Learning Objectives

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

- ✓ Use pre-defined functions: (sqrt(), abs(), pow(), toupper(), tolower(), setw(), setprecision())
- ✓ Build independent functions or user-defined functions
- ✓ Define function's data type: void, int, float, double, char, char*
- ✓ Understand the scope of global and local-variables
- ✓ Learn how to use function with parameter (value and reference)

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

A. Introduction to Function

- As programs get longer and complicated, it is common to group the program statements into its interrelated modules or segments or subprograms.
- Each module or segment will only perform a particular task. The module or segment of codes is referred to as function.

1. Modular Programming

- Modular programming is a way of breaking one module of program into sub-module of program in which each module will performs a particular task only.
- For example, a program may contain a module to perform each of the three main tasks in a program; that is ***input***, ***processing*** and ***output***.
- In C++, modules are called ***functions***.
- Some functions are predefined (built-in – and of course we have to include some necessary pre-processor directive to support that pre-defined function), which are referred to as ***library functions***, and others can be programmer-defined.
- As we have seen, each C++ program must contain `main` function, and it may also contain any number of additional functions which we are going to learn how to write it.

2. Function

- A function is a mini-program that performs a particular task. Each function may include its own variables and its own statements, just like writing the `main` function. This mini-program can be compiled and tested independently.

1. Benefit of Using Functions

i. The main program is simplified where it will contain the function call statements. By doing that planning, coding, testing, debugging, understanding and maintaining a computer program will be easier.

ii. The same function can be reused in another program, which will prevent code duplication and reduce the time of writing the program.

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

Example

The following program calculates and displays the wages of an employee based on the number of hours worked and the hourly rate of pay.

1. Example of program not using function

Problem definition

Input: name, hours worked, hourly rate

Process: calculate wages

Output: wages

Algorithm design

Get Input

Read name, hours_worked, hourly_rate

Calculate wages

Wages = hours_worked * hourly_rate

Display Output

Print wages

Program coding

```
#include <iostream.h>
//To display greeting and calculate wages without using function

void main()
{
       int hoursWorked;
       float hourlyRate;
       float wage;
       char name[25];

       cout << "Please enter name ";
       cin.getline(name, 25);
       cout << "Hello " << name << ". Welcome to this program." << endl;
       cout << "Enter the number of hours worked :";
       cin >> hoursWorked;
       cout << "\nEnter the hourly pay rate :";
       cin >> hourlyRate;
       wage = hourlyWorked * hourlyRate;
       cout << "\nThe wage is :RM" << wage << endl;
}
```


References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

Sample output

```
Please enter your name: Haizat
Hello Haizat. Welcome to this program.
Enter the number of hours worked: 8
Enter the hourly pay rate: 12.5
The wages is : RM100.0
```

After using function

- The above program is modified by applying functions to it. Two functions, `calWages()` and `greeting()` are added.

Algorithm design

main()

    greeting()

    Input hours worked, hourly rate

    calcWages()

    Display wages

greeting()

    Input name

calcWages()

    Calculate wages

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

Program coding

```
#include <iostream.h>
//To display greeting and calculate wages without using function

void calWages(int, int); //function prototype
void greeting();

void main()
{
       int hoursWorked;
       float hourlyRate;
       greeting();
       cout << "Enter the number of hours worked :";
       cin >> hoursWorked;
       cout << "\nEnter the hourly pay rate :";
       cin >> hourlyRate;
       calWages(hoursworked, hourlyRate);
}

void calWages (int hours, int rate)
{
       float wage = hours * rate;
       cout << "\nThe wage is :RM" << wage << endl;
}

void greeting()
{
       char name[25];
       cout << "Please enter name ";
       cin.getline(name, 25);
       cout << "Hello " << name << ". Welcome to this program." << endl;

}
```

Sample output

```
Please enter your name: Haizat
Hello Haizat. Welcome to this program.
Enter the number of hours worked: 8
Enter the hourly pay rate: 12.5
The wages is : RM100.0
```

Explanation

- `calWages()` – only performs the calculation of wages
- `greeting()` – display the greeting
- Each function does specific task, thus making it easier for tracing errors or debugging process.

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

3. Predefined functions

- Predefined or built-in functions are functions used by the programmers in order to speed up program writing.
- Programmers may use the existing code to perform common tasks without having to rewrite any code.
- These functions are predefined by the producer of a compiler, (C++) and are stored in the header files (.h files) called libraries
- In order to for a program to use a pre-defined function, the appropriate library file must be included in the program using the `#include` directive.

Syntax form

```
#include <header file name>
```

Example

To use the `getline()`, a program must include the directive

```
#include <string.h>
```

Some commonly used header files are `iostream.h`, `iomanip.h`, `string.h`, `char.h` (or `ctype.h`), `math.h`, and `stdlib.h`.

a. iostream.h

- `iostream.h` is a header file for input/output stream. The most commonly used built-in functions for input/output are `get()` and `put()`.

Commonly used functions

Name	Description
`get()`	for console input (keyboard) or type char
`put()`	for console output (screen) of type char

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

Example

The following program shows that when the input/output stream (`cin.cout`) are used, the header file `iostream.h` is included at the very beginning of the program file.

```
#include <iostream.h>
void main()
{
      char ch;
      cout << "Please enter a character: ";
      cin.get(ch);
      cout << "\nThe character is  " << put(ch);
}
```

Sample output

```
Please enter a character: a
The character is a
```

b. iomanip.h

- This header file defines a collection of functions to manipulate input and output data.
- Some of the commonly used functions are given below.

