CSC 128

TOPIC 5: FUNCTIONS

By : MOHD SAIFULNIZAM ABU BAKAR

Learning Outcomes

At the end of this chapter, you should be able to:

- Understand the two types of functions.
- Identify the three function elements.
- Develop a program using functions.

Introduction

- Programs can be as short as one line of code or millions of lines of code.
- Small programs can easily be developed using only one function, which is the main function.
- However, most computer programs that solve real-world problems are large and contain thousands to millions of lines of code.
- Thus, more than one programmer or a team of programmers are needed to develop such programs.

Introduction (cont)

- In order to develop and maintain large programs, an effective way to construct it is by using functions.
- Functions can efficiently break down large programs into smaller ones as they can divide problems into sub problems called modules.
- Every function is a program segment that can do specific tasks.
  - Each of the sub tasks can be coded as functions that must go together with the main function to build up a complete program.

Introduction (cont)

- Below are the advantages of using functions:
  1. Functions break the program into smaller chunks, making it easier for the programmer to code, test and debug the program.
  2. Each of the functions is reusable, where the function can be called many times.
  3. By using functions, repeating code in a program can be avoided.
- The explanations above refer to user-defined functions.
- There is another function known as the predefined function. available in the C++ standard library

Function Types

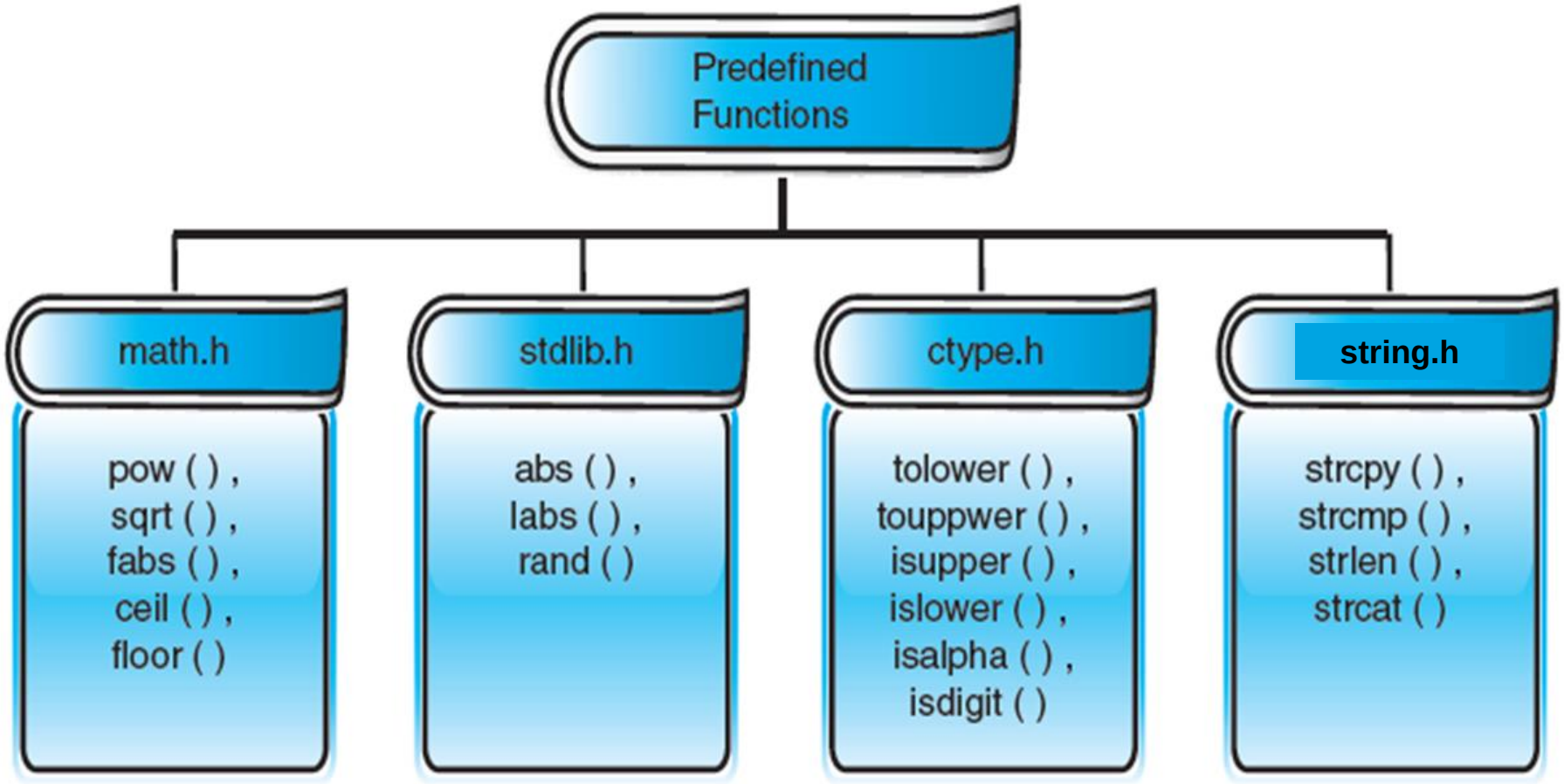
Pre-Defined Function

- C++ Standard Library contains many predefined functions to perform various operations
- These functions are available in the C++ standard library such as math.h and others
- Ready made

User-Defined Function

- Functions which are defined by the user.
- The functions that programmers create for specialized tasks.
- Custom made

Predefined Functions



Predefined Functions (cont.)

