GLS UNIVERSITY

UNIT – II

Functions in C++

- A function is a group of statements that together perform a task. Every C++ program has at least one function, which is main().
- If a problem involves lengthy and complex algorithms the length of the main() may increase to an extent where keeping track of the steps involved may become difficult task.
- In this case if problem is divided into number of logical units called function.
- **Each subprogram in C++ is called a function.**
- **A function is a group of statements that is given a name, and which can be called from some point of the program.**

Advantages of Functions

You can run one function many times. You can call a function to execute same lines of code multiple times without **re-writing** it.

In case of any **modification** in the code you can modify only the function without changing the structure of the program.

Saves the **memory space** when same code is used multiple times. Organize the program and divide into number of **subprograms** so that seperate **module** can be used whenever required.
Developmentof real life applications needs more than one programmer when project size is larger.

Project is divided into number of modules among programmer that builds **team work.**

Errrors can be easily identified so **testing and debugging becomes easy.**

Function Declaration and Calling

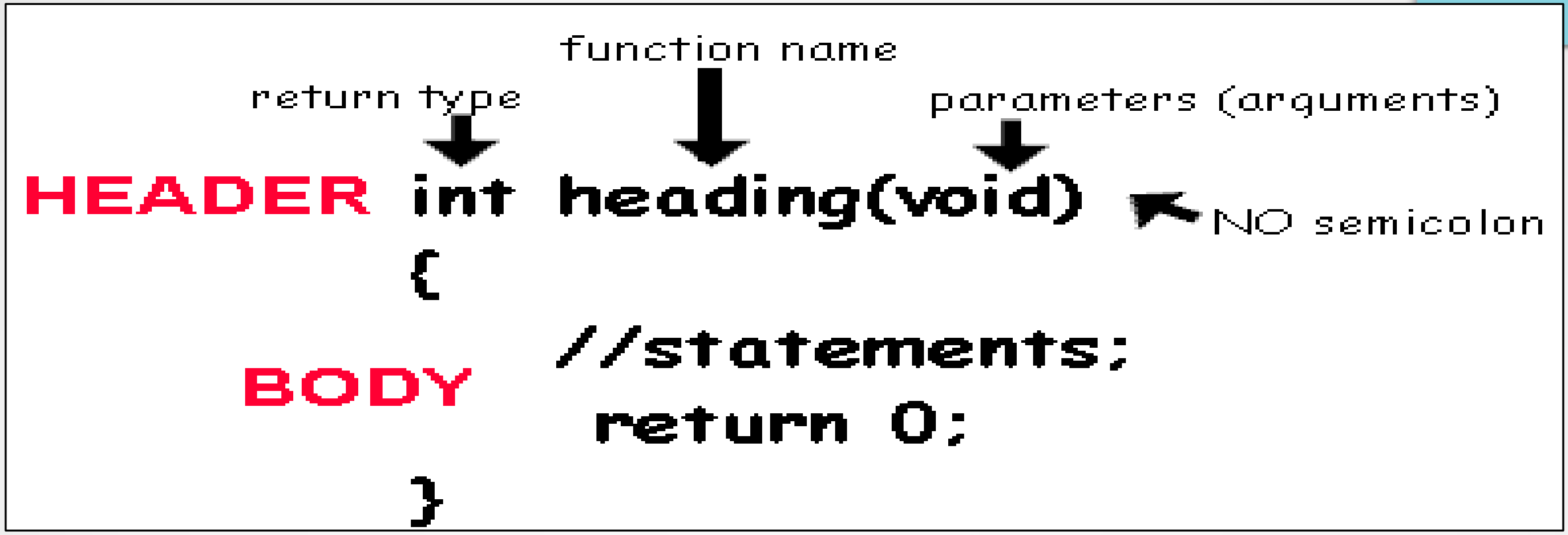
- A function declaration tells the compiler about a function's name, return type, and parameters.
- A function definition provides the actual body of the function.

