

COP 3223: C Programming Spring 2009

Functions In C – Part 2

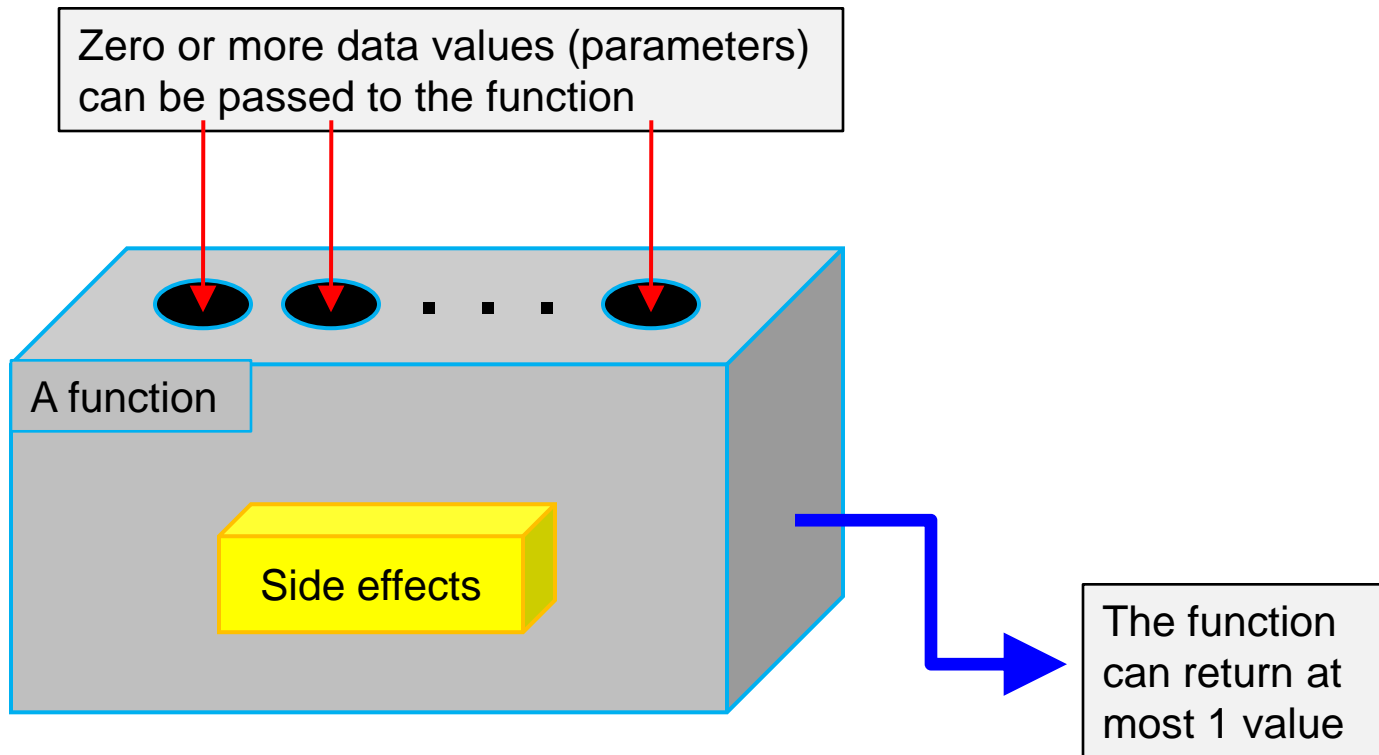
Instructor : Dr. Mark Llewellyn
markl@cs.ucf.edu
HEC 236, 407-823-2790
<http://www.cs.ucf.edu/courses/cop3223/spr2009/section1>

School of Electrical Engineering and Computer Science
University of Central Florida



More Functions In C

- The general purpose of a function is to receive zero or more pieces of data, operate on them, and return at most one piece of data.



More Functions In C

- If a function side effect occurs (there may not be any), it occurs during the execution of the function before the function returns.
- Side effects can involve accepting data from outside the program, sending data out of the program to the terminal or a file, or changing the value of a variable in the calling function.
- We'll write functions both with and without side effects. In general, for the safety of data, the programmer should attempt to minimize side effects that modify program data values and such side effects should be well-documented in the function.



More Functions In C

- Now we want to look in more detail at how different types of functions can be created in C.
- We noted before that a function might not return a value, in which case it is referred to as a `void function` (See Functions In C – Part 1 page 8). We also noted that a function might not require any parameters, in which case, the parameter list is replaced by the keyword `void`.
- Thus, there are four different categories of functions in C:
 1. Void functions that have no parameters.
 2. Void functions that have parameters.
 3. Functions that return a value that have no parameters.
 4. Functions that return a value that have parameters.



