FIT ICT Software Development

Lecture 9

Working With Strings

Lecturer : Charles Tyner

FIT ICT Software Development

Strings

In C programming, a string is an array of characters which has the special ***\0*** null character in its last element

```
char arr[6] = {'A','l','p','h','a'};
printf("%s", arr)  /*Outputs "Alpha" */
```

Six elements are required in the array to store the null character in the final element

String values can be read via ***scanf()*** function, but only until first space is encountered

To overcome this, we use ***fgets()*** function

Program listing that follows illustrates this

FIT ICT Software Development

Strings

```
#include <stdio.h>
int main()
{
   char str[51];
   printf("\nEnter up to 50 characters with spaces\n");
   fgets( str, sizeof(str), stdin);
   printf("fgets() read: ");
   puts(str);
   printf("\nEnter up to 50 characters with spaces\n");
   scanf("%s", str);
   printf("scanf() read: %s\n", str);
   return 0;
}
```

FIT ICT Software Development

Strings

fgets() and *puts()* are located in **stdio.h**

fgets() is used to read text, including input from a user

Accepts all characters, including spaces

Adds ***\0*** when user hits return/enter key

Takes 3 arguments:

1. name of character array to which to assign the string
2. number of characters to read
3. where to read from

FIT ICT Software Development

Strings

First parameter value provided by name of character array declared at start of block

Number of characters is returned by use of function ***sizeof()***

Note that for 50 characters, the array must have 51 elements, to account for terminating null

Third parameter ***stdin*** tells the function that it is to read standard input

puts() outputs a string specified as its argument and automatically adds a newline character at the end

FIT ICT Software Development

Copying Strings

Header file ***string.h*** contains string-handling functions

To be used, must be **included** at start of program

One useful function is ***strlen()*** which takes a string as its argument and returns an integer showing total number of characters – including spaces but excluding final null character

Two others are ***strcpy() and strncpy()***

FIT ICT Software Development

Copying Strings

strcpy()

Takes two arguments: name of target array to copy string into; and source array from where string is to be copied

strcpy(target-array, source-array);

All characters including null terminating character are copied into target array

Any remaining elements padded with nulls to overwrite any remnants from a previous, longer string

FIT ICT Software Development

Copying Strings

strncpy()

Similar to *strcpy()* but takes third argument tp specify how much of string to be copied

strcpy(target-array, source-array, length);

Will copy from first character but will stop at position specified by third argument

Will not include a null terminating character, which must be explicitly supplied

Implement program on slide following and carefully study output to see both functions used

FIT ICT Software Development

Copying Strings

```
#include <stdio.h>
#include <string.h>
int main()
{
  char s1[] = "Larger text string"; s2[] = "Smaller string";
  printf("\n%s: %d elements", s1 , sizeof(s1));
  printf(", %d characters\n", strlen(s1));
  strcpy( s1, s2);
  printf("\n%s: %d elements", s1 , sizeof(s1));
  printf(", %d characters\n", strlen(s1));
  strncpy( s1, s2, 5);
  s1[5] = '\0'
  printf("\n%s: %d elements", s1 , sizeof(s1));
  printf(", %d characters\n", strlen(s1));
  return 0;
}
```

FIT ICT Software Development

Copying Strings

Header includes standard i/o and string libraries.

In main function, two character array variables are declared and initialised.

Next, the contents of the first array, the number of elements and the length of string are output. Note the difference of one, accounted for by null

Entire contents of second string copied to first using ***strcpy()*** function

Previous code to output first array is repeated. Note that although it takes up fewer elements and the full array is output, no characters from the original string remain, having been over-written with nulls

FIT ICT Software Development

Copying Strings

Now, the first five characters only of second string copied to first array using ***strncpy()*** function

As this function does not automatically provide a terminating null character, this has to be manually added.

First five characters occupy elements 0-4 of the array, so we specify the fifth element to store the null

FIT ICT Software Development

Joining Strings

Known as “concatenation”

The target array, initially holding the first string but ultimately holding the concatenated string, must be large enough to accommodate the combined text or an error will occur.

