



# Top-Down Design with Functions

**Abdallah Karakra**

**Computer Science Department**

**Comp 230**

**Thursday, November 05, 2015**

# Functions

- Definition:

A function is a group of statements that together perform a task. **Every C program has at least one function, which is `main()`,** and all the most trivial programs can define additional functions

# Functions

- Two types:
  1. C library functions (**sqrt (x), abs (x),...**)
  2. User defined functions (Your own functions)

# Some Mathematical Functions

Function	Standard Header File	Example	Argument(s)	Result
<b>abs(x)</b>	<stdio.h>	x=-5 abs(x)=5	int	int
<b>ceil(x)</b>	<math.h>	x=45.23 ceil(x)=46	double	double
<b>cos(x)</b>	<math.h>	x=0.0 cos(x)=1.0	double (radians)	double
<b>exp(x)</b>	<math.h>	x=1.0 exp(x)=2.71828	double	double

# Some Mathematical Functions

Function	Standard Header File	Example	Argument(s)	Result
<b>fabs(x)</b>	<math.h>	x=-8.432 fab(x)=8.432	double	double
<b>floor(x)</b>	<math.h>	x=45.23 floor(x)=45	double	double
log(x)	<math.h>	x=2.71828 log(x)=1.0	double	double
log10(x)	<math.h>	x=100.0 log10(x)=2.0	double	double

# Some Mathematical Functions

Function	Standard Header File	Example	Argument(s)	Result
<b>pow(x,y)</b>	<math.h>	x=0.16 y=0.5 pow(x,y)=0.4	double double	double
<b>sin(x)</b>	<math.h>	x=1.5708 sin(x)=1.0	double (radians)	double
<b>sqrt(x)</b>	<math.h>	x=2.25 sqrt(x)=1.5	double	double
<b>tan(x)</b>	<math.h>	x=0.0 tan(x)=0.0	double (radians)	double

# Functions

- Why Functions:

1) Useful for C programmers to divide their programs into separate functions ( instead of big “chunk” ). This make it easy to debug the code and handling error.

2) reusability:

- Once a function is defined, it can be used over and over and over again.
- You can invoke the same function many times in your program.
- Use same function in several different (and separate) programs.

# Functions

- Types of functions:
  1. Function with **no arguments** and **no return value**.
  2. Function with **no arguments** but **return value**
  3. Function with **arguments** and **no return value**
  4. Function **with argument** and **a return value**



# Functions

- How to write a function:
  1. Function prototype
  2. Function Definition
  3. Function Call

# Functions

How to write a function: **Function prototype**

Tells the compiler about a function's name, return type, and parameters.

**return\_type** **function\_name** ( **parameter list** )

**int** **sum** ( **int** , **int** ); // **with parameters** and **return value**

**void** **printNum** ( **int** ); // **with parameters** and **no return value**

**float** **area** (); // **no parameters** and **with return value**

**double** **circumference** ( **double** ); // **with parameters** and **return value**

**void** **printChar** ( **char** ); // **with parameters** and **no return value**

**void** **printSquare** (); // **no arguments** and **no return value**

# Functions

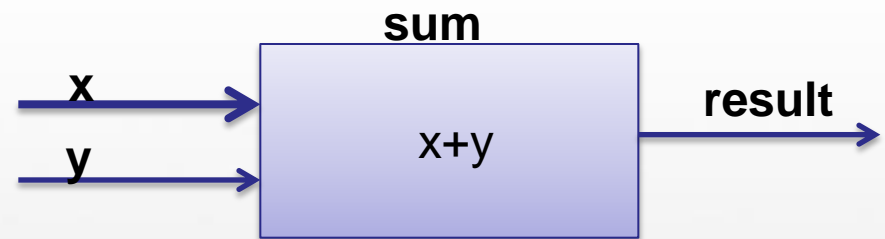
- How to write a function: **Function Definition**  
Provides the actual body of the function.

```
return_type function_name ( parameter list )  
{  
    body of the function  
}
```

# Functions

- How to write a function: **Function Definition**

```
int sum ( int x, int y)
{
    int result;
    result= x+y;
    return result;
}
```



# Functions

- How to write a function: **Function Definition**

```
void printNum ( int x)
{
    printf(“%d”, x);
}
```