- `stdlib.h` header

Function	Description	Example	Value
abs(x)	absolute value for x, x is an integer	abs (–15) abs (15)	15 15
labs(x)	absolute value for x, x is a long	labs (–25000) labs (25000)	25000 25000
rand( )	Any random number, integer type	rand()	any number

Predefined Functions (cont.)

stdlib.h

EXAMPLE 5.1

```
//functions using abs() and labs()

//header files
#include<iostream>
#include<stdlib.h>
using namespace std;

//main function
int main()
{
      long a;
      a=labs (-88000);

      cout<<"LABS: "<<a;
      cout<<"\nABS: "<<abs(-42);

      return 0;
}
```

OUTPUT:

```
LABS: 88000
ABS: 42
```

EXAMPLE 5.2

```
//function using rand()
//display 5 random numbers
//header files
#include<iostream>
#include<stdlib.h>
using namespace std;

//main function
int main()
{
      for(int i=1;i<=5;i++)
      {
            cout<<"\nRANDOM #"<< i<<" is "<<rand();
      }
      return 0;
}
```

OUTPUT:

```
RANDOM  #1  is  41
RANDOM  #2  is  18467
RANDOM  #3  is  6334
RANDOM  #4  is  26500
RANDOM  #5  is  19169
```

Predefined Functions (cont.)

EXAMPLE 5.3

stdlib.h

```
 function using rand()
//calculate the sum of five random numbers

//header files
#include<iostream>
#include<stdlib.h>
using namespace std;

//main function
int main()
{
      int total=0, i;
      for(i=1; i<=5;i++)
            total=total+rand();

      cout<<"The total sum of "<<i-1 <<" numbers is "<<total;
      return 0;
}
```

OUTPUT:
The total sum of 5 numbers is 70511

Predefined Functions (cont.)

- `math.h` header

Function	Description	Type	Example	Value
sqrt(x)	Square root $\sqrt{x}$	double	sqrt(49.0)	7.0
pow(x, y)	power x^y	double	pow(3.0,2.0)	9.0
fabs(x)	absolute value for double	double	fabs(–3.5) fabs(3.5)	3.5 3.5
ceil(x)	ceiling (round up)	double	ceil **(6.3)** ceil **(6.7)**	7.0 7.0
floor(x)	floor (round down)	double	floor(6.2) floor(6.8)	6.0 6.0

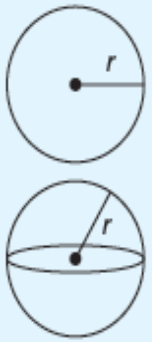
`math.h`

Predefined Functions (cont.)

EXAMPLE 5.4

Problem: Find the area of a circle and volume of a sphere given the formula A = πr² and V = 4/3 πr³. Prompt the user to enter the radius

```
//program using math.h header
//functions using pow() and floor()

//header files
#include<iostream>
#include<math.h>
using namespace std;

//main function
int main()
{

    double areaC, volumeS, radius;
    const double PI=3.142;

    cout<<"Enter radius:";
    cin>>radius;
    areaC=PI*pow(radius,2);
    volumeS=4/3.0*PI*pow(radius,3);

    cout<<"\nThe area of a circle is:"<<areaC;
    cout<<"\nThe volume of a sphere is:"<<volumeS;

    //to round down the answer
    cout<<"\n\nROUNDED ANSWER";
    cout<<"\nThe area of a circle is: "<<floor(areaC);
    cout<<"\nThe volume of a sphere is:"<<floor(volumeS);

    return 0;
}
```



OUTPUT:

```
Enter radius:3

The area of a circle is:28.278
The volume of a sphere is:113.112

ROUNDED ANSWER
The area of a circle is:28
The volume of a sphere is:113
```

`math.h`

Predefined Functions (cont.)

EXAMPLE 5.5

Problem: Find the distance between 2 points, (−4, 2) and (5, 3), given distance = $\sqrt{(x2 - x1)2 + (y2 - y1)^2}$

```
//program using math.h header
//functions using sqrt(), pow() and ceil()

//header files
#include<iostream>
#include<math.h>
#include<iomanip>
using namespace std;

//main function
int main()
{
    double x1=-4, x2=5, y1=2, y2=3;
    double distance;

    distance=sqrt (pow (x2-x1,2)+pow (y2-y1,2));

    cout.setf (ios: :fixed);
    cout.precision (2);
    cout<<"The distance between 2 points is "<<distance;

    //to round up the answer
    cout<<"The distance between 2 points is "
        <<ceil(distance);

    return 0;
}
```

OUTPUT:

```
The distance between 2 points is 9.06
The distance between 2 points is 10.00
```

Predefined Functions (cont.)

