VTC Academy
Think Ahead, Beyond Boundaries

BASIC PROGRAMMING LANGUAGE

LESSON 6

Functions

CONTENT

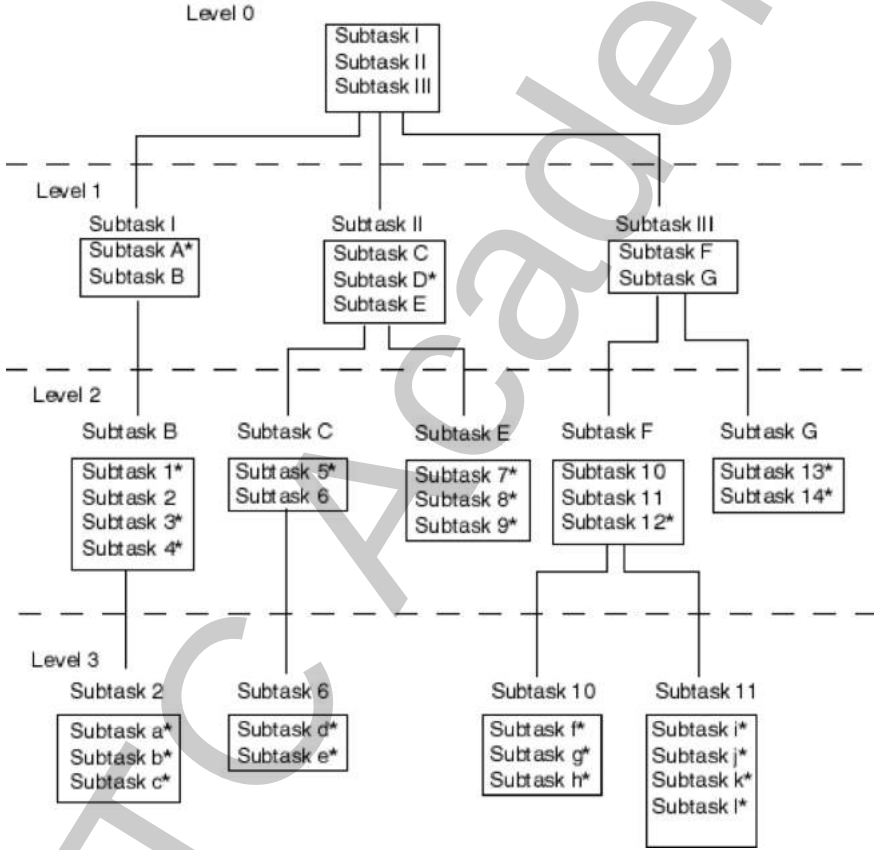
VTC Academy
Think Ahead, Beyond Boundaries

Top-Down Design

VTC Academy
Think Ahead, Beyond Boundaries

- A top-down approach is essentially the breaking down of a system to gain insight into the sub-systems that make it up.
- In a top-down approach an overview of the system is formulated, specifying but not detailing any first-level subsystems.
- Each subsystem is then refined in yet greater detail, sometimes in many additional subsystem levels, until the entire specification is reduced to base elements.
- Once these base elements are recognised then we can build these as computer modules.
- Once they are built we can put them together, making the entire system from these individual components.

Top-Down Design



Function Introduction

VTC Academy
Think Ahead, Beyond Boundaries

- A function is a self-contained program segment that carries out a specific, well-defined task.
- Functions are generally used as abbreviations for a series of instructions that are to be executed more than once.
- Functions are easy to write and understand.
- Debugging the program becomes easier as the structure of the program is more apparent, due to its modular form.
- Programs containing functions are also easier to maintain, because modifications, if required, are confined to certain functions within the program.

Types of Functions

VTC Academy
Think Ahead, Beyond Boundaries

- By using functions, programmer can reduce the time and cost of program development, reuse the code.
- Dividing a complex problem into smaller chunks makes our program easy to understand and reuse.
- In C, there 2 types of functions:
  - Standard library functions
  - User-defined functions

Standard Library Functions

VTC Academy
Think Ahead, Beyond Boundaries

- The standard library functions are built-in functions in C programming.
- These functions provide basic functions that help programmers easily and quickly focus on the business of the program.
- These library functions are divided into groups of similar functions, related to each other and declared in header files (.h).
- Example:
  - stdio.h: Store functions to perform standard I/O operations (scanf(), prinf(), …)
  - math.h: Store functions to perform math operations (sin(), cos(), sqrt(), …)

Standard Library Functions

VTC Academy
Think Ahead, Beyond Boundaries

C Header Files	Description
<assert.h>	Program assertion functions
<ctype.h>	Character type functions
<math.h>	Mathematics functions
<stdarg.h>	Variable arguments handling functions
<stdio.h>	Standard Input/Output functions
<stdlib.h>	Standard Utility functions
<string.h>	String handling functions
<time.h>	Date time functions

Standard Library Functions - Example

VTC Academy
Think Ahead, Beyond Boundaries

```c
#include <stdio.h>
#include <math.h>

int main()
{
    float num, root;
    printf("Enter a number: ");
    scanf("%f", &num);
    root = sqrt(num);
    printf("Square root of %.2f = %.2f", num, root);
    return 0;
}
```

Advantages of Standard Library Functions

VTC Academy
Think Ahead, Beyond Boundaries

- These functions have gone through multiple rigorous testing and are easy to use.
- The functions are optimized for performance.
- It saves considerable development time.
- The functions are portable.