# Functions

- How to write a function: **Function Definition**

**double** circumference (double r)

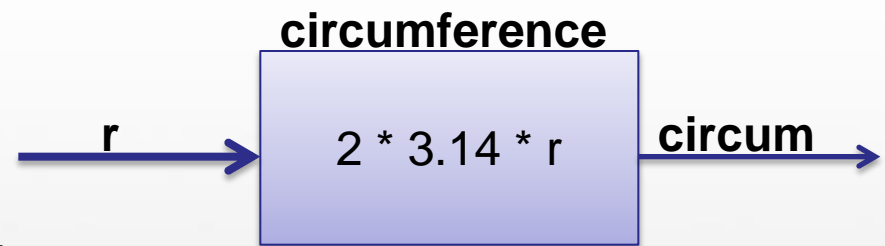
{

double circum;

circum= 2 \* 3.14 \* r;

**return** circum;

}



# Functions

- How to write a function: **Function Call**

To use a function, you will have to call that function to perform the defined task.

```
int mySum = sum (x,y);
```

```
double circum = circumference (r);
```

```
printNum(x);
```

# Functions

- How to write a function: **Terminology**

**Return Type**: A function may **return a value**. The **return\_type** is the data type of the value the function returns. Some functions perform the desired **operations without returning a value**. In this case, the **return\_type** is the keyword **void**.



# Functions

- How to write a function: **Terminology**

**Function Name:** This is the actual name of the function. The function name and the parameter list together constitute the function signature.

# Functions

- How to write a function: **Terminology**

**Parameters:** A parameter is like a placeholder. When a function is invoked, you pass a value to the parameter. This value is referred to as actual parameter or argument. The parameter list refers to the type, order, and number of the parameters of a function. **Parameters are optional**; that is, a **function may contain no parameters.**

# Functions

- How to write a function: **Terminology**

**Function Body:** The function body contains a collection of statements that define what the function does.

# Functions (Exercises)

- Write a C program to compute the **area** of a circle with radius  $r$ . (Recall that  $A = \pi r^2$ .)
- Write a C program to compute the **circumference** of a circle with radius  $r$ . (Recall that  $\text{circum} = 2 \pi r$ .)

```

#include <stdio.h>
#include <math.h>
#define PI 3.141593
// function prototype
double computeArea (double);
int main()
{
    double r, area; //Declare variables.
    //Enter the radius.
    printf("Enter the radius of the circle: \n");
    scanf("%lf",&r);
    area= computeArea(r); //call function
    // Print the value of the area.
    printf("The area of a circle with radius %5.3f is %5.3f. \n",r,area);
    // Exit program.
    return 0;
}
// Function Definition
double computeArea (double r)
{
    double area;
    // Compute the area of the circle.
    area = PI*pow (r,2);
    return area;
}

```

```

#include <stdio.h>
#define PI 3.141593
// function prototype
double computeCircumference (double,double);
int main()
{
    double r, circum; // Declare variables.
    // Enter the radius.
    printf("Enter the radius of the circle: \n");
    scanf("%lf",&r);
    circum= computeCircumference(r,PI); //call function
    // Print the value of the circumference
    printf("The circumference of a circle with radius %5.3f is %5.3f. \n",r,circum);
    // Exit program.
    return 0;
}
// Function Definition
double computeCircumference (double r,double pi)
{
    double circum;
    // Compute the circumference of the circle.
    circum = 2*pi*r;
    return circum;
}

```

```

#include <stdio.h>
#define PI 3.141593
// function prototype
double computeCircumference (double);
int main()
{
    double r, circum; // Declare variables.
    // Enter the radius.
    printf("Enter the radius of the circle: \n");
    scanf("%lf", &r);
    circum= computeCircumference(r); //call function
    // Print the value of the circumference
    printf("The circumference of a circle with radius %5.3f is %5.3f. \n", r, circum);
    // Exit program.
    return 0;
}
// Function Definition
double computeCircumference (double r)
{
    double circum;
    // Compute the circumference of the circle.
    circum = 2*PI*r;
    return circum;
}

```

# Functions (more practice)

Write a complete c program that asks the user to enter two numbers, finds and prints the sum of them. Your program should include at least one function called **sum** to return the sum of the two numbers.

