CS2313 Computer Programming

LT9 – String

Outline

- Character
  - Declaration and initialization
  - ASCII
  - Input and output
- String
  - Declaration and initialization
  - Copy and compare
  - Input and output

Syntax Summary

- Keywords
  - char
- Punctuators
  - [],’
- Constant
  - ‘\0’, ‘\n’

Character Data Type

- Written between single quotes, such as `'A', 'b', '*'.`
- In C++ language, a `char` type is represented by an **integer**.
- Therefore, a character can also be **printed** as an integer.

Character Type Examples

```
cout << 'a'; /* a is printed */
cout << (int)'a'; /* 97 is printed */
cout << (char)97; /* a is printed */
```

ASCII Table

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

ASCII value of '7' is 55 but not 7!

`Char` Features

A	B	...	Z	...	a	b	...	z
65	66	...	90	..	97	98	...	122

'A' + 1 has the value of 'B',
'B' + 1 has the value of 'C', ...

Test if character c is lower-case letter:

```
if ('a' <= c && c  <= 'z') ...
```

same as:

```
if (97 <=c && c <= 122) ...
```

If the variable `c` has the value of a lowercase letter, then the expression

```
c + 'A' - 'a' /* same as c + (65 -97)*/
```

has the value of the corresponding uppercase letter.

Example

```
#include <iostream>
using namespace std;

int main(){
     char c1='D', c2='e', c3,c4;
     c3=c1+'a'-'A'; //convert to lowercase
     c4=c2+'A'-'a'; //convert to uppercase
     cout << "c3=" << c3 << ", c4=" << c4 << endl;
      //c3=d, c4=E

     return 0;
}
```

Reading a Character

```
char c1,c2;
cin >> c1;
```

When `cin >> c1` is reached, the program will ask the user for input.

Suppose the character `'A'` is input, `c1` will evaluate to `65` (which is the ASCII code of `'A'`).

As a result, `65` will be assigned to the character `c1`. Therefore, `c1` holds the character `'A'`.

Some Facts About Keyboard Input

- Suppose >> is called to read a character.
- **Qn:** What if the user input more than one characters in a single line?
- **Ans**: The extra character will be stored in a buffer (certain memory location). The character will be retrieved later when >> is called to read more characters.

Some Facts About Keyboard Input

```
char c1,c2,c3;
cin >> c1; //enter the string "CS2313"
cin >> c2; //get the character 'S' from buffer
cin >> c3; //get the character '2' from buffer
```

	c1	c2	c3	Input buffer
(user input " cs2313")				C S 2 3 1 3 ↵
cin >> c1;	'C'			S 2 3 1 3 ↵
cin >> c2;	'C'	'S'		2 3 1 3 ↵
cin >> c3;	'C'	'S'	'2'	3 1 3 ↵

Printing a Character

```
char c1='A',c2='B';
cout << c1;
cout.put(c1);
```

Example

- Write a program which reads a character from the user and output the character type.

- The program should distinguish between the following types of characters
  - An upper case character (`'A'-'Z'`).
  - A lower case character (`'a'-'z'`).
  - A digit (`'0'-'9'`).
  - Special character (e.g. `'#', '$',` etc).

Answer

```
#include <iostream>
using namespace std;
int main(){
   char c;
   cin >> c;
   if (                    )
      cout << "An upper case character\n";
   else if (                    )
      cout << "A lower case character\n";
   else if (                    )
      cout << "A digit\n";
   else
      cout << "Special character\n";
    return 0;
}
```

Answer

```
#include<iostream>
using namespace std;
int main(){
   char c;
   cin >> c;
   if (c>='A' && c<='Z')
    cout << "An upper case character\n";
   else if (c>='a' && c<='z')
    cout << "A lower case character\n";
   else if (c>='0' && c<='9')
    cout << "A digit\n";
   else
    cout << "Special character\n";
    return 0;
}
```

