# Functions in

# C

**Instructor:** Maninder Kaur

**Email:** maninder.kaur2@sheridancollege.ca

**Course:** PROG20799

# C - Functions

- A function is a group of statements that together perform a task.
- Every C program has at least one function which is `main()`.
- Most of the programs can define additional functions.
- You can divide up your code into separate functions.
  - How you divide up your code among different functions is up to you, but logically the division usually is so each function performs a specific task.
- A function is known with various names like a method or a sub-routine or a procedure etc.

# C - Functions

- A function **declaration** tells the compiler about a function's:
  - Name
  - Return Type
  - Parameters

- A function **definition** provides the actual body of the function.

# Defining a Function

- The general form of a function definition in C programming language is as follows:

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

- A function definition in C programming language consists of a *function header* and a *function body*.

# Defining a Function

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

- Here are all the parts of a function:
- **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`.
- **Function Name:**
  - This is the actual name of the function.
  - The function name and the parameter list together constitute the *function signature*.
- **Parameters:**
  - 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.
- **Function Body:**
  - The function body contains a collection of statements that define what the function does.

# Example

- Following is the source code for a function called `sum()`.
- This function takes two parameters num1 and num2 and returns the sum of two.

```
int sum(int num1, int num2)
{
   int result;

   result = num1 + num2;

   return result;
}
```

# Calling a Function

- To use a function, you will have to call that function to perform the defined task.
- When a program calls a function, program control is transferred to the called function.
- A called function performs defined task and when its return statement is executed or when its function-ending closing brace is reached, it returns program control back to the main program.
- To call a function, you simply need to pass the required parameters along with function name.

# Ways to Define Functions

- Four ways to work with functions:
  - Without `return_type` and without parameters.
  - Without `return_type` and with parameters.
  - With `return_type` and without parameters.
  - With `return_type` and with parameters.

- It depends on the situation which out of these ways will be used in the programming.

# Without return_type and Without Parameters

```
#include <stdio.h>

void sum()
{
   int a, b, result;

   printf("Enter value of a and b: ");
   scanf("%d %d", &a, &b);

   result = a + b;

   printf("\nSum = %d", result);
}

main()
{
   sum();
}
```

# Without return_type and With Parameters

```
#include <stdio.h>

void sum(int x, int y)
{
   int result;

   result = x + y;

   printf("\nSum = %d", result);
}

main()
{
   int a, b;

   printf("Enter value of a and b: ");
   scanf("%d %d", &a, &b);

   sum(a, b);
}
```

# With return_type and Without Parameters

```
#include <stdio.h>

int sum()
{
   int a, b, result;

   printf("Enter two numbers: ");
   scanf("%d %d", &a, &b);

   result = a + b;

   return result;
}

main()
{
   int result;

   result = sum();

   printf("\nSum = %d", result);
}
```

# With return_type and With Parameters

```
#include <stdio.h>

int sum(int x, int y)
{
   int result;

   result = x + y;

   return result;
}

main()
{
   int a, b, result;

   printf("Enter value of a and b: ");
   scanf("%d %d", &a, &b);

   result = sum(a, b);

   printf("\nSum = %d", result);
}
```

# Function Declarations

- A function declaration tells the compiler about a function name and how to call the function.
- The actual body of the function can be defined separately.
- A function declaration has the following parts:

```
return_type function_name( parameter list );
```

- For the above defined function `sum()`, following is the function declaration:

```
int sum(int num1, int num2);
```

- Parameter names are not important in function declaration only their type is required, so following is also valid declaration:

```
int sum(int, int);
```

- Function declaration is required when you define a function in one source file and you call that function in another file. In such case you should declare the function at the top of the file calling the function.

# With return_type and With Parameters

```
#include <stdio.h>

int sum(int x, int y); /* Function declaration */

main()
{
   int a, b, result;

   printf("Enter value of a and b: ");
   scanf("%d %d", &a, &b);

   result = sum(a, b);

   printf("\nSum = %d", result);
}

int sum(int x, int y)
{
   int result;

   result = x + y;

   return result;
}
```

# Practice Problems

- Let’s try some programs in C language, using functions:
  - Program that finds the area of a rectangle using function and prints its.
  - Program to Convert Temperature from Degree Centigrade to Fahrenheit using Function.
  - Program to find the largest of three using functions.

# Any questions please?