# FUNDAMENTALS OF STRUCTURED PROGRAMMING

Lecture 8

Functions III (User-Defined: Pass by Reference)

Course Coordinator: Prof. Zaki Taha Fayed

Presented by: Dr. Sally Saad

SallySaad@gmail.com

DropBox folder link

https://www.dropbox.com/sh/85vnrgkfqgrzhwn/AABdwKLJZqZs26a7u-y0AFwia?dl=0

Credits to Dr. Salma Hamdy for content preparation

# **Quotes of the Day!**

I learned that computer science is not just about syntax and coding. We can make a difference in people's lives by developing applications ...

-Kyle Rector



# Procedural Abstraction -Functions III

- When you pass an <u>argument</u> variable, the value of the variable is plugged/copied into the function's formal <u>parameter</u>, hence the name call by value.
- Arguments vs. formal parameters

# Remember Calling by Value

```
#include <iostream>
 using namespace std;
void get_num(int, int, int);
¤void main ()
     int a = 10, b = 20, c = 30;
     get num(a, b, c);
     cout<<a<<"\t"<<b<<"\t"<<c<endl;
```

```
void get_num(int x, int y, int z)
{
    cout<<xx<<"\t"<<y<<"\t"<<z<<endl;
    x = 1; y = 2; z = 3;
    cout<<x<<"\t"<<y<<"\t"<<z<<endl;
}</pre>
```

Call by value:
Arguments and formal
parameters are totally different
places
Any change in the formal
parameters does not affect the
argument variables

```
#include <iostream>
using namespace std;
void get_num(int, int, int);
¤void main ()
   int a = 10, b = 20, c = 30;
     get_num(a, b, c);
     cout<<a<<"\t"<<b<<"\t"<<c<endl;
□void get num(int x, int y, int z)
     cout<<x<<"\t"<<y<<"\t"<<z<<endl;
     x = 1; y = 2; z = 3;
     cout<<x<<"\t"<<y<<"\t"<<z<<endl;
```

Memory location	value	
•••		
1006		
1008		
1010		
1012		
1014		
1016		
1018		
1020		
•••		

```
#include <iostream>
using namespace std;
void get_num(int, int, int);
⊡void main ()
     int a = 10, b = 20, c = 30;
     get_num(a, b, c);
     cout<<a<<"\t"<<b<<"\t"<<c<endl;
□void get num(int x, int y, int z)
     cout<<x<<"\t"<<y<<"\t"<<z<<endl;
     x = 1; y = 2; z = 3;
     cout<<x<<"\t"<<y<<"\t"<<z<<endl;
```

Memory location	value	
•••		
1006	10	a
1008		
1010	20	b
1012		
1014		
1016	30	С
1018		
1020		
•••		

```
#include <iostream>
using namespace std;

void get_num(int, int, int);

void main ()
{
   int a = 10, b = 20, c = 30;
   get_num(a, b, c);
   cout<<a<<"\t"<<b<<"\t"<<c<endl;
}</pre>
```

```
void get_num(int x, int y, int z)
{
    cout<<xx<<"\t"<<y<<"\t"<<z<<endl;
    x = 1; y = 2; z = 3;
    cout<<xx<<"\t"<<y<<"\t"<<z<<endl;
}</pre>
```

Memory location	value	
•••		
1006	10	a
1008		
1010	20	b
1012		
1014		X
1016	30	С
1018		у
1020		Z
•••		

```
#include <iostream>
using namespace std;
void get_num(int, int, int);
¤void main ()
     int a = 10, b = 20, c = 30;
     get_num(a, b, c);
     cout<<a<<"\t"<<b<<"\t"<<c<endl;
□voideget_num(int x, int y, int z)
     cout<<x<<"\t"<<y<<"\t"<<z<<endl;
     x = 1; y = 2; z = 3;
     cout<<x<<"\t"<<y<<"\t"<<z<<endl;
```

	value	
	10	a
	20	b
)	10	X
K	30	С
	<b>2</b> 0	у
	30	Z
		10 20 10 30 20

#### Remember Calling by Value

```
#include <iostream>
using namespace std;
void get_num(int, int, int);
⊡void main ()
     int a = 10, b = 20, c = 30;
     get num(a, b, c);
     cout<<a<<"\t"<<b<<"\t"<<c<endl;
□void get num(int x, int y, int z)
  cout<<x<<"\t"<<y<<"\t"<<z<<endl;</pre>
     x = 1; y = 2; z = 3;
```

cout<<x<<"\t"<<y<<"\t"<<z<<endl;

