Strings (Contd.)

ESC101: Fundamentals of Computing

Nisheeth

Announcements

- Next class on Saturday, 29th Feb at noon here in L20
- Tutorial on Friday, 28th Feb as usual

Strings

- String: A sequence of characters enclosed in double quotes " "
- A string can be declared and initialized as

```
char str[50] = "Hello World";
```

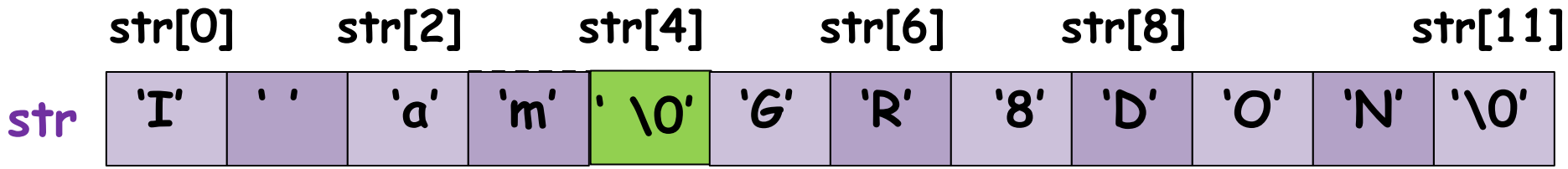
- Internally, a string is stored as a char array whose last element is '\0'

```
char str[50] = {'H','e','l','l','o',' ','W','o','r','l','d','\0'};
```

Equivalent to "Hello World"

The null character

Null character '\0' ends the string




```
char str[]="I am GR8DON";
str[4]='\0';
printf("%s", str);
```

Output

I am

Can still print all elements in the char array…

```
int i;
for (i=0; i < 11; i++) {
        putchar(str[i]);
}
```

Output

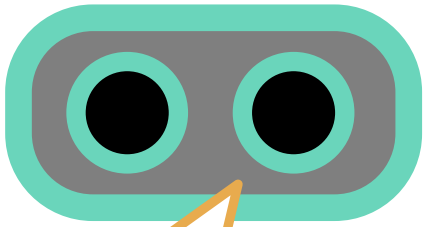
I amGR8DON

The character '\0' may be printed differently on screen depending on terminal settings.

Operations on Strings

Some common operations on strings

- Compute the *length* of a string.
- *Copy one string into another string variable*
- *Concatenate* one string with another.
- *Search* for a *substring* in a given string.
- *Reverse* a string
- Find first/last/k-th occurrence of a *character* in a string
  - … and more
- Case sensitive/*insensitive* versions

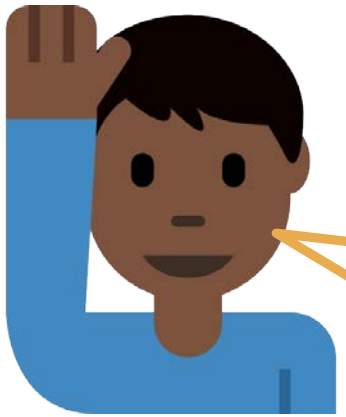


Can solve all these problems using loops, looking at the string char by char

Or by using pre-defined functions in a header file called string.h ☺

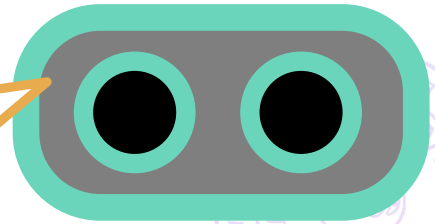
Computing the length of a string


```
char str[10] = “Hello”;
int i = 0;
while(str[i] != ’\0’)
    i++;
printf(“Length of %s is %d”,str,i);
```



How to find the length of not a fixed string but *any* string provided by user?