String

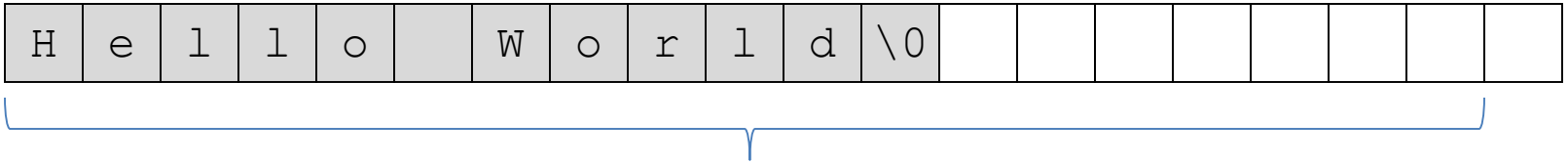
`cstring` VS `string` Object

- In C++, there are two types of strings.
- **`cstring`**: inherited from the c language.
- `string`: class defined in <string> library.

cstring

- A cstring is a `char` array, which is terminated by the end-of-string sentinel (or null character) **`'\0'`** .
- A character of array of size **`n`** may store a string with maximum length of **`n-1`**.
- Consider the following code:

```
char str[20]="Hello World";
```



This character array may store a string with maximum length of 19.

Rule of Safety

- Declare a string with one more character than needed.
- E.g.

```
char studentID[9];  //51234567
char HKID[11];     //a123456(7)
```

cstring - Declaration and Initialization

- String variable can be declared in one of two ways:
- Without initialization
  - char identifier[required size+1]
  - E.g.
    - `char name[12];  // name with 11 characters`
    - `char address[50];  // address with 49 characters`
- With initialization
  - char identifier[]=string constant
  - E.g.
    - `char name[]="John";  //name with 5 characters`
    - `char choice[]="a,b,c,d,e"; //choice with 10 characters`
- However, you cannot initialize a string after declaration

```
char name[10];
name="john";
```

error C2440: '=' : cannot convert from 'const char [5]' to 'char [10]'

cstring - Reading and Printing

```
#include<iostream>
using namespace std;
int main(){
    char word[20];
    //read a string
    cin >> word;
    cout << word; //print a string
    return 0;
}
```

The array `word` can store **20** characters but we can only use up to 19 character (the last character is reserved for `null` character).

No need to specify the brackets [] when printing.

Reading a Line of Characters

- `cin >> str` will terminate when whitespace characters (space, tab, linefeed, carriage-return, form-feed, vertical-tab and newline characters) is encountered.

- Suppose "hello world" is input:

```
char s[20];
cin >> s; //read "hello"
cin >> s; //read "world"
```

- How to read a line of characters?

cin.get()

```
#include <iostream>
using namespace std;

int main(){
   char c;
   do {
      cin.get(c);
      cout << c;
   }while (c!='\n');
    return 0;
}
```

`cin.get()`

- `get()`: member function of `cin` to read in one character from input.
- `>>` skips over whitespace but `get()` does not.
- Syntax:

```
char c;
cin.get(c);
```

cin.getline()

- Predefined member function of `cin` to read a line of text (including space).
- Two arguments as input:
  - `cstring` variable to receive the input.
  - size of the `cstring`.

```
#include <iostream>
using namespace std;
int main(){
     char s[20];

     cin.getline(s,20);
     cout << "\"" << s << "\"" << endl;

      return 0;
}
```

`cin.getline()`

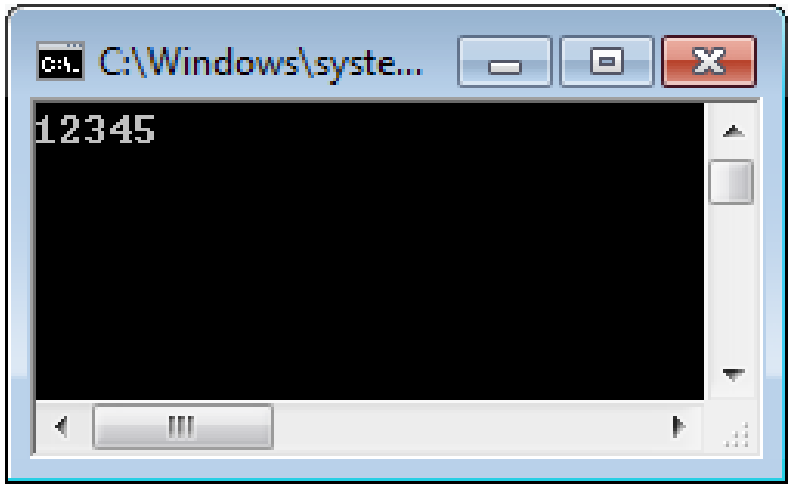
- What if
  - Input is longer than the string variable?
  - End of the source characters is reached?
  - Error occurred?