Commonly used functions

Name	**Description**
`setw(x)`	Set field width to x
`setfill(x)`	Set the fill character with x
`setprecision(x)`	Set the floating point precision to x
`setiosflags(ios::fixed)`	Display floating point values in decimal notation
`setiosflags(ios::left)`	Left justify output
`setiosflags(ios::right)`	Right justify output
`setiosflags(ios::showpoint)`	Display a decimal point

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

Example

The following program displays the integer, string, and floating point values according to the predetermined output setting.

```
#include <iostream.h>
#include <iomanip.h>

void main()
{
      int x = 2345;
      cout << "|" << setw(8) << setiosflags(ios::left)
           << x << "|" << endl;
      cout << "|" << setfill('*') << setw(8)
           << "HELLO" << "|" << endl;
      cout << "|" << setw(8) << setiosflags(ios::fixed)
           << setiosflags(ios::showpoint)
           << setprecision(2) << 86.2 << "|" << endl;
      cout << "|" << 5 << "|" << endl;

}
```

Output

```
|2345     |
|HELLO***|
|86.20 ***|
|5|
```

c. string.h

- This header file contains several string manipulation functions. Some of the commonly used functions are shown below.

Commonly used functions

Name	**Description**
`strcmp(s1, s2)`	Compares one string to another
`strcpy(s1, s2)`	Copies one string to another
`strlen(s)`	Calculates the length of a string
`strcat(s1, s2)`	Appends one string to another

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

Example

The following program accepts two full names; checks whether both names are equal, edit the names, get the length of the first name and concatenate two names together.

```
#include <iostream.h>
#include <string.h>
void main()
{
       char firsName[30], secName[30];
       cout << "Enter full name: " << endl;
       cin.getline(firstName, 30);
       cout << "Enter another full name: " << endl;
       cin.getline(secName, 30);

       if (strcmp(firstName, secName) == 0)
              cout << "The names are the same" << endl;
       else
              cout << "The names are different" << endl;

       strcpy(firstName, "Marissa");
       cout << "First name is now " << firstName << endl;
       cout << "The length of the first name is "
            << strlen(firstName) << endl;
       strcat(firstName, "Nurdia");
       cout << "First name is now " << firstName << endl;
}
```

Output

```
Enter full name: Nur Saidah
Enter another full name: Nurul Azua
The names are different
First name is now Marissa
The length of the first name is 7
First name is now Marissa Nurdia
```

d. char.h (ctype.h)

- This header file contains declarations of several character handling functions. Some of the most commonly used functions are given below.

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

Commonly used functions

Name	Description
`toupper(c)`	Converts character c from lowercase to uppercase letter
`tolower(c)`	Converts character c from uppercase to lowercase letter
`isupper(c)`	Return TRUE if c is an uppercase letter
`islower(c)`	Return TRUE if c is a lowercase letter
`isdigit(c)`	Return TRUE if c is digit
`isalpha(c)`	Return TRUE if c is an alphanumeric character
`isspace(c)`	Return TRUE if c is a space character

Example

The following program accepts a character, checks whether the character is an uppercase or a lowercase. If it is an uppercase, the lowercase of the character is then displayed and vice versa.

```
#include <iostream.h>
#include <ctype.h>

void main()
{
       char letter;
       cout << "Enter a character: ";
       cin >> letter;

       if(isalpha)
       {      if (isdigit(letter))
              {
                     cout << "The character is a digit" << endl;
              }
              else if (isupper(letter))
              {
                     cout << "The character is an uppercase" << endl;
                     cout << "The lowercase is "
                        << (char) tolower(letter) << endl;
              }
              else
              {
                     cout << "The character is a lowercase" << endl;
                     cout << "The uppercase is "
                        << (char)toupper(letter) << endl;
              }
       }
       else if (isspace(letter))
       {
              cout << "The character is a space" << endl;
       }
}
```


References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

Output

```
Enter a character: Z
The character is an uppercase
The lowercase is z
```

e. math.h

- This header file contains declarations of several mathematic functions. Some of the most commonly used functions are given below.

Commonly used functions

Name	**Description**
`pow(x, y)`	Return x raised to the power of y
`sqrt(x)`	Square root of x
`ceil(x)`	Smallest integer >= x
`floor(x)`	Largest integer <= x
`sin(x)`	Sine of x (x is in radians)
`cos(x)`	Cosine of x (x is in radians)
`tan(x)`	Tangent of x (x is in radians)
`asin(x)`	Arc sine or $\sin^{-1}$ of x
`acos(x)`	Arc cosine or $\cos^{-1}$ of x
`atan(x)`	Arc tangent or $\tan^{-1}$ of x
`log(x)`	Natural logarithm of x or ln(x), x>0
`sinh(x)`	Hyperbolic sine of x
`cosh(x)`	Hyperbolic cosine of x
`tanh(x)`	Hyperbolic tangent of x
`log10(x)`	Common log (base 10) of x or lg(x)
`exp(x)`	e raise to the x power or e^x
`fabs(x)`	Absolute value |x|

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

Example

The following program accepts a number, gets the square root of the number and multiplies the number by using the `sqrt()` and `pow()`.

```
#include <iostream.h>
#include <math.h>

void main()
{
      double number;
      cout << "Enter a number: ";
      cin >> number;
      cout << "The square root of " << number << " is "
           << sqrt(number) << endl;
      cout << number << " raised to the power of 2 is "
           << pow(number,2) << endl;
}
```

Output

```
Enter a number: 9
The square root of 9 is 3
9 raised to the power of 2 is 81
```

f. stdlib.h

- This header file contains declaration of several miscellaneous (mixed) functions. Some of the most commonly used functions are given below.

