Strings

ESC108A Elements of Computer Science and Engineering

B. Tech. 2017

Course Leaders:

Roopa G.

Ami Rai E.

Chaitra S.

Objectives

- At the end of this lecture, student will be able to
  - Use standard string manipulation functions
  - Create programs that manipulate strings

Contents

Character and String Constants

- A character constant
  - an int value represented as a character in single quotes
  - The value of a character constant is the integer value of the character in the machine's character set
  - 'z' represents the integer value of z, 122 in ASCII
  - '\n' the integer value of newline, 10 in ASCII

- String literals, or string constants
  - written in double quotation marks
  - "John Q. Doe" (a name)
  - "99999 Main Street" (a street address)

Strings

- Character arrays
  - A sequence of zero or more characters surrounded by double quotes
  - May include letters, digits and various special characters such as +, -, *, / and $

- Internal representations of a string has a null character '\0‘ at the end

```
char names[]=“hello”;
```

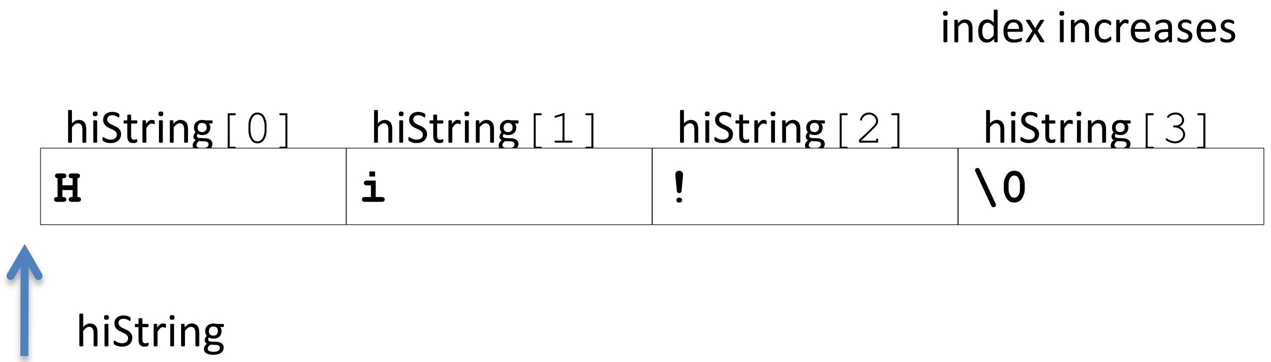
  - The string contains 6 elements

Strings as Arrays

- Strings are 1-Dimensional ***char*** arrays
  - – Always end with '\0' character

- Representation of string in memory

  char hiString[4] = "Hi!";   //creates a 4-element array hiString containing the characters 'H', 'i', '!', and '\0



Declaring and Initialising String Variables

- The general form of declaration of a string variable

    char string_name [size];

- Example

    char city[10];

- Initialisation

  char name[10] = {'M','S','R','U','A','S','\0'};

  or

  char name[10] = {"MSRUAS"};

  char name[10] = "MSRUAS"; //automatically adds null character at the end of the string

Reading Strings

- Reading Strings Using scanf
  - Reads characters from the keyboard until the first whitespace character is encountered

```
 scanf("%s", oneWord);


 printf("\nString is : %s“,oneWord);
```

  – Note there is no “&” as string itself has address
    - Array name is the address of the start of the array
  – Remember to add one extra place in size for '\0‘
  – Accepting a string with blank space is not possible

Reading Strings contd.

- Reading strings with white spaces

  scanf("%[^\t\n]s",str); //characters specified after the circumflex (^) are not permitted in the input string

  printf("\nString is : %s",str);

- Reading Strings using gets() and puts()

  gets(str); //reads everything from the keyboard until the ENTER key is pressed

  printf("\nString is : %s",puts(str)); //newline character will not be displayed

Strings - Example

```
char string1[20];  /* reserves 20 characters */
char string2[] = "string literal"; /* reserves 15 characters */
int i; /* counter */
 /* read string from user into array string1 */
printf("Enter a string: ");
scanf( "%s", string1); /* input ended by whitespace character */
/* output strings */
printf( "string1 is: %s \n string2 is: %s \n", string1, string2);
/* output characters until null character is reached */
for ( i = 0; string1[i] != '\0'; i++) {
        printf( "%c ", string1[i]);
} /* end for */
```