Function prototype

```
int sum (int x, int y)
```



# Functions (more practice)

- write the **prototype** of average, a function that returns the average of its two type double input parameters.

```
double average (double, double );
```

- write a definition for the above function prototype.

```
double average (double n1, double n2 )
```

```
{
```

```
    return ((n1 + n2) / 2.0);
```

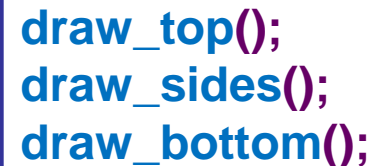
```
}
```

# Functions (more practice)

Write a function call for each function prototype.

```
#include <stdio.h>
/*Functions Prototypes */
void draw_top();
void draw_sides(void);
void draw_bottom(void);
```

```
int main(void)
{
    /*Functions calls */
    return (0);
}
/* Functions Definitions */
.....
```



```
draw_top();
draw_sides();
draw_bottom();
```

# Functions (more practice)

Rewrite the following mathematical expression using **C** functions

$$x = b^2 + c^2 - 2bc$$

```
double x, b, c;
```

```
x = pow(b,2)+pow(c,2)-2*b*c;
```

# Functions (more practice)

Rewrite the following mathematical expression using **C** functions

$$a^2 = b^2 + c^2 - 2bc \cos \alpha, \text{ where } \alpha \text{ in degree}$$

```
double a, b, c, alpha;
```

```
a=sqrt(pow(b,2)+pow(c,2) - 2 * b* c* cos(alpha * PI / 180.0));
```

*converting from degrees to radians is to simply multiply the number of degree by  $\pi / 180^\circ$*

# Functions (more practice)

1. Write a complete c program to do the following.

$$Y = x^3 + x^2 + x$$

Your program should include two functions, **cubic to return x to the power of three** and **square to return x to the power of two**.

2. Write a complete c program with a function that takes a number and prints it..

3. Write a complete c program with a function that reads a number and then prints it..

```

#include <stdio.h>
int cubic    (int);
int square   (int);
int main()
{
    int x,y;
    printf("Please enter the value of x: ");
    scanf ("%d",&x);
    y= cubic(x) + square(x) +x;
    printf("y = %d ",y);
    return 0;
}
int cubic    (int x)
{
    return  (x * x * x);
}
int square   (int x)
{
    return  (x *x );
}

```

```
#include <stdio.h>
void printNumber (int);
int main()
{
    int number;
    printf("please enter a number");
    scanf("%d", &number);
    printNumber (number);
    return 0;
}
void printNumber (int x)
{
    printf("%d", x);
}
```

```
#include <stdio.h>
void printNumber ();
int main()
{
    printNumber ();
    return 0;
}
void printNumber ()
{
    int number;
    printf("please enter a number");
    scanf("%d", &number);
    printf("%d", number);
}
```



# Functions (more practice)

What will be the output if you execute the following C code?

```
#include <stdio.h>
int f(int , int , int );
int main ()
{
    int q;
    q = f(3, 3, 4);
    printf ("q is %d ", q);
}
int f(int q, int b, int c)
{
    int p;
    p = q * b + 2 * c;
    return (p);
}
```

Main function

q

f function

q=3 , b=3 , c=4

p=??

Output (screen):

q is 17

# Functions (more practice)

What will be the output if you execute the following C code?

```
#include <stdio.h>
int f(int , int , int );
int main ()
{
    int q;
    q = f(3, 3, 4);
    printf ("q is %d ", q);
}
int f(int q, int b, int c)
{
    int p;
    p = q * b + 2 * c;
    return (p);
}
```

Main function

q

f function

q=3 , b=3 , c=4

p=??

Output (screen):

q is 17

# **Let us review the concepts:**

**Choose the best answer :**

- 1. When using a function, what is the first thing you must do?**
  - a) prototype**
  - b) declare**
  - c) initialize**
  
- 2. Where should the prototype be?**
  - a) after int main()**
  - b) before int main()**
  - c) a prototype isn't necessary**
  
- 3. Here is a function, double numbers (int x), what is the name of this function?**
  - a) double**
  - b) int x**
  - c) numbers**

# *Let us review the concepts:*

Choose the best answer :

4. From question 3, what data type will this function return?
  - a) int
  - b) double
  - c) char
  
5. From question 4, what data type will this function take in?
  - a) int
  - b) double
  - c) char
  
6. `int my_function (double a)`, what type of data will this functions take in?
  - a) double
  - b) int & double
  - c) int

# **Let us review the concepts:**

**Choose the best answer :**

- 7. Say we have a function, double subtract (double x, double y), what is the correct way to call this function in the main program?**  
a) subtract (x)    b) subtract (y)    c) subtract (x,y)
- 8. If a variable is declared inside a function, what kind of variable is this?**  
a) global variable    b) local variable    c) extended variable
- 9. If we have a function int stop (int n) , are we able to send it a different variable in the main program or does it have to be n. For example, stop (x) .**  
a) yes    b) no