Count the length char by char using getchar in a loop, or use strlen function in string.h



Read a string and also compute its length


```
int main() {
    char str[100];
    char ch;
    int i = 0;
    ch = getchar();
    while(1){
        if(ch==‘\n’) break;
        str[i] = ch;
        ++i;
        ch = getchar();
    }
    str[i] = ‘\0’;
    printf("Length of %s is %d",str,i);
    return 0;
}
```

Will store the length

Read the first character

I will enter a string and end it with newline. Please store it in a string named str and compute its length

If found a newline, break

Not a newline. Store the read character at index i of str

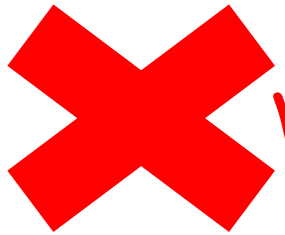
Read the next character

Let’s put ‘\0’ in the end to mark the end of string

Copying a string

- We cannot copy content of one string variable to other using assignment operator

```
char str1[] = "Hello";
char str2[] = str1;
```



Array type is not assignable.

C Pointers needed (will see this later)!

  - This is true for any array variable.
  - Error because array initializer must be a list (comma separated values in {}) or a string.
- We need to do element-wise copying

Copying a string element-by-element

- Goal: Copy contents of string src into string dest.
- Declare dest with size at least as large as src.
- Use a loop to copy elements one-by-one
- Note the use of '\0' for loop termination

```
char src[100] = "Hello World";
char dest[100];
int i;
for (i = 0; src[i] != '\0'; i++)
    dest[i] = src[i];
dest[i] = '\0';
```

Part of the loop

Not part of the loop

string.h

- Many string operations are already implemented in string.h
- It is a header file ("h" for header) with various functions on Strings
- strlen(s): returns length of string s (without '\0')
- strcpy(d, s): copies s into d
- strcat(d, s): appends s at the end of d ('\0' is moved to the end of result)

string.h

- strcmp(s1, s2): return an integer less than, equal to, or greater than zero if s1 is found, respectively, to be less than, to match, or be greater than s2.

- Example:

```
char str1[] = "Hello", str2[] = "Helpo";
int i = strcmp(str1,str2);
printf("%d", i);
```

- Prints the value 'l'-'p' which is -4.

string.h

- strncpy(d, s, n)
- strncat(d, s, n)
- strncmp(d, s, n)
  - restrict the function to "n" characters at most (argument n is an integer)
  - first two functions-- Truncate the string s to the first "n" characters.
  - third function-- Truncate the strings d, s to the first "n" characters.

```
char str1[] = "Hello", str2[] = "Helpo";
printf("%d",strncmp(str1,str2,3));
```

0

string.h

- strcasecmp, strncasecmp:
  case insensitive comparison.
- Example:

```
char str1[] = "HELLO", str2[] = "Helpo";
int i = strcmp(str1,str2);
int j = strcasecmp(str1,str2);
printf("%d %d", i, j);
```

-32 -4

- strcmp gives -32 because 'E' < 'e' .
  - 'E'-'e' = -32 .
- strcasecmp finds the first 3 characters the same (ignoring case) gives -4 because 'l' - 'p' = -4

string.h

- Many more utility functions.

- strupr(s) : converts lower to upper case.
- strlwr(s) : converts upper to lower case.
- strstr(S,s) : searches string s in string S (example: strstr("Hello","ll");). Returns a pointer(memory address) to the first occurrence.

- All functions depend on '\0' as the end-of-string marker.

Another useful string function

atoi(str): converts a string into integer, e.g. atoi("123") will return 123 (integer). So we can write int a = atoi("123");

atoi(str) will keep reading the string str **until it finds a non-digit character** in str and will return the integer containing the read digit characters in str

atoi("12abc3") will return 12

If no digit symbols found in str, atoi(str) will return 0

If the read integer is larger than the range of integers, garbage will be returned

Other function atof(str), atol(str) – return float and long

Can also convert int/long to string (itoa, ltoa, etc)

EOF (end of file)

A very special non-character – cannot be printed

Signals end of input (no more characters in input)

stdio.h gives you a named constant EOF for convenience

Has no ASCII value – but internally stored as -1

Recall characters have ASCII values from 0 to 127 only

getchar() will read EOF as a character but not scanf/gets

NULL and \n are valid characters with proper ASCII values

Be careful, do not confuse EOF with NULL and \n.

Strings: Summary

- Strings are character arrays
- The last character is '\0' (null character) and marks end of string
- Many direct operations (e.g., assignment) not possible for strings. Have to be done element-wise (e.g., using a loop)
- string.h contains many useful functions (so you don't need to write functions for basic operations, such as finding the length of string, copying one string into another, etc)