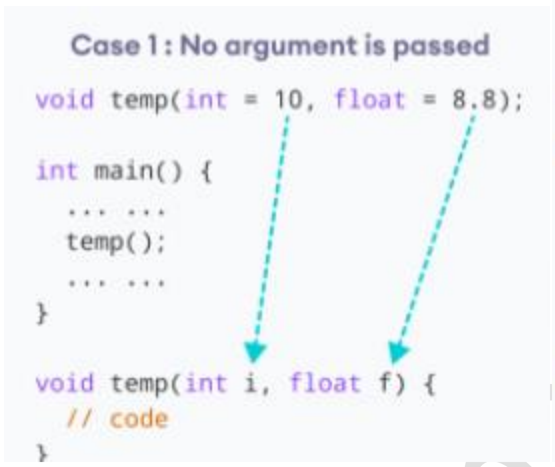
In C++ programming, we can provide default values for function parameters. If a function with default arguments is called without passing arguments, then the default parameters are used.

However, if arguments are passed while calling the function, the default arguments are ignored.

Working of default arguments

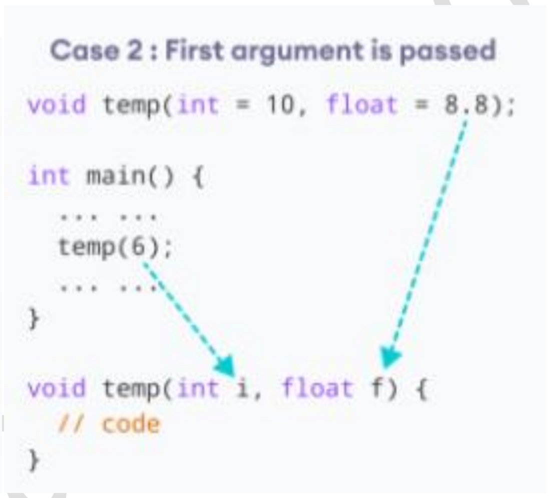


Case 1 : No argument is passed

```
void temp(int = 10, float = 8.8);

int main() {
  ... ...
  temp();
  ... ...
}

void temp(int i, float f) {
  // code
}
```

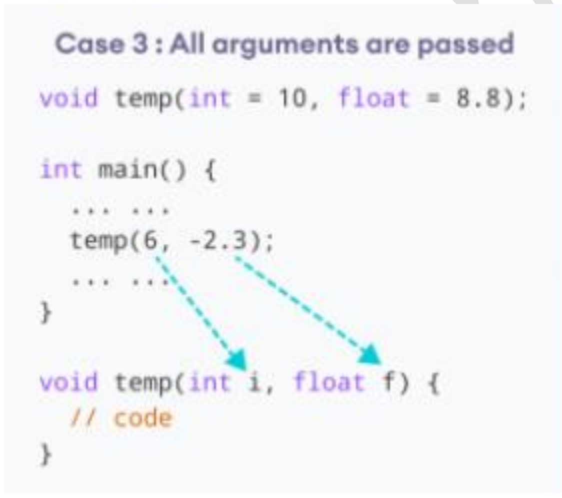


Case 2 : First argument is passed

```
void temp(int = 10, float = 8.8);

int main() {
  ... ...
  temp(6);
  ... ...
}

void temp(int i, float f) {
  // code
}
```

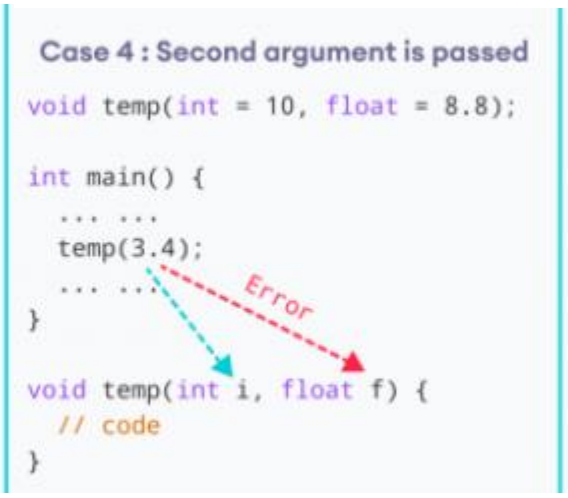


Case 3 : All arguments are passed

```
void temp(int = 10, float = 8.8);

int main() {
  ... ...
  temp(6, -2.3);
  ... ...
}

void temp(int i, float f) {
  // code
}
```