Memory location	value	
•••		
1006	10	a
1008		
1010	20	b
1012		
1014	10	X
1016	30	С
1018	20	у
1020	30	Z
•••		

```
#include <iostream>
using namespace std;
void get_num(int, int, int);
⊡void main ()
    int a = 10, b = 20, c = 30;
    get_num(a, b, c);
    cout<<a<<"\t"<<b<<"\t"<<c<endl;
□void get num(int x, int y, int z)
    cout<<x<<"\t"<<y<<"\t"<<z<<endl;
  cout<<x<<"\t"<<y<<"\t"<<z<<endl;
```

Memory location	value	
•••		
1006	10	a
1008		
1010	20	b
1012		
1014	1	X
1016	30	С
1018	2	у
1020	3	Z
•••		

#### Remember Calling by Value

```
#include <iostream>
using namespace std;
void get_num(int, int, int);
¤void main ()
     int a = 10, b = 20, c = 30;
     get_num(a, b, c);
     cout<<a<<"\t"<<b<<"\t"<<c<endl;
□void get num(int x, int y, int z)
     cout<<x<<"\t"<<y<<"\t"<<z<<endl;
     x = 1; y = 2; z = 3;
```

cout<<x<<"\t"<<y<<"\t"<<z<<endl;

Memory location	value	
•••		
1006	10	a
1008		
1010	20	b
1012		
1014	1	X
1016	30	С
1018	2	у
1020	3	Z
•••		

#### Remember Calling by Value

```
#include <iostream>
using namespace std;
void get_num(int, int, int);
¤void main ()
     int a = 10, b = 20, c = 30;
     get_num(a, b, c);
     cout<<a<<"\t"<<b<<"\t"<<c<endl;
□void get num(int x, int y, int z)
     cout<<x<<"\t"<<y<<"\t"<<z<<endl;
     x = 1; y = 2; z = 3;
```

cout<<x<<"\t"<<y<<"\t"<<z<<endl;

Memory location	value	
•••		
1006	10	a
1008		
1010	20	b
1012		
1014		
1016	30	С
1018		
1020		
•••		

# (9) Passing Different Types

• Formal parameters, and hence arguments, can be of simple types (int, float, char, ...), or of aggregate types (struct,...).

### (10) Returning Different Types

 A function's return value, can be of simple type (int, float, char, ...), or of aggregate type(struct,...).

#### Example 1 - Remember Lec8Ex1.cpp

- Function to output structure fields.
- Function to input (fill) structure fields.

#### Example 2 - Lec8Ex2.cpp

- Function to output array elements.
- Function to input (fill) array elements.
- Overloading the display functions. (HOW?)

EX	cample 2 - Lec8Ex2.cpp	Memory	value	
12	<pre>int scores[SCORES] = {0};</pre>	location		
13	// input	•••		
14		1006		0
15	// processing	• 1000		
16	<pre>display(scores);</pre>	1008		0
17	myFun(scores);	1010		0
18	<pre>display(scores);</pre>	1012		
19		1012		
20	return 0;	_ 1014	1006	scores
21	} // end main	1016		
22		1019		
23	<pre>void myFun(int a[])</pre>	1018		
24	{	1020		
25	cout< <a[0]<<endl;< td=""><td>•••</td><td></td><td></td></a[0]<<endl;<>	•••		
26	a[0]++;			

