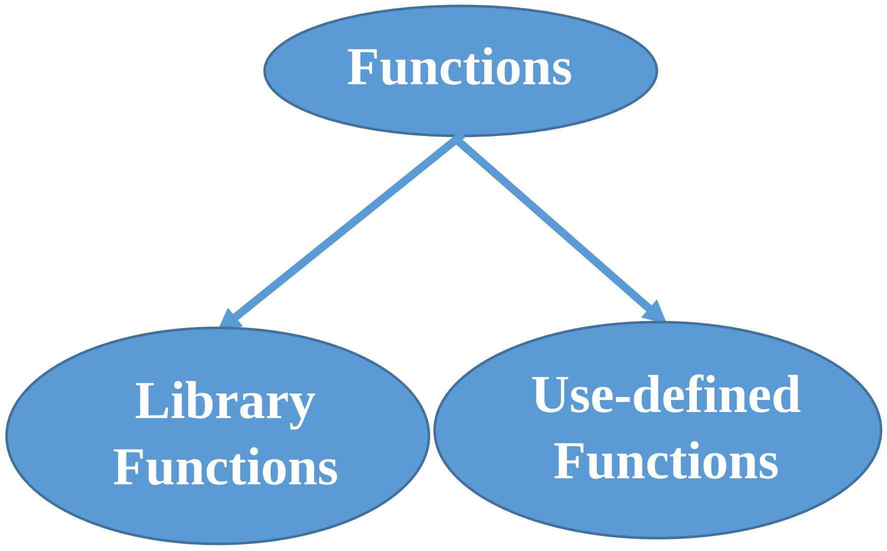
C Functions

1. Objective

- What is C function?
- Types of C functions
- How to invoke functions?
- Local variables in C functions.
- Parameter passing in C
- Functions the do not return any values.

2. Function Definition

- The length of your program can be reduced.
- It becomes easy
- Functions can be called several times within your program.
- There are two types of functions in C:



- All variables declared inside a function are local variables and are not accessible outside the function.
- Syntax:

***return-value-type function-name**(**parameter-list**)*
{
declarations and statements
}

Where

- **function-name**: any valid identifier
- **Return-value-type**:
  - data type of the result (default **int**)
  - **void** – indicates that the function returns nothing
- **Parameter-list**:
  - Also called **formal parameters**.
  - A list of variable, comma separated list, declares parameters:
  - A type must be listed explicitly for each parameter unless, the parameter is of type **int**
- **Declarations and statements**: function body (block)

        - Variables can be declared inside blocks (can be nested)
        - You cannot create functions within inside other functions.
    - Returning control
        - If nothing returned
            - **return;**
            - or, until reaches right brace
        - If something returned
            - **return** *expression***;**

- **Example:**

```
int findMax(int a,int b){

   if (a <b)
      return (b);
   else return (a);
}
```

3. Function Prototypes

- Function prototype is also called function signature defines the header of a function declaration:

  ***return-value-type function-name*(*parameter-list*);**

- A prototype functions is only used when its implementation comes after the main function.

- Example:

  int findMax(int a,int b);

4. Calling Functions

- Used when invoking functions
- If the function returns a value:
  - Given the following function:

  ***return-value-type function-name*(*parameter-list*)**
  {
  declarations and statements
  }

The call of this function is:

var = ***function-name****(list-values);*

Where var is of type: ***return-value-type***

List-values are also called **actual parameters.**

- If the function does not return any value:

  void function-name*(* ***parameter-list*** *)*
  {
  declarations and statements
  }

  The call of this function is:

  function-name*(list-values);*

  - Example:
    printf()

- Once a function is completely executed, control is passed back to the calling environment when the closing brace of the body is encountered.

- Values are passed to functions using one of the following modes:
    - Call by value
        - Copy of argument passed to function
        - Changes in function do not effect original
        - Use when function does not need to modify argument
        - In case you do not want to change the content of the original variables.

    - **Call by reference**
        - To pass original values
        - It changes the original variables

- We will now focus on Call by Value and we will revisit Call by Reference later.

- Example:

```
//gcc 5.4.0

#include  <stdio.h>
int findMax(int a,int b){

   if (a <b)
      return (b);
   else return (a);
}


int main(void)
{
   int x =5;
   int y = 10;
   printf ("Max of %d and %d is %d\n", x, y, findMax(x,y));

   x = 100; y = -1;
   printf ("Max of %d and %d is %d\n", x, y, findMax(x,y));

   return 0;
}
```

5. Questions/Practice

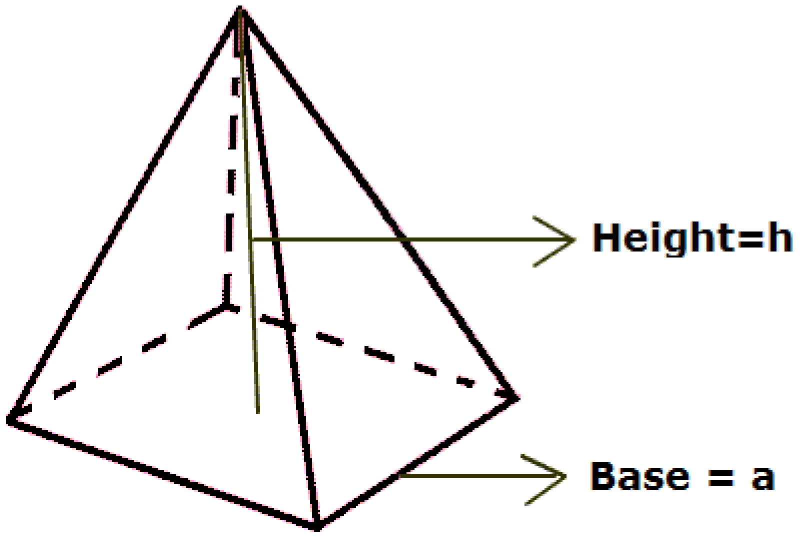
- Add the elements of an array:

```
//gcc 5.4.0
#include  <stdio.h>
int sumaray(int myparam[], int limit){
    int i;
    int sum=0;
    for(i=0; i < limit; ++i)
          sum += myparam[i];
    return (sum);
}


void main(void)
{
    int myarr1[5] = {1,2,3,4,5};
    int myarr2[8] = {1,2,3,4,5,6};

    printf ("sum=%d\n", sumaray(myarr1, 5));
    printf ("sum=%d\n", sumaray(myarr2, 8));
}
```

