C-strings

Anna Simon

September 11, 2015

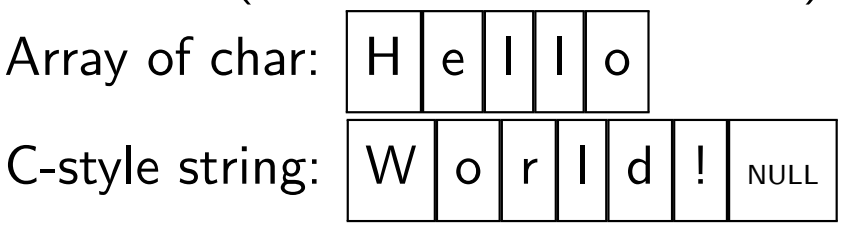
Sometimes one needs to read some information from an input that looks for example like this:

γ(^{152}Sm)

E_{level}	E_γ[#]	I_γ[#]	Mult.[gh]	δ[g]	α	$I_{\gamma+ce}$	Comments
121.7818	121.7817 *3*	100	E2		1.155		B(E2)(W.u.)=145.0 *16*
366.4793	244.6974 *8*	100	E2		0.1073		B(E2)(W.u.)=209.5 *22*
684.751	562.98 *3*	100.0 *19*	E2		0.00941		B(E2)(W.u.)=33.3 *12*
	684.85 *20*		E0			1.30 *14*	ϱ^2(E0)=0.051 5. **$I_{\gamma+ce}$:** from 9.3-h Eu ε decay relative to Iγ(563γ).
706.928	340.45 *3*	100	E2		0.0382		B(E2)(W.u.)=240 *4*
810.453	125.64 *7*	0.599 *22*	[E2]		1.034		B(E2)(W.u.)=170 *12*
	444.00 *3*	34.8 *13*	E2		0.01772		B(E2)(W.u.)=18.0 *12*
	688.670 *5*	100.0 *6*	E0+M1+E2		0.0434 *13*		B(E2)(W.u.)=5.7 *4*; B(M1)(W.u.)=1.5×10^{-5} *7* **δ:** δ(E2/M1)=+19 *+5-4* (1982La26); I(ce(K))(E0)/I(ce(K))(E2)=6.5 3 (^{152}Eu ε decay (13.517 y)). **α:** from ^{152}Eu ε decay (13.517 y).
	810.451 *5*	37.0 *3*	E2		0.00393		B(E2)(W.u.)=0.94 *6*
963.358	152.77 *16*	0.0126 *11*	[E1]		0.0872		B(E1)(W.u.)=0.000225 *27*
	278.7 *3*		[E1]		0.0177		**I_γ:** weak (^{152}Eu ε decay (9.3116 h)).
	841.570 *5*	100.0 *18*	E1		0.00144		B(E1)(W.u.)=0.0106 *9*
	963.367 *5*	82.3 *13*	[E1]		0.00111		B(E1)(W.u.)=0.0058 *5*
1022.970	212.43 *11*	14.4 *4*	E2		0.1706		B(E2)(W.u.)=2.5×10^{+2} *4*
	316.13 *13*	7.00 *4*	(E2)		0.0478		B(E2)(W.u.)=17 *3*
	656.489 *5*	100.0 *15*	E2+M1+E0		0.0568 *20*		B(E2)(W.u.)=5.0 *+10-7*; B(M1)(W.u.)=9.0×10^{-4} *25* **δ:** δ(E2/M1)=2.1 3 (1982La26, Coulomb excitation). I(ce(K))(E0)/I(ce(K))(E2)=10.0 6 (^{152}Eu ε decay (13.517 y)). **α:** from ^{152}Eu decay (13.517 y).
	901.19 *5*	59.2 *17*	E2		0.00311		B(E2)(W.u.)=0.74 *12*

C-style string

C-style string is an array of `char` ended by a null terminator, i.e. a '\0' character (which has ASCII code 0).



