

# Ngôn ngữ lập trình C

## Bài 8. Ký tự và chuỗi ký tự

*Soạn bởi: TS. Nguyễn Bá Ngọc*

# Nội dung

- Ký tự
- Chuỗi ký tự

# Nội dung

- Ký tự
- Chuỗi ký tự

# Sơ lược về hằng ký tự

- Các ngôn ngữ trên thế giới có rất nhiều ký tự và trong các hệ thống máy tính cũng có rất nhiều bảng mã, trong số đó phổ biến nhất có lẽ là bảng mã ASCII và bảng mã Unicode.
  - Các ký tự trong bảng mã ASCII là phần đầu tiên của bảng mã Unicode.
    - Có thể coi bảng mã Unicode hàm chứa bảng mã ASCII
  - Các ký tự cho tiếng Việt có trong Unicode
    - TCVN 6909:2001
- Mã số của các ký tự trong bảng mã ASCII:
  - Được bảo toàn trong bảng mã Unicode
  - Có thể được mã hóa bằng 7 bits
  - Có thể được lưu trong 1 biến kiểu **char** mà không phụ thuộc vào dấu của kiểu **char**.

# Sơ lược về hăng ký tự<sub>(2)</sub>

- Mã số của các ký tự trong bảng mã Unicode có thể được mã hóa với nhiều đơn vị khác nhau theo cùng 1 giải thuật
  - UTF-8 sử dụng đơn vị 8 bits - 1 byte
  - UTF-16 sử dụng đơn vị 16 bits - 2 bytes
  - UTF-32 sử dụng đơn vị 32 bits - 4 bytes
  - *(UTF - The Unicode Transformation Format - về bản chất là phương pháp mã hóa với độ dài linh hoạt).*
- Tùy theo kích thước của đơn vị mã hóa UTF và giá trị số của mã số, có thể cần sử dụng nhiều đơn vị để biểu diễn 1 mã số Unicode

# Sơ lược cú pháp hằng ký tự

- Hằng ký tự được viết trong cặp dấu nháy đơn ''
- Trong trường hợp đơn giản nhất có thể đặt 1 ký tự bất kỳ thuộc *tập ký tự mã nguồn* ngoại trừ dấu nháy đơn ', dấu gạch trái \, và dấu xuống dòng trong cặp dấu nháy đơn '',
  - Ví dụ các hằng 1 ký tự: 'a', 'b', 'C', v.v.
- Hằng chuỗi chuyển ký tự/Alphabetic escape sequence:
  - '\', '\"', '\?', '\\'. Riêng '\"' và '\?' tương đương với '"' và '?'.
  - '\a' - Tín hiệu cảnh báo hình ảnh hoặc âm thanh/Alert/Bell
  - '\b' - Xóa 1 ký tự bên trái/Backspace
  - '\f' - Sang trang/Form feed
  - '\n' - Xuống dòng/New line;
  - '\r' - Lùi về đầu dòng/Carriage return;
  - '\t' - Tab ngang/Horizontal tab;
  - '\v' - Tab dọc/Vertical tab;

# Sơ lược cú pháp hằng ký tự<sub>(2)</sub>

- Hằng chuỗi chuyển HCS 8
  - Định dạng chuỗi chuyển HCS 8: \các chữ số HCS 8
  - Ví dụ: '\101' ('A')
- Hằng chuỗi chuyển HCS 16
  - Định dạng chuỗi chuyển HCS 16: \x các chữ số HCS 16
  - Ví dụ: '\x41' ('A')
- Hằng ký tự cũng có thể được tạo bằng tên phổ quát của ký tự/Universal Character Name (UCN)
  - Ví dụ: '\u0024' (\$)
  - *(Tham khảo thêm)*

*Có thể viết bất kỳ hằng ký tự nào bằng chuỗi chuyển với mã ký tự.*

# Số nguyên và hằng ký tự

*Trong C các hằng ký tự có kiểu `int`, và có thể được sử dụng trong các biểu thức tính toán như các số nguyên.*

- Xác định giá trị số của hằng ký tự trong trường hợp tổng quát có thể là vấn đề phức tạp vì phụ thuộc vào nhiều bảng mã và triển khai trình biên dịch.
- Trong giới hạn các bảng mã được sử dụng là ASCII hoặc bảng mã có hàm chứa bảng mã ASCII:
  - Hằng thuần chỉ chứa 1 ký tự ASCII có giá trị số bằng mã số tương ứng, ví dụ 'a' có giá trị = 97.
  - Hằng chuỗi thoát ký tự có giá trị số bằng mã số của ký tự tương ứng trong bảng mã ASCII, ví dụ: '\n' có giá trị = 10