- A square pyramid is defined as follows:

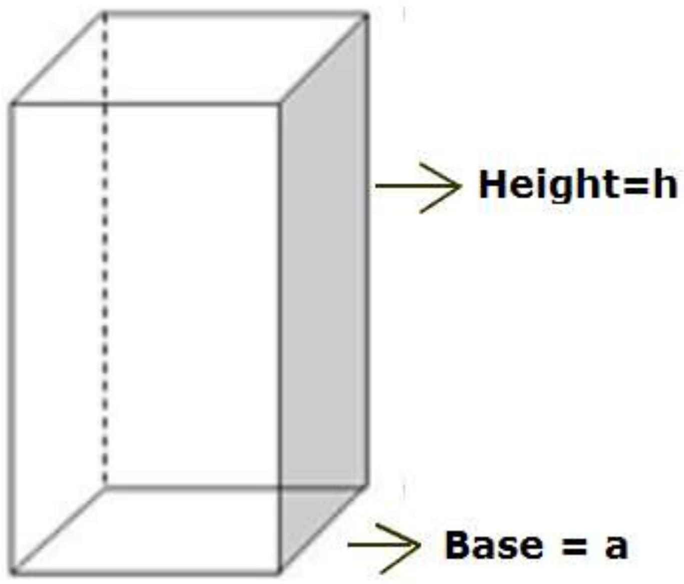


Where a is the length of the base and h is the height of the pyramid
The volume of a square pyramid is:

$$V = a^2 * \frac{1}{3} h$$

Write a program that calculates V. First write a function to calculate the area of square and call this function in another function that calculate the volume. This function should be called from main function.

- A Right square prism is defined as follows:



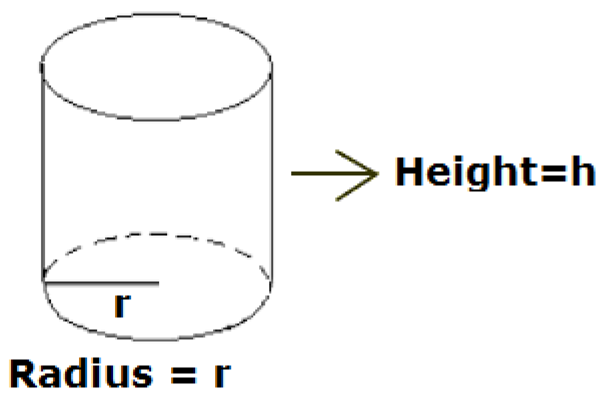
Where a is the length of the base and h is the height of the prism.

The volume of a square prism is:

$$V = a^2 * h$$

Use the square function you wrote in the previous problem to write a function that calculates V and call this function from main function.

- A cylindrical tank is defined as follows:



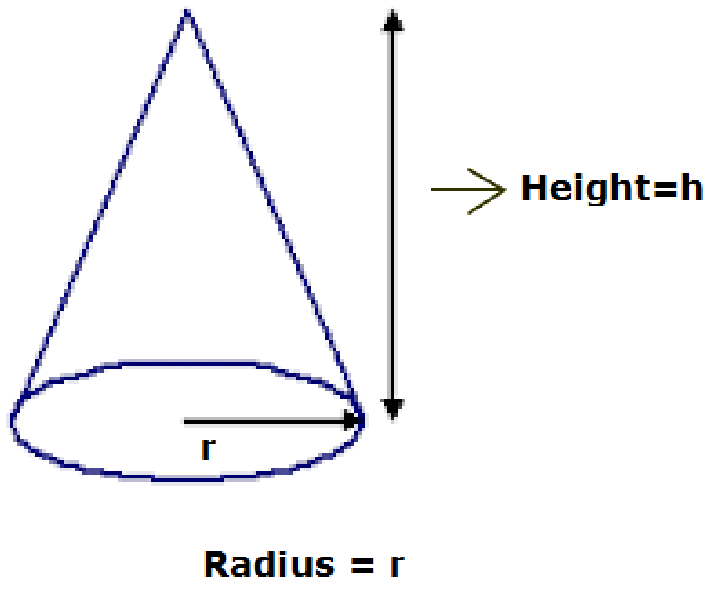
The volume of cylindrical tank is:

$$V = \pi r^2 * h$$

Write a C program that calculate the volume of a cylindrical tank. You program should include two functions:

- One for the area of a circle that will be called for the volume function.
- A function for calculating the volume. This function should be called from the main function.

- A cone is defined as follows:



The volume of cone is:

$$V = \pi r^2 * \frac{h}{3}$$

  - Write a C program that calculate the volume of a cone. You program should include two functions:
    - One for the area of a circle that will be called for the volume function.
    - A function for calculating the volume. This function should be called from the main function.