- `ctype.h` header

Function	Description
tolower (x)	returns the lowercase value of x
toupper (x)	returns the uppercase value of x
isupper (x)	determines if a character is uppercase
islower (x)	determines if a character is lowercase
isalpha (x)	determines if a character is a letter (a–z, A–Z)
isdigit (x)	determines if a character is a digit (0–9)

`ctype.h`

Predefined Functions (cont.)

EXAMPLE 5.6

Problem: Find the uppercase and lowercase values of a letter entered by the user.

```
//functions using tolower() and toupper()

//header files
#include<iostream>
#include<ctype.h>
using namespace std;
```

(Continued)

```
//main function
int main()
{
      char letter1, letter2, x, y;

      //lowercase to uppercase
      cout<<"Enter any character in lowercase:";
      cin>>letter1;
      x=toupper (letter1);
      cout<<"The character in uppercase is:"<<x;

      //uppercase to lowercase
      cout<<"\n\nEnter any character in uppercase:";
      cin>>letter2;
      y=tolower (letter2);
      cout<<"The character in lowercase is:"<<y;

      return 0;
}
```

OUTPUT:

```
Enter any character in lowercase: r
The character in uppercase is R

Enter any character in uppercase: H
The character in lowercase is h
```

`ctype.h`

Predefined Functions (cont.)

EXAMPLE 5.7

Problem: Determine whether the characters that the user enters are uppercase or lowercase

```
//functions using isupper() and islower()

//header files
#include<iostream>
#include<ctype.h>
using namespace std;

//main function
int main()
{
      char letter;
      int m, n;

      cout<<"Enter any character:";
      cin>>letter;

      m=isupper(letter);
      n=islower(letter);

      cout<<"\nUppercase:"<<m;
      cout<<"\nLowercase:"<<n;

      cout<<"\n\nIs the letter in uppercase or lowercase?";

      if (m>0||n==0)
            cout<<"\nThe letter is in uppercase";
      else if(m==0||n>0)
            cout<<"\nThe letter is in lowercase";
      return 0;
}
```

OUTPUT:

```
Enter any character: a

Uppercase:  0
Lowercase:  2

Is the letter in uppercase or lowercase?
The letter is in lowercase
```

`ctype.h`

Predefined Functions (cont.)

EXAMPLE 5.8

Problem: Determine whether characters that the user enters are letters or digits

```
//functions using isalpha() and isdigit()
//header files
#include<iostream>
#include<ctype.h>
using namespace std;

//main function
int main()
{
      char letter1,letter2;
      int a, b;
      cout<<"Enter first and second characters:";
      cin>>letter1>>letter2;

      a=isalpha(letter1);
      b=isdigit(letter2);
      cout<<"\nLetter 1:"<<a;
      cout<<"\nLetter 2:"<<b;

      cout<<"\n\nAre the characters letters or digits?";
      if (a>0)
      {
            cout<<"\nThe first character is a letter";
            if (a==0)
                  cout<<"\nThe first character is a digit";
      }
      if(b>0)
      {
            cout<<"\nThe second character is a digit";
            if(b==0)
                  cout<<"\nThe second character is a letter";
      }
      return 0;
}
```

(Continued)

```
OUTPUT:
Enter first and second characters:
M
5
Letter 1:  1
Letter 2:  1

Are the characters letters or digits?
The first character is a letter
The second character is a digit
```

Predefined Functions (cont.)

- `string.h` header

Function	Description
strcmp(string1, string2)	returns true if the two strings are different; otherwise returns 0
strcpy(string1, string2)	assigns the value of string2 into string1
strlen(string)	returns the length of the string
strcat(string1,string2)	combines both strings and assigns it to string1

string.h Predefined Functions (cont.)

EXAMPLE 5.9

Problem: Enter two names. Copy the second name and store it into the first name

```
//predefined function string.h
//using function strcpy()
#include<iostream>
#include<string.h>
using namespace std;

int main()
{
      char name1 [20] ;
      char name2 [20] ;

      cout<<"Enter first name: ";
      cin.getline(name1, 15);

      cout<<"\nEnter second name: ";
      cin.getline(name2, 15);

      //using strcpy()
      //copy name2 and store in name1
      strcpy(name1, name2);
      cout<<"\nNAME 1 is: "<<name1;
      cout<<"\nNAME 2 is: "<<name2;

      return 0;
}
```

```
OUTPUT:
Enter first name: SYAHIRAH
Enter second name: AISYAH
NAME 1 is: AISYAH
NAME 2 is: AISYAH
```

string.h Predefined Functions (cont.)