  Syntax functi on definition:

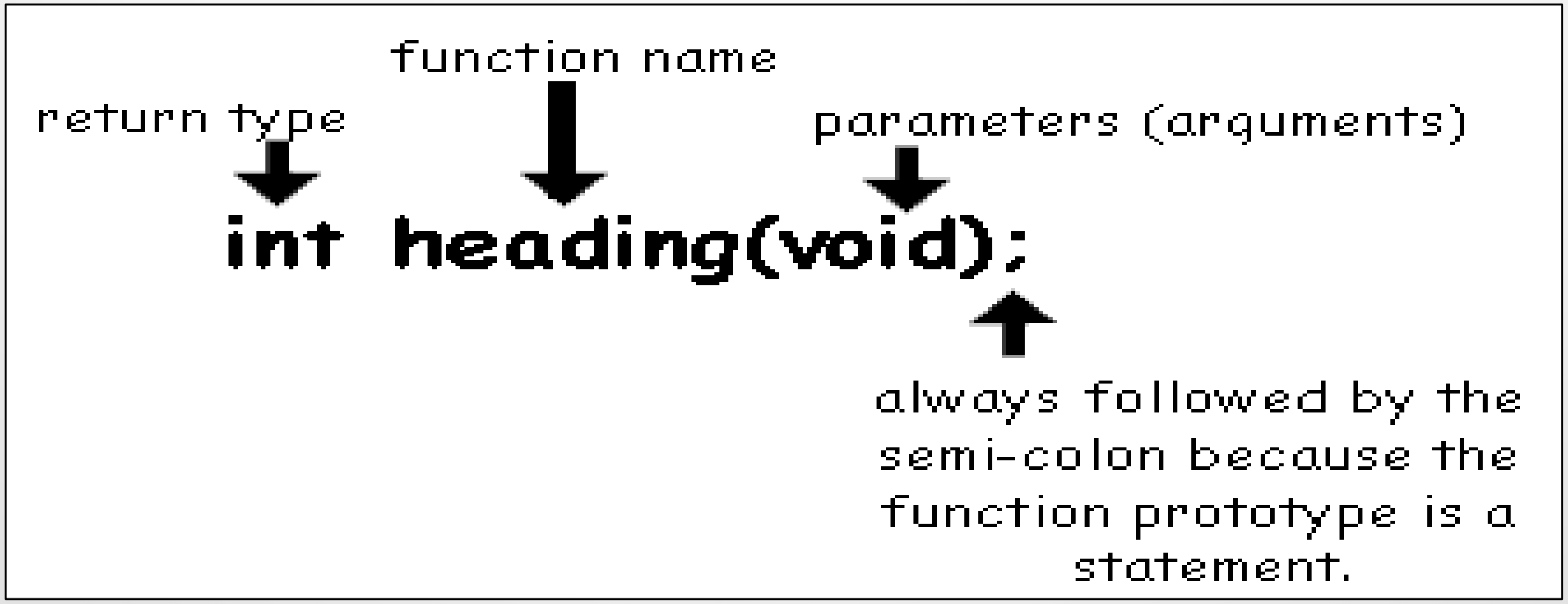
  The first line consists of return_type, function_name,

  parameter list/arguments referred to as **function header.**

Defining function-1, Declaring function-2



```
int heading(void)
{
    //statements;
    return 0;
}
```


function name
return type
parameters (arguments)
always followed by the semi-colon because the function prototype is a statement.

```
int heading(void);
```

Return Type: A function may return a value. The return_type is the data type of the value the function returns.

Some functions perform the desired operations without returning a value. In this case, the return_type is the keyword void.

Function Name: This is the actual name of the function.

The function name and the parameter list together constitute the function signature.

Parameters are optional; that is, a function may contain no parameters.

Function Body: The function body contains a collection of statements that define what the function does.