More Functions In C

- Each of the different types of functions as a role to play in various C programs, although not every type of function is necessarily used in every C program.
- Let's look at each of the different types of functions in C separately and see how they can be applied to certain problems and how they must be used.
- Void functions must be used only as statements; they cannot be used as expressions because they do not return a value and every expression must have a value.
- Void functions are typically only constructed and called for their side effects. In the following example, the function's side effect is to print something to the screen.



```
1 //Functions In C - Part 2 - Void function created for its side effects only
2 //The function in this program prints out header information in the output
3 //February 26, 2009    Written by: Mark Llewellyn
4
5 #include <stdio.h>
6 #define N 10
7
8 void printOutputHeader(void)
9 {
10     printf("The program that produced this data was written by: \n");
11     printf("Mark Llewellyn on February 26, 2009\n\n");
12 }//end printOutputHeader function
13
14 int main()
15 {
16     int i; // loop counter
17     int sum = 0; //variable to hold sum
18
19     printOutputHeader();
20     for (i = 1; i <= N; ++i) {
21         sum += i;
22         printf("The sum of the first %d integers is: %d\n", i, sum);
23     }//end for stmt
24
25     printf("\n\n");
26     system("PAUSE");
27     return 0;
28 }//end main function
29
```

A void function with no parameters



More Functions In C


```
C:\Courses\COP 3223 - C Programming\Spring 2009\COP 3223 P...
The program that produced this data was written by:
Mark Llewellyn on February 26, 2009
The sum of the first 1 integers is: 1
The sum of the first 2 integers is: 3
The sum of the first 3 integers is: 6
The sum of the first 4 integers is: 10
The sum of the first 5 integers is: 15
The sum of the first 6 integers is: 21
The sum of the first 7 integers is: 28
The sum of the first 8 integers is: 36
The sum of the first 9 integers is: 45
The sum of the first 10 integers is: 55
Press any key to continue . . .
```

The side effect produced by the void function is this header information printed in the output stream to the terminal

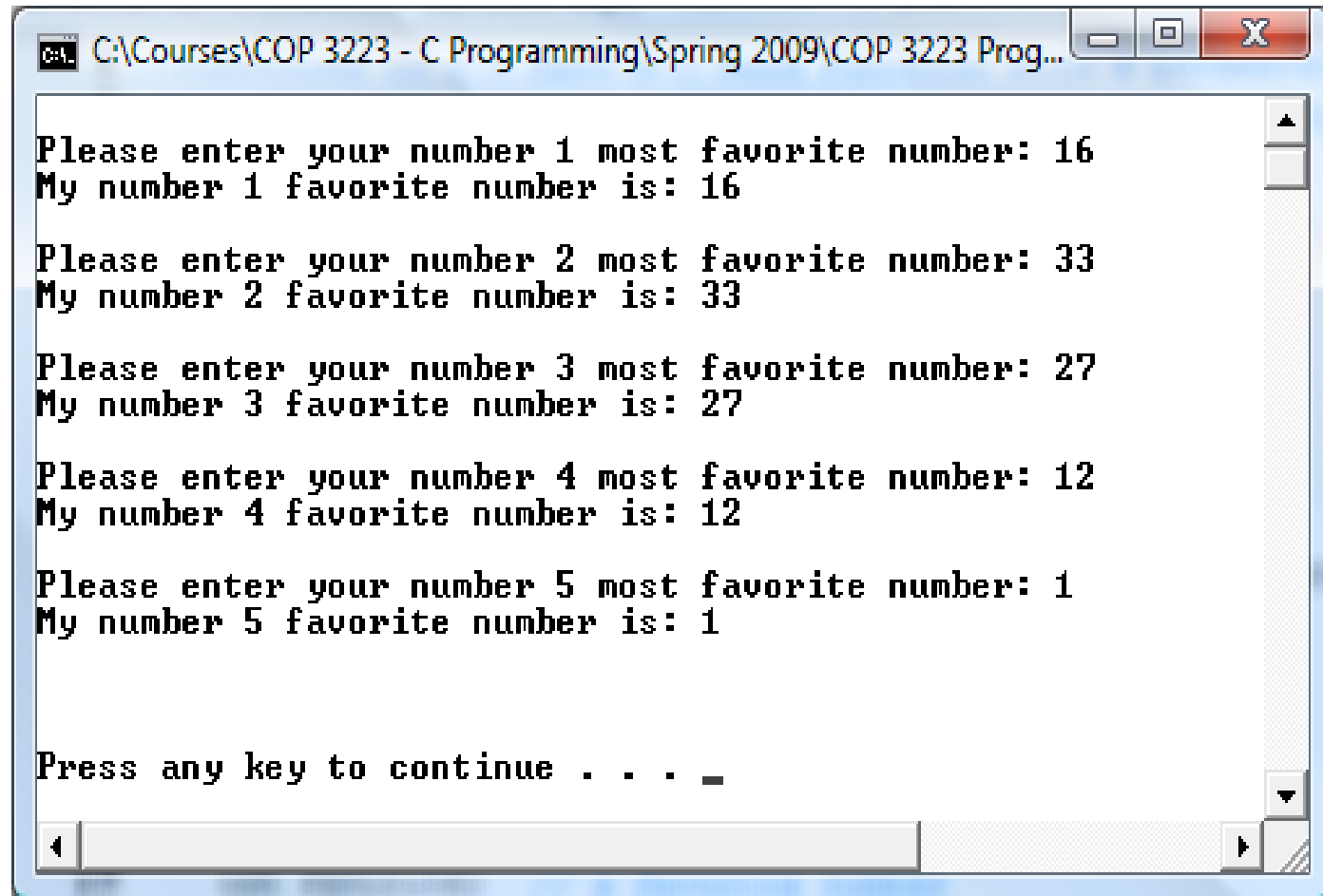


```
1 //Functions In C - Part 2 - Void function with a parameter
2 //another void function, this one with a parameter, but again the function's
3 //only real purpose is its side effect.
4 //February 26, 2009    Written by: Mark Llewellyn
5
6 #include <stdio.h>
7 #define MAX 5
8
9 void printFavoriteNumber(int level, int number)
10 {
11     printf("My number %d favorite number is: %d\n\n", level, number);
12 }//end printFavoriteNumber function
13
14 int main()
15 {
16     int i; //a counter
17     int favorite; // a favorite number
18
19     printf("\n");
20     for (i = 1; i <= MAX; ++i) {
21         printf("Please enter your number %d most favorite number: ", i);
22         scanf("%d", &favorite);
23         printFavoriteNumber(i, favorite);
24     }//end for stmt
25
26     printf("\n\n");
27     system("PAUSE");
28     return 0;
29 }//end main function
30
```