# Số nguyên và hằng ký tự<sub>(2)</sub>

- Hằng chuỗi thoát số với chuỗi thoát tương ứng với 1 mã số ASCII có giá trị số = mã số tương ứng, ví dụ:
  - '\x20' có giá trị số = 32.
- Các hằng chuỗi thoát số có thể biểu diễn được bằng 1 byte có giá trị bằng giá trị của 1 biến kiểu `char` với dãy bits tương tự chuỗi thoát sau khi ép kiểu thành `int`, ví dụ:
  - '\102' và '\x42' đều có giá trị là 66.
  - '\377' và '\xFF' có giá trị số phụ thuộc vào triển khai - nếu `char` là `unsigned char` thì giá trị số = 255, nếu ngược lại (`char` là `signed char`) thì giá trị số = -1.
- *(Chúng ta tạm thời chưa xét các trường hợp khác như hằng ký tự chứa nhiều hơn 1 ký tự, chuỗi thoát số có giá trị ngoài phạm vi biểu diễn bằng 1 byte, v.v..)*

# 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>]

# Các hàm phân loại ký tự tiêu biểu (ctype.h)

*Các nguyên mẫu giản lược:*

- `int isascii (int c);` // Ký tự thuộc bảng mã ASCII?
  - (có trong mở rộng GNU)
  - Đúng nếu độ rộng của c không quá 7 bits.
- `int islower(int c);` // Ký tự là chữ cái thường?
  - Mặc định hàm trả về giá trị đúng cho các ký tự trong 26 chữ cái Latin viết thường

<b>a</b>	<b>b</b>	<b>c</b>	<b>d</b>	<b>e</b>	<b>f</b>	<b>g</b>	<b>h</b>	<b>i</b>	<b>j</b>	<b>k</b>	<b>l</b>	<b>m</b>
<b>n</b>	<b>o</b>	<b>p</b>	<b>q</b>	<b>r</b>	<b>s</b>	<b>t</b>	<b>u</b>	<b>v</b>	<b>w</b>	<b>x</b>	<b>y</b>	<b>z</b>

- `int isupper(int c);` // Ký tự là chữ cái hoa?
  - Mặc định trả về giá trị đúng cho các ký tự trong 26 chữ cái Latin viết hoa

<b>A</b>	<b>B</b>	<b>C</b>	<b>D</b>	<b>E</b>	<b>F</b>	<b>G</b>	<b>H</b>	<b>I</b>	<b>J</b>	<b>K</b>	<b>L</b>	<b>M</b>
<b>N</b>	<b>O</b>	<b>P</b>	<b>Q</b>	<b>R</b>	<b>S</b>	<b>T</b>	<b>U</b>	<b>V</b>	<b>W</b>	<b>X</b>	<b>Y</b>	<b>Z</b>

# Các hàm phân loại ký tự tiêu biểu (ctype.h)<sub>(2)</sub>

- `int isalpha(int c);` // Chữ cái thường hoặc hoa?
  - Mặc định trả về đúng nếu **isupper** hoặc **islower** đúng với c.
- `int isdigit(int c);` // Ký tự là chữ số (HCS 10)?
  - Trả về đúng nếu c là 1 trong các chữ số HCS 10.

**0    1    2    3    4    5    6    7    8    9**

- `int isalnum(int c);` // Ký tự là chữ cái hoặc chữ số?
  - Đúng nếu **isalpha** hoặc **isdigit** đúng.
- `int isxdigit(int c);` // Ký tự là chữ số HCS 16 (Hexa)?
  - Đúng nếu c là 1 chữ số HCS 16.

**0    1    2    3    4    5    6    7    8    9**

**a    b    c    d    e    f**

**A    B    C    D    E    F**

# Các hàm phân loại ký tự tiêu biểu (ctype.h)<sub>(3)</sub>

- `int isgraph(int c);` // Ký tự nhìn thấy ngoại trừ dấu cách?
  - Trả về đúng cho ký tự in, không bao gồm dấu cách
- `int isprint(int c);` // Ký tự nhìn thấy, bao gồm dấu cách?
  - Trả về đúng cho ký tự in, bao gồm cả dấu cách
- `int isblank(int c);` // Ký tự là khoảng trống?
  - Mặc định trả về đúng cho dấu cách ' ' và '\t'.
- `int iscntrl(int c);` // Ký tự điều khiển?
  - Trả về đúng nếu c là ký tự điều khiển
- `int isspace(int c);` // Ký tự là khoảng trắng?
  - Mặc định trả về đúng cho dấu cách ' ', '\f', '\n', '\r', '\t', '\v'
- `int ispunct(int c);` // Ký tự là dấu?
  - Mặc định trả về đúng cho ký tự in mà cả isspace và isalnum đều sai

