MİMAR SİNAN FINE ARTS UNIVERSITY

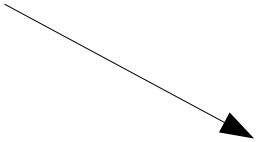
Bilgisayar Programlama II
Ders 4
MSGSU Fizik Bölümü
Ferhat ÖZOK

- **Kullanılacak kaynaklar:**
  http://www.cplusplus.com/doc/tutorial/
  Published by Juan Soulié

- **C++ ile ileri programlama**
  Paul Deitel
  Harvey Deitel

Bilgisayar Programlama II
Ders 4
MSGSU Fizik Bölümü
Ferhat ÖZOK

Şablon(template) fonksiyonlar tanımlayabildiğimiz gibi Şablon(template) sınıflarda oluşturabiliriz



```
template <class T>
class mypair {
    T values [2];
  public:
    mypair (T first, T second)
    {
      values[0]=first; values[1]=second;
    }
};
```

```
mypair<int> myobject (115, 36);
```

Aynı sınıf başka türden objeleroluşturmak içinde kullanılabilir

```
mypair<double> myfloats (3.0, 2.18);
```

MİMAR SİNAN FINE ARTS UNIVERSITY

Bilgisayar Programlama II
Ders 4
MSGSU Fizik Bölümü
Ferhat ÖZOK

Örnek:

Üye fonksiyonların tanımlanması:
template <class T>
T mypair<T>::getmax ()

```
// class templates
#include <iostream>
using namespace std;
template <class T>
class mypair {
    T a, b;
  public:
    mypair (T first, T second)
      {a=first; b=second;}
    T getmax ();
};
template <class T>
T mypair<T>::getmax ()
{
  T retval;
  retval = a>b? a : b;
  return retval;
}
int main () {
  mypair <int> myobject (100, 75);
  cout << myobject.getmax();
  return 0;
}
```

Bilgisayar Programlama II
Ders 4
MSGSU Fizik Bölümü
Ferhat ÖZOK

Şablon sınıflarda belli türler için özel tanımlamalar yapmak mümkündür.
Tanımlama şekli örnek olarak char tipi için özeltanımlama:
template <> class mycontainer <char> { ... };

Normal şablon ileözel tanımlama arasındaki fark:

template <class T> class mycontainer { ... };
template <> class mycontainer <char> { ... };

Örnek

```
// template specialization
#include <iostream>
using namespace std;
// class template:
template <class T>
class mycontainer {
    T element;
  public:
    mycontainer (T arg) {element=arg;}
    T increase () {return ++element;}
};
// class template specialization:
template <>
class mycontainer <char> {
    char element;
  public:
    mycontainer (char arg) {element=arg;}
    char uppercase ()
    {
      if ((element>='a')&&(element<='z'))
      element+='A'-'a';
      return element;
    }
};
int main () {
  mycontainer<int> myint (7);
  mycontainer<char> mychar ('j');
  cout << myint.increase() << endl;
  cout << mychar.uppercase() << endl;
  return 0;
}
```

Bilgisayar Programlama II
Ders 4
MSGSU Fizik Bölümü
Ferhat ÖZOK

Örnek:

```
//stack.h
#pragma once
template <class T>
class Stack
{
public:
        Stack(int = 10) ;
        ~Stack() { delete [] stackPtr ; }
        int push(const T&);
        int pop(T&) ;  // pop an element off the stack
        int isEmpty()const { return top == -1 ; }
        int isFull() const { return top == size - 1 ; }
private:
        int size ;  // Number of elements on Stack
        int top ;
        T* stackPtr ;
} ;
//constructor with the default size 10
template <class T>
Stack<T>::Stack(int s)
{
        size = s > 0 && s < 1000 ? s : 10 ;
        top = -1 ;  // initialize stack
        stackPtr = new T[size] ;
}
 // push an element onto the Stack
template <class T>
int Stack<T>::push(const T& item)
{
        if (!isFull())
        {
                stackPtr[++top] = item ;
                return 1 ;  // push successful
        }
        return 0 ;  // push unsuccessful
}
// pop an element off the Stack
template <class T>
int Stack<T>::pop(T& popValue)
{
        if (!isEmpty())
        {
                popValue = stackPtr[top--] ;
                return 1 ;  // pop successful
        }
        return 0 ;  // pop unsuccessful
}
```

MİMAR SİNAN FINE ARTS UNIVERSITY

Bilgisayar Programlama II
Ders 4
MSGSU Fizik Bölümü
Ferhat ÖZOK

```
#include <iostream>
#include "stack.h"
using namespace std ;
int main()
{
        typedef Stack<float> FloatStack ;
        typedef Stack<int> IntStack ;
        FloatStack fs(5) ;
        float f = 1.1 ;
        cout << "Pushing elements onto fs" << endl ;
        while (fs.push(f))
        {
                cout << f << ' ' ;
                f += 1.1 ;
        }
        cout << endl << "Stack Full." << endl
        << endl << "Popping elements from fs" << endl ;
        while (fs.pop(f))
                cout << f << ' ' ;
        cout << endl << "Stack Empty" << endl ;
        cout << endl ;
        IntStack is ;
        int i = 1 ;
        cout << "Pushing elements onto is" << endl ;
        while (is.push(i))
        {
                cout << i << ' ' ;
                i += 1 ;
        }
        cout << endl << "Stack Full" << endl
        << endl << "Popping elements from is" << endl ;
        while (is.pop(i))
                        cout << i << ' ' ;
        cout << endl << "Stack Empty" << endl ;
}
```

Bilgisayar Programlama II
Ders 4
MSGSU Fizik Bölümü
Ferhat ÖZOK

```
#ifndef ARRAY_H
#define ARRAY_H

#include <assert.h> // for assert()
using namespace std;
template <class T> // This is a template class, the user will
provide the data type for T
class Array
{
private:
    int m_length;
    T *m_data;

public:
    Array()
    {
        m_length = 0;
        m_data = nullptr;
    }
```

Devamı sol tarafta !!!

```
    Array(int length)
    {
        m_data = new T[length];
        m_length = length;
    }

    ~Array()
    {
        delete[] m_data;
    }

    void Erase()
    {
        delete[] m_data;
        // We need to make sure we set m_data to 0 here, otherwise it will
        // be left pointing at deallocated memory!
        m_data = nullptr;
        m_length = 0;
    }


    T& operator[](int index)
    {
        assert(index >= 0 && index < m_length);
        return m_data[index];
    }

    // The length of the array is always an integer
    // It does not depend on the data type of the array
    int getLength(); // templated getLength() function defined below
};
template <typename T> // member functions defined outside the class need their own template declaration
int Array<T>::getLength() { return m_length; } // note class name is Array<T>, not Array

#endif
```

MİMAR SİNAN FINE ARTS UNIVERSITY

Bilgisayar Programlama II
Ders 4
MSGSU Fizik Bölümü
Ferhat ÖZOK

```
// test2.cpp
```

g++ -std=c++11 test2.cpp -o test2

```
#include <iostream>
#include "Array.h"
using namespace std;
int main()
{
	Array<int> intArray(12);
	Array<double> doubleArray(12);

	for (int count = 0; count < intArray.getLength(); ++count)
	{
		intArray[count] = count;
		doubleArray[count] = count + 0.5;
	}

	for (int count = intArray.getLength()-1; count >= 0; --count)
		std::cout << intArray[count] << "\t" << doubleArray[count] << endl;

	return 0;
}
```