Function Declarations:

- A function declaration tells the compiler about a function name and how to call the function.
  The actual body of the function can be defined separately.
- A function declaration has the following parts:
  return_type function_name(parameter list);
- **Example:**
  int max(int num1, int num2);
  int max(int, int);

- A function doesnot execute by itself, it is to be invoked by another function which can be main() or any other function.
- **A function which invokes another function is called calling function and the function which is invoked is termed as called function.**

Calling a Function:

When a program calls a function, program control is transferred to the called function. A called function performs defined task and when it’s return statement is executed or when its function-ending closing brace is reached, it returns program control back to the main program.

To call a function, you simply need to pass the required parameters along with function name, and if function returns a value, then you can store returned value.

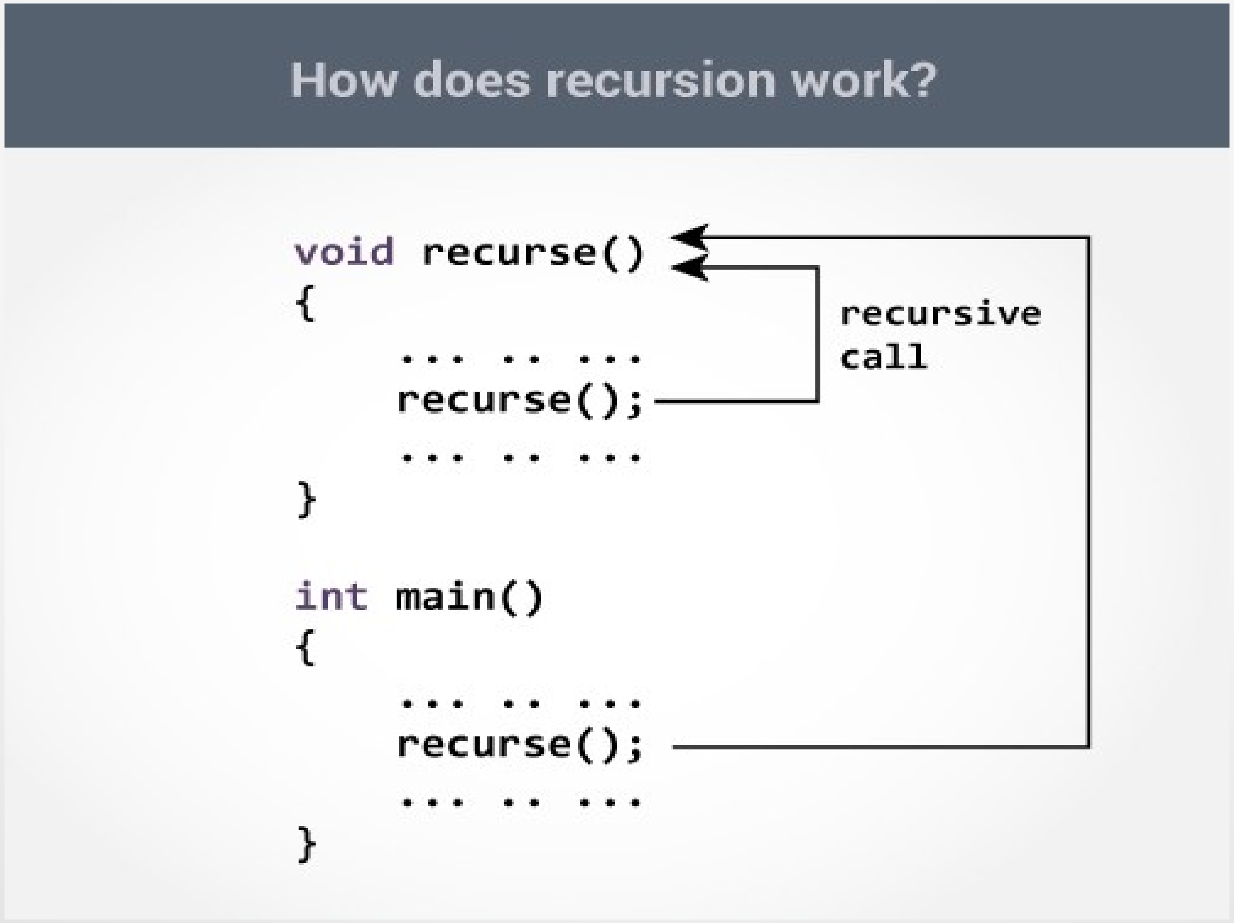
Classification of Functions

- Functions are categorised into two types:
    - Built-in functions
    - User-define functions
-
- Built in functions are those which are already made available as part of C++ library.
