

# Ngôn ngữ lập trình C

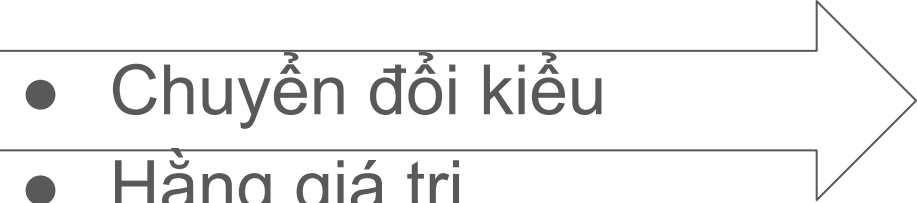
## Bài 3. Chuyển đổi kiểu, hằng giá trị

*Soạn bởi: TS. Nguyễn Bá Ngọc*

# Nội dung

- Chuyển đổi kiểu
- Hằng giá trị

# Nội dung

- 
- Chuyển đổi kiểu
  - Hằng giá trị

# Sơ lược về chuyển đổi kiểu

- NNLT C là một NNLT *định kiểu lỏng*:
  - Có thể kết hợp các kiểu khác nhau (ví dụ số nguyên và số thực) trong cùng 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 cần tự động/ngầm định tạo thêm các lệnh chuyển đổi kiểu khi cần.
  - Trong một số trường hợp có thể gây ra hành vi bất thường (khác với chủ định của người lập trình).
  - (Chi tiết sẽ tiếp tục được cung cấp trong phần biểu thức)
- Ngoài chuyển đổi kiểu ngầm định, người lập trình cũng có thể (chủ động/tường minh) viết lệnh chuyển đổi kiểu khi cần
  - Cú pháp: (Kiểu đích) đối-tượng
  - Ví dụ: (double)10

*Lưu ý thêm: Giá trị ban đầu có thể thay đổi sau khi ép kiểu*

# Chuyển đổi kiểu ngầm định

Chuyển đổi 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):
  - Các quy tắc ngầm định cho đại số phổ thông/Usual arithmetic conversions được áp dụng cho phần lớn các trường hợp tính toán.
  - Yêu cầu mở rộng biểu diễn cho toán hạng có kiểu số nguyên với hạng nhỏ hơn hạng của kiểu int (và unsigned int).
- Trong phép gán.
- Khi gọi hàm.
- Khi trả về giá trị cho hàm.

*Chúng ta sẽ học về hàm sau, các trường hợp với hàm có nhiều điểm tương đồng với trường hợp phép gán.*

# Quy tắc chuyển đổi cho đại số phổ thông

- Các quy tắc sau được áp dụng cho phần lớn phép toán thông dụng với 2 toán hạng có kiểu khác nhau:
  - Nếu có 1 toán hạng có kiểu long double thì toán hạng còn lại được chuyển đổi sang kiểu long double.
  - Trong trường hợp ngược lại, nếu có 1 toán hạng có kiểu double thì toán hạng còn lại được chuyển đổi sang kiểu double.
  - Trong trường hợp ngược lại, nếu có 1 toán hạng có kiểu float thì toán hạng còn lại được chuyển đổi sang kiểu float.
  - Trong trường hợp ngược lại - Cả 2 toán hạng đều có kiểu số nguyên, thực hiện nâng kiểu số nguyên.
- Nếu các toán hạng có cùng kiểu (**và có hạng  $\geq \text{rank}(\text{int})$  đối với số nguyên**) thì không chuyển đổi.

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

# Ví dụ chuyển đổi kiểu với kiểu số thực

Chúng ta xét một số trường hợp sau:

```
float f;
```

```
double lf;
```

```
long double llf;
```

```
long int li;
```

```
llf + lf; // chuyển lf thành kiểu long double
```

```
lf + f; // chuyển f thành kiểu double
```

```
f + li; // chuyển li thành kiểu float
```

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

# Hạng của kiểu số nguyên

- Chúng ta ký hiệu  $\text{rank}(T)$  là hạng của  $T$ , với  $T$  là kiểu số nguyên;  $\text{len}(T)$  là số bits trong biểu diễn của kiểu  $T$ .
- Một số quy ước cơ bản được mô tả trong quy chuẩn:
  - 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 chỉ có  $\text{len}(T1) \leq \text{len}(T2)$ .
  - 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<sub>(2)</sub>

- Với các kiểu số nguyên trong quy chuẩn chúng ta có:  
 $\text{rank}(\_Bool) < \text{rank}(\text{signed char}) < \text{rank}(\text{short}) < \text{rank}(\text{int}) < \text{rank}(\text{long}) < \text{rank}(\text{long long})$   
 $\text{rank}(\text{int}) = \text{rank}(\text{unsigned int})$   
...
- Với T1 và T2 là 2 kiểu số nguyên có 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.

