2510 : C Lecture 4

January 16, 2017

DESIGN BY CONTRACT

A function can be regarded as the contract between its user and its implementers.
- only will work if pre condition is met

↳ **pre condition**: condition that must be met before the function will work correctly
(ie. it meets the post conidition)

↳ **post condition**: the condition that function promises to satisfy (if the precondition is met)

* They are usually specified as comments
* Making pre- & post-conditions explicit can reduce bugs

ERRORS VS. BUGS

errors: are exceptional conditions that we expect may happen & typically we write code to handle them
(ie. failure to open a file)

bugs: are exceptional conditions that should never happen
(ie. failure to meet precondition)

° in C, we can use the assert macro to check precondition for debugging purposes

```
#include <assert.h>
int arr_max (const int a [], size_t n) {
    size_t i;
    int max;
    assert (n>0);   *same code as /*an assertion*/
```

if assertion fails, program terminants with a message

2510 : C Lecture 4

- We can turn off the assertion once we finish debugging

cmd : gcc -ansi -W -Wall -pedantic -DNDEBUG

Strings

- There are no separate string types in C.
- A string is just an array of characteristic terminated by the null character (denoted by '\0' or 0)

Examples

1) "HELLO" : .....
   This already has a null character at the end. To store it we need an array of at least 6 characters. However, we stay that this string has a length of 5.
   (ie. length does not count the null character)

2) char s[6] = {'h', 'e', 'l', 'l', 'o', '0'};
   ↳ in this case, we can change the character's cms

3) char s2[] = {'h', 'e', 'l', 'l', 'o', '0'};
   /* compiler counts the characters */

4) char s3[] = "hello"
   ↳ compare with int n = 123

STANDARD IDIOM TO PROCESS A STRING

```
s_string ( )
size_t i;
for (i = 0; s[i]!='\0'; i++)
    /* process s[i] */
```

Note: this doesn't process the null character

Examples:

① Length of String

```
size_t str_length (const char s []) {
    size_t i;
    for (i = 0; s[i] != '\0'; i++)
        ;   i is just used as a counter
    return i;
}
```

② Changing a String to all upper case

```
void str_uppercase (char s []) {
    size_t i;
    for (i = 0; s[i] != '\0'; i++)      // toupper('a') = 'A'
        s[i] = toupper(s[i]);           // toupper('*') = '*'
}                  ( #include <ctyle.h>
```

↳ str_uppercase ("hello") leads to undefined behavior
we cannot modify a string constant

∴ we store it in an array then pass it on

... Lecture 4 ||

- Examples cont.

③ Testing whether a string consists entirely of alphabets

```
int str_all_alpha (const char s[]) {
    size_t i;
    for (i = 0; s[i] != '\0'; i++)   /* '\0' means null */
        if (! isalpha (s[i])
            return 0;
    return 1;
}
```

once we see non-alphabet stop the loop immediately

need to include #include <ctype.h>

④ Looking for a character in a string

```
size_t str_find (const char s[], int x) {
    size_t i;
    for (i = 0; s[i] != '\0'; i++)
        if (s[i] == x)
            return i;
    return -1;
}
```

Why is x an int?
It is traditional to declare it as an int.

what?

...Lecture 4

- Examples cont.

⑤ Making a copy of a string

```
// we will need 2 arrays
// (destination, source which can be const)

void str_copy (char dest [], const char src []){
    size_t i;
    for(i = 0; src [i] != '\0'; i++) // Exit loop when src = null
        dest [i] = src [i];
    dest [i] = '\0';                 //terminate dest w/ null
```

How do we actually call this function?

```
char s [100];
str_copy (s, "hello");
```

The caller is responsible for ensuring that dest is large enough to store the src
↳ either won't store or buffer overflow

... Lecture 4 ||

Some string Functions in the Standard C Library

```
#include <string.h>
```

① `size_t strlen (const char s[]);`
returns the length of the string s

② `strcpy (dest, src);`
strcpy used to copy a string

SAFER VERSION

`strncpy (dest, src, n);`
↳ n is the maximum number of characters to copy
* need to create own null char.

example how to use strncpy:

```
char dest [100];
strncpy (dest, s, 99);   Leave a space for null char
        /* source */
dest [99] = '\0';
```

③ `int strcmp (const char s1[], const char s2[]);`
Used to compare s1 and s2:
returns NEGATIVE if s1 < s2
0 if s1 == s2
POSITIVE if s1 > s2

strcmp ("hell", "hello") returns negative
strcmp ("hello", "hell") positive

... Lecture 4

- Standard C Library cont.

③ example of strcmp

```
char s[] = "hello";
str_uppercase(s);
CHECK (strcmp(s, "HELLO") == 0);
```

cannot use == to compare strings

- Tips for commenting code

```
/*
  /*
  */
*/
```

comments can't be nested, once closing brackets appear, turns off all comment

```
#if 0
...
#endif
```

Lab 1 Feedback

- [ ] use indentation