EXAMPLE 5.10

Problem: Compare between the names of two students to check whether they are similar

```
//predefined function string.h
//using function strcmp()

#include<iostream>
#include<string.h>
using namespace std;

int main()
{
      char name1 [20];
      char name2 [20];
      int compare;

      cout<<"Please enter first student's name: ";
      cin.getline(name1, 15);
```

(Continued)

```
      cout<<"\nPlease enter second student's name: ";
      cin.getline(name2, 15);

      //using strcmp()
      //compare name1=name2

      compare = strcmp(name1, name2);

      if(compare!=0)
            cout<<"\nName 1 and name 2 are different";
      else
            cout<<"\nName 1 and name 2 are the same";

      return 0;
}
```

OUTPUT:

```
Please enter first student's name:  MIRA
Please enter second student's name:  MIRA
Name 1 and name 2 are the same
```

string.h Predefined Functions (cont.)

EXAMPLE 5.11

Problem: Find the length of the names of two people

```
//predefined function string.h
//using function strlen()

//header files
#include<iostream>
#include<string.h>
using namespace std;

//main function
int main()
{
    char name1 [20];
    char name2 [20];
    int length1, length2;

    cout<<"Enter employee's name:";
    cin.getline(name1, 20);

    length1=strlen(name1);

    cout<<"\nThe length of the name is: "<<length1;

    cout<<"\n\nEnter another employee's name: ";
    cin.getline(name2, 20);

    length2=strlen(name2);

    cout<<"\nThe length of the name is: "<<length2;

    return 0;
}
```

OUTPUT:

```
Enter employee's name: Mia Azmi
The length of the name is: 8

Enter another employee's name: Haziq
The length of the name is: 5
```

`string.h` Predefined Functions (cont.)

EXAMPLE 5.12

Problem: Find the longest name between three people

```
//predefined function string.h
//functions using strlen()
//header files
#include<iostream>
#include<string.h>
using namespace std;

//main function
int main()
{
      char name1 [20];
      char name2 [20];
      char name3 [20];
            int l1, l2, l3;

      cout<<"Enter first employee's name: ";
      cin.getline(name1, 20);
      l1=strlen(name1);

      cout<<"\nEnter second employee's name: ";
      cin.getline(name2, 20);
      l2=strlen(name2);

      cout<<"\nEnter third employee's name: ";
      cin.getline(name3, 20);
      l3=strlen(name3);

      if (l1>l2  && l1>l3)
            cout<<"\nLongest name is: "<<name1;
      else if (l2>l1 && l2>l3)
            cout<<"\nLongest name is: "<<name2;
      else
            cout<<"\nLongest name is: "<<name3;

      return 0;
}
```

OUTPUT:

```
Enter first employee's name: IESHA
Enter second employee's name: IENA
Enter third employee's name: SOFIEYA
Longest name is: SOFIEYA
```

string.h Predefined Functions (cont.)

EXAMPLE 5.13

Problem: Combine the first and second name

```
//predefined function string.h
//using function strcat()

//header files
#include<iostream>
#include<string.h>
using namespace std;

//main function
int main()
{
      char name1 [20];
      char name2 [20];

      cout<<"Enter your daughter's first name: ";
      cin.getline(name1, 20);

      cout<<"\nEnter your daughter's second name: ";
      cin.getline(name2, 20);

      strcat(name1, name2);

      cout<<"\nThe combined name is: "<<name1;

      return 0;
}
```

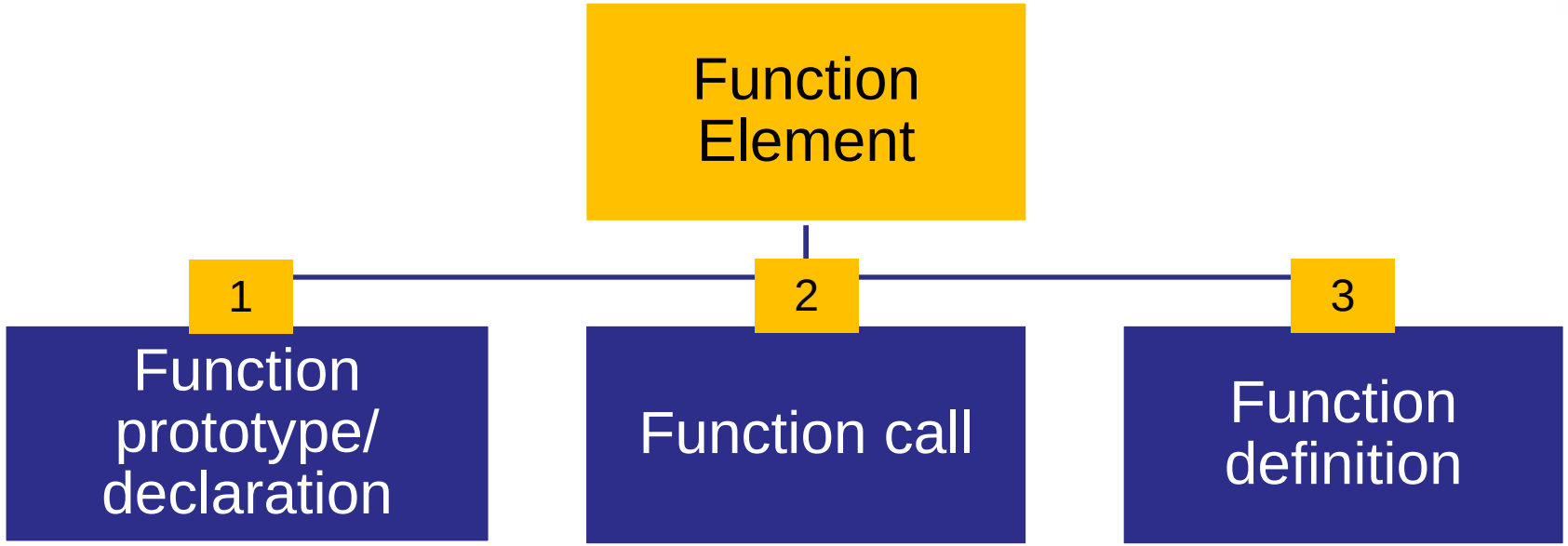
OUTPUT:

```
Enter your daughter's first name: AIN
Enter your daughter's second name: NADHIRAH
The combined name is: AINNADHIRAH
```