A void function with parameters



More Functions In C



A screenshot of a Windows command prompt window. The title bar reads "C:\Courses\COP 3223 - C Programming\Spring 2009\COP 3223 Prog...". The window contains the following text:

```
Please enter your number 1 most favorite number: 16  
My number 1 favorite number is: 16  
  
Please enter your number 2 most favorite number: 33  
My number 2 favorite number is: 33  
  
Please enter your number 3 most favorite number: 27  
My number 3 favorite number is: 27  
  
Please enter your number 4 most favorite number: 12  
My number 4 favorite number is: 12  
  
Please enter your number 5 most favorite number: 1  
My number 5 favorite number is: 1  
  
Press any key to continue . . . _
```



```
5
6 #include <stdio.h>
7 #define MAX 5
8
9 //this function reads integers from the keyboard
10 int getNumber(void)
11 {
12     int aNumber; //integer read in from keyboard
13     printf("Please enter an integer value: ");
14     scanf("%d", &aNumber);
15     return aNumber;
16 } //end getNumber function
17
18 int main()
19 {
20     //int aValue; //value returned from the function getNumber
21     int sum = 0; //sum of values returned so far
22     int i; //loop variable
23
24     for (i = 1; i <= MAX; ++i) {
25         sum = sum + getNumber();
26         printf("The sum so far is: %d \n\n", sum);
27     } //end for stmt
28
29     printf("\n\n");
30     system("PAUSE");
31     return 0;
32 } //end main function
33
```

A regular function with no parameters



```
C:\Courses\COP 3223 - C Programming\Spring 2...
Please enter an integer value: 1
The sum so far is: 1

Please enter an integer value: 9
The sum so far is: 10

Please enter an integer value: 3
The sum so far is: 13

Please enter an integer value: 4
The sum so far is: 17

Please enter an integer value: 6
The sum so far is: 23

Press any key to continue . . .
```