# Các hàm chuyển đổi chữ cái hoa/thường

- `int tolower(int c);` // Trả về chữ cái thường tương ứng của c
  - Nếu `isupper` đúng với c và có ít nhất 1 ký tự tương ứng với c mà `islower` đúng thì `tolower` trả về 1 ký tự tương ứng với c; Nếu ngược lại thì `tolower` trả về c - không thay đổi.
  - Mặc định:
    - `tolower('A')` trả về 'a',
    - `tolower('1')` trả về '1', v.v..
- `int toupper(int c);` // Trả về chữ cái hoa tương ứng của c
  - Nếu `islower` đúng với c và có ít nhất 1 ký tự tương ứng với c mà `isupper` đúng thì `toupper` trả về 1 ký tự tương ứng với c; Nếu ngược lại thì `toupper` trả về c - không thay đổi.
  - Mặc định:
    - `toupper('c')` - trả về 'C',
    - `toupper('2')` - trả về '2', v.v..

# Nhập & xuất ký tự

- Đọc ký tự:

- `int getchar(void);`
  - Đọc 1 ký tự trong luồng nhập chuẩn - stdin. Hàm trả về ký tự đọc được hoặc EOF nếu đã hết dữ liệu nhập hoặc phát sinh lỗi.
- `// Đọc 1 ký tự trong luồng nhập chuẩn`
- `char c = getchar();`
- `scanf("%c", &c);`

- Xuất ký tự:

- `int putchar(int c);`
  - Xuất ký tự ra luồng xuất chuẩn - stdout. Hàm trả về ký tự đã xuất hoặc EOF nếu phát sinh lỗi.
- `// Xuất 1 ký tự ra luồng xuất chuẩn`
- `putchar(c);`
- `printf("%c", c);`



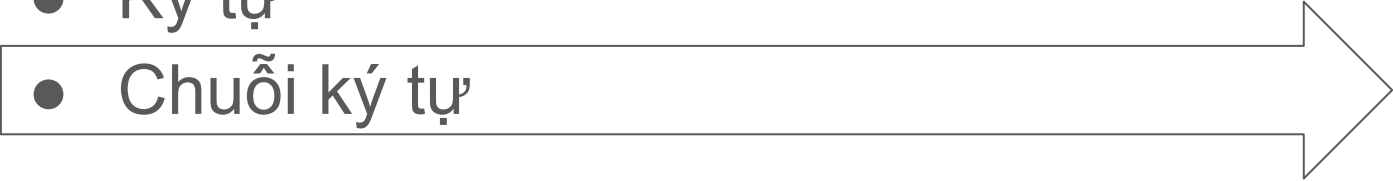
# Phân lớp ký tự ASCII với cấu hình mặc định

**Các lớp:** *cntrl* - điều khiển, *space* - khoảng trắng, *punct* - dấu, *alnum* - chữ số hoặc chữ cái, *print* - ký tự in.

