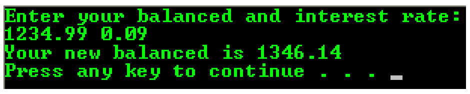
To:
Tenouk

C LAB WORKSHEET 13a_1
C & C++ Functions Part 6: With Return Values

1. Function with return value and array.
2. Activities with function sample codes.
3. C/C++ functions that receive and/or return values practices.
4. Tutorial references that should be used together with this worksheet are: C and C++ function part 1, function part 2, function part 3 and function part 4.
5. For main() function complete story please refer to C and C++ main() story.

2. The function AddInterest() will receive the Balance and InterestRate, both floats, from the calling function. It will return the new balance with the amount of interest added to it. For example, if 100.0 and 0.05 are received in the function, then 105.0 will be returned.

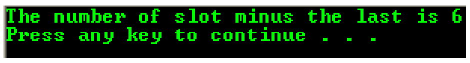
```
#include <stdio.h>

float AddInterest(float, float);

void main(void)
{
    float newValue, Balanced, Interest;

    printf("Enter your balanced and interest rate: \n");
    // scanf("%f%f", &Balanced, &Interest);
    scanf_s("%f%f", &Balanced, &Interest, sizeof(float), sizeof
(float));
    newValue = AddInterest(Balanced, Interest);
    printf("Your new balanced is %.2f\n", newValue);
}

float AddInterest(float BalancedHold, float InterestRateHold)
{
    float value;

    value = BalancedHold + (InterestRateHold*BalancedHold);
    return value;
}
```



```
Enter your balanced and interest rate:
1234.99 0.09
Your new balanced is 1346.14
Press any key to continue . . .
```

3. The function is to receive an array of floats that has a zero in its last slot. It is to return the number of slots preceding the zero. For example, if an array's contents are 3, 7, 2, 7, 1, 0, and then it should return 5. Call the function ArrayLen().

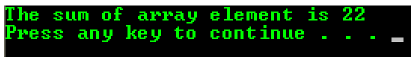
```
#include <stdio.h>

int ArrayLen(float [ ]);

void main(void)
{
    // the f modifier used to force to float
    float myArr[7] = {4.1f,7.2f,1.1f,8.5f,10.0f,2.2f,0.0f};
    int value;

    value = ArrayLen(myArr);
    printf("The number of slot minus the last is %d\n", value);
}

int ArrayLen(float num[ ])
{
    int count = 0, i;

    for(i=0;num[i] != 0;i++)
        count = count+1;
    return count;
}
```



```
The number of slot minus the last is 6
Press any key to continue . . .
```

4. The function AddArray() is to add up all the integers in the array until a 0 is encountered. The sum is to be returned. For example, in the array 3, 7, 2, 7, 1, 0, the returned value should be 20.

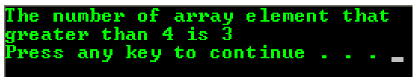
```
#include <stdio.h>

int AddArray(int [ ]);

void main(void)
{
    int myArr[7] = {4,2,1,8,5,2,0};
    int value;

    value = AddArray(myArr);
    printf("The sum of array element is %d\n", value);
}

int AddArray(int num[ ])
{
    int sum = 0, i;

    for(i=0;num[i] != 0;i++)
        sum = sum + num[i];
    return sum;
}
```



```
The sum of array element is 22
Press any key to continue . . .
```

5. The function CountArray() is to count how many integers n the array are greater than the first number. For example, in the array 3, 7, 2, 7, 1, -7, the two 7's are greater than 3, so 2 is returned. A negative number means that it is the end of the array.