Case 4 : Second argument is passed

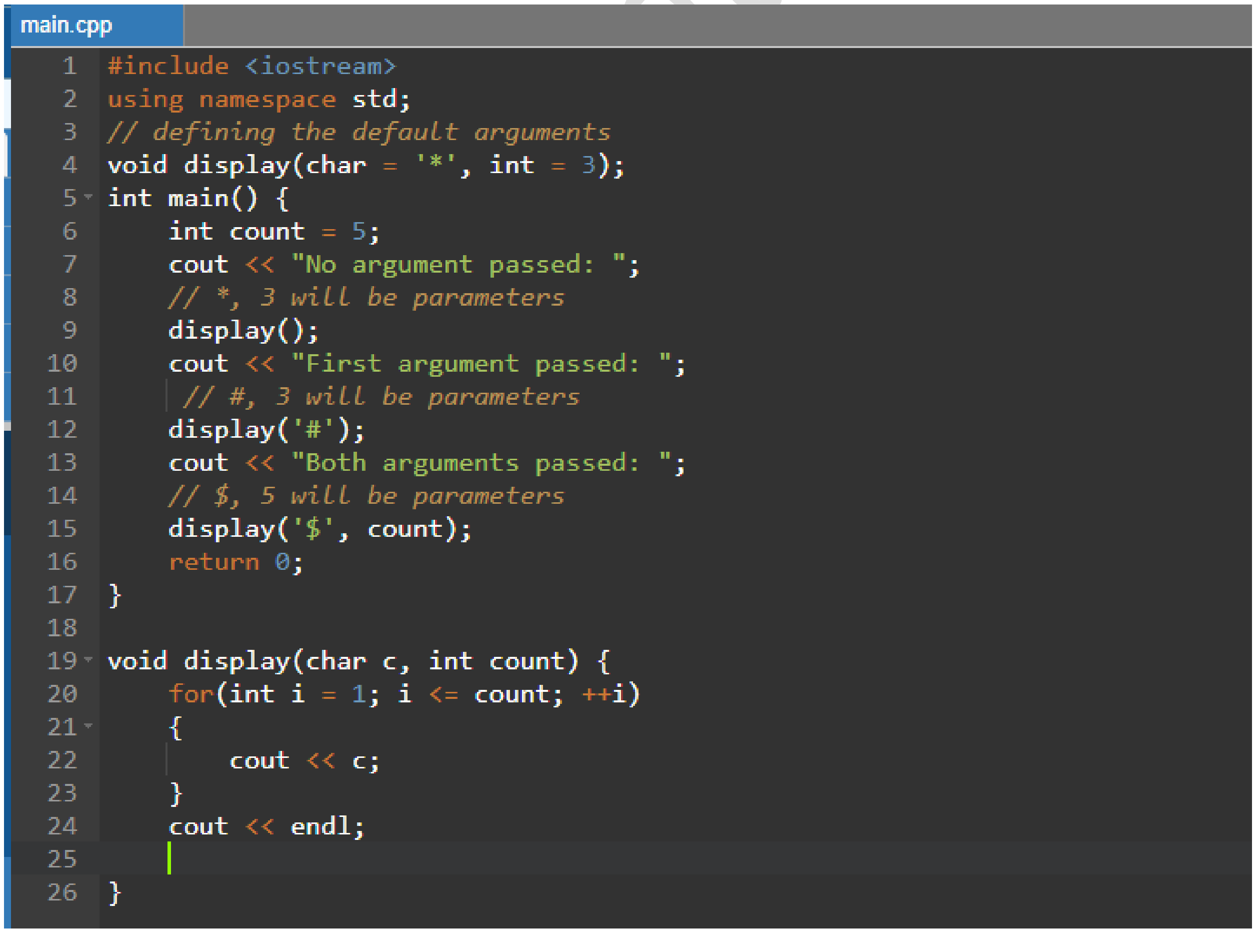
```
void temp(int = 10, float = 8.8);

int main() {
  ... ...
  temp(3.4);
  ... ...
}

void temp(int i, float f) {
  // code
}
```

We can understand the working of default arguments from the image above:

1. When `temp()` is called, both the default parameters are used by the function.
2. When `temp(6)` is called, the first argument becomes `6` while the default value is used for the second parameter.
3. When `temp(6, -2.3)` is called, both the default parameters are overridden, resulting in `i = 6` and `f = -2.3`.
4. When `temp(3.4)` is passed, the function behaves in an undesired way because the second argument cannot be passed without passing the first argument. Therefore, `3.4` is passed as the first argument. Since the first argument has been defined as `int`, the value that is actually passed is `3`.


main.cpp

```cpp
#include <iostream>
using namespace std;
// defining the default arguments
void display(char = '*', int = 3);
int main() {
    int count = 5;
    cout << "No argument passed: ";
    // *, 3 will be parameters
    display();
    cout << "First argument passed: ";
     // #, 3 will be parameters
    display('#');
    cout << "Both arguments passed: ";
    // $, 5 will be parameters
    display('$', count);
    return 0;
}

void display(char c, int count) {
    for(int i = 1; i <= count; ++i)
    {
        cout << c;
    }
    cout << endl;
    
}
```

Things to Remember

1. Once we provide a default value for a parameter, all subsequent parameters must also have default values. For example,

```
// Invalid
void add(int a, int b = 3, int c, int d);

// Invalid
void add(int a, int b = 3, int c, int d = 4);

// Valid
void add(int a, int c, int b = 3, int d = 4);
```

2. If we are defining the default arguments in the function definition instead of the function prototype, then the function must be defined before the function call.

```
// Invalid code

int main() {
    // function call
    display();
}

void display(char c = '*', int count = 5) {
    // code
}
```