C FUNCTIONS

FUNCTIONS

- Experience has shown that the best way to develop and maintain a large program is to construct it from smaller pieces or modules each of which is more manageable than the original program.
- Modules in C are called functions
- Functions perform a specific purpose in the program and they can be called in another part of the program
- When a function is called, its functionality is applied in the section of the program where it has been called.

Two kinds of a function

Functions in C standard library: C standard library provides a rich collection of functions for performing common mathematical calculations, string manipulations, input/output e.t.c (Check document attached)

Programmer defined functions: A programmer writes functions to define specific tasks.

Functions are invoked by a function call, which specifies the function name and provides information(as arguments)that the called function needs in order to perform its designated task.

Cont.

- Functions are normally used in a program by writing the name of the function followed by a left parenthesis followed by the arguments (or a comma separated list of argument)of the function followed by a right parenthesis.
- e.g a programmer desiring to calculate and print the square root of 900.0 might use a standard library function and write it as . printf(“%2f”,sqrt(900.0));

Reasons for using functions

- The divide and conquer approach makes program development more manageable
- Software reusability-existing functions are used as building blocks to create new programs.
- To avoid repeating code in a program-packaging code as a function allows the code to be executed from several locations in a program simply by calling the function

NB: Each function should be limited to performing a single, well defined task, and the function name should effectively express the task. This facilitates abstraction and promotes software reusability.

Function definitions

- Each program we have presented has consisted of a function called *main* that called standard library functions to accomplish its tasks.
- We now consider how programmers write their own customized function
- Syntax

```
return_value_type          function_name (parameter_list)
{
definitions statements
}
```

Syntax Explained

- The function_name is any valid identifier (variable).
- The return_value_type is the data type of the result returned to the caller.
  - The return_value_type *void* indicates that a function does not return a value.
  - An unspecified return_value_type is assumed by the compiler to be *int*.
- However omitting the return type is discouraged.
- Together, the return_value_type,function_name and parameter_list are sometimes referred to as the ***function header.***

Cont.

- The parameter_list is a comma-separated list that specifies the parameters received by the function when it is called.
- if a function doesn’t receive any values, parameter_list is void.
- A type must be listed explicitly for each parameter unless the parameter is of type int.
- If a type is not listed, int is assumed.

Example 1: Creating a function known as fun and using it to print value of x

```
#include <stdio.h>
void fun(int x)
{
printf("x = %d", x);
}
int main(void)
{
    int x = 20;
    fun(x);
    return 0;
}
```

Example 2: function

// An example function that takes two parameters 'x' and 'y'

// as input and returns max of two input numbers

See next slide

Example 2

```
#include <stdio.h>
int max(int x, int y)
{
    if (x > y)
      return x;
    else
      return y;
}
```

```
int main(void)
{
    int a = 10, b = 20;
    // Calling above function to find
max of 'a' and 'b'
    int m = max(a, b);
    printf("m is %d", m);
    return 0;
}
```

Example 3: Using functions to find the area of a rectangle

```
#include <stdio.h>
int area (int length, int width)
{
return  length*width;
}
```

```
int main() {
        int length=10;
        int width=4;
        int a=area(length, width);
        printf("%d", a);
        return 0;
}
```

Assignment

- Research on the difference between call-by-value and call-by-reference
- Write a program that uses functions to classify a student as either pass if the student gets 50 marks and above and fail if the student gets below 50 marks

Arrays

Arrays

- An array is defined as the collection of similar type of data items stored at contiguous memory locations.
- Arrays are the derived data type in C programming language which can store the primitive type of data such as int, char, double, float, etc.
- It also has the capability to store the collection of derived data types, such as pointers, structure, etc.
- The array is the simplest data structure where each data element can be randomly accessed by using its index number.

Cont.

- C array is beneficial if you have to store similar elements.
- For example, if we want to store the marks of a student in 6 subjects, then we don't need to define different variables for the marks in the different subject.
- Instead of that, we can define an array which can store the marks in each subject at the contiguous memory locations.
- By using the array, we can access the elements easily. Only a few lines of code are required to access the elements of the array.

Properties of Arrays

- Each element of an array is of same data type and carries the same size, i.e., int = 4 bytes.
- Elements of the array are stored at contiguous memory locations where the first element is stored at the smallest memory location.
- Elements of the array can be randomly accessed since we can calculate the address of each element of the array with the given base address and the size of the data element.

Advantages of Arrays

- **Code Optimization**: Less code to the access the data.
- **Ease of traversing**: By using the for loop, we can retrieve the elements of an array easily.
- **Ease of sorting**: To sort the elements of the array, we need a few lines of code only.
- **Random Access**: We can access any element randomly using the array.

Declaration of C Array

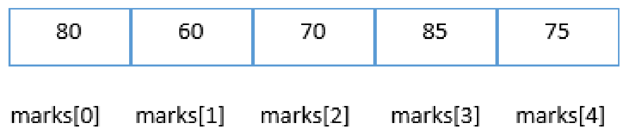
- data_type array_name[array_size]= { list of values }

- Example
  - **int** marks[5];
  - Here, int is the *data_type*, marks are the *array_name*, and 5 is the *array_size*.
- Index of an array starts from 0 to size-1. i.e if an array is of size 5 the first element will be stored in index [0] while the last element will be stored in index [4]

Initialization of C Array

- The simplest way to initialize an array is by using the index of each element.
- We can initialize each element of the array by using the index. Consider the following example.