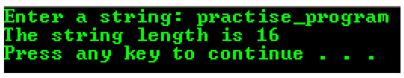
```
#include <stdio.h>

int CountArray(int [ ]);

void main(void)
{
    int myArr[7] = {4,2,10,8,5,2,-6};
    int value;

    value = CountArray(myArr);
    printf("The number of array element that\n"
              "greater than %d is %d\n", myArr[0], value);
}

int CountArray(int num[ ])
{
    int count = 0, i;

    for(i=0;num[i] >= 0;i++)
        if(num[i] > num[0])
    count = count + 1;
    return count;
}
```



```
The number of array element that
greater than 4 is 3
Press any key to continue . . .
```

6. The function receives a string and returns the number of characters in it. For example, if the string is "NOODLE", then the function should return 6. Call the function StringLen().

```
#include <stdio.h>

int StringLen(char [ ]);

void main(void)
{
    // initialize to dummy value
    char myArr[20] = "abc";
    int value;

    printf("Enter a string: ");
    scanf(" %s", &myArr);
    scanf_s(" %s", &myArr, sizeof(myArr));
    value = StringLen(myArr);
    printf("The string length is %d\n", value);
}

int StringLen(char holdStr[ ])
{
    int i;

     for(i=0;holdStr[i] != '\0';i++)
        // this is needed to complete the for loop definition
        ;
    return i;
}
```

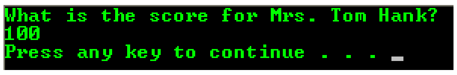


```
Enter a string: practise_program
The string length is 16
Press any key to continue . . .
```

More Practice on Functions

1. From the following code, will main() be able to get the value of score from GetScore() if this program is executed? And why?

```
#include <stdio.h>

void GetScore(char [ ]);

void main(void)
{
  GetScore("Mrs. Tom Hank");
}

void GetScore(char person[ ])
{
  int Score;

  printf("What is the score for %s?", person);
  scanf_s("%d", &Score, 1);
}
```



```
What is the score for Mrs. Tom Hank?
100
Press any key to continue . . .
```

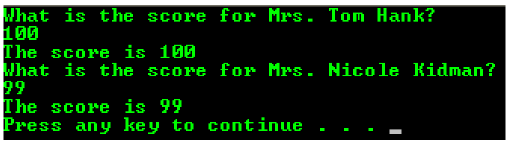
No. Because the stored score's value is not returned to main().

2. Let us change the program in practice 1 so that main() can receive a value. First, change the voids for GetScore() to int since the data type of the returned value will be of type int. Then add a statement at the end of the definition of GetScore() that will be return score: This will return the value to the calling function. Lastly, store this returned value in main() in a variable called x. This is done by assigning the function call to x. How does your program look now? Complete this task and show the output.

```
#include <stdio.h>

int GetScore(char [ ]);

void main(void)
{
   int x;
   x = GetScore("Mrs. Tom Hank");
}

int GetScore(char person[ ])
{
   int Score;

   printf("What is the score for %s?\n", person);
   // scanf("%d", &Score);
   scanf_s("%d", &Score, 1);
   return Score;
}
```

3. Next, rewrite only the main() so that it prints the value of x. Then it calls GetScore () again by passing “Mrs. Nicole Kidman”, also receives and prints her score.

```
#include <stdio.h>

int GetScore(char [ ]);

void main(void)
{
   int x;
   x = GetScore("Mrs. Tom Hank");
   printf("The score is %d\n", x);
   x = GetScore("Mrs. Nicole Kidman");
   printf("The score is %d\n", x);
}

int GetScore(char person[ ])
{
   int Score;

   printf("What is the score for %s?\n", person);
   // scanf("%d", &Score);
   scanf_s("%d", &Score, 1);
   return Score;
}
```

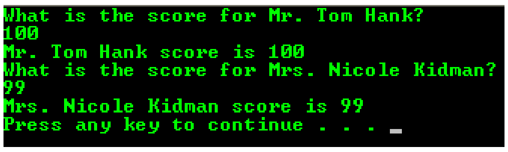


```
What is the score for Mrs. Tom Hank?
100
The score is 100
What is the score for Mrs. Nicole Kidman?
99
The score is 99
Press any key to continue . . .
```

```c
#include <stdio.h>

int GetScore(char [ ]);

void main(void)
{
   printf("Mr. Tom Hank score is %d\n", GetScore("Mr. Tom
Hank"));
   printf("Mrs. Nicole Kidman score is %d\n", GetScore("Mrs.
Nicole Kidman"));
}
```