dec	char	cntrl	space	punct	alnum	print	dec	char	cntrl	space	punct	alnum	print	dec	char	cntrl	space	punct	alnum	print	dec	char	cntrl	space	punct	alnum	print
0	null	T	F	F	F	F	32	Spac	F	T	F	F	T	64	@	F	F	T	F	T	96	`	F	F	T	F	T
1	SOH	T	F	F	F	F	33	!	F	F	T	F	T	65	A	F	F	F	T	T	97	a	F	F	F	T	T
2	STX	T	F	F	F	F	34	"	F	F	T	F	T	66	B	F	F	F	T	T	98	b	F	F	F	T	T
3	ETX	T	F	F	F	F	35	#	F	F	T	F	T	67	C	F	F	F	T	T	99	c	F	F	F	T	T
4	EOT	T	F	F	F	F	36	\$	F	F	T	F	T	68	D	F	F	F	T	T	100	d	F	F	F	T	T
5	ENQ	T	F	F	F	F	37	%	F	F	T	F	T	69	E	F	F	F	T	T	101	e	F	F	F	T	T
6	ACK	T	F	F	F	F	38	&	F	F	T	F	T	70	F	F	F	F	T	T	102	f	F	F	F	T	T
7	BEL	T	F	F	F	F	39		F	F	T	F	T	71	G	F	F	F	T	T	103	g	F	F	F	T	T
8	BS	T	F	F	F	F	40	(	F	F	T	F	T	72	H	F	F	F	T	T	104	h	F	F	F	T	T
9	TAB	T	T	F	F	F	41	)	F	F	T	F	T	73	I	F	F	F	T	T	105	i	F	F	F	T	T
10	LF	T	T	F	F	F	42	*	F	F	T	F	T	74	J	F	F	F	T	T	106	j	F	F	F	T	T
11	VT	T	T	F	F	F	43	+	F	F	T	F	T	75	K	F	F	F	T	T	107	k	F	F	F	T	T
12	FF	T	T	F	F	F	44	,	F	F	T	F	T	76	L	F	F	F	T	T	108	l	F	F	F	T	T
13	CR	T	T	F	F	F	45	-	F	F	T	F	T	77	M	F	F	F	T	T	109	m	F	F	F	T	T
14	SO	T	F	F	F	F	46	.	F	F	T	F	T	78	N	F	F	F	T	T	110	n	F	F	F	T	T
15	SI	T	F	F	F	F	47	/	F	F	T	F	T	79	O	F	F	F	T	T	111	o	F	F	F	T	T
16	DLE	T	F	F	F	F	48	0	F	F	F	T	T	80	P	F	F	F	T	T	112	p	F	F	F	T	T
17	DC1	T	F	F	F	F	49	1	F	F	F	T	T	81	Q	F	F	F	T	T	113	q	F	F	F	T	T
18	DC2	T	F	F	F	F	50	2	F	F	F	T	T	82	R	F	F	F	T	T	114	r	F	F	F	T	T
19	DC3	T	F	F	F	F	51	3	F	F	F	T	T	83	S	F	F	F	T	T	115	s	F	F	F	T	T
20	DC4	T	F	F	F	F	52	4	F	F	F	T	T	84	T	F	F	F	T	T	116	t	F	F	F	T	T
21	NAK	T	F	F	F	F	53	5	F	F	F	T	T	85	U	F	F	F	T	T	117	u	F	F	F	T	T
22	SYN	T	F	F	F	F	54	6	F	F	F	T	T	86	V	F	F	F	T	T	118	v	F	F	F	T	T
23	ETB	T	F	F	F	F	55	7	F	F	F	T	T	87	W	F	F	F	T	T	119	w	F	F	F	T	T
24	CAN	T	F	F	F	F	56	8	F	F	F	T	T	88	X	F	F	F	T	T	120	x	F	F	F	T	T
25	EM	T	F	F	F	F	57	9	F	F	F	T	T	89	Y	F	F	F	T	T	121	y	F	F	F	T	T
26	SUB	T	F	F	F	F	58	:	F	F	T	F	T	90	Z	F	F	F	T	T	122	z	F	F	F	T	T
27	ESC	T	F	F	F	F	59	;	F	F	T	F	T	91	[	F	F	T	F	T	123	{	F	F	T	F	T
28	FS	T	F	F	F	F	60	<	F	F	T	F	T	92	\	F	F	T	F	T	124		F	F	T	F	T
29	GS	T	F	F	F	F	61	=	F	F	T	F	T	93	]	F	F	T	F	T	125	}	F	F	T	F	T
30	RS	T	F	F	F	F	62	>	F	F	T	F	T	94	^	F	F	T	F	T	126	~	F	F	T	F	T
31	US	T	F	F	F	F	63	?	F	F	T	F	T	95	_	F	F	T	F	T	127	DEL	T	F	F	F	F



# Nội dung

- Ký tự
  - Chuỗi ký tự
- 

# Các khái niệm

*Chuỗi ký tự là dãy ký tự liên tiếp có chứa ký tự null (giá trị số = 0) và được kết thúc bởi ký tự null đầu tiên - điểm kết thúc chuỗi.*

- Trong học phần chúng ta sử dụng biểu diễn chuỗi ký tự với **mảng kiểu char** - có thể được gọi là chuỗi (kiểu) **char**.
  - *(Có thể tham khảo thêm các ký tự rộng - kiểu wchar\_t, và chuỗi ký tự rộng)*
  - **char** s1[] = {'H', 'i', '\0'}; // s1 là chuỗi ký tự hợp lệ
  - **char** s2[2] = {'H', 'i'}; // Về lô-gic s2 không phải chuỗi ký tự do không có ký tự kết thúc.
    - *(Mảng phải đủ lớn để lưu cả ký tự kết thúc chuỗi)*
- Con trỏ tới chuỗi ký tự là con trỏ tới ký tự đầu tiên của nó.
- Độ dài chuỗi ký tự là số lượng byte đứng trước ký tự null.
- Giá trị của 1 chuỗi là chuỗi các giá trị của các ký tự của nó, theo thứ tự.

# Hằng chuỗi ký tự

*Hằng chuỗi ký tự bao gồm các ký tự liên tiếp được đặt trong cặp dấu nháy kép "".*

- Các ký tự có thể là 1 ký tự bất kỳ trong *tập ký tự mã nguồn* ngoại trừ dấu nháy kép ", dấu gạch trái \, dấu xuống dòng.
- Các ký tự trong chuỗi cũng có thể được biểu diễn bằng các chuỗi thoát như các hằng ký tự.
- Chuỗi không chứa ký tự nào được gọi là chuỗi rỗng "".
- Ký tự null được tự động nối vào sau hằng chuỗi ký tự trong tiến trình biên dịch.
- Ví dụ:
  - "Hello world!";
  - "Hi\n\x43 Programming";
    - Các chuỗi thoát trong ví dụ tương ứng với các ký tự nào?