Commonly used functions

Name	**Description**
`abs(i)`	Converts to the absolute value of i
`rand()`	Generate a random positive integer between 0 and RAND_MAX(32767)
`srand(seed)`	Initialize random number generator where seed represent the starting point for the rand function

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

Example

The following program accepts a value of seed and generates five random values

```
#include <iostream.h>
#include <stdlib.h>
#include <iomanip.>

void main()
{
       unsigned seed;
       float randValue;
       cout << "Enter the value of seed ";
       cin >> seed;

       srand(seed); //generate the first seed value
       cout <<\n5 generated numbers are: " << endl;
       for (int i=1; i<=5; i++)
       {
              randValue = rand();
              cout << setw(8) << setfill('#')<< randValue << endl;
       }
}
```

Output

```
Enter the value of seed: 10
5 generated random numbers are:

####3463
###30957
###10345
###10368
####8444
```

4. Independent functions

- Programmer are able to create their own functions since the built-in function in C++ libraries are limited to perform other programming tasks.
- Independent function or also known as programmer-defined are functions whose task are determined by the programmer and defined within the program in which the function is used.
- The function definitions can be written in the same file as `main()` or compiled independently in separate files.
- To compile separately, the file that contains the definitions must be included and the following syntax form is used:

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

Syntax form

```
#include "<filename>.cpp"
```

Example

```
#include "myfile.cpp"
```

Program structure

The overall program structure when using the independent functions is shown below:

```
#include <iostream.h>
function prototype declarations;

void main()
{
      variable declarations;
      statements [including function calls];
}
function definition(s)
{
      statements;
}
```

- In order to use the independent function, a programmer must satisfy three requirements:
    i. Function prototype declaration
    ii. Function call
    iii. Function definition

5. Function Prototype Declarations

- All functions must be declared before they can be used in a program.
- Placed just before the main program and sometimes at the beginning of the main program, a function declaration specifies:
    i. The name of the function
    ii. The order and type of parameters
    iii. The return value
- The purpose of function prototype declaration allows the compiler to check for the existence of the function, the parameter list and the return type is correct.
- The syntax for the independent function prototype declaration is given below:

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

C++ syntax form

```
returnType functionName (type parameter_list);
```

Explanation

`returnType`

a. Indicates the type of value that the function will return
b. The return value may be of any valid data type:

```
int, float, double, char, long and void
```

c. void is used when a function does not return any value to the calling program

`functionName`

a. Is the name of function
b. It can be any names but the naming convention must follow the rules in naming the identifier

`type parameter_list`

a. Tells the calling function what type of data value the parameter will receive when the function is called
b. Tells the calling function the order of the values to be transmitted to the called function
c. If it contains more than one data type, parameter_list must be separated by comma(s)
d. Parameter names are optional
e. Parameter_list is also known as arguments

Function prototype examples

The following are examples of the function prototype:

```
void display();
int findSum(int, int);
float swap(float, float, int, char);
```


References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

Explanation

The following explains the declaration of the prototype:

a. `void display();`

   Declares that function `display()` does not receive any argument and does not return any argument to the calling function

b. `int findSum(int, int);`

   Declares that the function `findSum()`expects to receive two integer argument and return value to the calling function with data type integer.

c. `float swap(float, float, int, char);`

   Declares that the function `swap()` requires four arguments consisting of two floating-points argument, one integer argument and one character argument, in this order, and that it will return a floating-point value.

Parameters

There are two types of parameters:

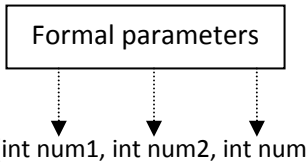
i. Formal parameter

ii. Actual parameter

a. Formal parameter

- A formal parameter is a placeholder variable which is defined at the called function and local function to the called function.
- The number of the formal parameters must agree with the number of actual parameters passed to the function.
- Example:

  The following function takes three formal parameters to calculate the sum.



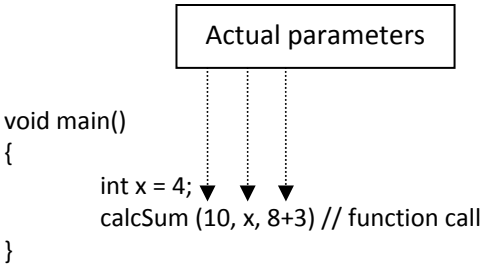

```
void calcSum (int num1, int num2, int num3)
{           cout << "Total : " << num1 + num2 + num3
}
```


References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

b. Actual parameter

- Actual parameter is a constant, variable or expression in a function call that corresponds to the formal parameter.
- Actual parameters contain values that will be passed to the function
- Example:

  The following function contains three values which will be passed to the `calcSum()`.



6. Function Definition

- A function must be defined before it can carry out the task assigned to it.
- In other words, the statements that perform the task are written inside the function definition.
- A function is written once in the program and can be used by any other function in the program.
- The function definition is placed either before the `main()` or after the `main()`.

Before the main()

The function prototype is not required.

```
int findMax()
{
      statements;
}

void main()
{
      findMax();
}
```


References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

After the main()

```
int findMax();      //function prototype

void main()
{
       findMax();
}
int findMax()
{
       statements;
}
```

- A function definition consist of a function header and function body as shown below:

Syntax form

```
returnType functionName (type parameter_list)
{
       declaration(s);
       statement(s);
       return expression;
}
```

Function header

Function body

Explanation

a. Function header

- `returnType functionName (type parameter_list)`
- The function header defines the return type, function name and list of parameters.
- It is written the same way as the function prototype, except that the header does not end with semicolon.
- Function header contains the following:
  - i. Type of data, if any, is returned from the function when operation has completed
  - ii. The name of the function
  - iii. Type of data, if any, is sent into the function.

b. Function body

- The function body contains declaration of local variables and executable statements needed to perform a task assigned to the function.