User-Defined Functions

VTC Academy
Think Ahead, Beyond Boundaries

- C programming language allows coders to define functions to perform special tasks.
- So, the functions which are created by the user are known as user-defined functions.
- User-defined functions have contained the block of statements which are written by the user to perform a task.
- You can also create functions as per your need.

Defining Functions

VTC Academy Think Ahead, Beyond Boundaries

- The general syntax of a function in C is:

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

  - Return type – A function may return a value. The return type is the data type of the value the function returns.
  - Function name – This is the actual name of the function. The function name and the parameter list together constitute the function signature.
  - Parameters – A parameter is like a placeholder. When a function is invoked, you pass a value to the parameter.
  - Function body – The function body contains a collection of statements that define what the function does.

Functions Example

VTC Academy
Think Ahead, Beyond Boundaries

```
/* function returning the max between two numbers */
int max(int num1, int num2) {

   /* local variable declaration */
   int result;

   if (num1 > num2)
      result = num1;
   else
      result = num2;

   return result;
}
```

Function Declaration / Prototypes

- 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.
- Syntax:

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

  - For the above defined function max(), the function declaration is as follows

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

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

```
int max(int, int);
```

Function Declaration / Prototypes

VTC Academy
Think Ahead, Beyond Boundaries

- 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.
- Example:

```
#include <stdio.h>
/* function declaration */
int max(int num1, int num2);
int main () { ... }
```

Calling Function

VTC Academy
Think Ahead, Beyond Boundaries

- While creating a C function, you give a definition of what the function has to do. To use a function, you will have to call that function to perform the defined task.
- When a program calls a function, the program control is transferred to the called function.
- A called function performs a defined task and when its return statement is executed or when its function-ending closing brace is reached, it returns the program control back to the main program.
- To call a function, you simply need to pass the required parameters along with the function name, and if the function returns a value, then you can store the returned value.

Calling Function - Example

VTC Academy
Think Ahead, Beyond Boundaries

```c
#include <stdio.h>

/* function declaration */
int max(int num1, int num2);

int main () {

   /* local variable definition */
   int a = 100;
   int b = 200;
   int ret;

   /* calling a function to get max value */
   ret = max(a, b);

   printf( "Max value is : %d\n", ret );

   return 0;
}
```

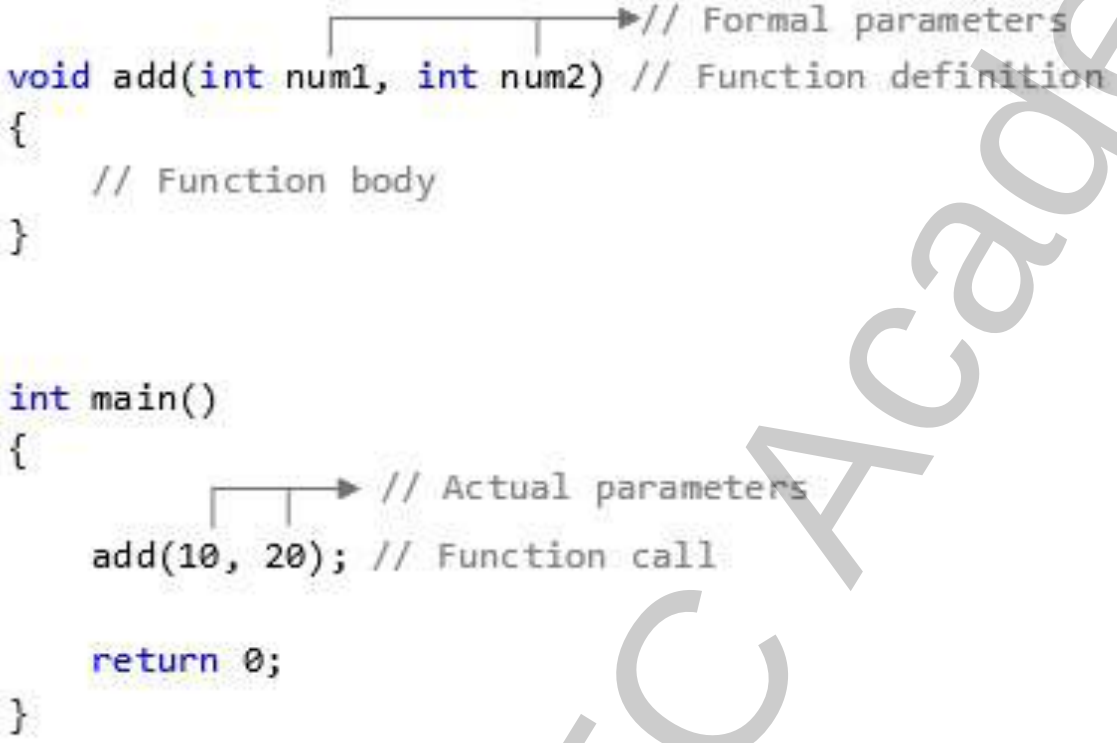
```c
/* function returning the max between two numbers */
int max(int num1, int num2) {

   /* local variable declaration */
   int result;

   if (num1 > num2)
      result = num1;
   else
      result = num2;

   return result;
}
```