# Hằng chuỗi ký tự<sub>(2)</sub>

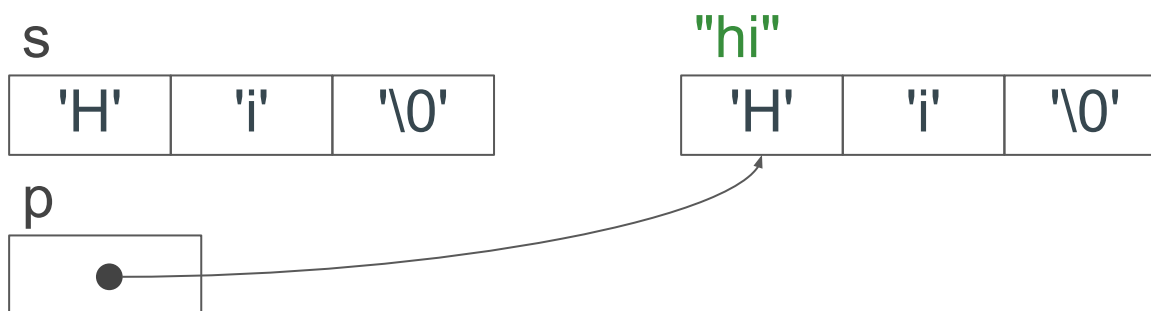
- Hằng chuỗi ký tự có thể bao gồm các ký tự Unicode, sau khi mã hóa UTF-8 có thể được lưu trong mảng kiểu `char`:
  - `char a[] = "Xin chào";`
    - `sizeof(a) == 10;` // > số lượng ký tự đọc được + 1
    - Ký tự à được biểu diễn bằng 2 bytes (*tham khảo thêm về UTF-8*)
- Hằng chuỗi ký tự được lưu trong mảng không tên với kích thước vừa đủ để lưu các ký tự (kể cả ký tự null được tự động thêm vào), có thể nằm trong phân vùng nhớ chỉ đọc.
  - Ví dụ cấu trúc bộ nhớ cho hằng ký tự `"Hi";`

'H'	'i'	'\0'
-----	-----	------

*Các phần tử có kiểu `char`*
  - Các hằng ký tự giống nhau có thể được lưu ở cùng 1 địa chỉ.
- Có thể khởi tạo mảng ký tự bằng hằng chuỗi ký tự:
  - `char s[] = "Hi";` tương đương với `char s[] = {'H', 'i', '\0'};`
  - (*nhưng không khởi tạo được mảng bằng 1 mảng khác*)

# Chuỗi ký tự và con trỏ tới chuỗi ký tự

- `char s[] = "hi";`
  - s là chuỗi ký tự được khởi tạo bằng "hi".
  - `s[0] = 'H';` // Ok - Nội dung của s bây giờ là "Hi"
- `char *p = "hi";`
  - p là con trỏ tới hằng chuỗi "hi".
  - `p[0] = 'H';` // Có thể phát sinh lỗi nếu dữ liệu được lưu trong phân vùng chỉ đọc.
  - Có thể sử dụng `const char *pc = "hi";` và không thay đổi nội dung hằng chuỗi được trỏ tới.



# Các hàm xử lý chuỗi tiêu biểu (string.h)

*Nguyên mẫu giản lược:*

- **size\_t** strlen (**const char** \*s);
  - Trả về độ dài của chuỗi được trỏ tới bởi s.
- **char** \*strcpy (**char** \*dest, **const char** \*src);
  - Sao chép nội dung chuỗi được trỏ tới bởi src sang vùng nhớ được trỏ tới bởi dest.
  - Hàm trả về con trỏ dest.
- **int** strcmp (**const char** \*s1, **const char** \*s2);
  - So sánh các chuỗi được trỏ tới bởi s1 và s2 theo giá trị của từng ký tự. *Các thuật ngữ chuỗi s1, s2 được sử dụng với*
  - Trả về: *nghĩa chuỗi được trỏ tới bởi s1, s2*
    - 1 số âm nếu chuỗi s1 được coi là < chuỗi s2
    - 1 số dương nếu chuỗi s1 được coi là > chuỗi s2
    - 0 - nếu chuỗi s1 được coi là == chuỗi s2

# Các hàm xử lý chuỗi tiêu biểu (string.h)<sub>(2)</sub>

Nguyên mẫu giản lược:

- `char *strcat (char *dest, const char *src);`
  - Nối tiếp chuỗi src vào cuối chuỗi dest.
  - Hàm trả về con trỏ dest.
- `const char *strstr (const char *str, const char *sub);`
  - Tìm chuỗi sub trong str và trả về con trỏ tới vị trí bắt đầu của sub trong str nếu tìm thấy, hoặc trả về NULL nếu không.
- `const char *strchr (const char *s, int c);`
  - Trả về con trỏ tới ký tự có giá trị == c trong s nếu có, hoặc NULL nếu không.

