Basic Computer Programming and Networking

Functions

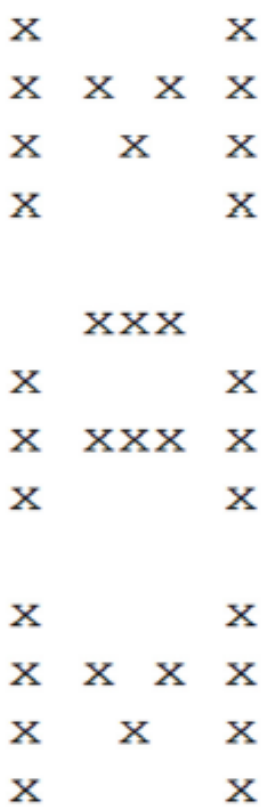
RHNS Jayathissa

Department of Computer Engineering

Faculty Of Computing

Exercise

- Create a C++ program to display the following *wordArt.(MAM)*



Solution

```
int main()
{
    cout<<"  x     x "<<endl;
    cout<<"  x x x x "<<endl;
    cout<<"  x  x  x "<<endl;
    cout<<"  x     x "<<endl;
    cout<<endl;
    cout<<"    xxx   "<<endl;
    cout<<"  x     x "<<endl;
    cout<<"  x xxx x "<<endl;
    cout<<"  x     x "<<endl;
    cout<<endl;
    cout<<"  x     x "<<endl;
    cout<<"  x x x x "<<endl;
    cout<<"  x  x  x "<<endl;
    cout<<"  x     x "<<endl;
    return 0;
}
```

Functions

- Functions are a basic building block for writing C/C++ programs.
- Breaking a program up into separate functions, each of which performs a particular task, makes it easier to develop and debug a program
- Solution for code reuse
- Functions allow for breaking down the program into discrete units

Function

- A function is a group of statements that together perform a task
- ***Example***
  - Every C++ program has at least one function, which is **main()**

```
int main()
{
    cout << "Hello world!" << endl;
    return 0;
}
```

- A C++ function is a simply a chunk of C++ code that has
  - ***A descriptive function name***
  - ***Parameters (Optional)***
  - ***A returning value***

Function

- A function is a block of code which only runs when it is called.
- You can pass data, known as parameters, into a function.
- Functions are used to perform certain actions, and they are important for reusing code: Define the code once, and use it many times.
- To create (often referred to as declare) a function, specify the name of the function, followed by parentheses ().

Functions

- Two types:
  - Standard library functions
  - User defined functions
- Two varieties
  - Those with return values
    - Produces a value that can be assigned to a variable
  - Those with none

Standard library functions

- is a collection of classes and functions, which are written in the core language and part of the C++ ISO Standard itself
- **Example**
  - **printf** and **scanf** to do input and output all three are functions.

Standard library functions

1. **stdio.h: I/O functions:**
    a. **getchar()** returns the next character typed on the keyboard.
    b. **putchar()** outputs a single character to the screen.
    c. **printf()** as previously described
    d. **scanf()** as previously described
2. **string.h: String functions**
    a. **strcat()** concatenates a copy of str2 to str1
    b. **strcmp()** compares two strings
    c. **strcpy()** copys contents of str2 to str1
3. **ctype.h: Character functions**
    a. **isdigit()** returns non-0 if arg is digit 0 to 9
    b. **isalpha()** returns non-0 if arg is a letter of the alphabet
    c. **isalnum()** returns non-0 if arg is a letter or digit
    d. **islower()** returns non-0 if arg is lowercase letter
    e. **isupper()** returns non-0 if arg is uppercase letter

Standard library functions

4. **math.h: Mathematics functions**
    a. **acos()** returns arc cosine of arg
    b. **asin()** returns arc sine of arg
    c. **atan()** returns arc tangent of arg
    d. **cos()** returns cosine of arg
    e. **exp()** returns natural logarithim e
    f. **fabs()** returns absolute value of num
    g. **sqrt()** returns square root of num
5. **time.h: Time and Date functions**
    a. **time()** returns current calender time of system
    b. **difftime()** returns difference in secs between two times
    c. **clock()** returns number of system clock cycles since program execution
6. **stdlib.h:Miscellaneous functions**
    a. **malloc()** provides dynamic memory allocation, covered in future sections
    b. **rand()** as already described previously
    c. **srand()** used to set the starting point for rand()

