

Function Templates

In this lesson, we will learn about function templates and their uses.

WE'LL COVER THE FOLLOWING



- Calling Function Template by Passing Arguments
- Instantiation
- Overloading

A function template is defined by placing the keyword `template` in front of the function template followed by type or non-type parameters.

- The keyword `class` or `typename` declares the parameters.
- The name T is usually used for the first parameter.
- The parameter can be used in the body of the function.

Let's take a look at an example of function templates:

```
template <typename T>
void xchg(T&x , T&y){
...

template <int N>
int nTimes(int n){
...

```



Calling Function Template by Passing Arguments

#

In the given code snippet, let's take a look at how can we call the initialized variables with our template. In the example below, look at lines 2, 5, 8, and 11 for better understanding.

```
template <typename T>
void xchg(T&x , T&y){

```



```
void xchg(T& x, T& y){ ...  
  
int a, b;  
  
xchg(a, b);  
  
template <int N>  
int nTimes(int n){ ...  
  
int n = 5;  
nTimes<10>(n);
```

The function arguments `x` and `y` in the function `xchg` must have the same type. By providing two type parameters, `template <typename T, typename T1>`, the types of the argument can be different.

Instantiation


The process of substituting the template parameter by the template arguments is called **instantiation**.

The compiler

- automatically creates an instance of the function template.
- will automatically create a function template if the template parameter is derived from the function arguments.



If the compiler cannot deduce the template arguments from the function arguments, you must specify the arguments explicitly.

 implicit

 explicit

```
template <typename T>  
void xchg(T& x, T& y){ ...  
int a, b;  
xchg(a, b);
```



Overloading

Function templates can be overloaded.

The following rules hold:

1. Templates support no automatic type conversion.

2. If a free function is a better or an equally good match as a function template, the free function will be used.
3. You can specify the type of the function template:

```
func<type>(...).
```

4. You can specify that you are only interested in function templates:

```
func<>(...)
```

Let's take a look at a few examples of function templates in the next lesson.