Template classes

Template classes

Classes can contain fields, the types of which are specified by the programmer for each instance of the class.

The declaration of a template class is preceded by the keywords template<typename T1, ... typename TN> with the required number of template parameters.

Template classes

Since the methods of a template class are instantiated only after the class instance specification for specific template parameter values, the implementation of template classes cannot be performed in source code files (*.cpp).

Template classes

Typically, the declaration of a class is done ın the header file, similar to non-template classes.

The implementation of a template class is done in a file with the extension *.hpp. This file should be located in the "Header Files" section.

When using template classes, the corresponding *.hpp file is included in the project.

Template classes

Complex.h

```
template<typename T>
class Complex {
	T Re;
	T Im;
public:
	Complex();
	Complex(T R, T I);
	~Complex();
	T GetRe();
	T GetIm();
	template<typename T>
	friend std::ostream& operator<<(std::ostream& Out, Complex<T>& C);
};
```

Template classes

Complex.hpp

```
#include “Complex.h”

template<typename T> Complex<T>::Complex(){
     this->Re = 0;
     this->Im = 0;
}

template<typename T> Complex<T>::Complex(T Re, T Im){
     this->Re = Re;
     this->Im = Im;
}
```

Template classes

Complex.hpp

```
template<typename T> Complex<T>::~Complex(){ }

template<typename T> T Complex<T>::GetRe() {
	return this->Re;
}

template<typename T> T Complex<T>::GetIm() {
	return this->Im;
}

template<typename T> std::ostream& operator<<(std::ostream& Out, Complex<T>& C) {
	Out << C.Re << “+i” << C.Im;
	return Out;
}
```

Template classes

main.cpp

```
#include <iostream>
#include “Complex.hpp”

int main()
{
    Complex<int> c(5, 7);
    int re = c.GetRe();                re = 5
    int im = c.GetIm();                im = 7
    std::cout << c << std::endl;       5 + i7
    return 0;
}
```