Two Dimensional Array of Characters

- To process a group of strings
- Arrays of arrays
- Consists of strings as its individual elements

- Example

  char colours[3][10] = {“blue”,”yellow”,”red”};

  – a two-dimensional array of 3 strings each of 10 characters long
  – colours[1] would be yellow

Character Handling in Standard Library

- Includes functions to perform
  - tests on characters
  - manipulations of character data
- Each function receives a character or **EOF** as an argument

String Handling Functions

string.h

1. strlen()
   - Returns the number of characters in the string
   - Does not include ‘\0’
2. strcmp()
   - Compares 2 strings character by character
   - Returns one of the three values {-1,0,1}
3. strcpy()
   - Used to copy one string to the other
4. strcat()
   - Used to concatenate 2 strings

String Handling Functions - Example

char name1[]=“MSR”,name2[]=“UAS” ;

int count;

- strlen(“MSR”)=3
- count=strlen(name2);
  - count=3
- strcmp(name1,name2)=-1
- strcpy(name1,name2);
  - name1=“UAS”,name2=“UAS”
- strcat(name1,name2);
  - name1=“MSRUAS”,name2=“UAS”

Character Handling Functions

<ctype.h>

Prototype	Description
int isdigit(int c)	Returns **true** if **c** is a digit and **false** otherwise.
int isalpha(int c)	Returns **true** if **c** is a letter and **false** otherwise.
int isalnum(int c)	Returns **true** if **c** is a digit or a letter and **false** otherwise.
int isxdigit(int c)	Returns **true** if **c** is a hexadecimal digit character and **false** otherwise.
int islower(int c)	Returns **true** if **c** is a lowercase letter and **false** otherwise.
int isupper(int c)	Returns **true** if **c** is an uppercase letter; **false** otherwise.
int tolower(int c)	If **c** is an uppercase letter, **tolower** returns **c** as a lowercase letter. Otherwise, **tolower** returns the argument unchanged.
int toupper(int c)	If **c** is a lowercase letter, **toupper** returns **c** as an uppercase letter. Otherwise, **toupper** returns the argument unchanged.
int isspace(int c)	Returns **true** if **c** is a white-space character—newline (**'\n'**), space (**' '**), form feed (**'\f'**), carriage return (**'\r'**), horizontal tab (**'\t'**), or vertical tab (**'\v'**)—and **false** otherwise
int iscntrl(int c)	Returns **true** if **c** is a control character and **false** otherwise.
int ispunct(int c)	Returns **true** if **c** is a printing character other than a space, a digit, or a letter and **false** otherwise.
int isprint(int c)	Returns **true** value if **c** is a printing character including space (**' '**) and **false** otherwise.
int isgraph(int c)	Returns **true** if **c** is a printing character other than space (**' '**) and **false** otherwise.

Standard I/O Library Functions

- Functions in **<stdio.h>**
  - Used to manipulate character and string data

Function prototype	Function description
int getchar(void);	Inputs the next character from the standard input and returns it as an integer.
char *gets(char *s);	Inputs characters from the standard input into the array **s** until a newline or end-of-file character is encountered. A terminating null character is appended to the array.
int putchar(int c);	Prints the character stored in **c**.
int puts(const char *s);	Prints the string s followed by a newline character.
int sprintf(char *s, const char *format, ...);	Equivalent to printf, except the output is stored in the array s instead of printing it on the screen.
int sscanf(char *s, const char *format, ...);	Equivalent to scanf, except the input is read from the array s instead of reading it from the keyboard.

Summary

- Strings can be represented using character arrays or character pointers
- Internal representations of a string has a null character '\0‘ at the end
- C provides Character and String manipulation functions
- C standard library also provides functions to convert a string to number

Further Reading

Kernighan, B. W. and Richie, D. (1992) *The C Programming Language.* 2nd ed., New Delhi:PHI.