Example

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

int main()
{
    cout << "This is a sample program" << endl;
    getch();
    system("cls");
    cout << "OK" << endl;

    return 0;
}
```

Steps to define a function

- Return type of the function (void / other)
  - No return type (void) or add some…
- Function Name
  - Add meaningful name
- Arguments
  - No argument () or some…
- Example
  - Write a function to display some message

Syntax

```
void myFunction() {
  // code to be executed
}
```

myFunction() is the name of the function

void means that the function does not have a return value. You will learn more about return values later in the next slides

inside the function (the body), add code that defines what the function should do

Call a Function

- Declared functions are not executed immediately. They are "saved for later use", and will be executed later, when they are called.
- To call a function, write the function's name followed by two parentheses () and a semicolon ;
- In the following example, myFunction() is used to print a text (the action), when it is called:

Example

Inside `main`, call `myFunction()`:

```
// Create a function
void myFunction() {
  cout << "I just got executed!";
}

int main() {
  myFunction(); // call the function
  return 0;
}

// Outputs "I just got executed!"
```

A function can be called multiple times:

Example

```
void myFunction() {
  cout << "I just got executed!\n";
}

int main() {
  myFunction();
  myFunction();
  myFunction();
  return 0;
}

// I just got executed!
// I just got executed!
// I just got executed!
```

Function Declaration and Definition

A C++ function consist of two parts:

Declaration: the return type, the name of the function, and parameters (if any)

Definition: the body of the function (code to be executed)

```
void myFunction() { // declaration
  // the body of the function (definition)
}
```

User defined Functions

- To use a function, you must
  - Provide a function prototype (before Main)

```
void printMe();
```

  - Provide a function definition

```
Void printMe()
{
        cout <<"Print function";
}
```

  - Call the function

```
printme();
```

Example

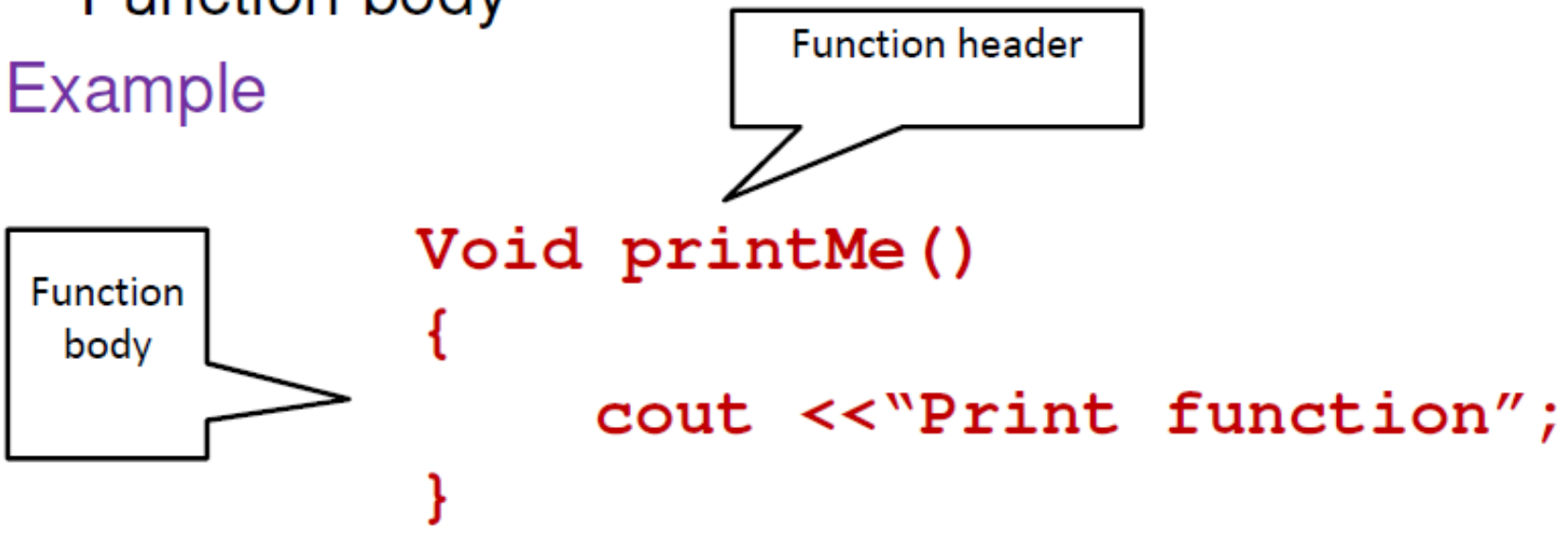
```
#include <iostream>

using namespace std;