# Yêu cầu mở rộng biểu diễn với số nguyên nhỏ

- Trong **nhiều** biểu thức đại số chứa các toán hạng có kiểu số nguyên với hạng nhỏ hơn hoặc bằng hạng của kiểu int (và unsigned int). Nếu kiểu int có thể biểu diễn được tất cả các giá trị của các kiểu ban đầu của các toán hạng đó thì các toán hạng đó được chuyển thành kiểu int, nếu ngược lại thì chuyển thành kiểu unsigned int.

Ví dụ:

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

# Quy tắc nâng kiểu số nguyên

- Các quy tắc sau được áp dụng cho phép toán với 2 kiểu số nguyên khác nhau:
  - Nếu cả 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 cao hơn hoặc bằng hạng của kiểu có dấu, thì toán tử 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 nâng 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.*

# Quy tắc chuyển đổi cho phép gán

- Trong phép gán giá trị ở vế phải được chuyển đổi thành kiểu của biến ở vế trái nếu cần.

- Ví dụ:

```
char c;
```

```
int i;
```

```
float f;
```

```
i = 80.115; // i == 80
```

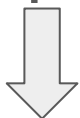
```
c = 300; // không chắc về kết quả
```

```
f = i; // f == 80
```

# Giá trị thu được sau khi chuyển đổi kiểu

Có khá 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<sub>(2)</sub>

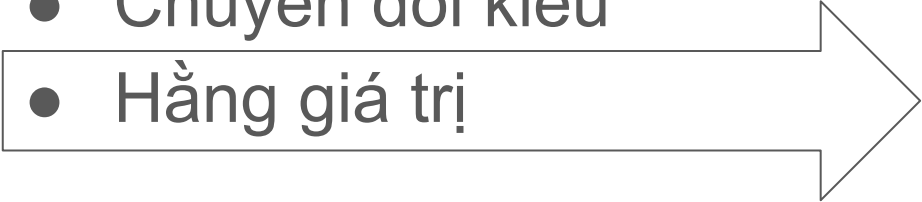
- Nếu không thể biểu diễn chính xác giá trị ban đầu bằng kiểu đích thì có thể có nhiều kịch bản khác nhau:
  - Số nguyên => số nguyên không dấu n bits: Giá trị mới bằng phần dư của giá trị ban đầu khi chia cho  $2^n$  (*= lặp cộng hoặc trừ với  $2^n$  cho tới khi thu được giá trị trong phạm vi*).
  - 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.

# Ví dụ giá trị sau khi chuyển đổi 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*



# Nội dung

- Chuyển đổi kiểu
  - Hằng giá trị
- 

# 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 != 0
  - Hằng HCS 8 bắt đầu với 0
  - Hằng HCS 16 bắt đầu với 0x hoặc 0X
- Hằng số nguyên có thể có hậu tố (không bắt buộc) để mô tả kiểu:
  - u hoặc U => có kiểu thuộc nhóm unsigned/không dấu
  - l hoặc L => có kiểu thuộc nhóm long
    - Có thể kết hợp với u hoặc U
  - ll hoặc LL => 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.

# Hậu tố và kiểu của hằng số nguyên

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

# Ví dụ 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 hằng số thực

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

# Các hằng liệt kê

- Các hằng liệt kê có kiểu int
- Giá trị mặc định theo thứ tự liệt kê và bắt đầu từ 0.
- Ví dụ:

```
enum Bool {False, True};
```

=> False == 0 và True == 1

*Thông tin chi tiết hơn về kiểu liệt kê được cung cấp sau.*

# Sơ lược về hằng ký tự

- Trên thế giới có rất nhiều ký tự và nhiều bảng mã, nhưng phổ biến nhất có lẽ là bảng mã ASCII và bảng mã Unicode.
  - Trong phạm vi học phần này chúng ta tạm thời chỉ xử lý bảng mã ASCII.
    - Bảng mã ASCII không chứa các ký tự cho tiếng việt, vì vậy việc tìm hiểu thêm về Unicode là cần thiết.
  - Chúng ta cũng giả sử giá trị số của các ký tự được quy đổi bằng bảng mã ASCII.
    - Như vậy sẽ làm giảm đáng kể sự phức tạp trong xử lý ký tự.
- Các ký tự trong bảng mã ASCII được mã hóa bằng 7 bits và trong C chúng ta có thể lưu 1 ký tự trong bảng mã ASCII bằng 1 biến kiểu char.