User-defined Functions

- All C++ programs must contain at least one function, which is the main( ).
- Functions must be assigned a name. Similar to variables, each function is given a valid identifier name.
- A function name must be followed by parentheses( ).
- The characteristics of user-defined functions are:
  1. A function name must be unique
  2. A function performs a specific task
  3. A function is independent
  4. A function may receive and return values to the calling function

User-defined Functions



EXAMPLE 5.14 General form of program without functions

```
#include.<iostream>

//using only main function
int main()
{
    statement;
    return 0;
}
```

User-defined Functions

No need to declare function if function definition on top of main function

EXAMPLE 5.15 General form of program with 3 function elements

```
#include<iostream>
using namespace std;
//function declaration/prototype
int x (int,double);
void y();

int main()
{
    //function call
    x (a, b);
    y ();
}

//function definition with 2 parameters
//function must return value
int x (int a,double b)
{
        statements;
}

//function definition without parameters
//function definition does not return value
void y ()
{
        statements;
}
```

1 Function prototype/ declaration

2 Function call

3 Function definition

3 Function definition

3 Function definition

EXAMPLE 5.15 General form of program with 3 function elements

```
#include<iostream>
using namespace std;

//function definition with 2 parameters
//function must return value
int x (int a,double b)
{
        statements;
}

//function definition without parameters
//function definition does not return value
void y ()
{
        statements;
}

int main()
{
    //function call
    x (a, b);
    y ();
}
```

3 Function definition

3 Function definition

2 Function call

User-defined Functions

EXAMPLE 5.16 General form of program with only function call and function definition

```
#include<iostream.h>
//function definition without parameters
//function must return value
int a()
{
        statements;
}
//function definition with 1 parameter
//function definition does not return value
void b (char k)
{
        statements;
}
```

3

Function definition

3

Function definition

1 Function Declaration/Prototype

- To make programs more readable, C++ programmers usually insert functions after the main function.
- If the function is placed after the main function, the function declaration, also known as function prototype, is required and should be placed on top of the program above the main function.
- Similar to declaring variables, the purpose of prototyping or declaring functions is to notify the compiler on the existence of the function, thus memory will be allocated to store the function.

1 Function Declaration/Prototype

- If a function definition is placed below the function main and is not prototyped, an error will occur.
- A function prototype can be written according to the following form:

```
function_type function_name (parameter list);
```

  - `function_type` is any data type such as `int`, `double`, `float`, `char` or `void`
  - `function_name` is any identifier name as the name must follow the rules in naming identifiers.
  - `parameter list` is any data type which belongs to variables or constants that is passed to the function.
- Similar to statements, a function declaration/prototype must end with a semicolon(;).

3 Function Definition

- A function is a sub program which can perform specific tasks.
- Function definition will not be used if it is not called by the function call.
- To write a function definition, below is the correct form:

```
function_type function_name (formal parameter list)
{
    function_body
}
```

  - `function type` is any data type such as `int`, `double`, `float`, `char` or `void`.
  - `function_name` is any identifier name as the name must follow the rules in naming identifiers
  - `parameter list` is any data type and identifier name that is being passed to the function

3 Function Definition

- A parameter list in a function definition is known as formal parameter list.
- If there is more than one parameter list, separate them with commas (,).
- function_body is enclosed in braces and composed of executable statements such as input, process and output.
- A function definition which starts with the function type int, double, float and char must return its value to the function call, whereas if it starts with function type void, the function does not have to return a value