// Function Prototype
void printMe();

int main()
{
    printMe();
    return 0;
}
// Function Definition
void printMe()
{
    cout<< "This is a Function";
}
```

Function Prototype

- A function prototype tells you all you need to write a call to the function.
  - What types are involved
  - Describes the function interface
    - The name
    - How many arguments the function needs
    - What type the arguments should be
- Example
  - `void printMe();`

Function Definition

- Includes the code for the function workings
- A function definition consists of
  - Function header
  - Function body
- Example



Example 1

```
using namespace std;
// Function Prototype
void printMe();

int main()
{
    printMe();
    return 0;
}
// Function Definition
void printMe()
{
    cout<< "This is a Function";
}
```

Parameters and Arguments

Information can be passed to functions as a parameter. Parameters act as variables inside the function.

Parameters are specified after the function name, inside the parentheses. You can add as many parameters as you want, just separate them with a comma:

Syntax

```
void functionName(parameter1, parameter2, parameter3) {
  // code to be executed
}
```

Example

```
void myFunction(string fname) {
  cout << fname << " Refsnes\n";
}

int main() {
  myFunction("Liam");
  myFunction("Jenny");
  myFunction("Anja");
  return 0;
}

// Liam Refsnes
// Jenny Refsnes
// Anja Refsnes
```

The following example has a function that takes a string called fname as parameter.

When the function is called, we pass along a first name, which is used inside the function to print the full name.

Example 2

```
#include <iostream>

using namespace std;
// Function Prototype
void printAge(int age);

int main()
{
    printName(23);
    return 0;
}
// Function Definition
void printAge(int age)
{
    cout<< "Age is "<< age;
}
```

Function Prototype

In C++, the code of function declaration should be before the function call. However, if we want to define a function after the function call, we need to use the function prototype. For example,

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

int main() {
    // calling the function before declaration
    add(5, 3);
    return 0;
}

// function definition
void add(int a, int b) {
    cout << (a + b);
}
```

Function Prototype

```
void add(int, int);
```

Example 3

```
#include <iostream>
#include <string>

using namespace std;
// Function Prototype
void printMe(string name,int age);

int main()
{
    string name = "saman";
    int age = 23;
    printMe(name, age);
    return 0;
}
// Function Definition
void printMe(string name, int age)
{
    cout<< "Name is "<< name;
    cout<< "Age is " << age;
}
```

Example 4

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

void calculateSum(int v1, int v2);

