EECS 10: Computational Methods in Electrical and Computer Engineering
Lecture 12

Rainer Dömer

doemer@uci.edu

The Henry Samueli School of Engineering
Electrical Engineering and Computer Science
University of California, Irvine

Lecture 12: Overview

- Passing arguments to functions
  - Pass by value
  - Pass by reference
- Character Arrays: Strings
  - Input and output
  - ASCII table
  - Example: Sort strings alphabetically
    - Task
    - Approach
    - Algorithm *Bubble Sort*
    - Program **BubbleSort.c**

EECS10: Computational Methods in ECE, Lecture 12   (c) 2018 R. Doemer   2

Passing Arguments to Functions

- Pass by Value
  - only the *current value* is passed as argument
  - the parameter is a *copy* of the argument
  - changes to the parameter *do not* affect the argument
- Pass by Reference
  - a *reference* to the object is passed as argument
  - the parameter is a *reference* to the argument
  - changes to the parameter *do* affect the argument
- In ANSI C, ...
  - ... basic types are passed by value
  - ... arrays are passed by reference

EECS10: Computational Methods in ECE, Lecture 12   (c) 2018 R. Doemer   3

Passing Arguments to Functions

- Example: Pass by Value

```
void f(int p)
{
   printf("p before modification is  %d\n", p);
   p = 42;
   printf("p after modification is   %d\n", p);
}

int main(void)
{
   int a = 0;

   printf("a before function call is %d\n", a);
   f(a);
   printf("a after function call is  %d\n", a);
}
```

```
a before function call is 0
p before modification is  0
p after modification is   42
a after function call is  0
```

Changes to the parameter *do not* affect the argument!

EECS10: Computational Methods in ECE, Lecture 12   (c) 2018 R. Doemer   4

Passing Arguments to Functions

- Example: Pass by Reference

```c
void f(int p[2])
{
   printf("p[1] before modification is  %d\n", p[1]);
   p[1] = 42;
   printf("p[1] after modification is   %d\n", p[1]);
}

int main(void)
{
   int a[2] = {0, 0};

   printf("a[1] before function call is %d\n", a[1]);
   f(a);
   printf("a[1] after function call is  %d\n", a[1]);
}
```

```
a[1] before function call is 0
p[1] before modification is  0
p[1] after modification is   42
a[1] after function call is  42
```

Changes to the parameter *do* affect the argument!

EECS10: Computational Methods in ECE, Lecture 12   (c) 2018 R. Doemer   5

Character Arrays: Strings

- Text is represented by character arrays (aka. *strings*)
  - Strings are null-terminated arrays of characters
  - String output
    - `printf()` format specifier: "%s"
- Example:

```c
char s1[] = {'H','e','l','l','o',0};

printf("s1 is %s.\n", s1);
```

	s1
0	'H'
1	'e'
2	'l'
3	'l'
4	'o'
5	0

```
s1 is Hello.
```

EECS10: Computational Methods in ECE, Lecture 12   (c) 2018 R. Doemer   6

Character Arrays: Strings

- Text is represented by character arrays (aka. *strings*)
  - Strings are null-terminated arrays of characters
  - String output
    - `printf()` format specifier: `"%s"`
- Example:

```
char s1[] = {'H','e','l','l','o',0};
char s2[] = "Hello";

printf("s1 is %s.\n", s1);
printf("s2 is %s.\n", s2);
```

```
s1 is Hello.
s2 is Hello.
```

s2

0	'H'
1	'e'
2	'l'
3	'l'
4	'o'
5	0

EECS10: Computational Methods in ECE, Lecture 12   (c) 2018 R. Doemer   7

Character Arrays: Strings

- Text is represented by character arrays (aka. *strings*)
  - Strings are null-terminated arrays of characters
  - String output
    - `printf()` format specifier: `"%s"`
- Example:

```
char s1[] = {'H','e','l','l','o',0};
char s2[] = "Hello";

printf("s1 is %s.\n", s1);
printf("s2 is %s.\n", s2);
s1[1] = 'i';
s1[2] = 0;
printf("Modified s1 is %s.\n", s1);
```

```
s1 is Hello.
s2 is Hello.
Modified s1 is Hi.
```

s1

0	'H'
1	'i'
2	0
3	'l'
4	'o'
5	0

EECS10: Computational Methods in ECE, Lecture 12   (c) 2018 R. Doemer   8

Character Arrays: Strings

- Text is represented by character arrays (aka. *strings*)
  - Strings are null-terminated arrays of characters
  - String input
    - **scanf()** format specifier: "**%Ns**",
      where **N** specifies maximum field width = array size - 1
    - address argument can be **&string[0]**

