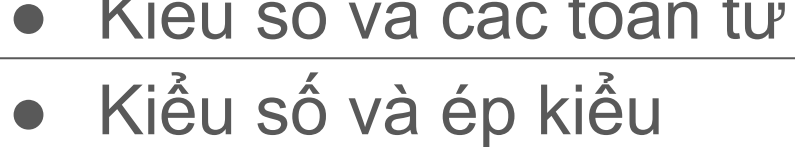
- Định dạng:
 - sizeof a
 - sizeof(a)
- Điều kiện:
 - Toán hạng là biểu thức hoặc
 - kiểu dữ liệu trong cặp dấu ()
- Ý nghĩa:
 - sizeof(a) cho kết quả là kích thước của kiểu của a tính bằng bytes và có kiểu size_t
 - size_t là 1 kiểu số nguyên không dấu được định nghĩa trong stddef.h
- Ví dụ:
 - sizeof(char) cho kết quả = 1
 - sizeof 10 cho kết quả = 4
 - = sizeof(int)

sizeof là toán tử 1 ngôi dạng tiền tố.

Quy tắc tính biểu thức phức tạp

- Trong biểu thức phức tạp, các biểu thức thành phần được tạo thành từ các toán tử có độ ưu tiên cao hơn được coi như các toán hạng của các biểu thức thành phần được tạo thành từ các toán hạng có độ ưu tiên thấp hơn.
 - Các toán tử có độ ưu tiên cao hơn được thực hiện trước, các toán tử có độ ưu tiên thấp hơn được thực hiện sau.
- Các toán tử có cùng mức ưu tiên có thể tạo thành chuỗi toán tử và được thực hiện theo thứ tự được quy định riêng cho từng toán tử.
- Thứ tự thực hiện các toán hạng chỉ được quy định cho các toán tử: gọi hàm (), &&, ||, ?:, và toán tử , (*sẽ học sau*).
Thứ tự thực hiện các toán hạng cho các toán tử còn lại phụ thuộc vào triển khai.

Nội dung

- Kiến thức đã học
 - Hằng giá trị
 - Kiểu số và các toán tử
 - Kiểu số và ép kiểu
- 

Sơ lược vấn đề ép kiểu

- C là một NNLT *định kiểu lỏng*:
 - Có thể kết hợp các toán hạng khác kiểu (ví dụ số nguyên và số thực) trong 1 biểu thức tính toán. v.v..
 - Tuy nhiên máy tính thường chỉ thực hiện các phép toán với các đối số cùng kiểu...
- Để hỗ trợ *định kiểu lỏng* trình biên dịch tự động tạo thêm các lệnh *ép kiểu* (chuyển đổi giá trị sang kiểu khác) khi cần
 - Ép kiểu tự động được thực hiện theo quy tắc được mô tả trong quy chuẩn ngôn ngữ
 - Giúp lập trình đơn giản hơn nhưng trong 1 số trường hợp có thể tạo ra hành vi ngoài ý muốn.
- Người lập trình cũng có thể (chủ động) ép kiểu khi cần
 - Cú pháp: (Kiểu đích) đối-tượng
 - Ví dụ: (double)10 - ép kiểu giá trị 10 thành double

!Lưu ý: Giá trị sau khi ép kiểu có thể khác giá trị ban đầu

Vấn đề ép kiểu tự động

Ép kiểu có thể được thực hiện tự động trong một số trường hợp tiêu biểu sau:

- Trong biểu thức đại số hoặc lô-gic (đối với các toán hạng):
 - Quy tắc ép kiểu số/Usual arithmetic conversions được áp dụng cho phần lớn trường hợp tính toán với 2 toán hạng.
 - Quy tắc nâng hạng số nguyên/integer promotions được áp dụng cho toán hạng kiểu số nguyên với hạng nhỏ hơn hạng của kiểu int.
- Trong phép gán.
- Khi gọi hàm.
- Khi trả về giá trị cho hàm.

Các nội dung chi tiết về hàm sẽ được cung cấp sau.

Quy tắc ép kiểu số

Thường được áp dụng trong phép toán 2 toán hạng để xác định kiểu chung cho các toán hạng và kết quả:

- Nếu có 1 toán hạng có kiểu long double thì toán hạng còn lại được ép kiểu long double.
- Trong trường hợp ngược lại, nếu có 1 toán hạng có kiểu double thì toán hạng còn lại được ép kiểu double.
- Trong trường hợp ngược lại, nếu có 1 toán hạng có kiểu float thì toán hạng còn lại được ép kiểu float.
- Trong trường hợp ngược lại (cả 2 toán hạng đều có kiểu số nguyên), thì thực hiện nâng hạng số nguyên cho cả 2 toán hạng và sau đó áp dụng quy tắc ép kiểu số nguyên.