int main()
{
    calculateSum(12, 34);
    return 0;
}
// Function Definition
void calculateSum(int v1, int v2)
{
    int total;
    total = v1 + v2;
    cout<< "Sum is " + total;
}
```

Example 2: Function with Parameters

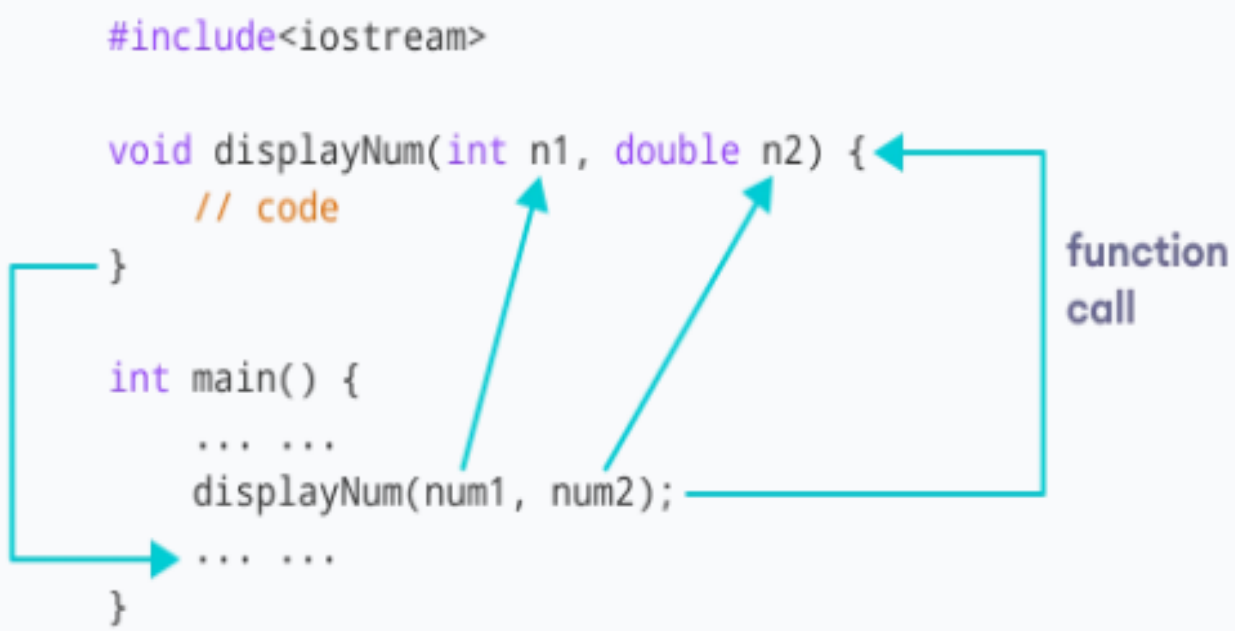
```
// program to print a text

#include <iostream>
using namespace std;

// display a number
void displayNum(int n1, float n2) {
    cout << "The int number is " << n1;
    cout << "The double number is " << n2;
}

int main() {

     int num1 = 5;
     double num2 = 5.5;

    // calling the function
    displayNum(num1, num2);

    return 0;
}
```

Output

```
The int number is 5
The double number is 5.5
```

In the above program, we have used a function that has one `int` parameter and one `double` parameter.

We then pass `num1` and `num2` as arguments. These values are stored by the function parameters `n1` and `n2` respectively.



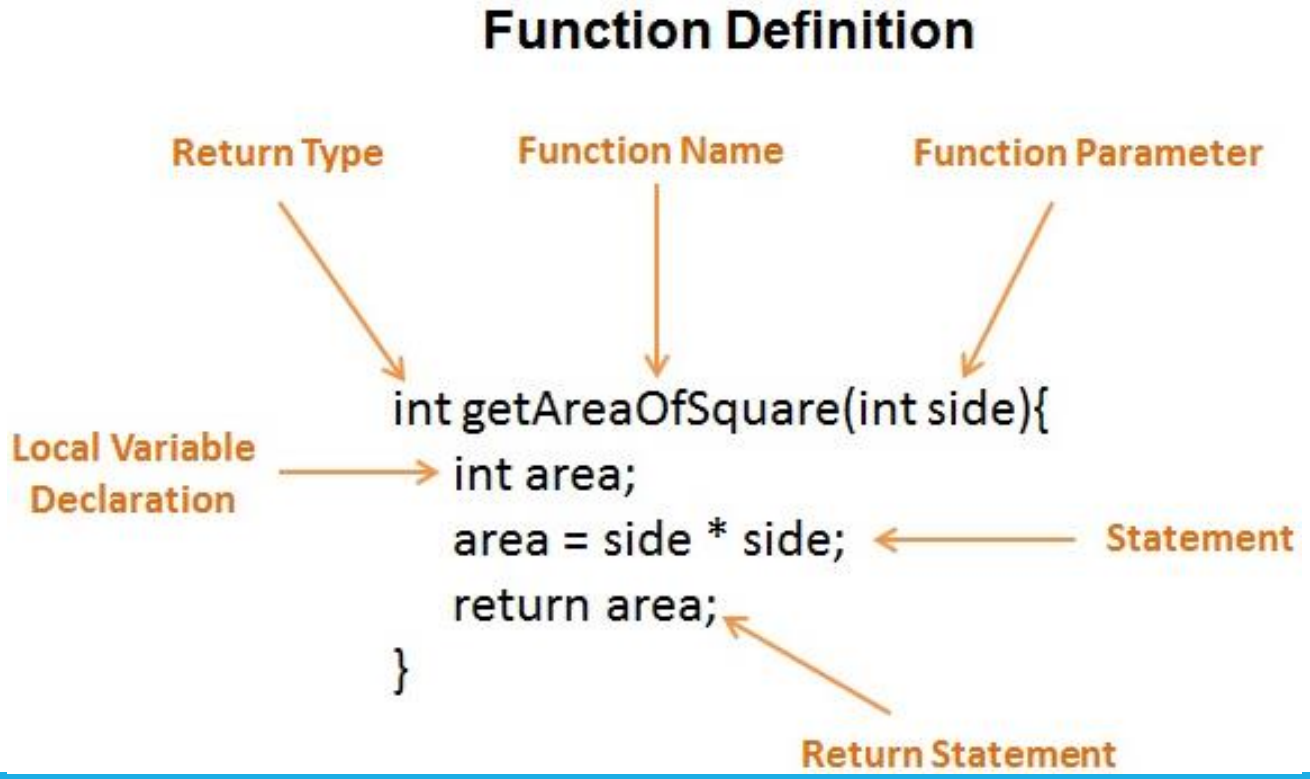
Return Function

- If a function returns a value, it must have a return statement that specifies the value to return.
- Example