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

Example

An example of function definition that accepts three values, calculates the total and return the total to the main function.

```
int sum3Num(int a, int b, int c)
{
      int total = a + b + c;
      return total;
}
```

Explanation

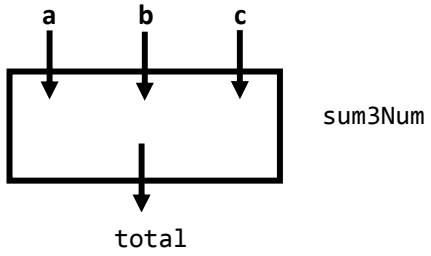
- The keyword `int` in the function header indicated that the function will return an integer value**.**
- The parameter-list `int a, int b` and `int c` indicated that three integer values will be sent into and used in the function sum3Num
- `Return total` indicated that the value of total will be returned to the calling function

Function representation

The above example can be represented as follows:

The values `a, b` and `c` are sent into the function

The function `sum3Num`
Calculates the `total`



The function returns the total

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

The keyword void

- A function that does not return a value after completing its assigned task is called a void function.
- When there is no value to be returned to the calling function, and no value accepted from the calling function we may use the following syntax form:

```
void functionName( )
{
        statements;
}
```

- When the `parameterList` is empty, it shows that no data will be transmitted into the function when it is run. We may use either `functionName()` or `functionName (void)` for that purpose.

Example of void functions

```
void sum();          //receives no values and returns no value
void calcSum(int a, int b);       //receives 2 integers and returns no value
```

Example 1

```
#include <iostream.h>

void sum();  //function prototype declaration

void main()
{
       cout << "Welcome!";
       sum(); //function call
       cout << "\nEnd of program";
}

void sum()   //function definition
{
       int num1, num2, total;
       cout << "Enter two numbers (separate by space): ";
       cin >> num1 >> num2;
       total = num1 + num2;
       cout << "\nThe sum is : " << total << endl;
}
```


References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

Example 2

```
#include <iostream.h>

void sum(int, int); //function prototype declaration

void main()
{
       int num1, num2;
       cout << "Welcome!";
       cout << "Enter two numbers (separate by space): ";
       cin >> num1 >> num2;
       sum(num1, num2);    //function call
       cout << "\nEnd of program";
}

void sum(int n1, int n2)  //function definition
{
       int total;
       total = n1 + n2;
       cout << "\nThe sum is : " << total << endl;
}
```

Output

```
Welcome!
Enter two numbers: 3 5
The sum is : 8
End of program
```

Explanation for example 1

- In the above the example, the `sum()` receives no values from the main() and returns no value to the `main()`.
- When the function `sum()` is called in the `main()`, C++ will search for the `sum()` and execute the statements inside the `sum()`.

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

7. Function Call

- A function is made to perform its designated task by a function call in main function or in other function.
- When a function is called, the program control is passed to the called function and the statements inside the function will be executed to the end until control is passed back the calling function.
- To call a function, specify the function name and the values of the parameters that the function needs to do it job.
- The following syntax is used when calling a function:

```
Function-name(actual parameters);
```

Example of function calls

```
findMin(num1, num2);
getSum(firstNum, secNum);
```

Explanation

a. `findMin()`

- A call to the function `findMin()` is made by the statement `findMin(num1, num2);`.
- The value residing in the variables num1, and num2 will be sent to the function definition `findMin()`.
- The function `findMin`() uses the values received to perform its assigned task. Here, the control is transferred to the function defintion `findMin()` and after execution is completed in the function definition `findMin()`, control is passed back to the calling function.

b. `getSum()`

- The `getSum()` causes the values if `firstNum` and `secondNum` to be sent to function definition `getSum()`.
- After the execution in the function definition `getSum` is completed, the control is passed back to the calling function.

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

Example

The following program shows the function calls used for the displayAvg() and calcSum()

```
#include <iostream.h>
//function prototype declaration
void displayAvg(int, int, int);
void calcSum(int, int, int);

void main()
{
	int num1, num2, num3; // local variable to the main function

	cout << "Enter three numbers (separate by space): ";
	cin >> num1 >> num2 >> num3;

	//function calls
	calcSum (num1, num2, num3); // calls calcSum()
	displayAvg(num1, num2, num3);
}

//function definition
int displayAvg(int x, int y, int z)
{
	float avg;
	avg = (x + y + z) / 3;
	cout << "\nThe average is : " << avg << endl;
}

void calcSum(int a, int b, int c)
{
	int sum = 0;
	sum = a + b + c;
	cout << "\nThe sum is : " << sum << endl;
}
```

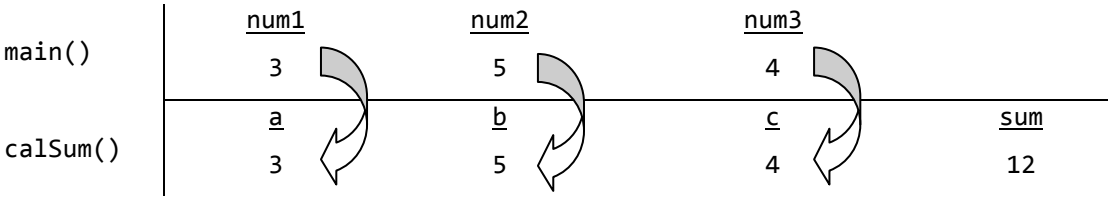
Example

```
Enter 3 numbers (separate by space): 3 4 5
The sum is : 12
The average is 4.0
```


References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

a. Making a function call

The following diagram shows the process that occurs when making a function call in statement `calcSum(num1, num2, num3)`.



Explanation

- When the statement `calcSum(num1, num2, num3);` is executed, values stored in `num1, num2,` and `num3` will be transmitted to the statement `calcSum().`
- The `calcSum()` then must determine where the values are to be stored.
- As shown the `calcSum()` function definition, the variables in the parameter list of the `calcSum()`, i.e., int `a, b` and `c` specifies that the values received from the `main()` will be stored into the location `a, b` and `c` as shown in the above diagram.
- The values received from the `main()` will be used perform the addition operation as specified in the `calcSum()`.
- As a result, after the statement `sum = a + b;` is executed, the value 12 is assigned to the variable `sum.`
- The latest statement `cout << "\nThe sum is : " << sum << endl;` causes the value of sum, i.e. 12 to be printed on the screen.

b. Completing a function call

i. Function call that returns no value

ii. Function call that returns a value

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

c. Function call without value returned

- This type of function does not return any value to the calling function.
- Upon completing the last statement in the function definition, the control is passed back to the statement that calls the function in the calling function.
- Thus, the next statement in the calling function will be executed.

