

- Examples

In this lesson, we look at some examples of function templates.

WE'LL COVER THE FOLLOWING ^

- Example 1
 - Explanation
- Example 2
 - Explanation

Example 1

```
// templateFunctionsTemplates.cpp

#include <iostream>
#include <string>
#include <vector>

template <typename T>
void xchg(T& x, T& y){
    T t = x;
    x =y;
    y =t;
}

template <int N>
int nTimes(int n){
    return N * n;
}

int main(){

    std::cout << std::endl;

    bool t = true;
    bool f = false;
    std::cout << "(t, f):  (" << t << ", " << f << ") "<< std::endl;
    xchg(t, f);
    std::cout << "(t, f):  (" << t << ", " << f << ") "<< std::endl;

    std::cout << std::endl;

    int int2011 = 2011;
```

```

int int2014 = 2014;
std::cout << "(int2011, int2014): (" << int2011 << ", " << int2014 << ")" << std::endl;
xchg(int2011, int2014);

std::cout << "(int2011, int2014): (" << int2011 << ", " << int2014 << ")" << std::endl;

std::cout << std::endl;

std::string first{"first"};
std::string second{"second"};
std::cout << "(first, second): (" << first << ", " << second << ")" << std::endl;
xchg(first, second);
std::cout << "(first, second): (" << first << ", " << second << ")" << std::endl;

std::cout << std::endl;
std::vector<int> intVec1{1, 2, 3, 4, 5};
std::vector<int> intVec2{5, 4, 3, 2, 1};

for (auto v: intVec1)std::cout << v << " ";
for (auto v: intVec2)std::cout << v << " ";
std::cout << std::endl;
xchg(intVec1, intVec2);
for (auto v: intVec1)std::cout << v << " ";
for (auto v: intVec2)std::cout << v << " ";

std::cout << "\n\n";

std::cout << "nTimes<5>(10): " << nTimes<5>(10) << std::endl;
std::cout << "nTimes<10>(5): " << nTimes<10>(5) << std::endl;

std::cout << std::endl;
}

```



Explanation

- In the example above, we declared two function templates `xchg` and `nTimes` in lines 8 and 15. `xchg` swaps the values passed as arguments. `nTimes` returns the `N` multiplied by the number passed, `n`.
- We have initialized multiple instances to check for functions in lines (31, 32) and (39, 40).

Example 2

```

// templateFunctionsTemplatesOverloading.cpp

#include <iostream>

void xchg(int& x, int& y){ // 1
    int t = x;

```



```

    x = y;
    y = t;
}

template <typename T>           // 2
void xchg(T& x, T& y){
    T t = x;
    x = y;
    y = t;
}

template <typename T>           // 3
void xchg(T& x, T& y, T& z){
    xchg(x, y);
    xchg(x, z);
}

int main(){

    std::cout << std::endl;

    int intA = 5;
    int intB = 10;
    int intC = 20;

    double doubleA = 5.5;
    double doubleB = 10.0;

    std::cout << "Before: " << intA << ", " << intB << std::endl;
    xchg(intA, intB);           // 1
    std::cout << "After: " << intA << ", " << intB << std::endl;

    std::cout << std::endl;

    std::cout << "Before: " << doubleA << ", " << doubleB << std::endl;
    xchg(doubleA, doubleB);     // 2
    std::cout << "After: " << doubleA << ", " << doubleB << std::endl;

    std::cout << std::endl;

    xchg<>(intA, intB);          // explicit 2
    xchg<int>(intA, intB);       // explicit 2: xchg<int>
    // xchg<double>(intA, intB); // ERROR explicit xchg<double>

    std::cout << "Before: " << intA << ", " << intB << ", " << intC << std::endl;
    xchg(intA, intB, intC);     // 3
    std::cout << "After: " << intA << ", " << intB << ", " << intC << std::endl;

    std::cout << std::endl;
}

```



Explanation

- In the above example, we used the concept of function overloading by

calling `xchg` with different arguments passed to the function.

- We used the `xchg` function with different data types by passing two and three arguments. The call `xchg<double, double>(intA, intB)` will operate when `xchg` takes its arguments by value.

Let's discuss class templates in the next lesson.