3 Function Definition

EXAMPLE 5.18 Writing function definition

```
//function display()
//function without parameter and does not return its value
//displays the area of a circle
void display()
{
        cout<<"Area of a circle is:"<<area<<endl;
}
```

3

Function definition

EXAMPLE 5.19 Writing function definition

```
//function volume()
//function with parameter that receives radius of a sphere
//calculate and display the volume of a sphere
void volume (int radius)
{
    double volume;
    volume=4/3.0*PI*pow(radius,3)
    cout<<"The volume of a sphere is:"<<volume<<endl;
}
```

3

Function definition

NOTE :

if no return statement, put void in front of your function name

3 Function Definition

EXAMPLE 5.20 Writing function definition

```
//function cal_price()
/*function with parameter that receives the price and quantity of an item*/
//calculate the total price and return its value

double cal_price(double price, int quantity)
{
      double total;

      total=price*quantity;

      return total;
}
```

3

Function definition with Return Statement

2 Function Call

- All function definitions must be called so that they can be useful.
- A function definition which is not called does not have the ability to perform its task.
- A function definition in a program can be called by any other functions including the main function.
- When calling a function, follow the form below:

```
function_name (actual parameter list);
```

  - function_name is any identifier name as the name must follow the rules in naming identifiers.
  - parameter list is any identifier name or value that is being passed to the function.
- A parameter list in a function call is known as an actual parameter list. If there is more than one parameter list, separate them with commas (,).

2 Function Call

EXAMPLE 5.21 Writing function call

```
//function call display()
//only suitable for function that does not return its value
display();

//function call volume()
//appropriate only for function that returns its value
r=volume(radius);

//function call cal_price()
//applicable only for function that returns its value
cout<<cal_price(price, quantity);
```

2 Function Call

EXAMPLE 5.22

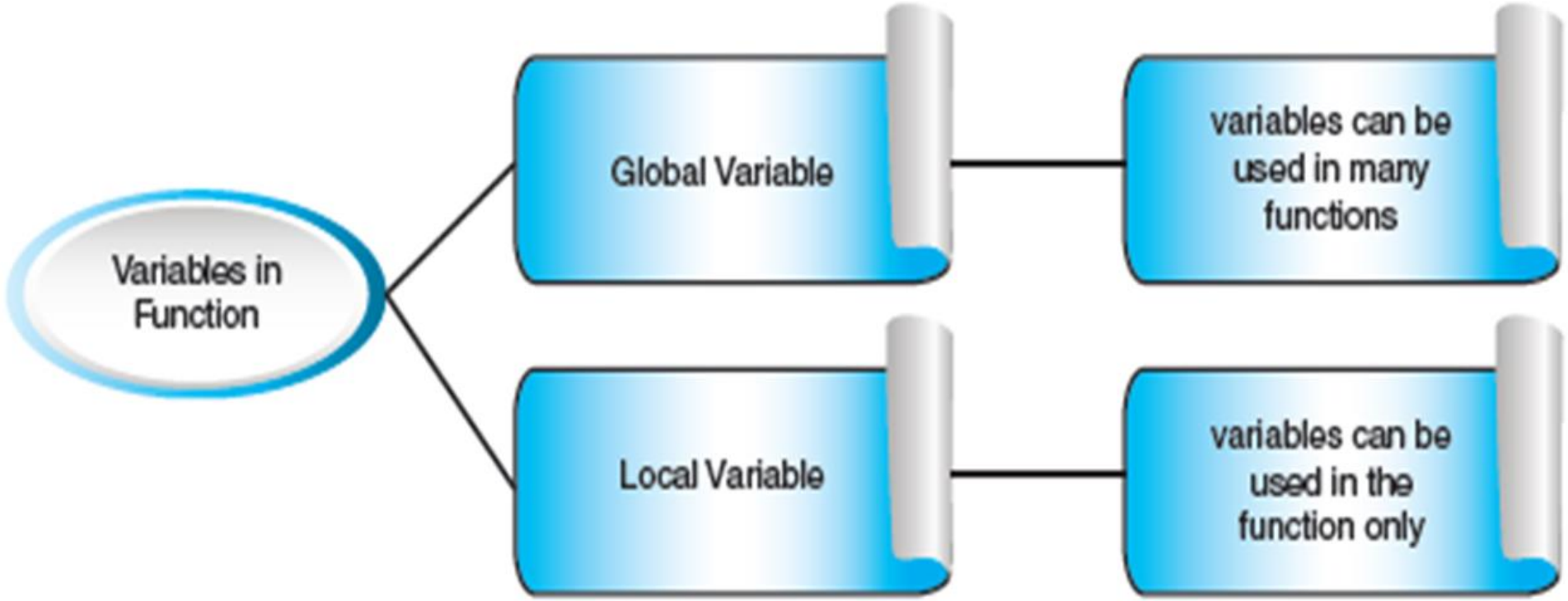
```
#include<iostream>
using namespace std;

//function declaration/prototype
void show_sign(char,int);

void main()
{
    //function can be called more than one time
    show_sign('x',4); //right function call
    show_sign(3,8); // wrong function call
    show_sign(5); // wrong function call
}
//function definition
void show_sign(char sign_type, int num)
{
    cout<<"The sign is:"<<sign_type<<endl;
    cout<<"The number is:"<<num<<endl;
}
```

1 Function Declaration

2 Function Call

3 Function Definition

Types of Variable and Their Scope



Function with Return Statement

- A function which type is int, char, double or float must return values to the function call.
- A function can only return one value at a time.
- The syntax to return a value is:

```
return expression;
```

- an expression can be in the form of values that match with their type of identifiers.