Example

```
//function prototype with no return value

void displayAvg(int, int, int);
void calcSum(int, int, int);'

void main()
{
	int num1, num2, num3;

	cout << "Enter 3 numbers : ";
	cin >> num1 >> num2 >> num3;

	//function calls
	calcSum(num1, num2, num3); // function call
	cout << "Next process…";
	displayAvg(num1, num2, num3); // function call
}

void calcSum(int a, int b, int c)
{
	int sum = 0;
	sum = a + b + c;
	cout << "\nThe sum is : " << sum << endl;
}

void displayAvg(int x, int y, int z)
{
	float avg;
	avg = (x + y + z) / 3;
	cout << "\nThe average is : " << avg;
}
```

Sample output

```
Enter 3 numbers: 3 4 5
The sum is : 12
Next process…
The average is : 4.0
```


References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

Explanation

- The statement `calcSum(num1, num2, num3);` causes the function `calcSum()` to be executed.
- Upon completing the execution in the `calcSum()`, control passed back to the calling statement in the main().
- The next statement, `cout << "Next process…";` will get executed.

d. Function call with value returned

- A function may also return a value. In this type of function, when the execution in the called function is complete, the control is passed back to the calling function with a value. This value will be used in the calling function.
- There are four ways how a returned value can be used in the calling function.
  i. In an arithmetic expression
  ii. In a logical expression
  iii. In an assignment statement
  iv. In an output statement

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

Example

```
//function prototype that returns a value
int calcSum(int, int, int);

void main()
{
	int num1, num2, num3, total;

	cout << "Enter 3 numbers :";
	cin >> num1 >> num2 >> num3;

	//in an assignment statement
	total = calcSum(num1, num2, num3);

	//in an output statement
	cout << "\nThe sum is :" << calcSum(num1, num2, num3);

	//in a logical expression
	if (calcSum(num1, num2, num) < 0
		//in an arithmetic expression
		total = calcSum(num1, num2, num3) / 3
}

//function definition
int calcSum(int a, int b, int c)
{
	int sum = 0;
	sum =  a + b + c;
	return sum;
}
```

B. Types of Independent Functions

In summary, when using functions, we can have four types of functions.

i. Function without value returned and parameter

ii. Function with value returned but without parameter

iii. Function without value returned but with parameter

iv. Function with value returned and parameter

	Not return value	**Return value**
Without parameters	void calcSum()	int calcSum()
With parameters	void calcSum(int a, int b)	int calcSum(int a, int b)

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

1. Function Without Value Returned and Without Parameter

- This function has no return value and the returned type is written as void. All inputs, calculation and outputs are done inside this function.

Example

The following program accepts radius from the keyboard, calculates and display the area of a circle.

```
#include <iostream.h>
void calcAreaCircle();    //function prototype
void main()  //calling function
{
      calcAreaCircle();
}

void calcAreaCircle()     //function definition
{
      float radius;
      cout << "Enter radius of circle";
      cin >> radius;
      cout << "\nThe area of the circle is " << 3.142 * radius * radius;
}
```

Sample output

```
Enter radius of circle: 2.1
The area of the circle is 13.85622
```

Explanation

- When the statement `calcAreaCircle();`is executed, the control from the calling function i.e. `main()` is passed to the called function i.e. `calcAreaCircle()`. Then the statement inside the `calcAreaCircle()` get executed.
- Upon completing the execution, the control is passed back to the calling function.

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

Example

The following program prints numbers in ascending and descending orders through `printUp()` and `printDown()`.

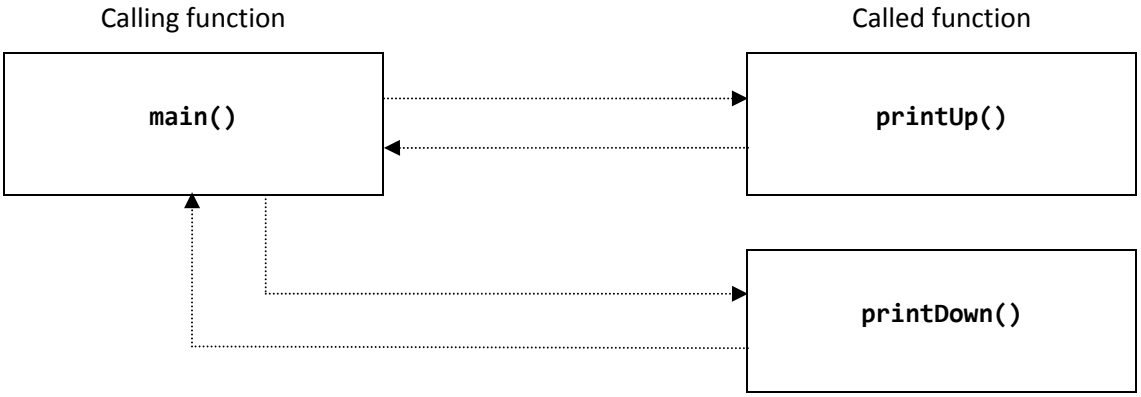
```
#include <iostream.h>
//function prototype
void printUp();
void printDown();

void main()
{
	//function call
	printUp();
	printDown();
}

//function definition
void printUp()	//printing numbers in ascending order
{
	int s = 1;
	while (s <= 10)
	{
		cout << s << " ";
		s++;
	}
	cout << endl;
}

void printDown()	//printing numbers in descending order
{
	int s = 10;
	while (s >= 1)
	{
		cout << s << " ";
		s--;
	}
	cout << endl;
}
```

Sample output

```
1 2 3 4 5 6 7 8 9 10
10 9 8 7 6 5 4 3 2 1
```


References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

Explanation



- When the statement `printUp();` is executed, the control is passed to the `printUp()`.
- The statements in the `printUp()` are executed and when completed the control is passed back to the calling function.
- Next, the statement `printDown()` get executed where the control is passed to the `printDown()`.
- Upon completing the execution in the `printDown()`, the control is passed back to the calling function.

