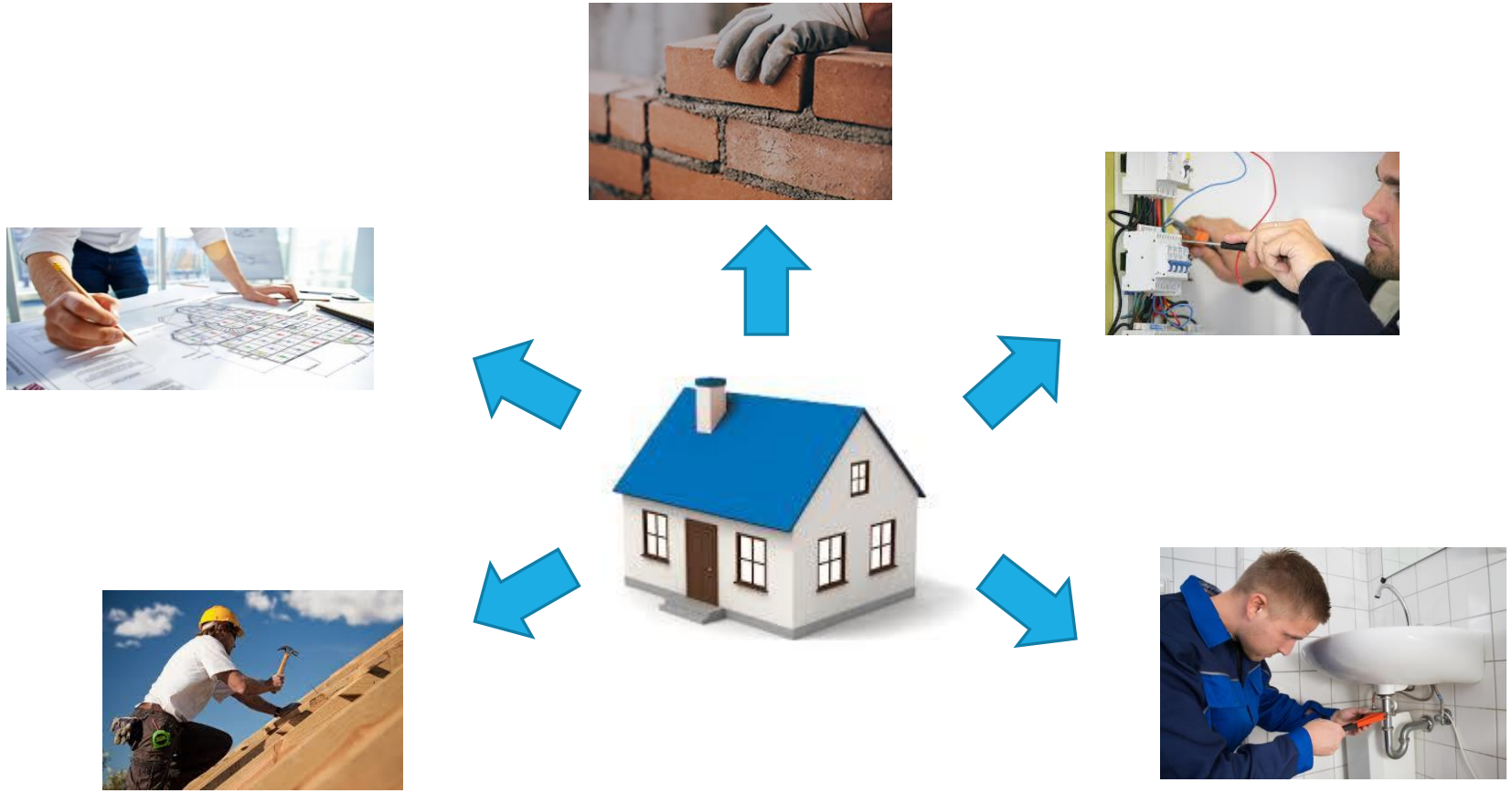
Fundamentals of Programming

Lecture 7

Chamila Karunatilake
Department of Information and Communication Technology
Faculty of Technology
University of Sri Jayewardenepura
chamilakarunatilake@sjp.ac.lk

C Functions



C Functions

- Functions break large computing tasks into smaller ones.
- Generally, function is a group of statements that together perform a specialized task.
- Every C program has at least one function, which is known as **main()** function.
- Most of the programs define additional functions.
- Functions can be reused whenever the same task must be done (instead of writing the whole program repeatedly from scratch).
- Also, functions hide details of operation from parts of the program that don't need to know about them.