# Độ dài chuỗi ký tự

*Độ dài chuỗi = số lượng byte đứng trước ký tự null.*

- Có thể triển khai hàm strlen như sau:

```
size_t my_strlen(const char *s) {  
    size_t len = 0;  
    while (*s != 0) { ++s; ++len; }  
    return len;  
}
```

...

```
char s[] = "ABC";
```

my\_strlen(s) trả về 3 - tương tự strlen(s).

*Khác với mảng, nhờ có quy ước ký tự kết thúc chuỗi chúng ta có thể duyệt các ký tự thuộc chuỗi chỉ với con trỏ tới chuỗi.*



# Sao chép chuỗi ký tự

- Có thể triển khai hàm strcpy như sau:

```
char *my_strcpy(char *dest, const char *src) {  
    char *p = dest;  
    while (*src) {*p = *src; ++p; ++src;}  
    *p = 0;  
    return dest;  
}
```

- Ví dụ:

- `char s1[20], s2[20] = "Hello world!";`
- `s1 = s2; // Lỗi - Không gán được mảng`
- `char *p1 = s2; // Đúng cú pháp - nhưng mang nghĩa khác, p1 trỏ đến chuỗi s2`
- `my_strcpy(s1, s2); // OK`

# So sánh các chuỗi ký tự

- Có thể triển khai hàm strcmp như sau:

```
int my_strcmp(const char *s1, const char *s2) {  
    for (;;) {  
        if (*s1 > *s2) { return 1; }  
        else if (*s1 < *s2) { return -1; }  
        else if (*s1 == 0) { break; }  
        ++s1; ++s2;  
    }  
    return 0;  
}
```

- Ví dụ:

- `char *s1 = "Abc", *s2 = "Abd", *s3 = "Abb";`
- `s1 < s2; // so sánh con trỏ`
- `my_strcmp(s1, s2)` - Trả về -1
- `my_strcmp(s1, s3)` - Trả về 1
- (Tương tự strcmp)

# Nhập & xuất chuỗi ký tự

- Có thể tự triển khai giải thuật nhập/xuất chuỗi dựa trên nhập/xuất từng ký tự..., hoặc sử dụng các hàm có trong thư viện chuẩn...
- Đọc chuỗi không chứa khoảng trắng:
  - `scanf("%s", s);`
- Đọc 1 dòng:
  - `char *fgets(char *s, int n, FILE *stream);` // Đọc 1 dòng cho tới hết ký tự '\n' và lưu ký tự '\n', hoặc tối đa n - 1 ký tự, sau đó điền ký tự null vào sau ký tự đọc được cuối cùng.
  - `scanf("%[^\\n]*c", s);` // Đọc 1 dòng cho tới hết ký tự '\n' nhưng không lưu ký tự '\n', ký tự null cũng được thêm vào sau ký tự cuối cùng - Tương tự `gets(s);`.
  - `char *gets(char *s);` // Đã bị loại khỏi thư viện chuẩn từ C11 - Không tiếp tục dùng `gets` trong mã nguồn mới.

# Nhập & xuất chuỗi ký tự<sub>(2)</sub>

- Có thể giới hạn số lượng ký tự (tương tự fgets) khi đọc 1 dòng với scanf bằng cách tạo động chuỗi định dạng:
  - `sprintf(fmt, "%%%d[^\n]%%*c", n - 1);`
  - `scanf(fmt, s);` // Tương tự `scanf("%[^\n]*c", s);` nhưng chỉ đọc tối đa  $n - 1$  ký tự
  - *(Đặc tả xuất %% - in ra dấu %)*
- Xuất chuỗi ký tự:
  - `int puts(const char *s);` // Xuất chuỗi s và thêm ký tự xuống dòng, trả về EOF nếu phát sinh lỗi hoặc giá trị không âm nếu ngược lại.
  - `printf("%s", s);` // Xuất chuỗi s theo định dạng.