2. Function With Value Returned but Without Parameter

- This function does not receive any values from the calling function but it will return a value (normally the result of a calculation) to the `main()` or other calling function.

Example

The following program accepts a radius from the keyboard, calculates and displays the area.

```
#include <iostream.h>
float calcAreaCircle();   //function prototype
void main()  //calling function
{
      cout << "\nThe area of the circle is " << calcAreaCircle();
}

float calcAreaCircle()    //function definition
{     float radius, area;
      cout << "\nEnter radius of circle: ";
      cin >> area;
      area = 3.142 * radius * radius;
      return area;
}
```


References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

Sample output

```
Enter radius of circle: 2.1
The area of the circle is 13.85622
```

Explanation

- When the following statement `cout << "\nThe area of the circle is " << calcAreaCircle();` is executed, C++ will search for the `calcAreaCircle()` first.
- Then the statements inside the function get executed. The last statement `return area;` indicates that a value stored in the location `area` is to be returned to the `main()`.
- The `cout` statement in the `main()` will display the value as shown in the sample output.

3. Function With Parameters and Without Value Returned

- This type of function transfers or copies data from the calling function to the called function through parameters but no value is returned from it.

Example

The following program calculates the average if three numbers and displays the value on the screen.

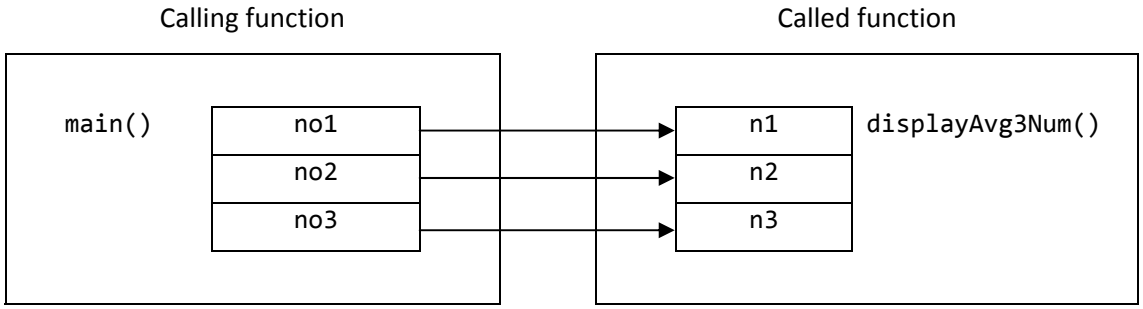
```
#include <iostream.h>
//function prototype declaration
void displayAvg3Num(int, int, int);
void main()
{      int no1, no2, no3;
       cout << "Enter three numbers";
       cin >> no1 >> no2 >> no3;
       displayAvg3Num(no1, no2, no3);   //function call
}
//function definition
void displayAvg3Num(int n1, int n2, int n3)
{      float average;
       average = (n1 + n2 + n3) / 3;
       cout << "\nThe average of the numbers is : " << average;
}
```


References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

Sample output

```
Enter three numbers: 5 10 15
The average of the numbers is 10.0
```

Explanation



- When the function `displayAvg3Num(no1, no2, no3);` is called, the values stored inside `no1`, `no2`, and `no3` are copied into the location `n1`, `n2` and `n3` inside the function `displayAvg3Num()`.
- With these values, the function `displayAvg3Num()` will use them to calculate and consequently display the average.

4. Function With Value Returned and With Parameters

- With this type of function, values are passed to the called function through parameters and one value is passed back to the calling function.

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

Example

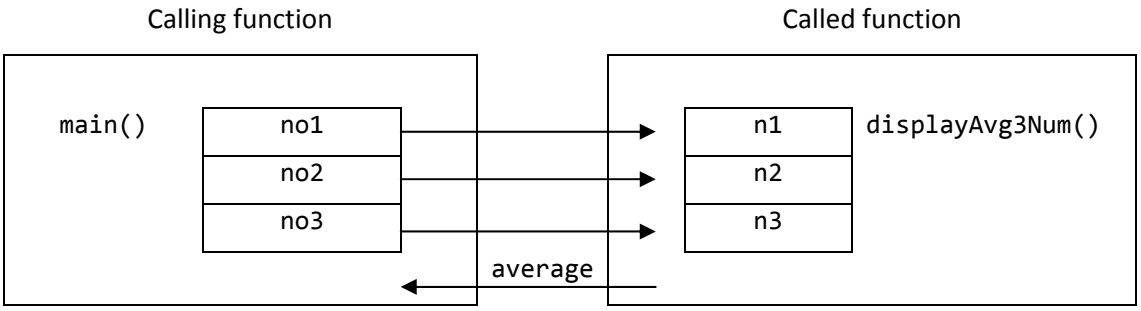
The following program calculates the average of three numbers and displays the value to the user.

```
#include <iostream.h>
//function prototype

float displayAvg3Num(int, int, int);
void main()
{
	int no1, no2, no3;
	float avg;
	cout << "Enter three numbers";
	cin >> no1 >> no2 >> no3;
	avg = displayAvg3Num(no1, no2, no3);
	cout << "\nThe average is: " << avg;
}

//function definition
float displayAvg3Num(int n1, int n2, int n3)
{
	float average;
	average = (n1 + n2 + n3) / 3;
	return average;
}
```

Sample output

```
Enter three numbers: 10 10 10
The average of the numbers is 10.0
```

Explanation



- When the statement `avg = displayAvg3Num(no1, no2, no3);` is executed, the function call to `displayAvg3Num()` is carried out first resulting the values of `no1`, `no2` and `no3` to be sent to the function respectively.