Quy tắc ép kiểu số nguyên được thiết lập dựa trên khái niệm hạng của kiểu số nguyên

Hạng của kiểu số nguyên

- Với T là kiểu số nguyên chúng ta ký hiệu $\text{rank}(T)$ là hạng của T ; $\text{len}(T)$ là số bits trong biểu diễn của kiểu T .
- Quy chuẩn C quy định:
 - Với $T1$ và $T2$ là 2 kiểu số nguyên:
 - Nếu $T1$ và $T2$ đều có dấu và $T1 \neq T2$ thì $\text{rank}(T1) \neq \text{rank}(T2)$
 - Nếu $\text{len}(T1) < \text{len}(T2)$ thì $\text{rank}(T1) < \text{rank}(T2)$
 - ...!Nhưng có khác biệt trong chiều ngược lại, nếu $\text{rank}(T1) < \text{rank}(T2)$ thì $\text{len}(T1) \leq \text{len}(T2)$.
 - Với S là kiểu số nguyên có dấu và U là kiểu không dấu tương ứng của S (ví dụ, `int` và `unsigned int`):
 - $\text{rank}(S) = \text{rank}(U)$
 - Quan hệ thứ hạng có tính chất bắc cầu. Với 3 kiểu số nguyên $T1$, $T2$ và $T3$ bất kỳ
 - Nếu $\text{rank}(T1) < \text{rank}(T2)$ và $\text{rank}(T2) < \text{rank}(T3)$ thì $\text{rank}(T1) < \text{rank}(T3)$.

Hạng của kiểu số nguyên₍₂₎

- Với T1 và T2 là 2 kiểu số nguyên cùng hình thức biểu diễn (2 kiểu số nguyên có dấu hoặc 2 kiểu số nguyên không dấu) và nếu $\text{rank}(T1) < \text{rank}(T2)$ thì tập giá trị của T1 là tập con của tập giá trị của T2.
- Với các kiểu số nguyên trong quy chuẩn chúng ta có:
 $\text{rank}(_Bool) < \text{rank}(\text{signed char}) < \text{rank}(\text{short}) < \text{rank}(\text{int}) < \text{rank}(\text{long}) < \text{rank}(\text{long long})$
 $\text{rank}(\text{signed char}) == \text{rank}(\text{unsigned char})$
 $\text{rank}(\text{short}) == \text{rank}(\text{unsigned short})$
 $\text{rank}(\text{int}) == \text{rank}(\text{unsigned int})$
...

Quy tắc nâng hạng số nguyên

- Trong biểu thức đại số với các toán hạng kiểu số nguyên có hạng $\leq \text{rank}(\text{int})$:
 - Nếu int có thể biểu diễn được tất cả các giá trị của các kiểu ban đầu của toán hạng thì toán hạng được ép kiểu int,
 - nếu ngược lại thì ép kiểu unsigned int.

Ví dụ:

```
char c;  
short s;  
unsigned u;  
c + s; // c và s được ép kiểu int  
c + u; // c được ép kiểu unsigned int  
s + s; // s được ép kiểu int.
```

Quy tắc ép kiểu số nguyên

Trong phép toán với 2 kiểu số nguyên khác nhau:

- Nếu 2 kiểu có cùng hình thức biểu diễn, thì toán hạng có kiểu với hạng thấp hơn được chuyển đổi thành kiểu với hạng cao hơn.
- Trong trường hợp ngược lại, nếu kiểu không dấu có hạng \geq hạng của kiểu có dấu, thì toán hạng thuộc kiểu có dấu được chuyển đổi sang kiểu không dấu.
- Trong trường hợp ngược lại, nếu kiểu có dấu có thể biểu diễn được tất cả các giá trị thuộc kiểu của toán hạng có kiểu không dấu, thì toán hạng thuộc kiểu không dấu được chuyển đổi sang kiểu có dấu.
- Trong trường hợp ngược lại, cả 2 toán hạng được chuyển đổi sang kiểu không dấu tương ứng với kiểu có dấu.

Ví dụ 3.3. Ép kiểu số thực

Chúng ta xét một số trường hợp sau:

```
float f;  
double lf;  
long double llf;  
long int li;  
llf + lf; // chuyển kiểu lf thành long double  
lf + f; // chuyển kiểu f thành double  
f + li; // chuyển kiểu li thành float
```

Trình biên dịch có xu hướng chọn kiểu gần nhất có thể biểu diễn cả 2 toán hạng.

Ví dụ 3.4. Ép kiểu số nguyên

Chúng ta xét một số trường hợp sau:

```
int i;  
unsigned int u;  
long li;  
unsigned long uli;  
long long lli;  
li + lli; // li được chuyển thành kiểu long long.  
i < u; // i được chuyển thành kiểu unsigned int  
u + li; // u => long (giả sử len(int) < len(long))  
uli + lli; // uli và lli => unsigned long long
```

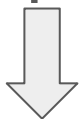
Giả sử $\text{len}(\text{long}) \stackrel{\uparrow}{=} \text{len}(\text{long long})$

Trình biên dịch có xu hướng chọn kiểu gần nhất có thể biểu diễn được cả 2 toán hạng.

