Functions

ESC108A Elements of Computer Science and Engineering

B. Tech. 2017

Course Leaders:

Roopa G.

Ami Rai E.

Chaitra S.

Objectives

- At the end of this lecture, student will be able to
  - Use modular programming using functions and multiple files
  - Identify flow chart elements and connectors that are associated with functions
  - Identify the constructs in algorithms that are associated with functions
  - Apply functions to solve a problem
  - Express functions in C programming language

Contents

- Functions
- Standard library functions
- Header files

Question

- Simple projects in C programming language have thousands of lines of code. How do programmers manage writing and maintaining such code?

Solution

- Divide and conquer
  - Break large computing tasks into smaller ones
  - Construct a program from smaller pieces of code
  - Each piece can be maintained independently

- Enable people to build on what others have done instead of starting from scratch

Modules in C

- Functions
  - Modules in C
  - A set of instructions to carry out a particular task

- Advantages
  - Reduce program development time
  - Reusability
  - Manageability - Debugging is easier
  - Avoid repeating code in a program

Types of C functions

1. Library function/Built-in functions
   - C standard library has many functions
   - E.g., printf(), scanf(),sqrt(), etc.,

2. User defined function/ Programmer defined function
   - Functions written by the user

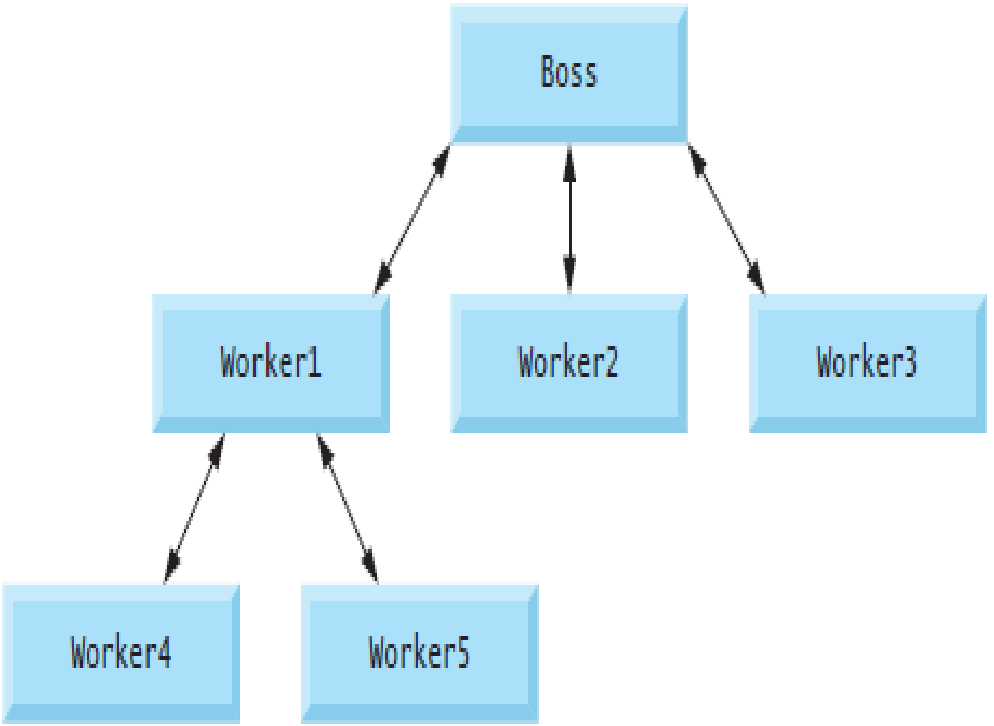
Programming in C

- When programming in C, you will be using
  - C standard library functions
  - Functions you create yourself
  - Functions other people have created and made available to you