Note that if the array is being initialised at the same time as being declared, by default the number of elements provided will be the number of characters (including spaces) in the string plus one for the null terminating character

FIT ICT Software Development

Joining Strings

Two functions used to concatenate strings included in *string.h*

strcat() takes 2 arguments specifying the strings to be concatenated

String named as second argument appended after string named in first

strcat(first-string, string-to-add)

Not clever enough to put in spaces between strings, so first must end with a space character or second must start with one for readability

FIT ICT Software Development

Joining Strings

strncat() takes 3 arguments specifying the strings to be concatenated and the number of characters from the second string to use

strncat(first-string, string-to-add, length)

By default, counting length will start at the first character of second string

This can be changed very simply by modifying the second parameter to indicate the position at which to start

strncat(first-string, string-to-add+places, length)

Implement program on slide following and carefully study output to see both functions used

FIT ICT Software Development

Joining Strings

```
#include <stdio.h>
#include <string.h>
int main()
{
  char s1[100] = "A Place for Everything ";
  char s2[] = "and Everything in its Place";
  char s3[100] = "The Truth is Rarely Pure ";
  char s4[] = "and Never Simple. – Oscar Wilde";
  strcat(s1, s2);
  printf("\n%s\n", s1);
  strncat(s3, s4, 17);
  printf("\n%s\n", s3);
  strncat(s3, (s4 + 17), 14 );
  printf("\n%s\n", s3);
  return 0;
}
```

FIT ICT Software Development

Joining Strings

Header includes standard i/o and string libraries.

In main function, four character array variables are declared and initialised.

Next, the contents of the second are appended to the first and the new value of the first string is output.

The first 17 characters of the fourth string are now appended to the third and the new value of the third string is output.

Then a subset of the fourth string, beginning at the seventeenth element of the array and continuing for 14 places, is appended and the final contents of string 3 are output.

FIT ICT Software Development

Finding Substrings

A string can be searched to see if it contains a specified sequence of characters (substring)

Function ***strstr()*** is provided by **string.h** for this purpose

Takes 2 arguments

strstr(string-to-search, substring-to-find);

If no match found, returns NULL; otherwise returns hex address in memory of occurrence of first character of substring – in other words NOT NULL

FIT ICT Software Development

Finding Substrings

Two strings can also be compared to see if they are identical

Function ***stramp()*** is provided by **string.h** for this purpose

Takes 2 arguments

strcmp(string1, string2);

When strings are identical in all respects, including matching case, functions returns 0; otherwise a non-zero integer

FIT ICT Software Development

Finding Substrings

```
#include <stdio.h>
#include <string.h>
int main()
{
  char str[] = "No Time Like the Present";
  char sub[] = "Time";
  if( strstr(str,sub) == NULL)
  {
    printf("Substring \"Time\" Not Found\n");
  }
  else
  {
    printf("Substring \"Time\" Found at %p\n",  strstr ( str, sub ));
    printf("Element index number %d\n\n",  strstr ( str, sub ) -  str );
  }
```

FIT ICT Software Development

Finding Substrings

```
printf("%s compared with\"Time\":  %d\n",
        sub , strcmp(sub , "Time"));
printf("%s compared with\"time\":  %d\n",
        sub , strcmp(sub , "time"));
printf("%s compared with\"TIME\":  %d\n",
        sub , strcmp(sub , "TIME"));
return 0 ;
}
```

FIT ICT Software Development

Finding Substrings

NOTES:

1. Function ***strstr()*** stops searching after finding first occurrence of substring.
2. In statement

   printf("Substring \"Time\" Not Found\n");

   the backslash character is used to escape the double quote; otherwise the string would be terminated prematurely

FIT ICT Software Development

Validating Strings

Standard library header file *ctype.h* provides useful functions for testing characters:

- ***isalpha()*** tests whether character is alphabetic
- ***isdigit()*** tests whether character is numeric
- ***ispunct()*** tests whether character is any other printable symbol
- ***isspace()*** tests for the space character
- ***isupper()*** and ***islower()*** tests character case
- ***toupper()*** and ***tolower()*** changes character case

Test functions return zero if character fails test; non-zero if it passes

Can be used to validate strings by looping through characters and setting flags

FIT ICT Software Development

Validating Strings

```
#include <stdio.h>
#include <ctype.h>
int main()
{
   char str[7];
   int i;
   int flag = 1;

   puts( "Enter six digits without any spaces…" ) ;
   gets( str );

    for( i = 0; i < 6; i++ )
    {
        if( !isdigit( str[i] ))
         {
```