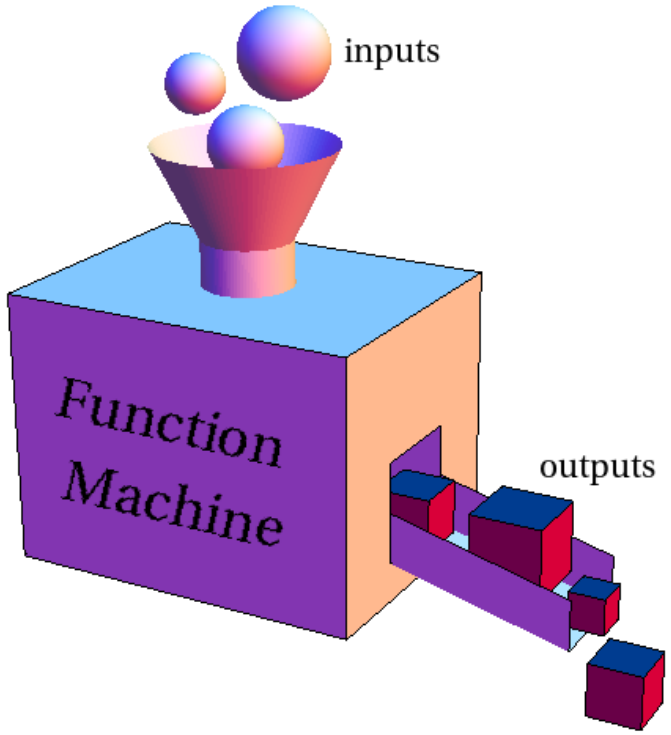
C Functions

When functions are used, several steps have to be followed.

- Function Definition
- Function Declaration
- Function Calling

- **Function Definition** creates the function (the actual body of the function)
- **Function Declaration** says the compiler that there is a function which has been created and ready to be used
- **Function Calling** is where the function's task is used. Without calling there is no use of that created function

C Function Definition



C Function Definition

```
Return_type Function_Name( Parameter list )
{
        body of the function
}
```

```
int addition(int num1, int num2)
{
        int result = num1 + num2; return
result;
}
```

```
int max(int num1, int num2)
{
        int result;
        if (num1 > num2)
                result = num1;
        else
                result = num2;
        return result;
}
```

C Function Definition

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

Return Type

- A function **may** return a value.
- The data type of that return value should be declared in the function header.
- It could be any data type or **void** in a case of no return value.

Function Name

- This is the actual name of the function.

C Function Definition

Parameters

- Parameter is a variable to hold the input value to the function.
- Data type of the parameter and name is given.
- Parameters are optional(a function may contain no parameters).
- The **function name** and the **parameter list** together constitute the **function signature**.

Function Signature

```
int addition(int num1, int num2)
```

C Function Definition

Function Body

- The function body contains a collection of statements that define what the function does.

Return Statement

- At the end of the body, there is a return statement as the return of the function.
- If return type is void, return statement is not necessary.
- Function can have only one return value.

C Function Definition

```
float calculateBonus(float salary)
{
        return salary  * (20/100);
}
```

```
void printValue(char name[])
{
        printf("%s\n", name);
}
```

```
char findResult(int marks)
{
        char result = 'F ';
        if(marks > 50)
        {
                result = 'F ';
        }
        return result;
}
```

```
void introduction()
{
        printf("Hi\n");
        printf("My name is Saman\n");
        printf("How are you?");
}
```

C Function Declaration

- A function **declaration** tells the compiler about a function which is defined and is going to be used.
- A function **prototype** should be declared at the top (before main()).
- Generally declaration consist of function **return type**, function **name** and **parameter list** followed by a **semicolon**.

```
Return_type Function_Name( Parameter list ) ;
```

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

- Parameter list may have both parameter types and names or just types.

```
int addition(int, int) ;
```

```c
#include <stdio.h>

int max(int num1, int num2);

int main ()
{
     .........
}

int max(int num1, int num2)
{
     int result;
     if (num1 > num2)
          result = num1;
     else
          result = num2;
     return result;
}
```

C Function Calling

- At this point, the function is defined and declared. However, it is not executed until it is **called** by main function(or sometimes other functions).
- Function calling is the point where the defined function is really being used.
- Function is called by its name. If there are values to be passed to the parameters of the function, they are also given in between two parentheses.

```
addition(234, 512) ;
```

- Those real values to be passed into the parameters are called **arguments.**

```
addition(234, 512) ;
```

Arguments

C Function Calling

```c
#include <stdio.h>

int max(int num1, int num2);

int main ()
{
      int a = 100;
      int b = 200;
      int ret;
      ret = max(a, b);
      printf( "Max value is : %d\n", ret );
      return 0;
}
```

```c
int max(int num1, int num2)
{
      int result;
      if (num1 > num2)
            result = num1;
      else
            result = num2;
      return result;
}
```

C Function Calling

```
int main ()
{
        .........
        ret = max(a, b);
        .........
}
```

```
int max(int num1, int num2)
{
        .........
}
```

C Function Execution

- The execution of a C program begins from the main() function.
- When the compiler encountered function call inside the main function, the control of the program jumps to the function.
- Then, the compiler starts executing the codes inside the function.
- At the end of the function execution, the control jumps back to the main() function, into the line where the function is called.
- If there is a return value from the function, it is also brought to the main function.
- The control of the program jumps to the next statement of the main function (executes until the end of the main function).

C Function Execution

```c
#include <stdio.h>

int max(int num1, int num2);

int main ()
{
    int a = 100;
    int b = 200;
    int ret;
    ret = max(a, b);
    printf( "Max value is : %d\n", ret );
    return 0;
}
```

```c
int max(int num1, int num2)
{
    int result;
    if (num1 > num2)
        result = num1;
    else
        result = num2;
    return result;
}
```

C Function Execution

```
int main ()
{
        .........
        ret = max(a, b);
        .........
}
```

```
int max(int num1, int num2)
{
        .........
        return result;
}
```

Questions?