A regular function with parameters

```
8 #include <stdio.h>
9 #define LIMIT 200
10
11 //this function generate the next number in a sequence given the previous
12 //number and the stepsize
13 int nextInSequence(int currentValue, int step)
14 {
15     return(currentValue + step);
16 }//end nextInSequence function
17
18 int main()
19 {
20     int startingValue; //initial value in the sequence
21     int currentValue; //current value in the sequence
22     int incrementValue; //stepsize in the sequence
23     int nextValue; //value returned by the function nextInSequence
24     int i; //loop control
25     int counter = 0; //print counter for nice output format
26
27     printf("Enter the initial value in the sequence: ");
28     scanf("%d", &startingValue);
29     currentValue = startingValue;
30     printf("\nEnter the step-size for the sequence: ");
31     scanf("%d", &incrementValue);
32     printf("\n\nThe sequence of values is:\n");
33     printf("%5d ",startingValue); //print first value in sequence
34     counter++; //we've printed the first value in the sequence
35     for (i = startingValue; i <= LIMIT; ++i) {
36         //generate next value in the sequence
```



```
35     for (i = startingValue; i <= LIMIT; ++i) {
36         //generate next value in the sequence
37         nextValue = nextInSequence(currentValue, incrementValue);
38         //print the next value in the sequence
39         printf("%5d  ", nextValue);
40         counter++;
41         if (counter % 11 == 0){
42             printf("\n");
43             counter = 0;
44         }//end if stmt
45         currentValue = nextValue;
46     }//end for stmt
47
48     printf("\n\n");
49     system("PAUSE");
50     return 0;
51 }//end main function
52
53
```



Enter the initial value in the sequence: 2

Enter the step-size for the sequence: 6

The sequence of values is:

2	8	14	20	26	32	38	44	50	56	62
68	74	80	86	92	98	104	110	116	122	128
134	140	146	152	158	164	170	176	182	188	194
200	206	212	218	224	230	236	242	248	254	260
266	272	278	284	290	296	302	308	314	320	326
332	338	344	350	356	362	368	374	380	386	392
398	404	410	416	422	428	434	440	446	452	458
464	470	476	482	488	494	500	506	512	518	524
530	536	542	548	554	560	566	572	578	584	590
596	602	608	614	620	626	632	638	644	650	656
662	668	674	680	686	692	698	704	710	716	722
728	734	740	746	752	758	764	770	776	782	788
794	800	806	812	818	824	830	836	842	848	854
860	866	872	878	884	890	896	902	908	914	920
926	932	938	944	950	956	962	968	974	980	986
992	998	1004	1010	1016	1022	1028	1034	1040	1046	1052
1058	1064	1070	1076	1082	1088	1094	1100	1106	1112	1118
1124	1130	1136	1142	1148	1154	1160	1166	1172	1178	1184
1190	1196									

Press any key to continue . . .



More Functions In C

- Now let's write a program that includes more than one function.
- Let's stick with a problem that you are now familiar with – generating Fibonacci numbers (ok, maybe you are not familiar with it yet, but you have heard of it and seen it!).
- In this program we'll write a function that generates $F(X)$ where $F(X)$ represents the x^{th} Fibonacci number.
- We'll also write a function that determines if a given Fibonacci number is prime.



Fibonacci number generator function

```
4
5 #include <stdio.h>
6 #define MAX 30
7
8 //function to generate Fibonacci numbers
9 int fibonacciGenerator(int value)
10 {
11     int i; //loop counter
12     int fib0 = 0; //first fibonacci number
13     int fib1 = 1; //second fibonacci number
14     int previous1, previous2; //
15     int result; //a fibonacci number
16
17     if (value == 0)
18         return 0;
19     else if (value == 1)
20         return 1;
21     else {
22         previous2 = 0;
23         previous1 = 1;
24         for (i = 2; i <= value; ++i) {
25             result = previous2 + previous1;
26             previous2 = previous1;
27             previous1 = result;
28         } //end for stmt
29         return result;
30     } //end else stmt
31 } //end fibonacciGenerator function
32
```




```
34 //function returns 0 if not prime and 1 if prime
35 int testPrimeFibonacci(int testValue)
36 {
37     int numfactors; //number of factors of testValue
38     int i; //loop control
39
40     if (testValue < 2){ //number is not prime
41         return 1;
42     }
43     else {
44         numfactors = 0;
45         for(i = 1; i <= testValue; ++i) {
46             if (testValue % i == 0) {
47                 numfactors++;
48             } //end if stmt
49         } //end for stmt
50         if (numfactors > 2){ //number is not prime
51             return 0;
52         } //end if
53         else {
54             return 1;
55         } //end else stmt
56     } //end else stmt
57 } //end testPrimeFibonacci
58
59
```