Ex	cample 2 - Lec8Ex2.cpp	Memory	value	
12	<pre>int scores[SCORES] = {0};</pre>	location		
13	// input			
14	scores	1006		0
15	// processing	1008		0
16	display(scores);			
17	<pre>myFun(scores);</pre>	1010		0
18 19	display(scores);	1012		
20	return 0;	1014	1006	score
21	} // end main			S
22		1016		
23	<pre>void myFun(int a[])</pre>	1018		
24	<b>{</b>	1020		
25	cout< <a[0]<<endl;< td=""><td>.0.20</td><td></td><td></td></a[0]<<endl;<>	.0.20		
26	a[0]++;	•••		

Ex	cample 2 - Lec8Ex2.cpp	Memory	value	
12	<pre>int scores[SCORES] = {0};</pre>	location		
13	// input	•••		
14	scores	1006		0
15	// processing	1000		
16	<pre>display(scores);</pre>	1008		0
17	myFun(scores);	1010		0
18	<pre>display(scores);</pre>	1012		
19		1012		
20	return 0;	1014	1006	scores
21	} // end main	1016		
22		1018		
23	<pre>void myFun(int a[])</pre>	1016		
24	{	1020		
25	cout< <a[0]<<endl;< td=""><td>•••</td><td></td><td></td></a[0]<<endl;<>	•••		
26	a[0]++:			

Ex	cample 2 - Lec8Ex2.cpp	Memory	value	
12	<pre>int scores[SCORES] = {0};</pre>	location		
13	// input	•••		
14	scores	1006		0
15	// processing			
16	<pre>display(scores);</pre>	1008		0
17	<pre>myFun(scores);</pre>	1010		0
18	display(scores);	1012		
19				
20	return 0;	1014	1006	scores
21	} // end main	1016		
22		1019		
23	<pre>void myFun(int a[])</pre>	1018		
24	{	1020		
25	cout< <a[0]<<endl;< td=""><td>•••</td><td></td><td></td></a[0]<<endl;<>	•••		
26	a[0]++;			XXX

```
Example 2 - Lec8Ex2.cpp
                                         Memory
                                                  value
                                         location
12
        int scores[SCORES] = {0};
        // input
13
                                          1006
14
                              scores
                                                           0
15
        // processing
                                          1008
                                                           0
        display(scores);
16
                                          1010
                                                           0
        myFun(scores);
17
                                          1012
        display(scores);
18
                                          1014
                                                  1006
                                                         scores
19
                                          1016
        return 0;
20
                                          1018
21
      // end main
                                          1020
22
   void myFun(int a[])
                                   NOTE that a is not an actual
24
```

25

26

cout<<a[0]<<endl;</pre>

a[0]++;

variable in memory. It is a

reference variable.

```
Example 2 - Lec8Ex2.cpp
                                       Memory location
                                                        value
12
        int scores[SCORES] = {0};
13
        // input
                                scores-
14
                                             1006
15
        // processing
                                             1008
                                                                0
        display(scores);
16
                                             1010
                                                                0
        myFun(scores);
17
                                            1012
        display(scores);
18
                                             1014
                                                        1006
                                                              scores
19
20
        return 0;
                                             1016
21
   } // end main
                                             1018
22
                                             1020

void myFun(int a[])

                                              •••
24
25
        cout<<a[0]<<endl;</pre>
                               NOTE that a is not an actual variable
```

26

a[0]++;

in memory. It is a reference variable.

Example 2 - Lec8Ex2.cp; Memory value						
12		Memory location	value			
12	<pre>int scores[SCORES] = {0};</pre>	tocacion				
13	∮ // input	•••				
14	scores —>	1006		0		
15	// processing a	1008		0		
16	<pre>display(scores);</pre>	1010		0		
17	myFun(scores);	1012				
18	display(scores);	1014	1006	scores		
19						
20	return 0;	1016				
21	} // end main	1018	1005	a		
22		1020		•		
23	pvoid myFun(int a[])	•••				
24	{ NOTE that a	a is not an act	ual varia	ble		
25	<pre>cout&lt;<a[0]<<endl; in="" memory.<="" pre=""></a[0]<<endl;></pre>	It is a referen	ce varial	ole.		

