CS 31 Discussion

ABDULLAH-AL-ZUBAER IMRAN

WEEK 6: C-STRINGS, STRUCT, ENUM AND PROJECT4

Recap

- Functions
  - Parameter passing: pass by reference
- Strings
  - Letter to digit and vice-versa
- Array
  - Declaration and initialization
  - 1D and 2D arrays
  - Arrays in a function

Discussion Objectives

Review and practice things covered during lectures

- C Strings
- Struct
- Enum

- Coding examples
- Project4
  - Required Functions
  - Important Points

Programming Challenge

Time for you to ask questions!

Functions for C Strings

`#include <cstring>`

Operation	What it does
`strlen(s)`	Returns the length of `s`, not counting `‘\0’`.
`strcpy(t,s)`	Copies the string `s` to `t`. (Notes: `t=s` won’t do the job. Also, `strcpy` doesn’t do the size check for you. You must make sure there’s enough space in t yourself.)
`strncpy(t,s,n)`	Copies the first n characters of `s` to `t`. (Note: No size check.)
`strcat(t,s)`	Appends `s` to the end of `t`. (Notes: No size check. `t += s` won’t do the job.)
`strcmp(t,s)`	Compares `t` and `s`. Returns `0` if they are equal, something greater than `0` if `t > s`, and something less than `0` if `t < s`. (Note: `t == s` or `t < s` won’t do the job.)

C Strings

Two alternatives to traverse a C string.

```
char s[10] = "HOWAREYOU";
for (int k = 0; t[k] != '\0'; k++)
  cout << t[k] << end;
```

```
#include<cstring>
...

char s[10] = "HOWAREYOU";
for (int k = 0; k < strlen(s); k++)
  cout << t[k] << end;
```

C Strings

An array of C strings

```
char s[10][20];
```

In s, we can store up to ___ C strings, and each C string can be at most ___ characters long.

10

19

```
s[3];
// refer to the string in position 3

s[3][5];
// refer to the letter in position 5 of the string in position 3
```

C Strings

Convert a C string into a C++ string, and vice versa.

```
char cs[10] = "hello";
string cpps;
cpps = cs;

string cpps = cs;
string cpps(cs);
```

C string to string

```
char cs[10];
string cpps = "hello";
strcpy(cs, cpps.c_str());
```

string to c string

You cannot:

```
char cs[10] = cpps.c_str();
```

```
char cs[10];
cs = cpps.c_str();
```

C Strings

Question: Will this code compile? What's the output?

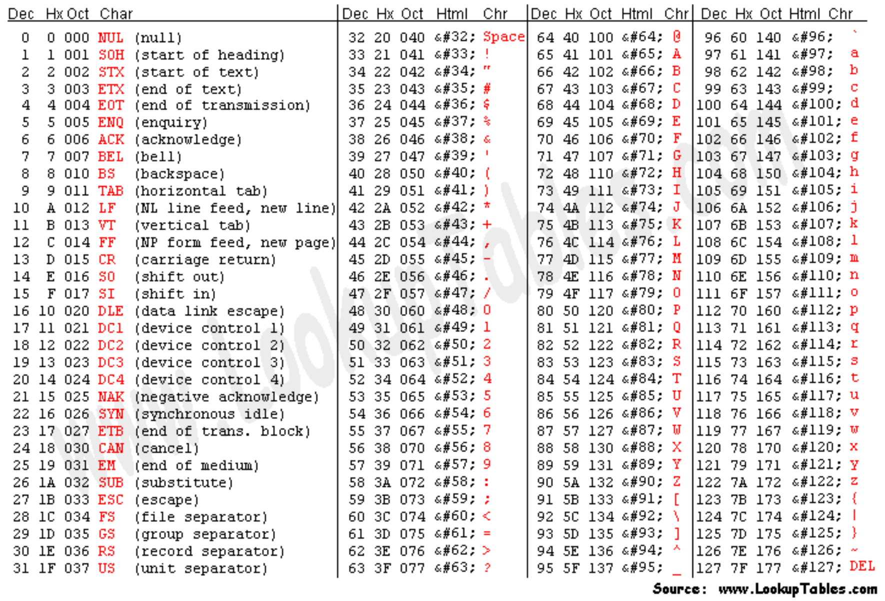
```
#include <iostream>
#include <cstring>

using namespace std;
int main () {
  char a[]= "sup";
  char A[] = "SUP";

  cout << strcmp(a, A) << endl;
  cout << strcmp(A, a) << endl;
}
```

Output:

```
32
-32
```



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

Source: www.LookupTables.com

ASCII table

Struct

Structs are objects in C++ that represent "data structures", or variables, functions, etc. that are organized under a categorizing identifier.