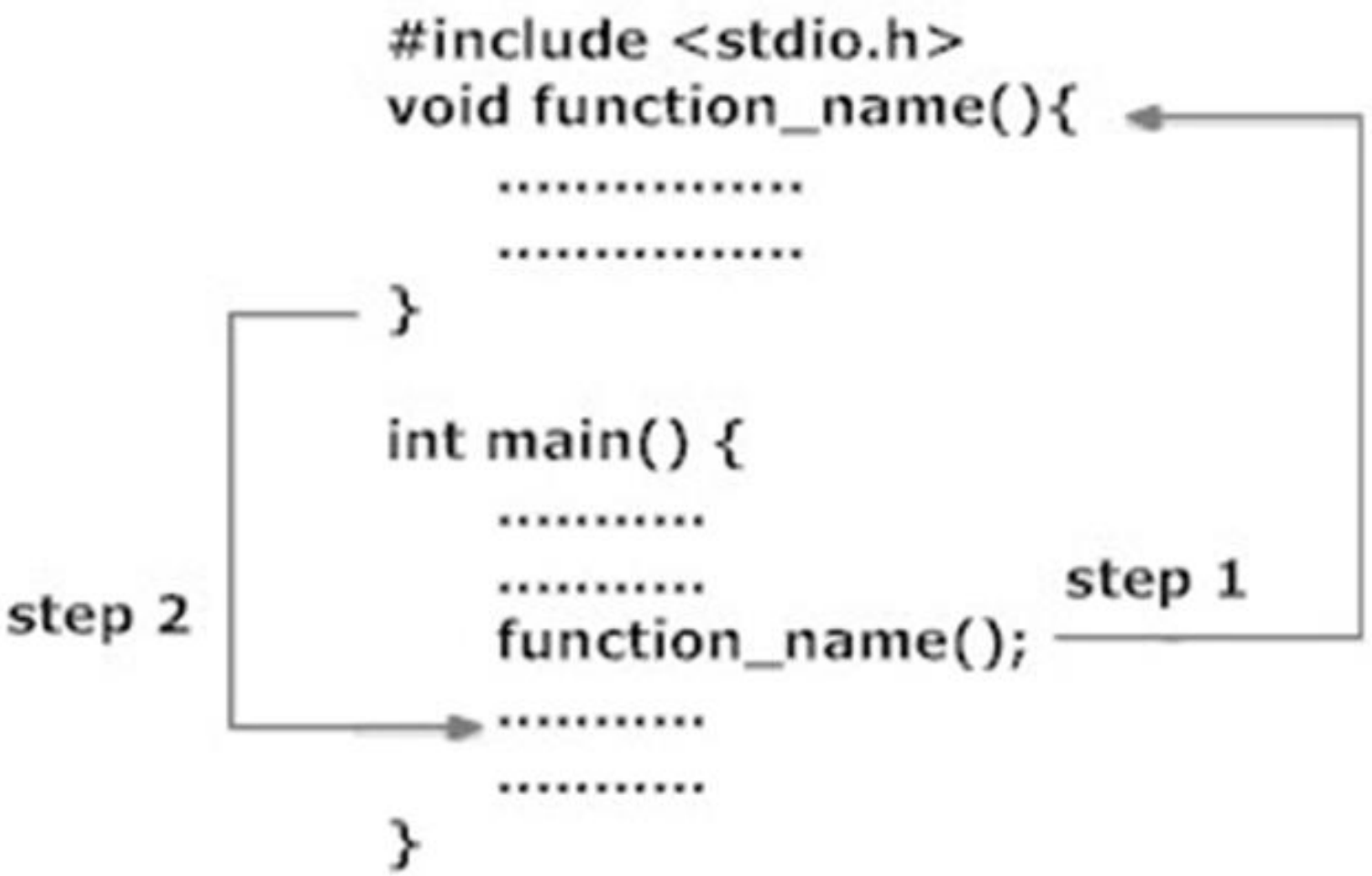
Functions

- Function declaration
- Function call
- Function definition

- Consider the hierarchical form of management



User Defined Function



Function Prototype

- Helps compiler know that such a function exists
- Function prototype tells the compiler
  - the type of data returned by the function
  - the number of parameters the function expects to receive
  - the types of the parameters
  - the order in which these parameters are expected

- Example

```
int product(int); //Takes 1 integer and returns an integer
int product(int a) ; //compiler ignores the names
```

Function Definition

- Writing an actual code for the function which performs a specific task
- A function cannot be defined inside another function
- Function definition format

```
<return value type>  function-name(<parameter list> ){
        statements
    }
```

- Example

```
int product(int n){
        int result;
        result=n *10;
        return result;
}
```

Function Definition contd.

