DEFAULT ARGUMENT :-

What are default arguments?

A default arguments is a predetermined value in a function's declaration. Without a particular argument for that parameter, the function uses the default setting whenever it is called. The default value, however, is overridden if a value is given when calling the function.

Example

```
#include <iostream>
using namespace std;
void sum(int x, int y = 10, int z = 20)
{
   cout << (x + y + z); // returns the sum of x, y, z
}

int main()
{
   sum(10); // outputs 40 as x=10, y=10 and z=20
   sum(10, 20, 30); // outputs 60 as x=10, y=20, z=30
}
```

Explanation

- The 'x', 'y', and 'z' parameters of the sum function have default values of 'y' and 'z,' respectively.

- We use two different sets of arguments to call the sum function twice in the main function.

- When we call sum(10) with just one input, 'x' is set to 10, and 'y' and 'z' are both set to their default values (10 and 20, respectively), producing an output of 40.

- When calling sum(10, 20, 30) with all three arguments supplied, the values of 'x', 'y', and 'z' are set to 10, 20, and 30, respectively, and the output is 60.

Characteristics of Default Argument in C++

Characteristics of default arguments in C++ are:

- The default argument values are not final; they can be changed during runtime.
- The function call involves copying the values in a left-to-right sequence.
- All the values with default value will be on the right.
- Default arguments enable flexibility by allowing functions to be called with fewer arguments.

Rules of Default Arguments in C++

Rules of default arguments in C++ are:

1. Argumentation

When defining a function, default values must come after parameters without default values.

```
// Correct: Non-defaulted parameter 'x' comes before defaulted parameter 'y'.
void e1(int x, int y = 5) {
   // Function implementation
}
// Incorrect: Defaulted parameter 'y' comes before non-defaulted parameter 'x'.
void e2(int y = 5, int x) {
   // Invalid function definition
}
```

2. Matching Declarations

When the function definition and function declaration don't match, function prototypes must utilise default values.

```
// Function declaration (prototype) with default argument values.
void print_Info(int a, int b = 5, int c = 10);

// Function definition with the same default arguments.
void print_Info(int a, int b, int c) {
   // Function implementation
}
```

3. Unique Overloads

A function with default arguments may be overloaded, but each overload must have a different set of parameters.

```
// Overloaded functions with unique parameter lists.
void overloadedEg(int x, int y = 10);
void overloadedEg(int x, double y = 3.14);
void overloadedEg(char c);
// Correct usage with unique parameter combinations.
overloadedEg(5);          // Calls int version with default 'y'.
overloadedEg(5, 3.14);    // Calls int/double version.
overloadedEg('A');        // Calls char version.
```

4. No Default for Reference Parameters

Since reference parameters must always connect to a proper object, they cannot have default arguments.

```
// Incorrect: Reference parameter 'ref' cannot have a default argument.
void invalidEg(int& ref = someValue) {
   // Invalid function definition
}
// Correct usage of reference parameters without default arguments.
void validEg(int& ref) {
   // Function implementation
}
```

Working of Default Argument in C++

In C++, default arguments allow you provide default settings for function parameter values. When a caller does not supply a value for a parameter in a function call, these default values are applied. Here is how C++ handles default arguments:

1. Function Declaration

In the function declaration (prototype), which is generally in the header file, you define the default arguments. The parameter list contains assignments that serve as default parameters.

void print(int a, int b = 4, int c = 6);
In this example, the print function has three parameters, but b and c have default values of 4 and 6, respectively.

2. Function Definition

You don't need to give default values once more in the function definition (often in the source file). Instead, you provide the entire set of parameters without any default values.

```
void printInfo(int x, int y, int z) {
   // Function implementation
}
```

3. Function Calling

When calling the function, you can choose to provide values for some or all of the parameters. The default value is applied when a value for a parameter with a default argument is omitted.

```
int main() {
   print(5);       // 'x' is 5, 'y' uses default (10), 'z' uses default (20)
   print(5, 15);   // 'x' is 5, 'y' is 15, 'z' uses default (20)
   print(5, 15, 25); // 'x' is 5, 'y' is 15, 'z' is 25
   return 0;
}
```

4. Reduction of Function Overloading

Default arguments might lessen the requirement for Function Overloading. You can offer default values for optional parameters in a single function rather than having many overloaded functions with distinct parameter lists.

Advantages of default argument in C++

Some advantages of default argument in C++ are:-

- **Simplified Function Calls:** Function calls can be made more brief and understandable by using default arguments, which allow you to call a function with less arguments.

- **Reduced Function Overloading:** By allowing you to offer many iterations of a function with various default values for the same parameters, default arguments lessen the requirement for function overloading.

- **Backward Compatibility:** Functions that already exist can have default parameters added to them without causing problems for older code that calls them with less arguments.

- **Flexibility:** They give you the opportunity to define default values for parameters, which is useful when working with optional arguments in functions.

- **Better Code Organization:** By combining numerous iterations of a function into a single function with default values, default arguments can streamline code.

Disadvantages of default argument in C++

Some disadvantages of default argument in C++ are:-

- **Ambiguity:** When there are several overloaded functions with similar parameter lists, default arguments might cause ambiguity in function calls if they are not used wisely.

- **Maintenance Complexity:** Complex default values or an overuse of default parameters can make code more difficult to comprehend and maintain.

- **Reduced Explicitness:** Using default parameters could make it less obvious which arguments are required for a function, which could cause confusion in the code.

- **Limitated to the Right:** Default arguments must be provided in a right-to-left order, therefore you cannot have a default argument for a parameter if the parameter immediately to the right of it lacks a default value.

- **Reference Parameters:** Reference parameters, like int&, must always relate to genuine objects, hence they are unable to have default arguments. This restriction may prevent some coding patterns.