26

a[0]++;

```
Example 2 - Lec8Ex2.cpp
                                         Memory
                                                  value
                                         location
12
        int scores[SCORES] = {0};
13
        // input
                              scores ->
14
                                          1006
15
        // processing
                                          1008
16
        display(scores);
                                          1010
                                                           0
        myFun(scores);
17
                                          1012
        display(scores);
18
                                          1014
                                                   1006
                                                         scores
19
                                          1016
20
        return 0;
21
      // end main
                                          1018
22
                                          1020
   pvoid myFun(int a[])
                                           •••
24
                             NOTE that a is not an actual variable
        cout<<a[0]<<endl
25
```

**●**a[0]++;

26

in memory. It is a reference variable.

```
Example 2 - Lec8Ex2.cpp
                                                  value
                                        Memory
                                        location
12
        int scores[SCORES] = {0};
13
        // input
14
                             scores ____
                                         1006
                                                           1
15
        // processing
                                         1008
                                                          0
        display(scores);
16
        myFun(scores);
17
                                         1010
                                                          0
      display(scores);
18
                                         1012
19
                                         1014
                                                  1006
                                                        scores
20
        return 0;
21
   } // end main
                                         1016
22
                                         1018

void myFun(int a[])

                                         1020
24
25
        cout<<a[0]<<endl;</pre>
        a[0]++;
26
```

- The return statement can only return one value.
- You can "return" more than one value by passing the address of a variable instead of its value (calling by reference). The parameter in this case is called a reference variable.

# (11) Returning More than One Value

```
#include <iostream>
using namespace std;

void get_num(intal) int, intal);

void main ()
{
   int a = 10, b = 20, c = 30;
   get_num(a, b, c);
   cout<<a<<"\t"<<b<<"\t"<<c<endl;
}</pre>
```

```
void get_num(irt& x, int y, int&z)
{
    cout<<x<<"\t"<<y<<"\t"<<z<<endl;
    x = 1; y = 2; z = 3;
    cout<<x<<"\t"<<y<<"\t"<<z<<endl;</pre>
```

#### Call by reference:

- Arguments and formal parameters are the same places in memory.
- Any change in the formal parameters WILL AFFECT the argument variables.

```
#include <iostream>
using namespace std;
void get num(int&, int, int&);
⊡void main ()
   int a = 10, b = 20, c = 30;
     get num(a, b, c);
     cout<<a<<"\t"<<b<<"\t"<<c<endl;
⊡void get num(int& x, int y, int& z)
{
     cout<<x<<"\t"<<y<<"\t"<<z<<endl;
     x = 1; y = 2; z = 3;
     cout<<x<<"\t"<<y<<"\t"<<z<<endl;
```

Memory location	value	
•••		
1006		
1008		
1010		
1012		
1014		
1016		
1018		
1020		
•••		

```
#include <iostream>
                                                      value
                                           Memory
 using namespace std;
                                           location
 void get_num(int&, int, int&);
                                            1006
                                                       10
                                                              a
⊡void main ()
                                            1008
     int a = 10, b = 20, c = 30;
                                            1010
                                                       20
                                                              b
     get num(a, b, c);
                                            1012
     cout<<a<<"\t"<<b<<"\t"<<c<endl;
                                            1014
                                            1016
                                                       30
pvoid get_num(int& x, int y, int& z)
                                            1018
 {
     cout<<x<<"\t"<<y<<"\t"<<z<<endl;
                                            1020
     x = 1; y = 2; z = 3;
     cout<<x<<"\t"<<y<<"\t"<<z<<endl;
```

(11) Returning More than One Value

```
#include <iostream>
using namespace std;
void get num(int&, int, int&);
⊡void main ()
     int a = 10, b = 20, c = 30;
     get num(a, b, c);
     cout<<a<<"\t"<<b<<"\t"<<c<endl;
□voieget_num(int& x, int y, int& z)
{
     cout<<x<<"\t"<<y<<"\t"<<z<<endl;
     x = 1; y = 2; z = 3;
     cout<<x<<"\t"<<v<<"\t"<<z<<endl:
```

varac		
Memory location	value	
•••		
1006	10	a
1008		
1010	20	b
1012		
1014	1006	3
1016	30	С
1018	20 \$	X
1020	1016	Z

