Class Template

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

- The parameters are declared by `class` or `typename` .
- The parameter can be used in the class body.
- You can define the member functions of the class template inside or outside the class.

```
template <typename T, int N>
class Array{
  T elem[N];
...
```

Class Template: Instantiation

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

- A class template (before C++17) cannot deduce its template arguments from its function arguments.

⇨ Each template argument must be explicitly specified.

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

int a, b;
xchg(a, b);
```

```
template <typename T, int N>
class Array{ ...

Array<double, 10> doubleArray;
Array<Account, 1000> accountArray;
```

templateClassTemplate.cpp

Member Function Template

Member function templates are function templates used in a class or class template.

- Member function templates can be defined inside or outside the class.

```
template <class T, int N>
class Array{
public:
  template <class T2>
  Array<T, N>& operator = (
...
```

```
template <class T, int N>
class Array{
public:
  template <class T2>
  Array<T, N>& operator = (const Array<T2, N>& a);
...
};
template<class T, int N>
  template<class T2>
   Array<T, N>& Array<T, N>::operator = (const Array<T2, N>& a{
     ...
```

⚠ The destructor and copy/move constructor cannot be a template.

templateClassTemplateMemberFunctions.cpp