Function to test if Fibonacci number is prime



```
59
60 int main()
61 {
62     int num; //user entered value
63     int i; //loop control
64     int fibonacciNumber[MAX] = {0}; //array of fibonacci numbers
65
66     for (i = 0; i < MAX; ++i) {
67         fibonacciNumber[i] = fibonacciGenerator(i);
68         printf("Fibonacci[%d] = %d\n", i, fibonacciNumber[i]);
69     } //end for stmt
70     printf("\n\n");
71     for (i = 0; i < MAX; ++i) {
72         if (testPrimeFibonacci(fibonacciNumber[i]) == 0){
73             printf("Fibonacci[%d] = %d is not prime\n", i, fibonacciNumber[i]);
74         } //end if stmt
75         else {
76             printf("Fibonacci[%d] = %d is prime\n", i, fibonacciNumber[i]);
77         } //end else stmt
78     } //end for stmt
79
80     printf("\n\n");
81     system("PAUSE");
82     return 0;
83 } //end main function
```

Main function



```
Fibonacci[0] = 0
Fibonacci[1] = 1
Fibonacci[2] = 1
Fibonacci[3] = 2
Fibonacci[4] = 3
Fibonacci[5] = 5
Fibonacci[6] = 8
Fibonacci[7] = 13
Fibonacci[8] = 21
Fibonacci[9] = 34
Fibonacci[10] = 55
Fibonacci[11] = 89
Fibonacci[12] = 144
Fibonacci[13] = 233
Fibonacci[14] = 377
Fibonacci[15] = 610
Fibonacci[16] = 987
Fibonacci[17] = 1597
Fibonacci[18] = 2584
Fibonacci[19] = 4181
Fibonacci[20] = 6765
Fibonacci[21] = 10946
Fibonacci[22] = 17711
Fibonacci[23] = 28657
Fibonacci[24] = 46368
Fibonacci[25] = 75025
Fibonacci[26] = 121393
Fibonacci[27] = 196418
Fibonacci[28] = 317811
Fibonacci[29] = 514229
```

```
C:\K:\COP 3223 - Spring 2009\COP 3223 Program Files\Functions in C - Part 2\fibonacci with ...
Fibonacci[0] = 0 is prime
Fibonacci[1] = 1 is prime
Fibonacci[2] = 1 is prime
Fibonacci[3] = 2 is prime
Fibonacci[4] = 3 is prime
Fibonacci[5] = 5 is prime
Fibonacci[6] = 8 is not prime
Fibonacci[7] = 13 is prime
Fibonacci[8] = 21 is not prime
Fibonacci[9] = 34 is not prime
Fibonacci[10] = 55 is not prime
Fibonacci[11] = 89 is prime
Fibonacci[12] = 144 is not prime
Fibonacci[13] = 233 is prime
Fibonacci[14] = 377 is not prime
Fibonacci[15] = 610 is not prime
Fibonacci[16] = 987 is not prime
Fibonacci[17] = 1597 is prime
Fibonacci[18] = 2584 is not prime
Fibonacci[19] = 4181 is not prime
Fibonacci[20] = 6765 is not prime
Fibonacci[21] = 10946 is not prime
Fibonacci[22] = 17711 is not prime
Fibonacci[23] = 28657 is prime
Fibonacci[24] = 46368 is not prime
Fibonacci[25] = 75025 is not prime
Fibonacci[26] = 121393 is not prime
Fibonacci[27] = 196418 is not prime
Fibonacci[28] = 317811 is not prime
Fibonacci[29] = 514229 is prime

Press any key to continue . . .
```

Practice Problems

1. Write a C program that uses a function to simulate throwing two standard dice and incorporate this function into the program where the main objective of the program is to play the game of craps. Have the program play 5 games in one execution of the program. The rules for craps are given below.

Two standard die are thrown by the player. If the sum of the two die on the first throw is 7 or 11, the player wins. If the sum is 2, 3, or 12 on the first throw (called "craps"), the player loses (i.e., the house wins). If the sum is 4, 5, 6, 8, 9 or 10 on the first throw, then that sum becomes the player's "point". To win from this point, the player must continue rolling the dice until you "make the point". The player loses by rolling a 7 before making the point.



GAME 1

Player rolled 2 + 4 = 6

Point is 6

Player rolled 1 + 3 = 4

Player rolled 5 + 4 = 9

Player rolled 3 + 2 = 5

Player rolled 4 + 6 = 10

Player rolled 3 + 4 = 7

Player loses

GAME 2

Player rolled 2 + 1 = 3

Player loses

GAME 3

Player rolled 5 + 5 = 10

Point is 10

Player rolled 6 + 4 = 10

Player wins

GAME 4

Player rolled 2 + 2 = 4

Point is 4

Player rolled 5 + 5 = 10

Player rolled 2 + 2 = 4

Player wins

GAME 5

Player rolled 1 + 6 = 7

Player wins