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

- The function then calculates the average of those values and return the average value to the `main()`.
- Upon returning to the `main()`, the value returned will be captured into the variable avg. the display statement following it will display the value kept in avg to the screen.

Another Example

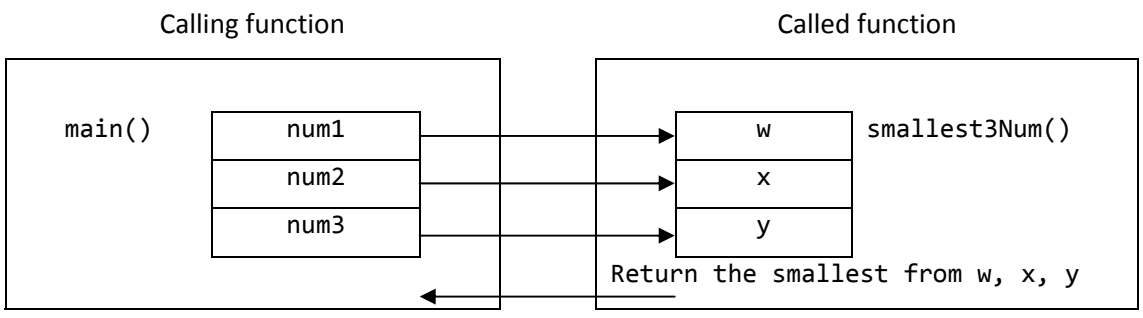
The following program accepts three floating-point values and displays the smallest value.

```
#include <iostream.h>
//function prototype

float smallest3Num(float, float, float);
void main()
{
	float num1, num2, num3;
	cout << "Enter 3 floating-point values: ";
	cin >> num1 >> num2 >> num3;
	cout << "\nThe smallest value is " << smallest3Num(num1, num2, num3);
}

//function definition
float smallest3Num(float w, float x, float y)
{
	if(w < x && w < y)
		return w;
	else if (x < w && x < y)
		return x;
	else
		return y;
}
```

Sample output

```
Enter 3 floating-point values: 5.0 10.5 15.5
The smallest values is 5.0
```

Explanation



References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

- The statement `cout << "The smallest values is " << smallest3Num (num1, num2, num3);` causes the function `smallest3Num()` to be called.
- The values of `num1`, `num2` and `num3` are passed to the function which will be used to determine the smallest value. The smallest value returned to the `main()` will be displayed on the screen.

Summary

	Without Value Returned	With Value Returned
Without Parameters	void calcSum( )	int calcSum( )
With Parameters	void calcSum(int a, int b)	int calcSum(int a, int b)

C. How Values are Passed Between Function

- A function communicates with another function by passing values between them.
- A function can send data to other function, and it can also accept values sent by other functions.
- Data can be passed between function through
    i. Global variable
    ii. Parameter passing
    iii. A return value

1. Global Variable

- When a variable is declared as global, its value can be changed at any point during the program execution because the values are accessible to all function definition in the program.
- The new value assigned to the global variable will replace value stored and this new value will then be used throughout the whole program.
- The global variable is typically declared outside of any function in the program and they remain in memory until the program ends.

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

Example

The following program accepts three values and displays the sum.

```
#include <iostream.h>

void calcSum();     //function prototype
int sum;     //global variable

void main()
{
       sum = 0;
       cout << "Sum is : " << sum << endl;
       calcSum();
       cout << "\nThe sum is :" << sum;
}

void calcSum()
{
       int no1, no2, no3;  //local variables
       cout << "\nEnter three value : ";
       cin >> no1 >> no2 >> no3;
       sum = no1 + no2 + no3;
}
```

Sample output

```
Sum is : 0
Enter three values: 2 5 7
The sum is 14
```

Explanation

sum	~~0~~ 14

main()			
calcSum()	no1 2	no2 5	no3 7

- When the statement `sum = 0` is executed in the `main()`, the global variable sum is assigned with the value `0`.
- Next the statement `calcSum()` causes the function `calcSum()` to be called and the statements in the function `calcSum()` are executed. Here, after the statement `sum = no1 + no2 + no3;` is carried out, `sum` gets a new value, 14.
- When the control is passed back to the `main()`, the display statement `cout << "\nThe sum is : " << sum;` displays new value assigned to the global variable `sum`.

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

- Local variables on the other hand, are used inside the function that declared them. Thus, the variable `no1`, `no2`, and `no3` are local variables in the function `calcSum()`.
- Global variables are sometimes useful in creating variables and constants that must be shared between many functions. However, declaring a variable as a global should be avoided due to the following drawbacks:
    i. It allows unintentional errors to occur when a function that does not need to access the function accidently changes the variable's contents.
    ii. Functions allow the tasks to be independently executed and reusable, on the other hand the use of global variables makes the program more dependent on them.
- Thus, if more than one function needs access to the same variable, it is better to create a local variable in one of the functions and then pass that variable only to the functions that need it.

2. Parameter passing

- Parameters can be passed from the calling function to the called function using two type of passing which are:
    i. Parameter passing by value
    ii. Parameter passing by reference

a. Parameter passing by value

- When passing by value, a copy of that value is passed from the calling function to the called function through the parameters.
- The changes made to the copied value do not change the original value.

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

Example

The following example shows that a value is passed to the `displayValue()` and displayed as statement.

```
#include <iostream.h>
void displayValue(int);   //function prototype

void main()
{      int num = 4;
       cout << "num BEFORE the function call : " << num << endl;
       displayValue(num);  //function call
       cout << "num AFTER the function call : " << num << endl;
}

void displayValue(int value)     // function definition
{
       cout << "Value PASSED to the function is : " << value << endl;
       value = value * 3;
       cout << "Value at the end of function : " << value << endl;
}
```

Sample output

```
num BEFORE the function call : 4
Value PASSED to the function is : 4
Value at the end of the function : 12
num AFTER the function call :  4
```

Explanation

Calling function	Called function
	Num
main()	4
	Value
displayValue()	4
	12

- When `displayValue(num);` is executed, the `displayValue(int value)` is called and then a copy of the value stored in the variable `num` is passed to the function. Thus, 4 is passed to the function and stored in the variable value.