Data Members are variable components of a given struct; they can be of any variable type.

```
struct <structName> {
  <member1_type> <member1_name>;
  <member2_type> <member2_name>;
  // ...etc.
}; // Remember the semicolon!
```

```
struct Student {
  string name;
  int id;
  string email;
  char grade;
};
```

Struct

```
struct Student {
  string name;
  int id;
  string email;
  char grade;
};
```

```
const int NUM_STUDENTS = 32;
Student st
Student students[NUM_STUDENTS];
```

The element/member selection operation (.)

```
st.name = "Joe Bruin";
// st's name is set to "Joe Bruin"
students[10].id = 123456789;
// the 10-th Student in students array is assigned an ID
cout << st.grade << endl;
// print the grade of st
cout << students[0].email << endl;
// print the 0-th Stduent's email address
```

Member Functions (Method)

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

struct Student {
  string name;
  int id;
  string email;
  char grade;

  Student() {
    name = "Jane Doe";
    id = 0;
    email = "aa@a.com";
    grade = 'A';
  }
  void printName() {
    cout << name << endl;
  }
};
```

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

struct Student {
  string name;
  int id;
  string email;
  char grade;

  Student() {
    name = "Jane Doe";
    id = 0;
    email = "aa@a.com";
    grade = 'A';
  }
};

void Student::printName() {
    cout << name << endl;
}
```

```
int main() {
  Student p;
  p.printName();
}
```

```
Jane Doe
```

Enumeration

- A user-defined data type consisting of multiple constants. An enumeration is defined using the keyword “enum”

enum <name> {<const1>, <const2>, <const3>};

Example: *enum quarter {winter, spring, summer, fall};*

By default, winter is 0, spring is 1 and so on. You can change the default values.

quarter current;

current = winter;

cout << “Next Quarter: ” << current + 1;

What will be printed?

Project4: Seven Functions

int locateMaximum(const string array[ ], int n);
→Return the index of the largest item found in the array or -1 if n <= 0.

bool hasDuplicates(const string array[ ], int n);
→If there is a value that is repeatedly found in the array, return true otherwise false or if n <= 0 return false.

---

int countSs(const string array[ ], int n);
→Return the number of 's' or 'S' characters found inside each of the elements of the passed array or if n <= 0 return -1.

int shiftLeft(string array[ ], int n, int amount, string placeholderToFillEmpties);
→ Adjust the items found in the array, shifting each value to the right by amount parameter, filling the resulting first amount elements of the array with the placeholder parameter and returning the number of times the placeholder value was used after all the shifting has been performed or -1 if the size or amount is less than zero.

bool isInDecreasingOrder(const string array[ ], int n);
→If every value in the array is smaller than the one that precedes it, return true otherwise false or if n <= 0 return false.

bool matchingValuesTogether(const string array[ ], int n);
→If all the duplicated values found in the array are located one right after the other, return true otherwise false.

int divide(string array[ ], int n, string divider);
→Rearrange the elements of the array so that all the elements whose value is < divider come before all the other elements, and all the elements whose value is > divider come after all the other elements. (Yes, there might be numerous correct rearrangements that are valid.) Return the position of the first element that, after the rearrangement, is not < divider, or 0 if there are none.

Important Notes

Time due: 9:00 PM Wednesday, February 19th

You cannot use any function templates from the algorithms portion of the Standard C++ library.

Your implementations must *not* use any global variables whose values may be changed during execution.

Your program must build successfully under both Visual C++ and either clang++ or g++.

You should write useful comments for any non-obvious code.

A zip containing two files: array.cpp and report.doc/report.txt.

Thanks!

Questions?

Today’s discussion slides can be found at

https://github.com/zubaerimran/W20-CS31-1J/blob/master/week6/winter20_cs31_w6.pdf

Some of the materials presented have been taken from earlier TA discussions