User Defined Function

Two types of functions:
- **void** function (with value and reference parameters)
- **value returning** function

Function Parameter
- **Parameter passing by value**
  - a copy of the data is created and placed in a local variable in the called function
  - regardless how the data is manipulated and changed in the called function, the original data in the called function are safe and unchanged
- **Parameter passing by reference : necessary when more than one value need to be passed back to the calling function**
  - sends the address of a variable to the called function, rather than sending its value
  - used when you want to change the content of a variable in the calling function
    Indicate reference parameter(s) by adding the address operator : **&**

Example 1 : Write a program to convert the currency from US Dollar to EURO, or vice versa.

// Declare the user-defined functions here

const float DOLLAR_TO_EURO = 0.87;   // currency conversion rate

```
int main( )
{
        float     amount;                       // amount entered by the user
        float     convertedAmount;          // amount converted to the second currency
        char      currency;              // currency type of the amount entered by the user

        DisplayWelcome();

        GetUserInput(amount, currency);

        // compute the converted amount based on the currency type provided
        if (currency=='d')
                convertedAmount = amount * DOLLAR_TO_EURO;
        else
                convertedAmount = amount / DOLLAR_TO_EURO;

        DisplayResults(amount, convertedAmount, currency);

     return 0;
}
```

//Define each user-defined functions below:

Example 2:

```cpp
void Exchange (int &, int &);
int main()
{
        int  num1= 3,  num2 = 5;

        cout << num1 << "\t" << num2 << endl;

        Exchange (num1, num2);

        cout << num1 << "\t" << num2  << endl;

        return 0;
}

void Exchange (int &  number1,  int &  number2)
{
        int  temp;

        temp = number1;
        number1 = number2;
        number2 = temp;

        return;
}
```

Example 3:

```cpp
void Divide(int, int, int&, int&);
int main( )
{
        int  num1, num2;
        int  quotient, remainder;

        cout << "Enter two integers\n" ;
        cin >> num1 >> num2;

        Divide(num1, num2, quotient, remainder);

        cout << "Quotient is " << quotient << "\t";
        cout << "Remainder is " << remainder
            << endl;

        return 0;
}

void Divide(int numerator, int denominator, int &  quotient,   int  & remainder)
{
        quotient  = numerator / denominator;
        remainder = numerator % denominator;

        return;
}
```

Example 1 revisied : Write a program to convert the currency from US Dollar to EURO, or vice versa. **The program reads the inputs from a data file instead**

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

// Declare the user-defined functions here


```
const float DOLLAR_TO_EURO = 0.87;   // currency conversion rate

int main( )
{
        float      amount;                      // amount entered by the user
        float      convertedAmount;             // amount converted to the second currency
        char       currency;           // currency type of the amount entered by the user
        ifstream   myIn;

        cout << "Welcome! This program converts your currency in US Dollar to Euro, or vice versa." <<
endl;

        // open data file
        myIn.open("../datafile");
        assert(myIn);

        ReadData(myIn, amount, currency);

        // compute the converted amount based on the currency type provided
        if (currency=='d')
                convertedAmount = amount * DOLLAR_TO_EURO;
        else
                convertedAmount = amount / DOLLAR_TO_EURO;

        DisplayResults(amount, convertedAmout, currency);

     return 0;
}
```

//Define each user-defined functions below:

Value returning function
- A value is explicitly returned using "return" statement
  The value can be of any C++ data type: char, int, float, bool, string
  The return type in the function header and function declaration should correspond to the type
  of the value returned
- A value returning function is activated/called within an expression.
  (The value returned will be used in evaluating the expression. Often, the function
activation/call is an expression by itself.)

<u>Example 1 – revisited</u> : **Write a value-returning function** to convert the currency from US Dollar to
<u>EURO, or vice versa.</u>

// Declare the user-defined functions here

const float DOLLAR_TO_EURO = 0.87;   // currency conversion rate

int main( )
{
       float   amount;   // amount entered by the user
       float   convertedAmount;   // amount converted to the second currency
       char   currency;   // currency type of the amount entered by the user

       DisplayWelcome();

       GetUserInput(amount, currency);

       convertedAmount = Convert(amount, currency);

       DisplayResults(amount, convertedAmount, currency);

      return 0;
}

//Define each user-defined functions below:

Practice Questions: Write a C++ value returning function that
1. takes the length of the two sides of a right triangle, and computes and returns the perimeter of the triangle
2. receives a floating-point number and returns the fractional part of that number. For example, if the incoming value of x is 16.753, the function returns the value 0.753.
3. returns the smallest of three integer parameters.
4. determines whether a character entered is an alpha numeric character. Returns true if it is an alpha numeric character, returns false otherwise.
5. determines whether an integer value is a prime number. Returns true if it is a prime number, returns false if it is not.
6. Determines whether an integer value is a perfect number or not

Questions:

o When to use value-returning function, and when to use void function?

o Can I use reference parameter with value returning function?

o Can I use a mixture of parameters passed by value and parameters passed by reference?

o Is file stream (ifstream or ofstream parameters) always passed by reference?