- Example:

s1

```
char s1[6];

printf("Enter a string: ");
scanf("%5s", &s1[0]);
printf("s1 is %s.\n", s1);
```

0	'T'
1	'e'
2	's'
3	't'
4	0
5	0

```
Enter a string: Test
s1 is Test.
```

EECS10: Computational Methods in ECE, Lecture 12   (c) 2018 R. Doemer   9

Character Arrays: Strings

- Text is represented by character arrays (aka. *strings*)
  - Strings are null-terminated arrays of characters
  - String input
    - **scanf()** format specifier: "**%Ns**",
      where **N** specifies maximum field width = array size - 1
    - address argument can be **&string[0]**
      or simply **string**

- Example:

s1

```
char s1[6];

printf("Enter a string: ");
scanf("%5s", s1);
printf("s1 is %s.\n", s1);
```

0	'T'
1	'e'
2	's'
3	't'
4	0
5	0

```
Enter a string: Test
s1 is Test.
```

EECS10: Computational Methods in ECE, Lecture 12   (c) 2018 R. Doemer   10

Character Arrays: Strings

- Text is represented by character arrays (aka. *strings*)
  - Strings are null-terminated arrays of characters
  - Characters are represented by numeric values
  - ASCII table defines character values 0-127
- Example:

```
char s1[] = "ABC12";
int i = 0;

while(s1[i])
  { printf("%c = %d\n",s1[i],s1[i]);
    i++; }
```

```
A = 65
B = 66
C = 67
1 = 49
2 = 50
```

s1	
0	'A'
1	'B'
2	'C'
3	'1'
4	'2'
5	0

EECS10: Computational Methods in ECE, Lecture 12   (c) 2018 R. Doemer   11

Character Arrays: Strings

- ASCII Table
  - American Standard Code for Information Interchange

0 NUL	1 SOH	2 STX	3 ETX	4 EOT	5 ENQ	6 ACK	7 BEL
8 BS	9 HT	10 NL	11 VT	12 NP	13 CR	14 SO	15 SI
16 DLE	17 DC1	18 DC2	19 DC3	20 DC4	21 NAK	22 SYN	23 ETB
24 CAN	25 EM	26 SUB	27 ESC	28 FS	29 GS	30 RS	31 US
32	33 !	34 "	35 #	36 $	37 %	38 &	39 '
40 (	41)	42 *	43 +	44 ,	45 -	46 .	47 /
48 0	49 1	50 2	51 3	52 4	53 5	54 6	55 7
56 8	57 9	58 :	59 ;	60 <	61 =	62 >	63 ?
64 @	65 A	66 B	67 C	68 D	69 E	70 F	71 G
72 H	73 I	74 J	75 K	76 L	77 M	78 N	79 O
80 P	81 Q	82 R	83 S	84 T	85 U	86 V	87 W
88 X	89 Y	90 Z	91 [	92 \	93]	94 ^	95 _
96 `	97 a	98 b	99 c	100 d	101 e	102 f	103 g
104 h	105 i	106 j	107 k	108 l	109 m	110 n	111 o
112 p	113 q	114 r	115 s	116 t	117 u	118 v	119 w
120 x	121 y	122 z	123 {	124		125 }	126 ~	127 DEL

EECS10: Computational Methods in ECE, Lecture 12   (c) 2018 R. Doemer   12

Character Arrays: Strings

- Case Study: *Bubble Sort*
    - Task: Sort an array of strings alphabetically
    - Input: Array of 10 strings entered by the user
    - Output: Array of 10 strings in alphabetical order
- Approach: Divide and Conquer
    - Step 1:   Let user enter 10 strings
    - Step 2:   Sort the array of strings
    - Step 3:   Output the strings in order

EECS10: Computational Methods in ECE, Lecture 12   (c) 2018 R. Doemer   13

Character Arrays: Strings

- Case Study: *Bubble Sort*
    - Task: Sort an array of strings alphabetically
    - Input: Array of 10 strings entered by the user
    - Output: Array of 10 strings in alphabetical order
- Approach: Divide and Conquer
    - Step 1:   Let user enter 10 strings
    - Step 2:   Sort the array of strings
        - Algorithm
            - in 9 rounds, compare all adjacent pairs of strings and swap the pair if they are not in alphabetical order
        - String comparison
            - compare character pairs alphabetically: use ASCII values!
        - String swap (exchange two strings in place)
            - swap each character pair in the two strings
    - Step 3:   Output the strings in order