# Ví dụ 8.1. Đọc chuỗi theo từng ký tự

```
1  #include <stdio.h>
2
3  // Đọc 1 dòng nhưng không quá n - 1 ký tự, không lưu
4  // ký tự '\n', trả về độ dài chuỗi kết quả
5  int my_gets(char *s, int n) {
6      int idx = 0, ch;
7      while (idx < n - 1 && (ch = getchar()) != EOF) {
8          if (ch == '\n') {
9              break;
10         }
11         s[idx++] = ch;
12     }
13     s[idx] = 0;
14     return idx;
15 }
16 int main() {
17     char s[10];
18     int cc = my_gets(s, 10);
19     printf("Chuỗi đã nhập: %s\ncc = %d\n", s, cc);
20 }
```

123  
Chuỗi đã nhập: 123  
cc = 3

1234567890  
Chuỗi đã nhập: 123456789  
cc = 9

**Thử:** Hiệu chỉnh hàm đọc chuỗi để lưu cả dấu xuống dòng?

## Ví dụ 8.2a. Các giải thuật nhập chuỗi<sub>(a)</sub>

*Sử dụng cùng 1 luồng nhập và so sánh kết quả đọc chuỗi trong các trường hợp*

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main() {
5      int n = 0;
6      char s[100];
7      scanf("%d", &n);
8      scanf("%s", s);
9      printf("s = %s, strlen(s) = %zu\n",
10             s, strlen(s));
11 }
```

1	0	\n	H	i	\n
---	---	----	---	---	----

10
Hi
s = Hi, strlen(s) = 2

Bỏ qua \n

## Ví dụ 8.2b. Các giải thuật nhập chuỗi<sub>(b)</sub>

*Sử dụng cùng 1 luồng nhập và so sánh kết quả đọc chuỗi trong các trường hợp*

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main() {
5      int n = 0;
6      char s[100];
7      scanf("%d", &n);
8      scanf("%[^\\n]*c", s);
9      printf("s = %s, strlen(s) = %zu\\n",
10             s, strlen(s));
11 }
```

1	0	\\n	H	i	\\n
---	---	-----	---	---	-----

s chưa được  
khởi tạo và  
không thay đổi

10  
s = , strlen(s) = 1

Dừng sau  
khi đọc \\n

## Ví dụ 8.2c. Các giải thuật nhập chuỗi<sub>(c)</sub>

*Sử dụng cùng 1 luồng nhập và so sánh kết quả đọc chuỗi trong các trường hợp*

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main() {
5      int n = 0;
6      char s[100];
7      scanf("%d", &n);
8      fgets(s, 100, stdin);
9      printf("s = %s, strlen(s) = %zu\n",
10             s, strlen(s));
11 }
```

s chứa dấu \n

1	0	\n	H	i	\n
---	---	----	---	---	----

```
10
s =
, strlen(s) = 1
```

Dừng sau  
khi đọc \n



## Ví dụ 8.2d. Các giải thuật nhập chuỗi<sub>(d)</sub>

Sử dụng cùng 1 luồng nhập và so sánh kết quả đọc chuỗi trong các trường hợp

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main() {
5      int n = 0;
6      char s[100];
7      scanf("%d", &n);
8      while (getchar() != '\n') ;
9      scanf("%[^\\n]*c", s);
10     printf("s = %s, strlen(s) = %zu\\n",
11            s, strlen(s));
12 }
```

1	0	\\n	H	i	\\n
---	---	-----	---	---	-----

10
Hi
s = Hi, strlen(s) = 2

Không lưu \\n

Dòng lệnh `while (getchar() != '\\n') ;` có tác dụng gì?

## Ví dụ 8.2e. Các giải thuật nhập chuỗi<sub>(e)</sub>

Sử dụng cùng 1 luồng nhập và so sánh kết quả đọc chuỗi trong các trường hợp

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main() {
5      int n = 0;
6      char s[100];
7      scanf("%d", &n);
8      while (getchar() != '\n') ;
9      fgets(s, 100, stdin);
10     printf("s = %s, strlen(s) = %zu\n",
11            s, strlen(s));
12 }
```

1	0	\n	H	i	\n
---	---	----	---	---	----

```
10
Hi
s = Hi
, strlen(s) = 3
```

Có lưu \n

`while (getchar() != '\n');` - đọc hết dòng hiện tại.

# Các mẫu vòng lặp

Duyệt các ký tự thuộc chuỗi s:

- Sử dụng chỉ số và độ dài:

```
for (int i = 0; i < strlen(s); ++i) {  
    /* Các xử lý ký tự s[i] */  
}
```

- Sử dụng chỉ số và ký tự null

```
for (int i = 0; s[i]; ++i) {  
    /* Các xử lý ký tự s[i] */  
}
```

- Sử dụng con trỏ và ký tự null:

```
for (char *p = s; *p; ++p) {  
    /* Các xử lý ký tự *p */  
}
```

## Ví dụ 8.3. Chuyển chữ hoa thành chữ thường

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <ctype.h>
4  int main() {
5      char s[20];
6      scanf("%[^\\n] %*c", s);
7      int cc = 0;
8      for (char *p = s; *p; ++p) {
9          if (isupper(*p)) {
10             cc++;
11             *p = tolower(*p);
12         }
13     }
14     printf("s = %s\\ncc = %d\\n", s, cc);
15 }
```

```
Hello WOrld!
s = hello world!
cc = 3
```

# Mảng chuỗi ký tự

*Chuỗi ký tự là mảng 1 chiều của ký tự, vì vậy có thể biểu diễn mảng chuỗi ký tự như mảng của mảng 1 chiều của ký tự.*

- Có thể sử dụng mảng 2 chiều với chiều thứ nhất là số lượng, chiều thứ 2 là độ dài của các chuỗi ký tự, ví dụ:
  - `char ss[10][20];` - `ss[i]` là chuỗi ký tự thứ `i`.
- Có thể lưu mảng chuỗi ký tự với độ dài khác nhau bằng mảng con trỏ `char *` tới các chuỗi *cấp phát động*, ví dụ:
  - `char *ss[10];`
  - `ss[i] = malloc(sizeof(char) * n[i]);`
- Cũng có thể lưu mảng con trỏ tới các hằng ký tự:
  - `const char *digits[] = {"Không", "Một", "Hai", "Ba", "Bốn", "Năm", "Sáu", "Bảy", "Tám", "Chín"};`
  - `digits[0]` trỏ tới hằng `"Không"`, v.v..

# Chuyển đổi giữa chuỗi và số

- Các hàm sscanf và sprintf được thiết kế tương tự scanf và printf nhưng cho phép nhập, xuất với chuỗi ký tự:
- Có thể sử dụng sscanf để chuyển đổi chuỗi (chứa giá trị số) thành số, ví dụ:
  - `const char *s = "3.1415";`
  - `double d;`
  - `sscanf(s, "%lf", &d); // d ~ 3.1415`
- Có thể sử dụng sprintf để chuyển đổi số thành chuỗi, ví dụ:
  - `double d = 3.1415;`
  - `char s[20];`
  - `sprintf(s, "%.3f", d); // s chứa "3.142"`

## Ví dụ 8.4. Đọc các chữ số của số nguyên

```
1  #include <stdio.h>
2  #include <string.h>
3  int main() {
4      const char *digits[] = { "Không", "Một", "Hai",
5          "Ba", "Bốn", "Năm", "Sáu", "Bảy", "Tám",
6          "Chín"};
7      int n = sizeof(digits)/sizeof(digits[0]), x;
8      printf("Nhập 1 số nguyên dương: ");
9      scanf("%d", &x);
10     char s[20];
11     sprintf(s, "%d", x);
12     for (char *p = s; *p != 0; ++p) {
13         printf("%s ", digits[*p - '0']);
14     }
15     printf("\n");
16 }
```

Nhập 1 số nguyên dương:

1230456879

Một Hai Ba Không Bốn Năm Sáu  
Tám Bảy Chín

## Ví dụ 8.5. Tách chuỗi ký tự

```
1  #include <stdio.h>
2  #include <string.h>
3  int main() {
4      char s[] = "Hello world!";
5      s[5] = '\0';
6
7      // ss là mảng con trỏ char * - tới các từ
8      char *ss[] = {s, s + 6};
9      int n = sizeof(ss)/sizeof(ss[0]);
10     for (int i = 0; i < n; ++i) {
11         printf("%s\n", ss[i]);
12     }
13 }
```

Hello  
world!

*Có thể lưu nhiều chuỗi ký tự trong cùng 1 mảng dài*



## Ví dụ 8.6. Xuất n chuỗi ký tự theo thứ tự ngược

```
1  #include <stdio.h>
2  int main() {
3      char ss[100][100];
4      int n;
5      printf("0 < n <= 100: ");
6      scanf("%d", &n);
7      printf("Nhập n chuỗi ký tự: \n");
8      while (getchar() != '\n') ;
9      for (int i = 0; i < n; ++i) {
10         printf("Chuỗi %d: ", i);
11         scanf("%[^\\n]*c", ss[i]);
12     }
13     printf("Các chuỗi theo thứ tự ngược: \n");
14     for (int i = n - 1; i >= 0; --i) {
15         printf("%s\\n", ss[i]);
16     }
17 }
```

0 < n <= 100: 3

Nhập n chuỗi ký tự:

Chuỗi 0: AAA

Chuỗi 1: BBB

Chuỗi 2: CCC

Các chuỗi theo thứ tự ngược  
lại:

CCC

BBB

AAA