- User defined function is written by the user to solve a problem. It gives flexibility in developing the code.
- Now we are going to develop our own functions to accomplish the tasks which we come across.
- Functions are classified into four categories:

1. **Function with no argument and no return value:** This type of function will not send any data and also not return any data to its calling place in the program.
2. **Function with argument but no return value:** This type of function will send data as arguments but it will not return any kind of data when it will call.
3. **Function with no argument but return value:** This type of function will not send any data but it willreturn some type of data when it will call. And it will return data to its calling place from where it is being called in the program
4. **Function with argument and return value:** This type of function will send and also return bunch of data toits calling place in the program. From where it’s being used to call for do something.

Recursion

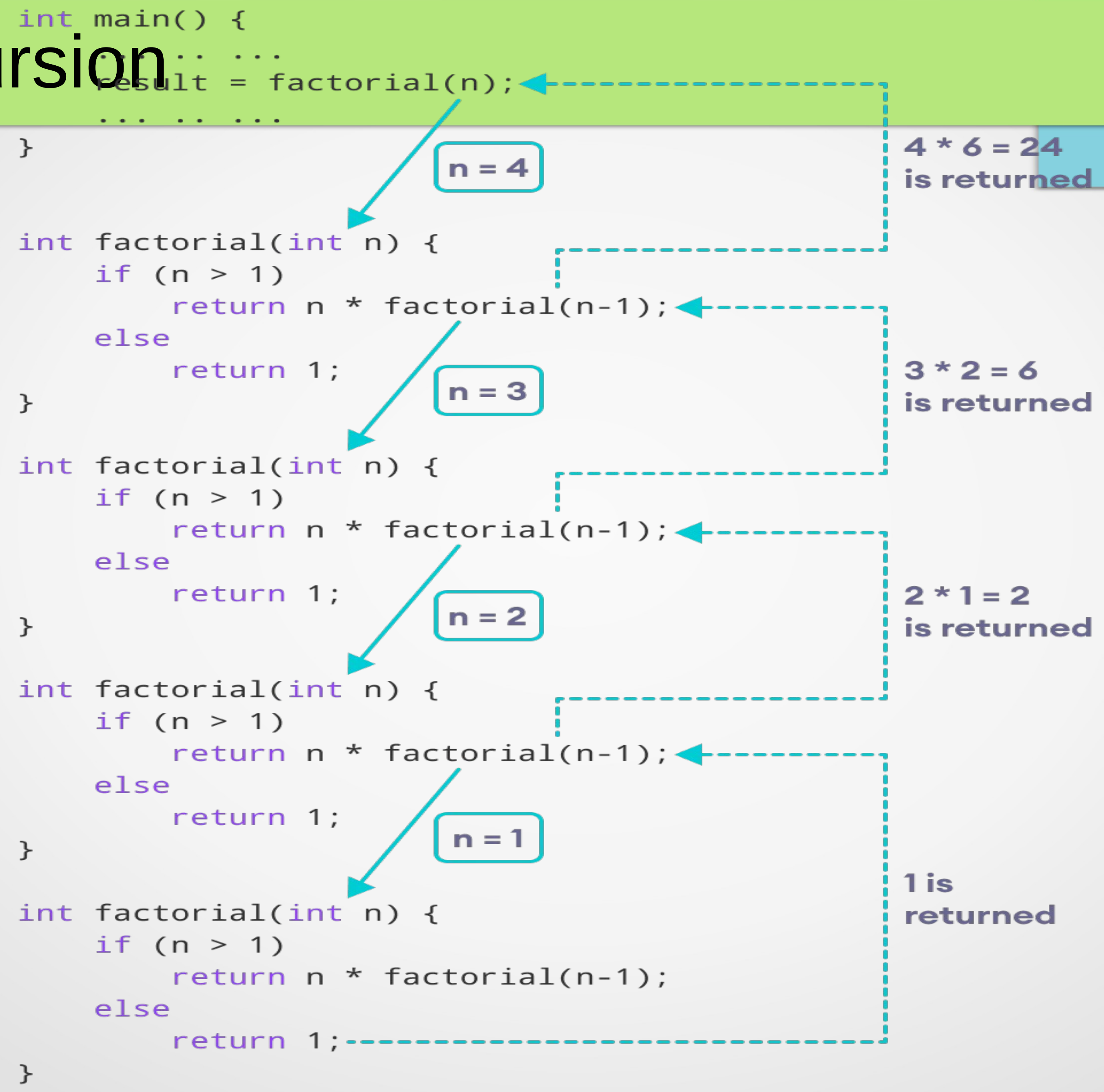
- Recursion is the technique of making a function call itself. This technique provides a way to break complicated problems down into simple problems which are easier to solve.



```
void recurse()
{
    ... .. ...
    recurse();
    ... .. ...
}

int main()
{
    ... .. ...
    recurse();
    ... .. ...
}
```