EECS10: Computational Methods in ECE, Lecture 12   (c) 2018 R. Doemer   14

Character Arrays: Strings

- Program example: **BubbleSort.c** (part 1/7)

```
/* BubbleSort.c: sort strings alphabetically   */
/* author: Rainer Doemer                        */
/* modifications:                               */
/* 11/01/06 RD  swap only adjacent elements     */
/* 11/06/04 RD  initial version                 */

#include <stdio.h>

/* constant definitions */

#define NUM 10  /* ten strings  */
#define LEN 20  /* of length 20 */

/* function declarations */

void EnterText(char Text[NUM][LEN]);
void PrintText(char Text[NUM][LEN]);
int CompareStrings(char s1[LEN], char s2[LEN]);
void SwapStrings(char s1[LEN], char s2[LEN]);
void BubbleSort(char Text[NUM][LEN]);
...
```

EECS10: Computational Methods in ECE, Lecture 12   (c) 2018 R. Doemer   15

Character Arrays: Strings

- Program example: **BubbleSort.c** (part 2/7)

```
...

/* function definitions */

/* let the user enter the text array            */

void EnterText(char Text[NUM][LEN])
{
   int i;

   for(i = 0; i < NUM; i++)
      { printf("Enter text string %2d: ", i+1);
        scanf("%19s", Text[i]);
      } /* rof */
} /* end of EnterText */

...
```

EECS10: Computational Methods in ECE, Lecture 12   (c) 2018 R. Doemer   16

Character Arrays: Strings

- Program example: **BubbleSort.c** (part 3/7)

```
...

/* print the text array on the screen          */

void PrintText(char Text[NUM][LEN])
{
   int i;

   for(i = 0; i < NUM; i++)
      { printf("String %2d: %s\n", i+1, Text[i]);
        } /* rof */
} /* end of PrintText */

...
```

Character Arrays: Strings

- Program example: **BubbleSort.c** (part 4/7)

```
...
/* alphabetically compare strings s1 and s2:    */
/* return -1, if string s1 < string s2          */
/* return  0, if string s1 = string s2          */
/* return  1, if string s1 > string s2          */
int CompareStrings(char s1[LEN], char s2[LEN])
{
   int i;

   for(i = 0; i < LEN; i++)
      { if (s1[i] > s2[i])
            { return(1); }
        if (s1[i] < s2[i])
            { return(-1); }
        if (s1[i] == 0 || s2[i] == 0)
            { break; }
        } /* rof */
   return 0;
} /* end of CompareStrings */
...
```

Character Arrays: Strings

- Program example: **BubbleSort.c** (part 5/7)

```
...

/* swap/exchange the strings s1 and s2 in place */

void SwapStrings(char s1[LEN], char s2[LEN])
{
   int  i;
   char c;

   for(i = 0; i < LEN; i++)
      { c = s1[i];
        s1[i] = s2[i];
        s2[i] = c;
      } /* rof */
} /* end of SwapStrings */

...
```

Character Arrays: Strings

- Program example: **BubbleSort.c** (part 6/7)

```
...
/* sort the text array by comparing every pair  */
/* of strings; if the pair of strings is not in */
/* alphabetical order, swap it                   */

void BubbleSort(char Text[NUM][LEN])
{
   int p, i;

   for(p = 1; p < NUM; p++)
      { for(i = 0; i < NUM-1; i++)
           { if (CompareStrings(Text[i], Text[i+1]) > 0)
                { SwapStrings(Text[i], Text[i+1]);
                  } /* fi */
            } /* rof */
        } /* rof */
} /* end of BubbleSort */

...
```

Character Arrays: Strings

- Program example: **BubbleSort.c** (part 7/7)

```
...
/* main function: enter, sort, print the text    */

int main(void)
{
   /* local variables */
   char Text[NUM][LEN]; /* NUM strings, length LEN */

   /* input section */
   EnterText(Text);

   /* computation section */
   BubbleSort(Text);

   /* output section */
   PrintText(Text);

   /* exit */
   return 0;
} /* end of main */

/* EOF */
```

EECS10: Computational Methods in ECE, Lecture 12   (c) 2018 R. Doemer   21

Character Arrays: Strings

- Example session: **BubbleSort.c**

```
% vi BubbleSort.c
% gcc BubbleSort.c -o BubbleSort -Wall -ansi
% BubbleSort
Enter text string  1: Charlie
Enter text string  2: William
Enter text string  3: Donald
Enter text string  4: John
Enter text string  5: Jane
Enter text string  6: Jessie
Enter text string  7: Donald
Enter text string  8: Henry
Enter text string  9: George
Enter text string 10: Emily
String  1: Charlie
String  2: Donald
String  3: Donald
String  4: Emily
String  5: George
String  6: Henry
String  7: Jane
String  8: Jessie
String  9: John
String 10: William
%
```

EE