  Int marks[5]= {80, 60, 70, 85, 75}



Initialization of Array

Printing specific values in an array

- To print specific values in an array you have to specify the index of the value that you wish to print.

For example

- printf("%d", marks[0]); // first value
- printf("%d", marks[1]);// second value
- printf("%d", marks[2]);// third value
- printf("%d", marks[3]);// fourth value
- printf("%d", marks[4]);// fifth value

Cont.

The following program prints the first value in the array called marks

```
#include<stdio.h>
void main()
{
    int marks[5] = {80, 60, 70, 85, 75};
        printf("%d", marks[0]);
}
```

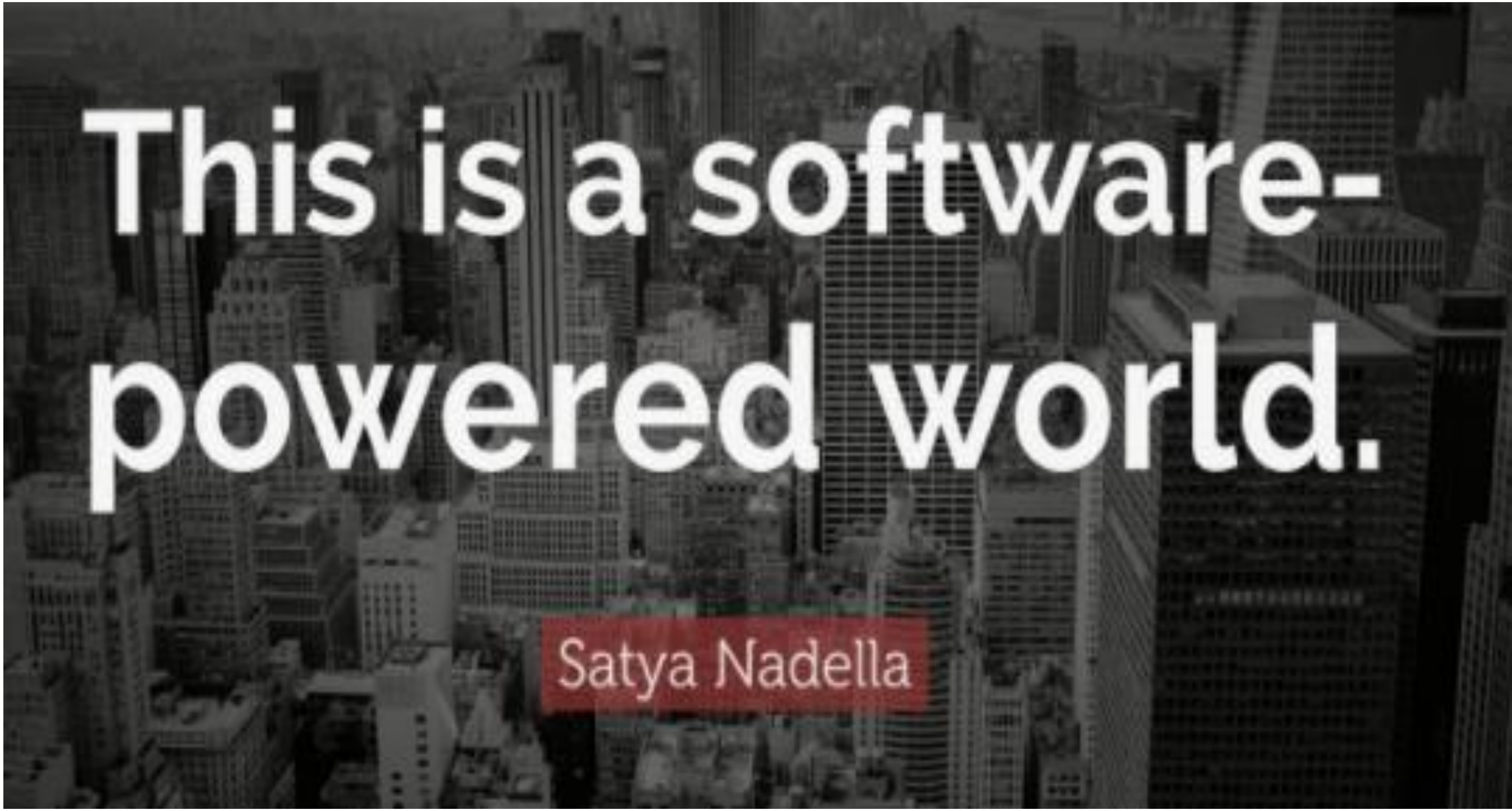
Example 1: Printing all values in an Array method 1

```
#include<stdio.h>
void main() {
int marks[5] = {80, 60, 70, 85, 75};
printf("%d\n", marks[0]); // first value
printf("%d\n", marks[1]);// second value
printf("%d\n", marks[2]);// third value
printf("%d\n", marks[3]);// fourth value
printf("%d\n", marks[4]);// fifth value
}
```

Example2 simplified way of printing all values in an array using a for loop

The following program prints all the values in the array called marks

```
#include<stdio.h>
void main()
{
   int i;
   int marks[5] = {80, 60, 70, 85, 75};
   for(i = 0 ; i < 5 ; i++)
   {
      printf("%d\t", marks[i]);
   }
}
```



This is a software-powered world.

Satya Nadella

THE END

“ I love the way programming changes human Imagination into useful products whose aim is to make life better and easier.” J. Kamiri