```
int getSum(int v1, int v2)
{
    int tot;
    tot = v1 + v2;
    return tot;

}
```

- The result of a function is called its return value and **the data type of the return value** is called the return type.

- Every function declaration and definition must specify a return type, whether or not it actually returns a value.



Example 3: Add Two Numbers

```
// program to add two numbers using a function

#include <iostream>

using namespace std;

// declaring a function
int add(int a, int b) {
    return (a + b);
}

int main() {

    int sum;

    // calling the function and storing
    // the returned value in sum
    sum = add(100, 78);

    cout << "100 + 78 = " << sum << endl;

    return 0;
}
```

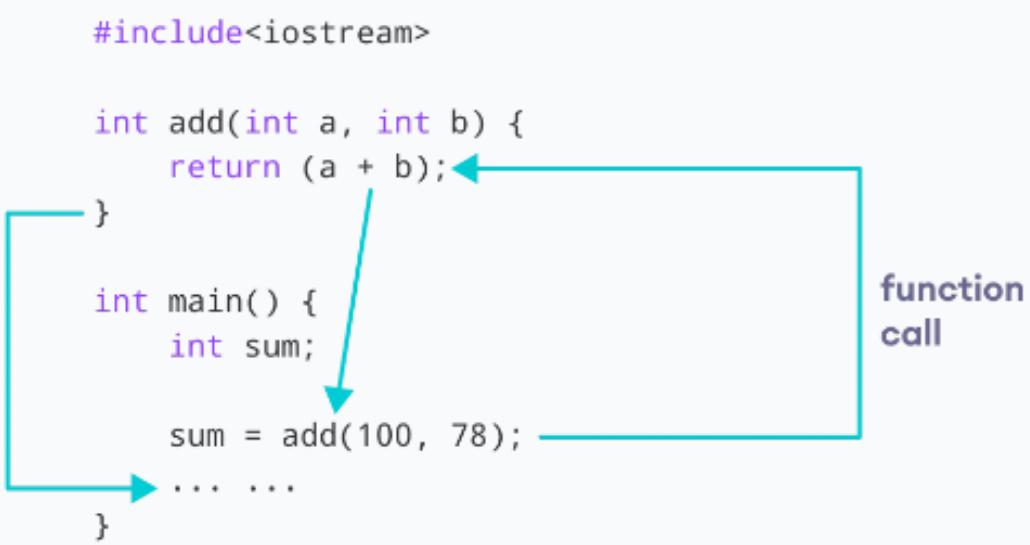
Output

```
100 + 78 = 178
```

In the above program, the `add()` function is used to find the sum of two numbers.

We pass two `int` literals `100` and `78` while calling the function.

We store the returned value of the function in the variable `sum`, and then we print it.



Working of C++ Function with return statement

Notice that `sum` is a variable of `int` type. This is because the return value of `add()` is of `int` type.

Thank You