Giá trị thu được sau khi ép kiểu

Có tương đối nhiều trường hợp khác nhau, vì vậy tìm hiểu dần theo thời gian có thể phù hợp hơn liệt kê một lượt tất cả các trường hợp với đầy đủ các quy tắc. Trong bài giảng này chúng ta xét một số trường hợp cơ bản:

- Khi chuyển đổi một giá trị đơn (scalar value) thành giá trị kiểu `_Bool`, kết quả = 0 nếu giá trị ban đầu bằng 0, kết quả = 1 nếu ngược lại - giá trị ban đầu khác 0.
- Khi chuyển đổi một giá trị thuộc kiểu số nguyên hoặc số thực sang kiểu số nguyên (khác kiểu `_Bool`) hoặc số thực khác: Nếu giá trị ban đầu có thể được biểu diễn chính xác bằng kiểu đích thì nó được giữ nguyên.



Kịch bản phổ biến khi tính toán, có thể yêu cầu các xử lý (ví dụ khi chuyển đổi giữa số nguyên và số thực)

Chuyển đổi kiểu₍₂₎

- Nếu không thể biểu diễn chính xác giá trị ban đầu bằng kiểu đích thì có thể có nhiều kịch bản khác nhau:
 - Số nguyên => số nguyên không dấu n bits: Giá trị mới bằng phần dư của giá trị ban đầu khi chia cho 2^n (*= lặp cộng hoặc trừ với 2^n cho tới khi thu được giá trị trong phạm vi*).
 - Số nguyên => số nguyên có dấu: Hành vi không xác định.
 - Số thực hữu hạn => số nguyên: Nếu phần nguyên của số thực có thể được biểu diễn chính xác bởi kiểu đích thì giá trị mới bằng phần nguyên của giá trị ban đầu (phần thập phân bị lược bỏ), nếu ngược lại thì hành vi là không xác định.
 - Số nguyên hoặc số thực => số thực: Nếu có thể biểu diễn (gần đúng) giá trị ban đầu bằng kiểu đích thì giá trị mới bằng giá trị hữu hạn gần nhất lớn hơn hoặc nhỏ hơn giá trị ban đầu tùy theo triển khai. Nếu không thể biểu diễn giá trị ban đầu bằng kiểu đích thì hành vi là không xác định.

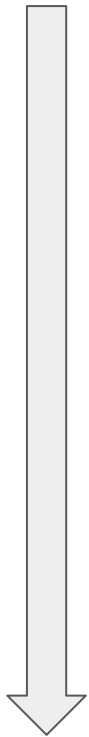
Giá trị sau khi ép kiểu

- Giá trị ban đầu được bảo toàn trong các trường hợp sau:
 - signed char => short => int => long => long long
 - (tương tự với các số nguyên không dấu)
 - float => double => long double
 - *Giá trị luôn được bảo toàn khi mở rộng phạm vi biểu diễn*
- Giá trị ban đầu không được bảo toàn trong các trường hợp sau:
 - 320 (int) => unsigned char: 64
 - -20(int) => unsigned char: 236;
 - 200 (int) => signed char: Hành vi không xác định
 - 1.8f(float) => int: 1
 - 300.0f(float) => unsigned char: Hành vi không xác định;
 - 987654321l(long) => float: 987656704 (GCC)
 - *Có thể phát sinh vấn đề khi thu hẹp phạm vi biểu diễn*

Thứ tự ưu tiên và chiều thực hiện chuỗi toán tử

Các toán tử trong cùng 1 dòng có cùng thứ tự ưu tiên. Các toán tử ở dòng trên có thứ tự ưu tiên cao hơn các toán tử ở dòng dưới.

Ưu tiên cao
(thực hiện trước)



Ưu tiên thấp
(thực hiện sau)

Tên toán tử	Ký hiệu & ví dụ	Thứ tự
Tăng 1 (hậu tố) Giảm 1 (hậu tố)	++ (x++) -- (x--)	<i>Trong phạm vi hiện tại hiếm khi cần quan tâm (thường chỉ áp dụng 1 lần cho 1 toán hạng)</i>
Tăng, giảm 1 (tiền tố) Toán tử dấu (tiền tố) Toán tử sizeof	++, -- (++x, --x) +, - (+x, -x) sizeof (sizeof(int))	
Ép kiểu	() (double)5	
Nhân Chia Phần dư	* (x * y) / (x / y) % (x % y)	
Cộng Trừ	+ (x + y) - (x - y)	Trái -> Phải
Dịch sang trái Dịch sang phải	<< (x << y) >> (x >> y)	Trái -> Phải
AND theo bit	& (x & y)	Trái -> Phải
XOR theo bit	^ (x ^ y)	Trái -> Phải
OR theo bit	(x y)	Trái -> Phải
Gán đơn giản Gán kết hợp	= *=, /=, %=, +=, -=, <=<=, >>=, &=, ^=, =	Phải -> Trái

Ví dụ 3.5. Các ví dụ tổng hợp

Giả sử #bits trong biểu diễn của kiểu int = 32.

Hãy cho biết giá trị và kiểu của các biểu thức sau:

`-0xFFFFFFFF`

`0x80000000 + 0x80000000`

Có vấn đề gì với biểu thức?

`0x7FFFFFFFFF + 0x7FFFFFFFFF`