Arguments and Parameters

VTC Academy
Think Ahead, Beyond Boundaries

- **Arguments** are referred to the values that are passed within a function when the function is called. These values are assigned to the variables in the definition of the function that is called.
- The type of the values passed in the function is the same as that of the variables defined in the function definition.
- **Parameters** are referred to as the variables that are defined during a function declaration or definition. These variables are used to receive the arguments that are passed during a function call.
- These parameters within the function prototype are used during the execution of the function for which it is defined.

Arguments and Parameters



Arguments and Parameters - Example

VTC Academy
Think Ahead, Beyond Boundaries

```c
#include <stdio.h>
// function definition, a & b are parameters
int sum(int a, int b)
{
    // returning the addition
    return a + b;
}
int main()
{
    int num1 = 10, num2 = 20, re;
    // sum() is called with num1 & num2 as arguments
    re = sum(num1, num2);
    // display the result
    printf("Sum: %d", re);
    return 0;
}
```

Arguments and Parameters

VTC Academy
Think Ahead, Beyond Boundaries

- While calling a function, there are two ways in which arguments can be passed to a function:
  - **Call by value:** This method copies the actual value of an argument into the formal parameter of the function. In this case, changes made to the parameter inside the function have no effect on the argument
  - **Call by reference:** This method copies the address of an argument into the formal parameter. Inside the function, the address is used to access the actual argument used in the call. This means that changes made to the parameter affect the argument.

Return Statement

- The return statement terminates the execution of a function and returns a value to the calling function.
- The program control is transferred to the calling function after the return statement.
- Syntax

```
return (expression);
```

- Example:

```
return a;
return (a+b);
```

Return Statement

VTC Academy Think Ahead, Beyond Boundaries

- The type of value returned from the function and the return type specified in the function prototype and function definition must match.
- Example:

```
int sum(int a, int b) {
    return (a + b); // same value type with return value (int)
}
```

- If you do not want function to return any value, you can set type of return type is void.

Variables Scope

- Local Variables
  - Declared inside a function
  - Created upon entry into a block and destroyed upon exit from the block
- Formal Parameters
  - Declared in the definition of function as parameters
  - Act like any local variable inside a function
- Global Variables
  - Declared outside all functions
  - Holds value throughout the execution of the program

Storage Classes

- Every C variable has a characteristic called as a storage class.
- The storage class defines two characteristics of the variable:
  - Lifetime – The lifetime of a variable is the length of time it retains a particular value
  - Visibility – The visibility of a variable defines the parts of a program that will be able to recognize the variable
- Storage class keyword:
  - automatic
  - external
  - static
  - register

Function Scope Rules

VTC Academy
Think Ahead, Beyond Boundaries

- Scope Rules - Rules that govern whether one piece of code knows about or has access to another piece of code or data
- The code within a function is private or local to that function
- Two functions have different scopes
- Two Functions are at the same scope level
- One function cannot be defined within another function

Functions in Multi-File Program

VTC Academy
Think Ahead, Beyond Boundaries

- Functions can also be defined as static or external.
- Static functions are recognized only within the program file and their scope does not extend outside the program file.

```
static fn _type fn_name(argument list);
```

- External function are recognized through all the files of the program.

```
extern fn_type fn_name(argument list);
```

Functions in Multi-File Program

VTC Academy
Think Ahead, Beyond Boundaries

```c
/* main.c Program File */

#include <stdio.h>
#include "max.c"

int main () {

   /* local variable definition */
   int a = 100;
   int b = 200;
   int ret;

   /* calling a function to get max value */
   ret = max(a, b);

   printf( "Max value is : %d\n", ret );

   return 0;
}
```

```c
/* max.c file */

/* function declaration */
static int max(int num1, int num2);

/* function returning the max between two numbers */
int max(int num1, int num2) {

   /* local variable declaration */
   int result;

   if (num1 > num2)
      result = num1;
   else
      result = num2;

   return result;
}
```

Summary

- A function is a self-contained program segment that carries out a specific, well-defined task.
- Functions are generally used as abbreviations for a series of instructions that are to be executed more than once.
- Function declaration is required when you define a function in one source file and you call that function in another file.
- Every C variable has a characteristic called as a storage class: automatic, external, static, register.

VTC Academy
Think Ahead, Beyond Boundaries