NOTE that x,z are not an actual variables in memory. They are reference variables.

value

Memory

(11) Returning More than One Value

```
location
 #include <iostream>
 using namespace std;
 void get num(int&, int, int&);
                                                            10
                                                  1006
                                                  1008
⊡void main ()
                                                            20
                                                  1010
     int a = 10, b = 20, c = 30;
                                                  1012
     get num(a, b, c);
     cout<<a<<"\t"<<b<<"\t"<<c<endl;
                                                  1014
                                                                   X
                                                  1016
□void get num(int& x, int y, int& z)
                                                  1018
                                                            20
   cout<<x<<"\t"<<y<<"\t"<<z<<endl;</pre>
                                                  1020
     x = 1; y = 2; z = 3;
     cout<<x<<~\t~<<y<<~\t~<<z<<end1;
```

NOTE that x,z are not an actual variables in memory. They are reference variables

(11) Returning More than One Value

```
value
                                                Memory
 #include <iostream>
                                                location
 using namespace std;
 void get num(int&, int, int&);
                                                 1006
                                                                  a
                                                 1008
¤void main ()
                                                 1010
                                                           20
                                                                  h
     int a = 10, b = 20, c = 30;
                                                 1012
     get num(a, b, c);
                                                 1014
     cout<<a<<"\t"<<b<<"\t"<<c<endl;
                                                 1016
pvoid get_num(int& x, int y, int& z)
                                                 1018
                                                 1020
     cout<<x<<"\t"<<y<<"\t"<<z<<endl;
     x = 1; y = 2; z = 3;
                                                   •••
     cout<<x<<"\t"<<v<<"\t"<<z<<endl:
```

NOTE that x,z are not an actual variables in memory. They are reference variables.

#### (11) Returning More than One Value

```
#include <iostream>
                                                Memory
                                                           value
 using namespace std;
                                                location
 void get num(int&, int, int&);
                                                 1006
                                                                   a
⊡void main ()
                                                 1008
                                                 1010
                                                            20
                                                                   b
     int a = 10, b = 20, c = 30;
     get num(a, b, c);
                                                 1012
     cout<<a<<"\t"<<b<<"\t"<<c<endl;
                                                 1014
                                                 1016
pvoid get_num(int& x, int y, int& z)
                                                 1018
     cout<<x<<"\t"<<y<<"\t"<<z<<endl;
                                                 1020
     x = 1; y = 2; z = 3;
    cout<<x<<"\t"<<y<<"\t"<<z<<end1;
```

NOTE that x,z are not an actual variables in memory. They are reference variables

```
#include <iostream>
using namespace std;
void get num(int&, int, int&);
⊡void main ()
     int a = 10, b = 20, c = 30;
     get num(a, b, c);
     cout<<a<<"\t"<<b<<"\t"<<c<endl;
□void get num(int& x, int y, int& z)
{
     cout<<x<<"\t"<<y<<"\t"<<z<<endl;
     x = 1; y = 2; z = 3;
     cout<<x<<"\t"<<y<<"\t"<<z<<endl;
```

Memory location	value	
•••		
1006	1	a
1008		
1010	20	b
1012		
1014		
1016	3	С
1018		
1020		
•••		

- In passing a reference to a variable you are not actually returning anything, you are just changing the values in that address (which are the actual arguments of the function).
- Any other function "looking" at that location will see the changed value (as if you returned it).
- Notice that you didn't need to dereference the address. It's done for you in the called function.

- Passing by reference is used to provide access to caller's actual argument.
- Caller's data can be modified by called function.
- Typically used for input functions.
- Specified by ampersand &, after the type in formal parameter list.
- Remember that passing an array variable is passing by reference without explicitly specifying.

#### (11) Returning More than One Value

Example 3 (\*BONUS) - Lec8Ex3.cpp

A function to return the sum, average, max, and min of three numbers.



A function that swaps two numbers (mathematically ...).

#### Example 5

A function to input an array of struct variable.

