

Functions

- Functional Decomposition with Void Functions
- Functional Decomposition with User-Defined Functions
- Flow of Control in Function Calls
- Function Parameters
- Function Calls
- Function Declarations and Definitions

Functional Decomposition with void Functions

- A **void** function does not return a function value, nor is it called from within an expression.
- Any module can be coded as a function, it's generally done to make the program easier to understand.
- A void function looks like the **main** function except that the function heading uses **void** rather than **int** as the data type of the function and there is no return statement.

Example

```

Main
  Print two lines of asterisks
  Print "Welcome Home"
  Print four lines of asterisks
Print 2 Lines
  Print "*****"
  Print "*****"
Print 4 Lines
  Print "*****"
  Print "*****"
  Print "*****"
  Print "*****"
  
```

Example Implementation

```

//*****
// Welcome program
// This program prints a "Welcome Home" message
//*****

#include <iostream>
using namespace std;

void Print2Lines();
void Print4Lines();
  
```

Example Implementation (Continued)

```

int main()
{
    Print2Lines();
    cout << "Welcome Home!" << endl;
    Print4Lines();
    return 0;
}

void Print2Lines()
{
    cout << "*****" << endl;
    cout << "*****" << endl;
}
  
```

Example Implementation (Continued)

```

void Print4Lines()
{
    cout << "*****" << endl;
    cout << "*****" << endl;
    cout << "*****" << endl;
    cout << "*****" << endl;
}
  
```

An Overview of User-Defined Functions

- When a function call is encountered, logical control is passed to the first statement in that function's body. After the last statement of a function is executed, control returns to the point immediately following the function call.
- Functions can have parameters, we could have written one function which has an input parameter which tells it how many lines of asterisks to print.

PrintLine Function with Input Parameter

```
void PrintLines( int numLines )

// This function prints lines of asterisks, where
// numLines specifies how many lines to print

{
    int count;        // Loop control variable
    count = 1;
    while (count <= numLines)
    {
        cout << "*****" << endl;
        count++;
    }
}
```

Parameters and Arguments

- The items listed in the call to a function are the arguments.
- The variables declared in the function heading are the parameters.

Syntax and Semantics of void Functions

- A function call in a program results in the execution of the body of the called function.
- Syntax Template
 - FunctionCall (to a void function)


```
FunctionName ( ArgumentList );
```
- The parentheses are required even if the argument list is empty. Multiple arguments must be separated by commas.
 - ArgumentList


```
Expression , Expression ..
```

Function Declarations and Definitions

- In C++, you must declare every identifier before it can be used. A function's declaration must physically precede any function call.
- Example from welcome program


```
#include <iostream>
using namespace std;
void Print2Lines();
void Print4Lines();
```
- A function declaration announces to the compiler the name of the function, the data type of the function's return value, and the data type of the parameters it uses.

Function Declarations (Prototypes) and Definitions

- FunctionPrototype (void function)


```
void FunctionName (ParameterList);
```
- ParameterList (in a function prototype)


```
DataType & VariableName , DataType & VariableName ...
```
- FunctionDefinition (void function)


```
void FunctionName (ParameterList)
{
    Statement
    :
}
```
- ParameterList (in a function definition)


```
DataType & VariableName , DataType & VariableName
:

```

Modified Example

```

//*****
// Welcome program
//*****
#include <iostream>
using namespace std;

void PrintLines(int);    function prototype
int main()
{
    PrintLines(2);
    cout << "Welcome Home!" << endl;
    PrintLines(4);
    return 0;
}

```

Modified Example
(continued)

```

void PrintLines( int numLines )

// This function prints lines of asterisks, where
// numLines specifies how many lines to print

{
    int count;    // Loop control variable
    count = 1;
    while (count <= numLines)
    {
        cout << "*****" << endl;
        count++;
    }
}

```

Modified Example

```

//*****
// Welcome program
//*****
#include <iostream>
using namespace std;

void PrintLines(int);  function prototype variable list
int main()
{
    PrintLines(2);
    cout << "Welcome Home!" << endl;
    PrintLines(4);
    return 0;
}

```

Modified Example
(continued)

```

void PrintLines( int numLines )

// This function prints lines of asterisks, where
// numLines specifies how many lines to print

{
    int count;    // Loop control variable
    count = 1;
    while (count <= numLines)
    {
        cout << "*****" << endl;
        count++;
    }
}

```

Modified Example

```

//*****
// Welcome program
//*****
#include <iostream>
using namespace std;

void PrintLines(int);
int main()
{
    PrintLines(2);
    cout << "Welcome Home!" << endl;
    PrintLines(4);
    return 0;
}

```

Modified Example
(continued)

```

void PrintLines( int numLines )    function definition

// This function prints lines of asterisks, where
// numLines specifies how many lines to print

{
    int count;    // Loop control variable
    count = 1;
    while (count <= numLines)
    {
        cout << "*****" << endl;
        count++;
    }
}

```

Modified Example

```

//*****
// Welcome program
//*****
#include <iostream>
using namespace std;

void PrintLines(int);
int main()
{
    PrintLines(2);
    cout << "Welcome Home!" << endl;
    PrintLines(4);
    return 0;
}

```

Modified Example
(continued)

```

void PrintLines( int numLines )    function definition
                                   parameter list
// This function prints lines of asterisks, where
// numLines specifies how many lines to print

{
    int count;    // Loop control variable
    count = 1;
    while (count <= numLines)
    {
        cout << "*****" << endl;
        count++;
    }
}

```

Modified Example

```

//*****
// Welcome program
//*****
#include <iostream>
using namespace std;

void PrintLines(int);
int main()
{
    PrintLines(2);           function call
    cout << "Welcome Home!" << endl;
    PrintLines(4);           function call
    return 0;
}

```

Modified Example
(continued)

```

void PrintLines( int numLines )

// This function prints lines of asterisks, where
// numLines specifies how many lines to print

{
    int count;    // Loop control variable
    count = 1;
    while (count <= numLines)
    {
        cout << "*****" << endl;
        count++;
    }
}

```