Function with Return Statement

EXAMPLE 5.23

Problem: Prompt user to enter the working hours and rate per hour. Function calSalary() will accept the working hours and rate per hour and will calculate and return the salary to the main function while the main function will display the salary.

```
#include<iostream>
using namespace std;

//no function declaration/prototype is needed

/*function definition with two parameters and return statement*/
double calSalary(int wh, double rh)
{
      double salary;
      salary=wh*rh;
      return salary;
}

int main ()
{
      int workhr;
      double ratehr;

      cout.setf(ios::fixed);
      cout.precision(2);
      cout<<"Enter work hour & rate per hour:";
      cin>>workhr>>ratehr;
                                  /*function call which passed
                                  the workhr and ratehr*/
      cout<<"Salary is:RM"<<calSalary(workhr,ratehr);
      return 0;
}
```

1. Call and send value

2. Return salary

3 Function definition

2 Function Call

EXAMPLE 5.24

```
double absolute_value(double no)
{
      if(no>=0)
            return no;
      else
            return 0;
}
```

Function with Return Statement

EXAMPLE 5.24

```
double absolute_value(double no)
{
        if(no>=0)
            return no;
        else
            return 0;
}
```

Function with Return Statement

- A function which starts with the word void in front does not have to return a value to the function call.
- However, this function can have the word return in it but the purpose is only to stop function execution and does not return any value.
- Any statement after return will not be executed.

Parameter Passing

- If a global variable is not used in a program, the variable needs to be declared as a local variable.
- Parameter passing must be done to pass the variables to the function.
- In parameter passing, the parameter data type needs to be written in the parameter list in the function declaration/prototype and it needs to be declared in the function definition.
- However, in the function call, the parameter which passed the parameter to the function definition must only pass the value or identifier name.

Parameter Passing

- Three important things about parameters:
  1. The number of actual parameters and formal parameters must both be the same in the function call and function definition.
  2. The relationship between the actual parameter and formal parameter is one-to-one. First, the actual parameter must be the same with the first formal parameter.
  3. The data type for every actual parameter must the same as the formal parameter or type that can be changed by the compiler.

Parameter Passing

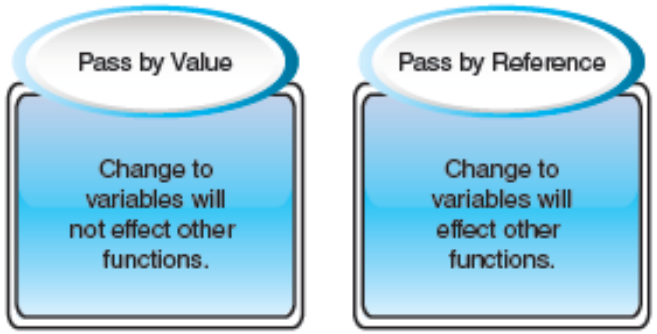
EXAMPLE 5.28

```
#include<iostream>
using namespace std;

//function declaration/prototype
void insert(char);

int main( )
{
        char symbol;

        cout<<"Enter any symbol:";
        cin>>symbol;

        insert(symbol);   //function call
        insert('&');   //function call

        return 0;
}

//function definition
void insert(char sign)

{
        cout<<"The sign is:"<<sign<<endl;
}
```

OUTPUT:

```
The sign is *
The sign is &
```

Parameter Passing



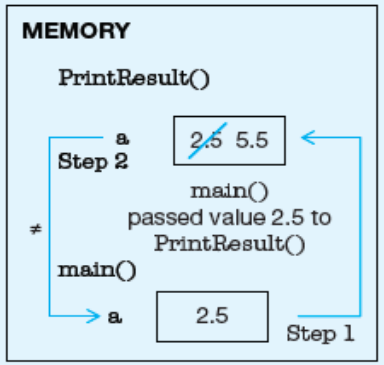
EXAMPLE 5.29

Pass By Value

```
#include<iostream>
using namespace std;

//function with parameter
//pass by value

void PrintResult (double a)
{
        a=5.5;
}
int main()
{
        double a =2.5;
        cout<<a<<endl;

        PrintResult(a);

        cout<<a;

        return 0;
}
```