- Return-value-type: data type of the result
  - *void* - function returns nothing
  - Default - *int*

- Returning control
  - If nothing returned
    - return;
    - or, until end of function block
  - If something returned
    - return *expression*;

Function Calls

- Invoking functions
- Function call includes
  - function name
  - arguments (data)
  - Function returns results
- Example: value=product(x);
- The compiler uses function prototype to validate function calls
  - A function call that does not match the function prototype is a syntax error
- Need not worry about how operations are performed
  - Procedural abstraction

Sample Program

```
int sum(int,int);  //function declaration
int main(int argc, char** argv) {
    int a,b,c;
    printf("Enter the numbers:");
    scanf("%d%d",&a,&b);
    c=sum(a,b);  //function call, a and b are arguments
    printf("\nSum is %d",c);
    return (EXIT_SUCCESS);
}
int sum(int x,int y) {  //function definition, x and y are parameters
    int z=x+y;
    return z;
}
```

Category of Functions - Demo

- Functions can be categorized into
  - No arguments, no return value
  - Arguments , no return value
  - No arguments, return value
  - Arguments, return value

Headers

- Each standard library has a corresponding header
- It contains
  - the function prototypes for all the functions in that library
  - definitions of various data types and constants needed by those functions

- You can create custom headers
  - Programmer-defined headers should also use the .h filename extension
  - A programmer-defined header can be included by using the #include preprocessor directive

Some Standard Library Headers

Header	Explanation
<assert.h>	Contains macros and information for adding diagnostics that aid program debugging
<float.h>	Contains the floating-point size limits of the system
<limits.h>	Contains the integral size limits of the system
<math.h>	Contains function prototypes for math library functions
<stdio.h>	Contains function prototypes for the standard input/output library functions, and information used by them
<stdlib.h>	Contains function prototypes for conversions of numbers to text and text to numbers, memory allocation, random numbers, and other utility functions
<string.h>	Contains function prototypes for string-processing functions
<time.h>	Contains function prototypes and types for manipulating the time and date

Math Library functions

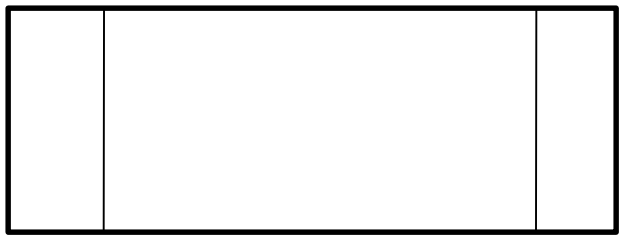
- Allow you to perform certain common mathematical calculations
- Include the math header by using the preprocessor directive #include <math.h>
- Example, a programmer desiring to calculate and print the square root of 900.0 might write

  printf("%.2f", **sqrt**(900.0));

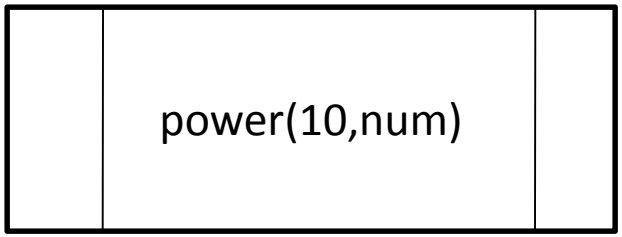
  – When this statement executes, the math library function sqrt is called
  – The number 900.0 is the argument of the sqrt function
  – The preceding statement would print 30.00

Flow Charts and Algorithms

- Predefined Process



- Examples



- Use function call
- For each function, define separate algorithm
- Examples
  - add(a,b);

```
Algorithm add
    (a,b:Integer):Integer
var c:Integer; {The result}
Begin
      c := a+b;
End
```

Summary

- Functions are used to increase reusability and maintainability of C code
- Functions focus on solving one aspect
- Functions take arguments and copy values to local parameters
- All variables declared within a function are local to that function
- Functions can be declared if use of function appears before the function definition

Further Reading

Kernighan, B. W. and Richie, D. (1992) *The C Programming Language.* 2nd ed., New Delhi:PHI.