# **Let us review the concepts:**

**Answers :**

- 1) a) prototype**
- 2) b) before int main()**
- 3) c) numbers**
- 4) b) double**
- 5) a) int**
- 6) a) double**
- 7) c) subtract (x,y)**
- 8) b) local variable**
- 9) a) yes**

# Extra Exercises

Given the following declarations:

**double x; int y;**

What value is assigned to x and y in the following statements:

1) **x= ceil (34.234);**

2) **x= ceil (34.534);**

3) **x= ceil (34.0);**

4) **x= ceil (34);**

5) **y=abs (-345);**

6) **x= floor (34);**

7) **x= floor (34.89);**

8) **x=fabs(-8.532);**

9) **x=pow(2,4);**

10) **x=floor(21.8 + 0.8);**

11) **x=floor(-7.5) ;**

12) **x=floor(-7.5) \* pow(3.0, 2.0);**

13) **x=ceil(-7.5) ;**

14) **x=ceil(-7.5) \* pow(3.0, 2.0);**

# Extra Exercises

Rewrite the following mathematical expression using **C functions**:

$$root = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

**Hint: Compute two roots**

```
double a, b, c;
```

```
double root_1=
```

```
double root_2=
```



# Extra Exercises

## Choose the best answer :

1. Which is not a proper prototype?
  - A. `int funct(char x, char y);`
  - B. `double funct(char x)`
  - C. `void funct();`
  - D. `char x();`
2. What is the return type of the function with prototype: `"int func(char x, float v, double t);"`
  - A. `char`
  - B. `int`
  - C. `float`
  - D. `double`
3. Which of the following is a valid function call (assuming the function exists)?
  - A. `funct;`
  - B. `funct x, y;`
  - C. `funct();`
  - D. `int funct();`

# Extra Exercises

**Choose the best answer :**

4. Which of the following is a correct function definition?

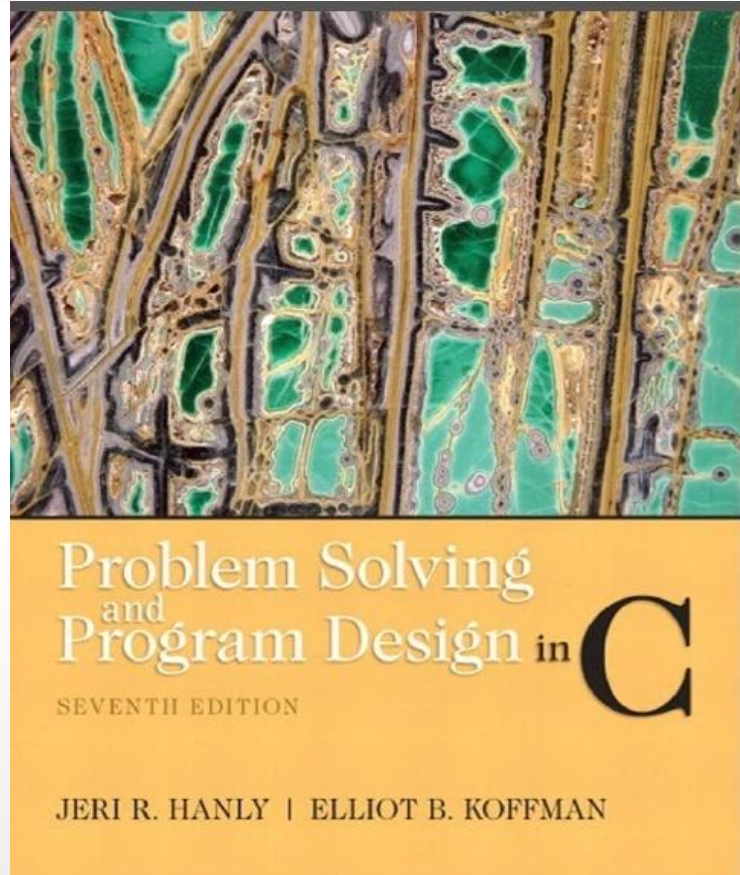
- A. `int funct();`
- B. `int funct(int x) {return x=x+1;}`
- C. `void funct(int) {printf("Hello");}`
- D. `void funct(x) {printf("Hello");}`

**Write a function to return the square of an integer number ?**

# Question?

A good question deserve a good grade...





## **References:**

***Problem Solving & Program Design in C (main reference)***

**[http://www.tutorialspoint.com/cprogramming/c\\_functions.htm](http://www.tutorialspoint.com/cprogramming/c_functions.htm)**

**<http://www.programiz.com/c-programming/types-user-defined-functions>**