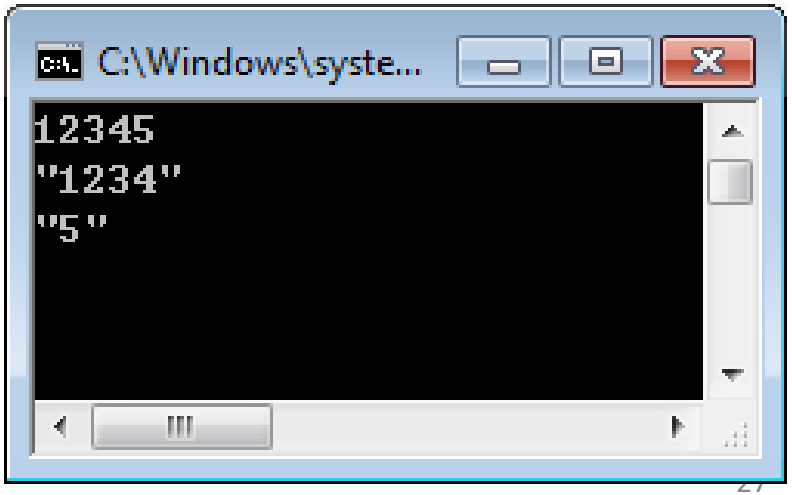
- Internal state flags (`eofbit`, `failbit`, `babbit`) of `cin` object will be set.

- To reset those flags, call method `clear()` of `cin`. E.g. `cin.clear();`

Example

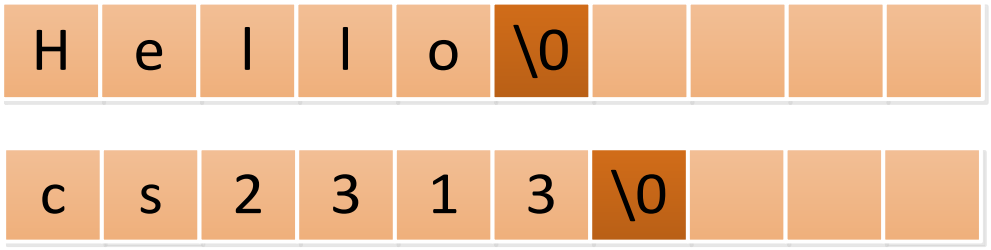
```
#include <iostream>
using namespace std;
int main(){
    char s[5];
    while (1){
        cin.getline(s,5);
        cin.clear();
        cout << "\"" << s << "\"" << endl;
    }
    return 0;
}
```





The Null Character '\0'

- The null character, '\0', is used to mark the end of a cstring.
- '\0' is a single character (although written in two symbols).
- It is used to distinguish a cstring variable from an ordinary array of characters (cstring variable must contain the null character).

