Structured Programming
- String

Donglong Chen

With Thanks to Dr. Xin Feng and Dr. Haipeng Guo

Outline

- String concept
- Input and output a string
- String functions

String

- A string is an array of chars terminated with a null character, ‘\0’.
- String declaration
  - char `str[10];`
  - This is a string that contains at most 9 characters

Initialize A String

- There are several ways to initialize a string
  - `char str[20] = {'H', 'e', 'l', 'l', 'o', '\0'};`
  - `char str[20] = "Hello";` // '\0' is automatically set to str[5]
  - `chat str[] = "Hello";` //str's size is 6
  - `char str[20]; str = "Hello";` (exception gcc)

Exercise

- Are the following initialization correct?
  - `char str[10] = {'y', 'e', 's'};`
  - `char str[10] = "Good Morning";`
  - `char str = "Hi";`
  - `char str[] = 'Good';`
  - `char str[] = "O";`

Output (Write) A String

- To output a string to the screen, two functions can be used
  - printf
  - puts
- Both functions are defined in stdio library, so we need to include stdio.h in the program

printf

- %s must be used in the format string to indicate a string is to be printed
  - E.g.,

```
char str[] = "This is a message";
printf("%s", str);
```

- printf stops printing when it meets '\0'.

Exercise

- What is the output of the following code

```
char str[15]="unix and c";

printf("%s", str);
printf("\n");
str[6]='\0';
printf("%s", str);
printf("\n");
str[2]='%';
printf("%s", str);
printf("\n");
```

puts

- Simpler than `printf`
  - E.g.,

```
char str[] = “This is a message”;
puts(str);
```

- puts stops printing when it meets ‘\0’.

Input (Read) A String

- To input a string from the keyboard, two functions can be used
  - scanf
  - gets
- Both functions are in stdio library, so we need to include stdio.h in the program

scanf

- %s or %*n*s is used in the format string to indicate a string is to be read
  - %s: scans up to but not include the next space character
  - E.g.,

```
char str1[20], str2[20], str3[20];
scanf("%s%s%s", str1, str2, str3);
```

Assume the input is

```
UIC Computer Science Students
```

Then,
```
str1 = "UIC"
str2 = "Computer"
str3 = "Science"
```

scanf

- %s or %*n*s is used in the format string to indicate a string is to be read
  - %*n*s: scans the next *n* characters or up to but not include the next space character, whichever comes first
  - E.g.,

```
char str1[20], str2[20], str3[20];
scanf("%2s%3s%5s", str1, str2, str3);
```

Assume the input is

```
UIC Computer Science Students
```

Then,

```
str1 = "UI"
str2 = "C"
str3 = "Compu"
```

Exercise

- What is the output of the following code?

```
char lName[81], fName[81];
int id_num;
puts("Enter the last name, firstname, ID number
       separated by spaces ");
puts("then press Enter \n");
scanf("%s%s%d", lName, fName, &id_num);
printf("3 items entered: %s %s %d\n",
       fName,lName,id_num);
```

The input is

```
Tony Towey 23456
```

gets

- gets( ) gets a line from the standard input.
  - E.g.,

```
char your_line[100];
printf("Enter a line:\n");
gets(your_line);
puts("Your input follows:\n");
puts(your_line);
```

- Be careful, do not overflow the string buffer (exceed the size of array)
  - `fgets(your_line, sizeof(your_line), stdin)`

https://www.tutorialspoint.com/c_standard_library/c_function_fgets.htm

String Functions

- Common string functions include functions
  - computing the length of a string (strlen)
  - copying strings (strcpy, strncpy)
  - comparing strings (strcmp, strncmp)
  - concatenating strings (strcat)
  - more …
- To use these functions, we must include the file string.h, e.g., #include <string.h>

IMPORTANT!!!
http://www.cplusplus.com/reference/cstring/

strlen

- Call: strlen(str);
- Objective : Calculating the length of *str*
- Return: length of str, not including '\0';
- E.g.,

```
char your_line[100] = "Hello"; int l;
l = strlen(your_line);
printf("The length of your_line is %d", l);
```

- `The length of your_line is 5`

strcpy

- Call: strcpy(destination, source);
- Objective : copy string from *source* to *destination*
- Return: same as destination
- E.g.,

```
char my_line[100];
char your_line[100] = “Hello”;
int l;
strcpy(my_line, your_line)
printf(“my_line is %s”, my_line);
```