FIT ICT Software Development

Validating Strings

```
        flag = 0 ;
        if( isalpha( str[i] ))
        {printf( "Letter %c found\n ", toupper(str[i] )) ; }
        else if( ispunct( str[i] ))
        {printf( "Punctuation found\n ") ; }
        else if( isspace( str[i] ))
        {printf( "Space found\n ") ; }
          }
    }
    ( flag ) ? puts("Entry valid") : puts("Entry invalid");
    return 0;
}
```

FIT ICT Software Development

Validating Strings

Header includes standard i/o and ctype libraries.

In main function, one character array and two integer variables are declared.

Second integer variable is to act as a flag for whether or not the entry is valid. It's initialised to 1, meaning True.

User is prompted for input in a particular format and this is read by function *gets()* and assigned to variable *str*

FIT ICT Software Development

Validating Strings

A loop examines each character in the array. If any character is not a digit, the flag is changed to false

Even if an invalid character is found in the first iteration, the code will continue to check the string. However, there is no mechanism to reset the flag to True, so the end result will be accurate (if not efficient)

If a character is not numeric, the nested if will identify and print out what kind of invalid character was encountered

FIT ICT Software Development

Converting Strings

Since numbers can be treated as either characters or integers, it is often useful to read as one data type but process as another.

Library file *stdlib.h* provides two useful functions to do this.

It must be included as a pre-processor instruction.

To convert from alpha to integer, use *atoi()* function.

To convert from integer to alpha, use *itoa()*.

FIT ICT Software Development

Converting Strings

Function *atoi()* takes string to be converted as its single argument.

If string is empty, or if its first character is not a number or a minus sign, returns zero.

Otherwise, string will be converted to *int* data type from first character to the end, or to the point where a non-numeric character is encountered.

FIT ICT Software Development

Converting Strings

Function *itoa()* is not part of ANSI standard but is supported by many compilers, including GNU.

Takes three arguments:

- number to be converted
- string to which conversion is assigned
- base to be used for conversion

Base allows for conversion to binary equivalent

FIT ICT Software Development

Converting Strings

ANSI-compliant alternative to ***itoa()*** in ***stdlib.h*** is ***sprintf()***.

Less powerful as doesn't allow selection of base

Takes three parameters:

- string to which conversion is to be assigned
- format specifier
- number to be converted

FIT ICT Software Development

Converting Strings

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   int n1, n2, n3;
   char s1[10] = "12eight", s1[10] = "-65.8", s1[10] = "x13";

   n1= atoi(s1);
   printf( "\nString %s converts to integer: %d\n ", s1, n1;
   n2= atoi(s2);
   printf( "\nString %s converts to integer: %d\n ", s2, n2;
   n3= atoi(s3);
   printf( "\nString %s converts to integer: %d\n ", s3, n3;
```

FIT ICT Software Development

Converting Strings

```
itoa(n1 , s1 ,  2);
printf( "\nDecimal %d is Binary: %s\n ", n1, s1;

n2= sprintf(s3, "%o", n1);
printf( "\nDecimal %d is Octal: %s  chars: %d\n ", n1, s3, n2;

n3= sprintf(s3, "%x", n1);
printf( "\nString %s is Hexadecimal: %s  chars: %d\n ", n1, s3, n3;

return 0;
```

FIT ICT Software Development

ASCII characters

American Standard Code for Information Interchange

Standard representation of characters by numerical code

Includes printable and non-printable characters such as space (32) and delete (127)

To find the ASCII value of a character, simply use the %d format string:

FIT ICT Software Development

ASCII characters

```
#include <stdio.h>
int main()
{
   char c;
   printf("Enter a character: ");
   scanf("%c",&c);        /* Takes a character from user */
   printf("ASCII value of %c = %d",c,c);
   return 0;
}
```

Output

Enter a character: G

ASCII value of G = 71

FIT ICT Software Development

ASCII characters

Warning:

If you cast from a **char** variable to an **int**, the resulting integer value is the numerical ASCII code. For instance, casting the character “1” will return 49.

Use function *atoi()* instead