Recursion



```
int main() {
    ... .. ...
    result = factorial(n);
    ... .. ...
}

int factorial(int n) {
    if (n > 1)
        return n * factorial(n-1);
    else
        return 1;
}

int factorial(int n) {
    if (n > 1)
        return n * factorial(n-1);
    else
        return 1;
}

int factorial(int n) {
    if (n > 1)
        return n * factorial(n-1);
    else
        return 1;
}

int factorial(int n) {
    if (n > 1)
        return n * factorial(n-1);
    else
        return 1;
}
```

Inline Function

C++ provides an inline functions to reduce the function call overhead.

When the inline function is called whole code of the inline **function gets inserted or substituted** at the point of inline function call.

This substitution is performed by the **C++ compiler** at compile time.

Inline function may **increase efficiency if it is small.**

Inline Function

Compiler may not perform inlining in such circumstances like:

1) If a function contains a loop. (for, while, do-while)
2) If a function contains static variables.
3) If a function is recursive.
4) If a function contains switch or goto statement.

Syntax:

```
inline return-type function-name(parameters)
{
   // function code
}
```

Default Argument

- The idea behind default argument is very simple.
- If a function is called by passing argument/s, those arguments
- are used bythe function.

Default Argument

- Default value/s are passed to argument/s in function prototype.
- A default parameter (also called an optional parameter or a default argument) is a function parameter that has a default value provided to it.
- If the user does not supply a value for this parameter, the default value will be used.
- sum (int x=0,int y); // Incorrect
- If you default an argument, then you will have to default all the ***subsequent*** arguments after that.

  sum (int x,int y=0);

  sum (int x,int y=0,int z); // This is incorrect

  sum (int x,int y=10,int z=10);// Correct
- You can give any value a default value to argument, compatible with its datatype.

Function Overloding

- If any class have multiple functions with same names but different parameters then they are said to be overloaded.
- Function overloading is usually used to enhance the readability of the program.
- If you have to perform one single operation but with **different number or types of arguments,** then you can simply overload the function.
- Function overloading can be considered as an example of polymorphism feature in C++.
- By changing number of Arguments.
- By having different types of Arguments.