# Sơ lược về hằng ký tự<sup>(2)</sup>

- Hằng ký tự được viết trong cặp dấu nháy đơn ('')
- Trong trường hợp đơn giản nhất có thể biểu diễn 1 hằng ký tự bằng chính ký tự đó (trong cặp dấu ''), ví dụ: 'a', 'b'.
- Chúng ta còn có thể biểu diễn hằng ký tự bằng các chuỗi chuyển (escape sequence):
  - Chuỗi chuyển ký tự (cho một số ký tự đặc biệt):
    - '\n' - Dấu xuống dòng/New line;
    - '\r' - Lùi về đầu dòng/Carriage return;
    - '\t' - Dấu tab/Vertical tab;
    - '\"' - Dấu nháy đơn/Single quote;
    - '\\' - Dấu gạch (ngghiêng) trái/Backslash,
    - v.v..
  - Chuỗi chuyển HCS 8, ví dụ: '\101' ('A')
  - Chuỗi chuyển HCS 16, ví dụ: '\x41' ('A')
  - (Ngoài ra còn có chuỗi chuyển sử dụng mã riêng của ký tự, universal character name - tham khảo thêm)

*Với các chuỗi chuyển  
số chúng ta có thể  
viết được bất kỳ hằng  
ký tự nào.*

# Số nguyên và hằng ký tự

- Trong C các hằng ký tự có kiểu là int, và có thể xuất hiện trong các tính toán như các số nguyên.
- Các hằng ký tự chỉ có 1 ký tự ASCII trong cặp dấu "", ví dụ 'a' có giá trị số bằng mã số của ký tự tương ứng trong bảng mã ASCII, ví dụ: 'a'  $\Rightarrow$  97
- Các hằng được biểu diễn bằng chuỗi thoát ký tự có giá trị số bằng mã số của ký tự tương ứng, ví dụ: '\n'  $\Rightarrow$  10
- Các hằng được biểu diễn bằng chuỗi thoát số có giá trị bằng giá trị của 1 biến kiểu char với dãy bits tương tự khi quy đổi sang kiểu int.
  - Ví dụ: '\102' và '\x42' đều có giá trị là 66 (ký tự 'B')
  - Nhưng '\377' và '\xFF' có giá trị phụ thuộc vào trình biên dịch. Nếu char là unsigned char thì giá trị số là, 255. Nếu char là signed char thì giá trị số là -1.

# Bảng mã ASCII

Dec	Hex	Oct	Ký tự	Dec	Hex	Oct	Ký tự	Dec	Hex	Oct	Ký tự	Dec	Hex	Oct	Ký tự
0	0	0	NULL	32	20	40	Space	64	40	100	@	96	60	140	`
1	1	1	SOH (Start Of Heading)	33	21	41	!	65	41	101	A	97	61	141	a
2	2	2	STX (Start Of Text)	34	22	42	"	66	42	102	B	98	62	142	b
3	3	3	ETX (End Of Text)	35	23	43	#	67	43	103	C	99	63	143	c
4	4	4	EOT (End Of Transmission)	36	24	44	\$	68	44	104	D	100	64	144	d
5	5	5	ENQ (Enquiry)	37	25	45	%	69	45	105	E	101	65	145	e
6	6	6	ACK (Acknowledge)	38	26	46	&	70	46	106	F	102	66	146	f
7	7	7	BEL (Bell)	39	27	47	'	71	47	107	G	103	67	147	g
8	8	10	BS (Backspace)	40	28	50	(	72	48	110	H	104	68	150	h
9	9	11	TAB (Horizontal Tab)	41	29	51	)	73	49	111	I	105	69	151	i
10	A	12	LF (Line Feed, NL-New Line)	42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13	VT (Vertical Tab)	43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14	FF (Form Feed, NP-New Page)	44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15	CR (Carriage Return)	45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16	SO (Shift Out)	46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17	SI (Shift In)	47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20	DLE (Data Link Escape)	48	30	60	0	80	50	120	P	112	70	160	p
17	11	21	DC1 (Device Control 1)	49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22	DC2 (Device Control 2)	50	32	62	2	82	52	122	R	114	72	162	r
19	13	23	DC3 (Device Control 3)	51	33	63	3	83	53	123	S	115	73	163	s
20	14	24	DC4 (Device Control 4)	52	34	64	4	84	54	124	T	116	74	164	t
21	15	25	NAK (Negative Acknowledge)	53	35	65	5	85	55	125	U	117	75	165	u
22	16	26	SYN (Synchronous Idle)	54	36	66	6	86	56	126	V	118	76	166	v
23	17	27	ETB (End of Transmission Block)	55	37	67	7	87	57	127	W	119	77	167	w
24	18	30	CAN (Cancel)	56	38	70	8	88	58	130	X	120	78	170	x
25	19	31	EM (End of Medium)	57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32	SUB (Substitute)	58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33	ESC (Escape)	59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34	FS (File Separator)	60	3C	74	<	92	5C	134	\	124	7C	174	
29	1D	35	GS (Group Separator)	61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36	RS (Record Separator)	62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37	US (Unit Separator)	63	3F	77	?	95	5F	137	_	127	7F	177	DEL

<https://github.com/bangoc/C0/blob/master/b01-bien-kieudulieu-hangso/kytu-ascii.c>