4. Next, instead of storing the score in x each time and then printing out x, call GetScore() directly from the printf()'s and print the score for each of these people.

```c
int GetScore(char person[ ])
{
   int Score;

   printf("What is the score for %s?\n", person);
   // scanf("%d", &Score);
   scanf_s("%d", &Score, 1);
   return Score;
}
```



```
What is the score for Mr. Tom Hank?
100
Mr. Tom Hank score is 100
What is the score for Mrs. Nicole Kidman?
99
Mrs. Nicole Kidman score is 99
Press any key to continue . . .
```

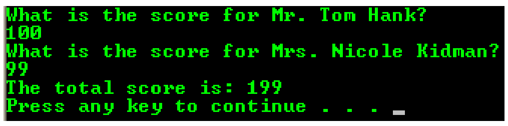
5. Do we need the variable x? If not, can you tell what is the disadvantage of not having it?

No. We don't need variable x. The disadvantage of not having x is that, once the score is printed, it is not available in main(). If we need that score later for some computation, then we would need to call get_score() again. In that case, it would have been to have saved the score in x at the beginning.

```c
#include <stdio.h>

int GetScore(char [ ]);

void main(void)
{
   int x;

   x = GetScore("Mr. Tom Hank");
   x = x + GetScore("Mrs. Nicole Kidman");
   printf("The total score is: %d\n",x);
}
```

6. By using x, rewrite main() so that it prints only the total score of both people.

```c
int GetScore(char person[ ])
{
   int Score;

   printf("What is the score for %s?\n", person);
   // scanf("%d", &Score);
   scanf_s("%d", &Score, 1);
   return Score;
}
```

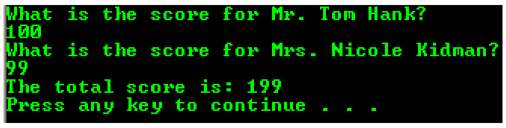


```
What is the score for Mr. Tom Hank?
100
What is the score for Mrs. Nicole Kidman?
99
The total score is: 199
Press any key to continue . . .
```

```c
#include <stdio.h>

int GetScore(char [ ]);

void main(void)
{
   int x;

   x = GetScore("Mr. Tom Hank");
   printf("The total score is: %d\n",x + GetScore("Mrs. Nicole
Kidman"));
}
```

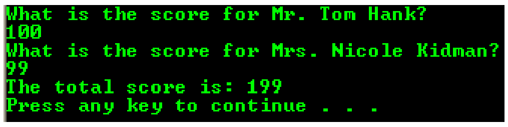
7. Rewrite your solution in 6 so that there is only one assignment statement instead of two. The second time you call GetScore(), it must be called from the printf().

```c
int GetScore(char person[ ])
{
   int Score;

   printf("What is the score for %s?\n", person);
   // scanf("%d", &Score);
   scanf_s("%d", &Score, 1);
   return Score;
}
```



```
What is the score for Mr. Tom Hank?
100
What is the score for Mrs. Nicole Kidman?
99
The total score is: 199
Press any key to continue . . .
```

8. Repeat solution 7 but use no x at all.



```
What is the score for Mr. Tom Hank?
100
What is the score for Mrs. Nicole Kidman?
99
The total score is: 199
Press any key to continue . . .
```

9. Write a function called PrintScore() that will receive a string and the score, an integer. It will print the string and the score and return nothing. Call the variable person and score.

```
void PrintScore(char person[ ], int score)
{
   printf("%s's score is %d\n", person, score);
}
```

10. Now write main() that will initialize person[ ] to “Mr. Krakatua” and call GetScore() by passing person[ ] to it. main() will receive the returned score in the variable called x. It will then call PrintScore() by passing person[ ] and x.

```
void main(void)
{
   char person[20] = "Mr. Krakatua";
   int x;

   x = GetScore(person);
   PrintScore(person);
}
```