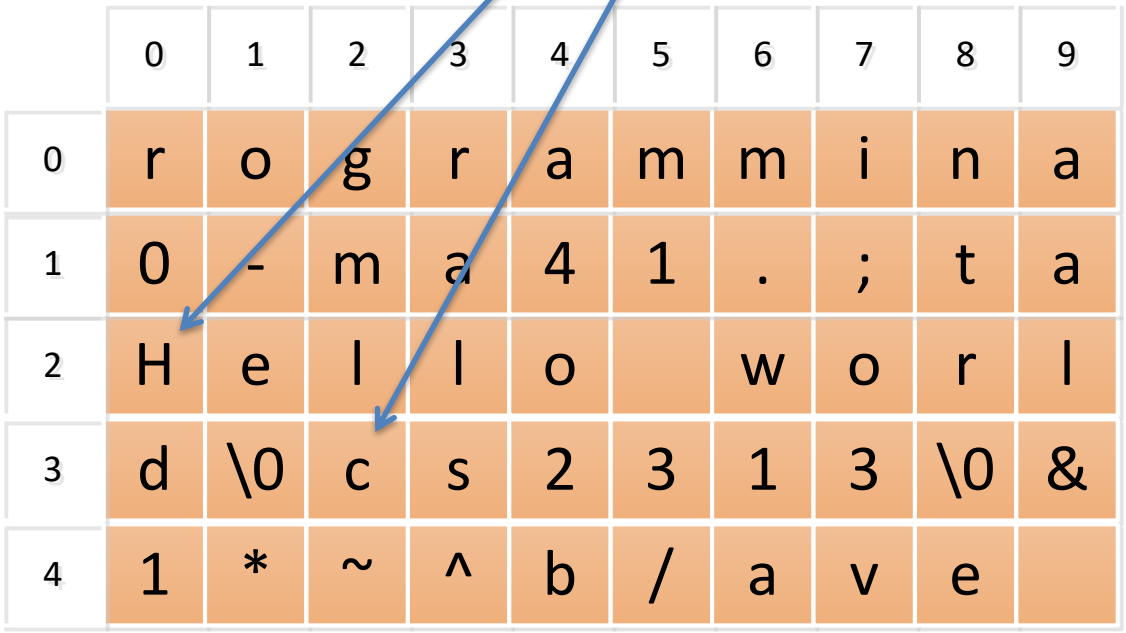


Why Need '\0'?

- `cstring` is stored in main memory continuously.
- Only the starting address of the `cstring` is stored in `cstring` variable.

```
char s1[]="Hello World"; // s1=20
char s2[]="cs2313";      // s2=32
```

- `'\0'` indicates the end of `cstring`.



Why Need '\0'?

- When a `cstring` variable is passed to a output function, i.e. `cout`, the function will print all the memory content until '\0' is encountered.

r	o	g	r	a	m	m	i	n	a
0	-	m	a	4	1	.	;	t	a
H	e	l	l	o		w	o	r	l
d	\0	c	s	2	3	1	3	\0	&
1	*	~	^	b	/	a	v	e	

Passing String to Functions

- Example:
  - Write a function to count the number of occurrences of a character in a string.
- Functions
  - `count`: given a character and a string as input, return the number of occurrences of the character in the string.
  - `main` function: call `count` function.

Function - count

The size 100 is optional

```
int count(char s[100], char c){
   int occurrence=0;
   int i; //counter
   for (i=0; s[i]!='\0'; i++){
      if (s[i]==c)
         occurrence++;
   }
   return occurrence;
}
```

The `main` Function

```
int main(){
   char str[]="Hong Kong is a very good place to live";
   int count1 = count(str, 'o');
   cout << "count = " <<count1<< " \n";

    return 0;
}
```

Another Example - String Length

```
#include<iostream>
using namespace std;

int strlength(char s[]){
    int i=0;
    while (s[i]!='\0')
        i++;
    return i;
}

int main(){
    char s[20]="Hello world";
    cout << strlength(s);
    return 0;
}
```

cstring - length

- To count the number of characters contained in a string
- We can use a loop to read characters one by one from the string until a null character ('\0') is read
- For each character read, increment a counter by one

```
#include<iostream>
using namespace std;

int main(){
   char s[20]="Hello world";
   int len;
   int i=0;
   while (s[i]!='\0'){
       i++;
   }
   len=i;
  return 0;
}
```

cstring-copy

- Suppose there are two strings, `str1` and `str2`.
- `str1` is initialized to "`Hello world`".
- We want to copy the value of `str1` to `str2`.
  - We should **NOT** use the following expression to copy array elements:

```
str2=str1;
```

  - Similarly, we should **NOT** use the following expression to compare strings:

```
if (str1==str2) ......
```

Idea

- Use a loop to read characters one by one from `str1` until a null character (`'\0'`) is read .
  - Copy each character to the corresponding position of `str2`.