OUTPUT:

```
2.5
2.5
```



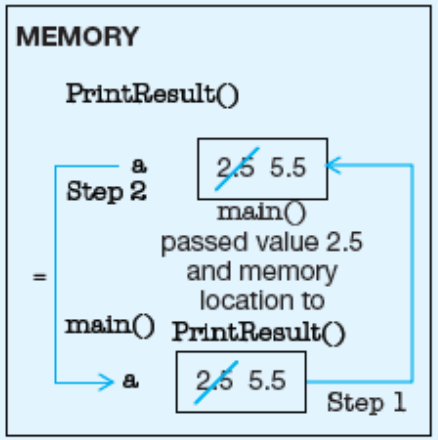
EXAMPLE 5.30

Pass By Reference
- & symbol

```
#include<iostream>
using namespace std;

//function with parameter
//pass by reference
//function with parameter
//pass by value

void PrintResult (double &a)
{
        a=5.5;
}

int main()
{
        double a=2.5;

        cout<<a<<endl;

        PrintResult(a);
        cout<<a;

        return 0;
}
```

OUTPUT:

```
2.5
5.5
```



Parameter Passing

EXAMPLE 5.31

```
//using pass by value

#include<iostream>
using namespace std;

int calc(int a, int b);

int main()
{
        int a=5, b=3,z;

        cout<<"\nBEFORE: ";
        cout<<a<<" "<<b<<endl;

        z=calc(a,b);

        cout<<"\nAFTER: ";
        cout<<a<<" "<<b<<" "<<z<<endl;

        return 0;
}
```

a=5 b=3

a=5 b=3 z = 11

a=5 b=3 j=5 k=3

```
int calc(int j, int k)
{
        int x;
        j=4, k=7;

        x=j+k;

        cout<<"\nINSIDE: ";
        cout<<j<<" "<<k<<" "<<x<<endl;
        return x;
}
```

j=4 k=7 x = 11

```
OUTPUT:

BEFORE: 5 3

INSIDE: 4 7 11

AFTER: 5 3 11
```

Parameter Passing

EXAMPLE 5.32

```
//using pass by reference

#include<iostream>
using namespace std;
void result(int, int &, int &);

int main()
{
        int i=5, j=10, k=3;

        cout<<"\nBEFORE: ";
        cout<<i<<" "<<j<<" "<<k<<endl;

        result(i,j,k);

        cout<<"\nAFTER: ";
        cout<<i<<" "<<j<<" "<<k<<endl;
        return 0;
}

void result(int a, int &b, int &c)
{
        a=b+c;
        b=a*2;
        c=b-a;

        cout<<"\nINSIDE: ";
        cout<<a<<" "<<b<<" "<<c<<endl;
}
```

OUTPUT:

BEFORE: 5 10 3

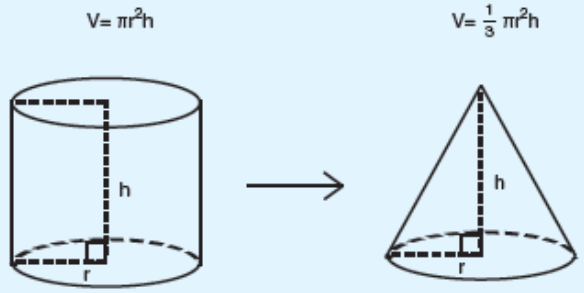
INSIDE: 13 26 13

AFTER: 5 26 13

Parameter Passing

EXAMPLE 5.36

Problem: Write a program to calculate the volume of a cone and cylinder. Use function getValue() to get the radius and height from the user. Function calVolume() will calculate and return the volume of the cone and cylinder. Then display the volume of a cone and cylinder in function display().




```
#include<iostream>
#include<math.h>
using namespace std;

void getValue (int &r, int &h);
void calVolume (int &r, int &h, double &cone, double &cylinder);
void display (double &c, double &s);

int main ()
{
        int radius, height;
        double cone, cylinder;

        getValue (radius, height);
        calVolume (radius, height, cone, cylinder);
        display (cone, cylinder);

        return 0;
}

void getValue(int &r, int &h)
{
        cout<<"Enter radius:";
        cin>>r;

        cout<<"Enter height:";
        cin>>h;
}

void calVolume(int &r, int &h, double &cone, double &cylinder)
{
        const double PI=3.142;

        cone=(1/3.0)*PI*pow(r,2)*h;
        cylinder=PI*pow(r,2)*h;
}

void display(double &c, double &s)
{
        cout<<"\nVolume of a cone is:"<<c<<endl;
        cout<<"\nVolume of a cylinder is:"<<s<<endl;
}
```

(Continued)

Conclusion

- Functions can divide problems into sub-problems.
- In order to develop programs using functions, the three function elements needed to be identified are function prototype/declaration, function call and function definition.
- Writing programs using functions provide many advantages as functions can break a program into smaller items which will be easier for the programmer to code, test and debug.
- Each of the functions is reusable where the function can be called many times.
- Repeating code in a program can be avoided.