11. Now instead of receiving the score into x, pass the received score directly to the PrintScore() function without using x. The GetScore() returns the score to PrintScore() which then prints it.

```
void main(void)
{
   char person[20] = "Mr. Krakatua";

   PrintScore(person, GetScore(person));
}
```

12. Write a program that measures the volumes of cylinders. To do that, we first need to write a function called Area(). This function will receive the radius of a circle and return its area. Use all floating point items. The area of a circle is about 3.1416 (π) times its radius squared or if put in formulae:

 Area = π * radius^2

```
#define PI 3.1416

float Area(float radius)
{
    return (PI * radius * radius);
}
```

13. Next, write a function called Volume(). It will receive the “radius” and the “height”. It will call Area() to receive its area, multiply this area by the height and return the volume. Use all floating points for those items.

```
float Volume((float radius, float height)
{
    return (Area(radius) * height);
}
```

14. Write main() that will pass a radius of 3.0 and a height of 5.5 to the Volume() function. This will be done within the printf() function in main(). Therefore, without the use of any variables, have main() print the volume of this cylinder. Notice that main() will call Volume() and Volume() will call Area().

```
void main(void)
{
       printf("%d\n", Volume(3.0, 5.5));
}
```

15. Write a function that will find the length of a string, that is, it will count the characters until a null character is encountered and return the count. Call the function StringLen(). The definition for such a function already exists in the string. h header file. Let us write our own. For example, if “Tale Story” is passed, then 10 will be returned. Here is the template of the function:

```
int StringLen(char s[ ])
{
     int i;
     ...
     ...
     ...
    return ...
  }
```

```
int StringLen(char s[ ])
{
   int i;
   for(i=0;s[i]  != '\0';i++)
     ;
    return i;
}
```

16. Write the RemoveStr() function. It will receive a string, the starting position in the string to be removed and the count, which is the total number of characters to be removed. For example, if str[] is “Bigger School” and this call is made, RemoveStr (str, 2, 3); then will become “Bir School”. Here is some of the code as your guide. Fill in the three blanks.

```
void RemoveStr(char s[ ], int start, int count)
{
int i, length;

length = StringLen(s);
for(i = start, j = start + _________; j <= __________; ++i, ++j)
     s[i] = ____________;
}
```

```
void RemoveStr(char s[ ], int start, int count)
{
int i, length;

length = StringLen(s);
for(i = start, j = start + count; j <= length; ++i, ++j)
     s[i] = s[j];
}
```

17. Using the example where str[ ] is “Bethel School” and the RemoveStr(str, 2, 3); call is made, answer the following questions.

    a. What will be the first value of i?
    b. What will be the first value of j?
    c. What will be the value of length?
    d. For the first time that s[i] = s[j]; is executed, what character is placed in which position? Or what character takes which character’s place in the string?
    e. How many times is i incremented? Is it as many times as j is incremented?

    a. The first value of i will be 2.
    b. The first value of j will be 5.
    c. The value of length will be 13.
    d. The 'l' of "Bethel" is placed in slot number 2 or the 'l' takes the place of 't'.
    e. i is incremented as many times as j is incremented that is 9 times. The last time, that is the ninth time, the condition j <= length becomes false and the loop is terminated.

System/pre-defined Functions

For commonly used routines, there are already functions for them such as printf()/printf_s() and scanf()/scanf_s(). These are pre-defined functions defined in the related header files. For the standard C or C++, the header files provided with the compiler that you use. A collection of the header files that perform certain categories of tasks normally supplied in a compiled form and the package called libraries, for example, you may have a C graphic library. For the non-standard or implementation specific, the header files normally supplied by the company that provide the compiler or from third party. So, before you create your own functions, check your compiler’s documentation for the related tasks need to be completed by functions. Finally don’t forget to learn about the mystery of main() function.