- `my_line is Hello`
- Attention: *destination* should have enough room to store the string.

strncpy

- Call: strncpy(destination, source, n);
- Objective : copy *n* characters from *source* to *destination*
- Return: same as *destination*
- E.g.,

```
char my_line[100];
char your_line[100] = "Hello";
strncpy(my_line, your_line, 2)
my_line[2] = '\0';
printf("my_line is %s", my_line);
```

- `my_line is "He"`
- Attention: destination should have enough room to store the string
- '\0' should be added to the end.

strcmp

- Call: strcmp(str1, str2);
- Objective : Compare *str1* and *str2* based on ASCII
- Return: < 0 if str1 is less than str2
  = 0 if str1 equals str2
  > 0 if str1 is greater than str2
- E.g.,

```
char my_line[100] = "Hello World";
char your_line[100] = "Hello world";
int l;
l = strcmp(my_line, your_line);
```

- `l > 0 or l < 0 or l = 0?`

strncmp

- Call: strncmp(str1, str2, n);
- Objective : Compare *n* characters of *str1* and *str2* based on ASCII
- Return: < 0 if str1 is less than str2
  = 0 if str1 equals str2
  > 0 if str1 is greater than str2
  (the beginning *n* characters)
- E.g.,

```
 char my_line[100] = "Hello World";
 char your_line[100] = "Hello world";
int L1, L2;
L1 = strncmp(my_line, your_line, 6);
L2 = strncmp(my_line, your_line, 7);
```

- L1? L2?

strcat

- Call: strcat(destination, source);
- Objective : add source to destination
- Return: concatenated string, same as destination
- E.g.,

```
 char my_line[100] = “Hello ”;
 char your_line[100] = “world”;
strcat(my_line, your_line);
printf(“The linked string is %s”, my_line);
```

- The linked string is `Hello world`

Char, String, Number

- We must distinguish clearly numbers, chars, strings.
  - ‘1’: char
  - “1”: string
  - 1: number
- Are the following expressions correct?
  - str[0] = “h”;
  - printf(‘\n’);
  - Str[10] = ‘hello’;

String to Number Functions

- Some functions can transfer number string to numbers
  - atoi
  - atol
  - atof
- If use these functions, must include stdlib.h, e.g., #include<stdlib.h>

IMPORTANT!!!

http://www.cplusplus.com/reference/cstdlib/

atoi

- Call: atoi(str)
- Objective: convert *str* to an int number, starting at beginning and continuing until something non-convertible is encountered
- Space and + are acceptable
- E.g.,

String	Value returned
"157"	157
"-1.6"	-1
"+50x"	50
"twelve"	0
"x506"	0

atol

- Same as atoi except that converting the string to a long number

atof

- Call: atof(str)
- Objective: convert *str* to a float number, starting at beginning and continues until something non-convertible is encountered
- Space and + are acceptable
- An E or e (exponent) is acceptable
- A decimal point is acceptable
- E.g.,

String	Value returned
"12"	12.000000
"-0.123"	-0.123000
"123E+3"	123000.000000
"123.1e-5"	0.001231

Exercise

```
 char name1[16] = "Frans Coenen";
 char name2[16] =
{'F','r','a','n','s', ' ','C','o','e','n','e','n','\0'};
 char name3[16] =
 {70,114,97,110,115,32,67,111,101,110,101,110,0};


  printf("name1 = %s\n",name1);
 printf("name2 = %s\n",name2);
 printf("name3 = %s\n",name3);
```

What is the output?

Array of Strings

- See the following example to understand how an array of string is used

```
#include< stdio.h>
void main(void)
{
  char names[2][8] = {"Frans", "Coenen"};
  printf("names = %s, %s\n",names[0],names[1]);
  printf("names = %s\n", names);
  printf("Initials = %c.%c.\n",names[0][0],names[1][0]);
 }
```

What is output?

Implementing String Functions

- See what the following program does.

```
void copyString(char dest[], char src[])
{
  int i = 0;
  while(src[i] != '\0') {
    dest[i] = src[i];
    i++;
  }
  dest[i] = '\0';
}
```

Try to write your own functions to serve the same objectives as strlen() and strcmp() !

Summary

- String is a set of characters.
- Library functions have been offered on the operations of the strings.