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

- The statement `value = value * 3;` causes the value to be assigned a new value, 12. After the display statement is carried out, the control is passed back to the `main()`. The display statement in the main() displays the value stored in the local variable num. Thus, the value stored i.e. 4 is displayed.

Another example

The following program finds the sum of four numbers where the function `add()` is called by value.

```
int add(int, int, int, int);     //function prototype
void main()
{
      int sum = 0;
      sum = add(10, 20, 15, 5); //function call
      cout << "The sum is : " << sum;
}

//function definition
int add(int a, int b, int c, int d)
{
      int s;
      s =  a + b + c + d;
      return s;
}
```

Sample output

```
The sum is : 50
```

Explanation

Calling function	Called function				
					sum
main()	10	20	15	5	~~0~~ 50
	a	b	c	d	s
add()	10	20	15	5	50

- The statement `sum = add(10, 20, 15, 5);` calls the `add(int a, int b, int c, int d)` and passes four value to the function. Thus in the `add()`, the variables `a`, `b`, `c`, and `d` get copy the value passed to each of them.

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

- When the arithmetic expression `s = a + b + c + d;` is executed, `s` is assigned the value of `50`. This value is returned to the `main()` and captured into the variable `sum`. The display statement displays the value then stored in `sum`.

b. Parameter passing by reference

- When there is more than one value to be returned to the calling function or changed during the program execution, reference parameter is used.
- When a parameter is sent by reference, the function actually gets the memory location of the actual parameter it can directly access the data.
- In other words, the address of the variable in the computer's memory is actually being passed. As a result, the changes made to the variable will affect the original value.
- When passing by reference, the actual parameter in the calling function must be variable and the formal parameter must use the reference operator, which is referred by the symbol ampersand (&). The reference operator forces the corresponding actual formal parameters to refer to the same location.
- In addition, the returned-type for passing parameters by reference is typically given the type void.

Example

The next program shows that four memory locations are passed to the `getMinMax()` where the last two parameters are referenced.

```
//function prototype
void getMinMax(int, int, int&, int&);

//function call
getMinMax(num1, num2, minimum, maximum);

//function definition
void getMinMax(int a, int b, int& min, int& max)
{     if(a > b)
      {     min = b;
            max = a;
      }
      else
      {     min = a;
            Max = b;
      }
}
```


References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

Example

The following program shows that the `displayValue()` is called by reference which causes the changes made after the call affect the original value.

```
#include <iostream.h>

void displayValue(int&);  //function prototype

void main()
{
      int num = 4;
      cout << "num BEFORE the function call : " << num << endl;
      displayValue(num);  //function call
      cout << "num AFTER the function call : " << num << endl;
}

void displayValue(int& value)    //function definition
{
      cout << "Value PASSED to the function is : " << value << endl;
      value = value * 3;
      cout << "Value at the end of the function : " << value << endl;
}
```

Sample output

```
num BEFORE the function call : 4
Value PASSED to the function is : 4
Value at the end of the function : 12
Num AFTER the function call : 12
```

Explanation

Calling function	Called function
`main()`	`num` 4 12
`displayValue()`	`&value` 4 12

Any changes in value will directly reflect num

- In the `main()`, the value is referenced as num while in the `displayValue()`, the value is referenced as `value`.
- The statement `value = value * 3;` causes the `value` to be assigned 12.
- This changes just made will affect the original value, thus `num` on the `main()` gets 12.

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

Example

The following program shows the use of function call by reference, to find the sum of four numbers.

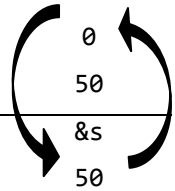
```
//function prototype
void add(int, int, int, int, int&);

void main()
{
	int sum = o;
	add(10, 20, 15, 5, sum);  //function call
	cout << "The sum is : " << sum;
}

//function definition
void add(int a, int b, int c, int d, int& s)
{
	s = a + b + c + d;
}
```

Sample output

```
The sum is : 50
```

Explanation

	10	20	15	5	sum
					0
main()					50
displayValue()	a	b	c	d	&s
	10	20	15	5	50



- In `main()` value referenced is `sum` and in the `displayValue()` the value is referenced as `value`.
- When `add(10, 20, 15, 5, sum);` is executed, `add()` is called and four values and a memory location are passed to the function.
- The memory is referenced as `s` in the `add()` and as `s` is assigned the value `50`, the changes reflect the original variable, `sum`. Thus `sum` gets a new value, `50`.

References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam

Example

```
#include <iostream.h>
#include <conio.h>

void change(int& num1, int& num2)      //function definition
{
      num1 += 10;
      num2 += 20;
}

void main()
{
      int firstnum, secnum;

      cout << "Enter two numbers : ";
      cin >> firstnum >> secnum;

      cout << "\nFirst number is : " << firstnum;
      cout << "\nSecond number is : " << secnum;

      change(firstnum, secnum); //function call

      cout << "\nFirst number is now : " << firstnum;
      cout << "\nSecond number is now  : " << secnum;

getch();
}
```

Sample output

```
Enter two numbers : 8 9
First number is : 8
Second number is : 9
First number is now : 18
Second number is now : 29
```


References:
Rosenah A. H., (2009), Introduction to Programming with C++, for Scientists and Engineers, UPENA UiTM Shah Alam
Norizan M., Mazidah, P., (2006), Problem Solving with C++, UPENA UiTM Shah Alam