- Put a null character at the end of `str2`.

The Program

```
#include<iostream>
using namespace std;
int main(){
   char str1[20]="Hello world";
   char str2[20];
   int i;

   /*be aware of the boundary condition!*/
   for (i=0; i<19 && str1[i]!='\0'; i++) {
       str2[i]=str1[i];
   }
   str2[i]='\0';
   cout << str2;
    return 0;
}
```

Common `cstring` Functions in `<cstring>`

Function	Description	Remarks
`strcpy(dest,src)`	Copy the content of string src to the string dest.	No error check on the size of dest is enough for holding src.
`strcat (dest,src)`	Append the content of string src onto the end of string dest.	No error check on the size of dest is enough for holding src.
`strcmp(s1,s2)`	Lexicographically compare two strings, s1 and s2, character by character.	0: s1 and s2 is identical >0: s1 is greater than s2 <0: s1 is less than s1.
`strlen(string)`	Returns the number of characters (exclude the null character) contain in the string.	

Examples

```
#include <iostream>
using namespace std;
int main(){
    char s1[50]="String 1";
    char s2[50]="cs2311 computer programming";
    cout <<"s1:"<< s1 <<endl;
    cout <<"s2:" << s2<<endl;

    strcpy(s1,"cs2313");
    cout <<"s1:"<< s1 <<endl;

    strcat(s1," computer programming");
    cout <<"s1:"<< s1 <<endl;

    cout <<"Compare s1 vs s2:" << strcmp (s1,s2) <<endl;
    cout <<"Comprare s2 vs s1:" << strcmp (s2,s1) <<endl;
    cout <<"Compare s1 vs \"cs2313 computer programming\":"
<< strcmp (s1,"cs2313 computer programming") <<endl;
    return 0;
}
```

Example

- Write a program to print a word backward.
- Example:
  - Input:
    - `hello`
  - Output:
    - `olleh`

The Program

```
#include<iostream>
using namespace std;

int main(){
   char    ____; //define an array input with size 50
   int n; //length of str
   int i;

   cin >>____;
   n=_______; //compute string length
   for (i=____; i > 0 ;i--)
      cout <<____ ;
    return 0;
}
```

Answer

```
#include<iostream>
using namespace std;

int main(){
   char input[50]; //define an array input with size 50
   int n; //length of str
   int i;

   cin >> input;
   n=strlen(input); //compute string length

   for (i=n-1; i>=0; i--)
      cout << input[i];
   return 0;
}
```

Exercise

- Write a program to let the user to input a line of string.
- The program should reverse the case of the input characters and print the result.
  - Lowercase characters are changed to uppercase.
  - Uppercase characters are changed to lower case.
- Example input/output:

```
Hello World
hELLO wORLD
```

The Program

```
#include<iostream>
using namespace std;
int main(){
   char s[20];
   int i;
   cin.getline(s,20);

   for (i=0; s[i]!='\0'; i++){
      if (                  ) //uppercase letter
         cout <<              ; //convert to lowercase
      else if (s[i]>='a' && s[i]<='z') //lowercase letter
         cout <<              ; //convert to uppercase
      else //other characters
         cout <<   ;
   }
   return 0;
}
```

Answer

```
#include<iostream>
using namespace std;
int main(){
   char s[20];
   int i;
   cin.getline(s,20);

   for (i=0; s[i]!='\0'; i++){
    if (s[i]>='A' && s[i]<='Z') //uppercase letter
        cout << (char)(s[i]+'a'-'A'); //convert to
   lowercase
    else if (s[i]>='a' && s[i]<='z') //lowercase letter
        cout << (char)(s[i]+'A'-'a'); //convert to
   uppercase
    else //other characters
        cout << s[i]  ;
   }
    return 0;
}
```

Summary

- Character is an integer.
- String is an array of character terminated by `'\0'`.
- Accessing string content is similar to access individual element of an array.
- String copy, comparison, concatenate can be done by functions provided by `<cstring>` library (also defined in `<iostream>`).