Declaration of C-style string

```
1  //uses only part of the string
2  char sStr1[20] = "string"; // \0 automatically added at the end
3
4  //string of length 0 (just null terminator)
5  char sStr2[20] = "";
6
7  //compiler will figure out the length
8  char sStr3[] = "string";
9
10 cout << sStr3 << endl;
```

The C-style string has to end with a null terminator, if the user overwrites the null terminator, CPU won't know where the string ends. Printing out the string may lead to unexpected behavior. Most likely the program will print everything in the adjacent memory until reaching a zero value.

Accessing C-string elements

Since C-style string is an array, its elements can be accessed individually:

```
1 char sString[] = "string";
2
3 //replace a single character
4 sString[1] = 'p';
5
6 //print out the whole C-string to the screen
7 cout << sString;
```

NOTE:

A single char (eg. ‘a’) is typically only allocated one byte, but the equivalent string (eg. “a”) is allocated two bytes — one for the char, and one for the null terminator.

Reading in a C-string form user's input

```
1 char sString[255];
2 cin >> sString;
3 cout << "You entered: " << sString << endl;
```

- Requires declaration of the string with a fixed size
- If user types in a longer string → buffer overflow
- If the input string contains space, `cin` will stop there

Use `cin.getline()` to read until '\n' or a specified character or a fixed length:

```
1
2 //dangerous may overflow sString
3 cin.getline(sString);
4
5 //read 254 characters  + \0
6 cin.getline(sString, 255);
7
8 //read up to 29 characters or until you find a '-' (dash)
9 cin.getline(sString, 30, '-');
```

- Spaces are read as single characters

Manipulating a C-style string

E.g.: copy one string into the other:

```
1 char sSource[] = "Copy this!";
2 char sDest[50];
3 strcpy(sDest, sSource); //copy string
4 strncpy(sDest, sSource, 49); // copy only 49 chars
```

The first option may result in buffer overflow if the destination string is not large enough.
The second one will copy 49 characters + a null pointer into the destination string.

Manipulating C-style strings

`strcpy(sDest,sSrc)` - copies a string from a destination to the source string

`strncpy(sDest,sSrc,nChar)` - copies a specified number of characters from one string to the other

`strcat(sDest,sSrc)` — Appends one string to another (dangerous)

`strncat(sDest,sSrc,nChar)` — Appends one string to another (with buffer length check)

`strcmp(sStr1,sStr2)` — Compare two strings (returns 0 if equal)

`strncmp(sStr1,sStr2,nChar)` — Compare two strings up to a specific number of characters (returns 0 if equal)

`strlen(sStr)` — Returns the length of a string (excluding the null terminator)

`sprintf(sStr,`*format*`,...)` - composes a C string and stores it in sStr buffer, e.g. `sprintf(sStr,"%d + %d = %d", 2,3,5)`

NOTE: Another option is to use string class (std::string) from the `string` header. We'll discuss it later.

sprintf format parameters (some examples)

```
1 int age=20;
2 char name[20]="John";
3 sprintf(sStr,"%s is %d years old.",name,age);
4 cout << sStr;
```

specifier	Output	Example
d or i	Signed decimal integer	392
u	Unsigned decimal integer	7235
f	Decimal floating point, lowercase	392.65
F	Decimal floating point, uppercase	392.65
e	Scientific notation (mantissa/exponent), lowercase	3.9265e+2
E	Scientific notation (mantissa/exponent), uppercase	3.9265E+2
g	Use the shortest representation: %e or %f	392.65
G	Use the shortest representation: %E or %F	392.65
c	Character	a
s	String of characters	sample
%	%% character will write a single % to the stream.	%

Testing the type of character and converting it to a numerical type

From <cstdlib>

atoi	Convert string to integer
atol	Convert string to long integer
atof	Convert string to double

From <cctype> - examples

isalnum	Check if character is alphanumeric
isalpha	Check if character is alphabetic
isdigit	Check if character is decimal digit
islower	Check if character is lowercase letter
isupper	Check if character is uppercase letter
ispunct	Check if character is a punctuation character
isspace	Check if character is a white-space
tolower	Convert uppercase letter to lowercase
toupper	Convert lowercase letter to uppercase