MİMAR SİNAN FINE ARTS UNIVERSITY

Bilgisayar Programlama II
Ders 2
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 2
MSGSU Fizik Bölümü
Ferhat ÖZOK

Sınıf (Class):

C++ Sınıflar(Class) data yapıları(structures) kavramının genişletilmiş halidir. Data yapılarına benzer şekilde data üyelerini barındıracağı gibi yapısında fonksiyonlarıda barındırabilir

```
class class_name {
  access_specifier_1:
    member1;
  access_specifier_2:
    member2;
  ...
} object_names;
```

Örnek:

```
class Rectangle {
    int width, height;
  public:
    void set_values (int,int);
    int area (void);
} rect;
```

Sınıf Bileşenleri data yapılarından farklı olarak Erişimin Denetlenmesi tabidir. Sınıflarda sınıf bileşenlerinin erişim türlerine göre türleri

Private:

Herhangi bir tanımlama yok ise varsayılan tanımlama private'dir. Private üyeler sadece o sınıfa ait üyeler veya o sınıfın friend üyeleri

tarafından erişilebilir.

Protected :

o sınıfa ait üyeler, o sınıftan türetilmiş sınıflarınüyeleri veya o sınıfın friend üyeleri tarafından erişilebilir.

Public:

sınıf nesnesinin görünür olduğu heryerden erişilebilir

Bilgisayar Programlama II
Ders 2
MSGSU Fizik Bölümü
Ferhat ÖZOK

Örnek 1:

```
/ classes example
#include <iostream>
using namespace std;
class Rectangle {
    int width, height;
  public:
    void set_values (int,int);
    int area() {return width*height;}
};
void Rectangle::set_values (int x, int y) {
  width = x;
  height = y;
}
int main () {
  Rectangle rect;
  rect.set_values (3,4);
  cout << "area: " << rect.area();
  return 0;
}
```

Örnek 2:

```
// example: one class, two objects
#include <iostream>
using namespace std;
class Rectangle {
    int width, height;
  public:
    void set_values (int,int);
    int area () {return width*height;}
};
void Rectangle::set_values (int x, int y) {
  width = x;
  height = y;
}
int main () {
  Rectangle rect, rectb;
  rect.set_values (3,4);
  rectb.set_values (5,6);
  cout << "rect area: " << rect.area() << endl;
  cout << "rectb area: " << rectb.area() << endl;
  return 0;
}
```

Bilgisayar Programlama II
Ders 2
MSGSU Fizik Bölümü
Ferhat ÖZOK

Constructors

(KURUCU): Sınıf için özel bir fonksiyondur.

Sınıf için oluşturulan her bir nesnenin tanımlanması ile otomatik olarak çağrılır. Ve Sınıf bileşenlerinin ilk değerlerini atar yada hafızada yer ayırır

```
// example: class constructor
#include <iostream>
using namespace std;
class Rectangle {
    int width, height;
  public:
    Rectangle (int,int);
    int area () {return (width*height);}
};
Rectangle::Rectangle (int a, int b) {
  width = a;
  height = b;
}
int main () {
  Rectangle rect (3,4);
  Rectangle rectb (5,6);
  cout << "rect area: " << rect.area() << endl;
  cout << "rectb area: " << rectb.area() << endl;
  return 0;
}
```

Bilgisayar Programlama II
Ders 2
MSGSU Fizik Bölümü
Ferhat ÖZOK

```
// overloading class constructors
#include <iostream>
using namespace std;
class Rectangle {
    int width, height;
  public:
    Rectangle ();
    Rectangle (int,int);
    int area (void) {return (width*height);}
};
Rectangle::Rectangle () {
  width = 5;
  height = 5;
}
Rectangle::Rectangle (int a, int b) {
  width = a;
  height = b;
}
int main () {
  Rectangle rect (3,4);
  Rectangle rectb;
  cout << "rect area: " << rect.area() << endl;
  cout << "rectb area: " << rectb.area() << endl;
  return 0;
}
```

Bir sınıf için birden fazla değişik parametreler içieren constructor tanımlanabilir

MİMAR SİNAN FINE ARTS UNIVERSITY

Bilgisayar Programlama II
Ders 2
MSGSU Fizik Bölümü
Ferhat ÖZOK

Üye bileşenlere ilk değer verilmesi:

```
Rectangle::Rectangle (int x, int y) { width=x; height=y; }

Rectangle::Rectangle (int x, int y) : width(x) { height=y; }

Rectangle::Rectangle (int x, int y) : width(x), height(y) { }
```

```
// member initialization
#include <iostream>
using namespace std;
class Circle {
    double radius;
  public:
    Circle(double r) : radius(r) { }
    double area() {return radius*radius*3.14159265;}
};
class Cylinder {
    Circle base;
    double height;
  public:
    Cylinder(double r, double h) : base (r), height(h)
{}
    double volume() {return base.area() * height;}
};
int main () {
  Cylinder foo (10,20);
  cout << "foo's volume: " << foo.volume() << '\n';
  return 0;
}
```

MİMAR SİNAN FINE ARTS UNIVERSITY

Bilgisayar Programlama II
Ders 2
MSGSU Fizik Bölümü
Ferhat ÖZOK

Pointers to classes

Rectangle * prect;

```
// pointer to classes example
#include <iostream>
using namespace std;
class Rectangle {
  int width, height;
public:
  Rectangle(int x, int y) : width(x), height(y) {}
  int area(void) { return width * height; }
};
int main() {
  Rectangle obj (3, 4);
  Rectangle * foo, * bar, * baz;
  foo = &obj;
  bar = new Rectangle (5, 6);
  baz = new Rectangle[2] { {2,5}, {3,6} };
  cout << "obj's area: " << obj.area() << '\n';
  cout << "*foo's area: " << foo->area() << '\n';
  cout << "*bar's area: " << bar->area() << '\n';
  cout << "baz[0]'s area:" << baz[0].area() << '\n';
  cout << "baz[1]'s area:" << baz[1].area() << '\n';
  delete bar;
  delete[] baz;
  return 0;
}
```

MİMAR SİNAN FINE ARTS UNIVERSITY

Bilgisayar Programlama II
Ders 2
MSGSU Fizik Bölümü
Ferhat ÖZOK

expression	can be read as
`*x`	pointed to by x
`&x`	address of x
`x.y`	member y of object x
`x->y`	member y of object pointed to by x
`(*x).y`	member y of object pointed to by x (equivalent to the previous one)
`x[0]`	first object pointed to by x
`x[1]`	second object pointed to by x
`x[n]`	(n+1)th object pointed to by x

MİMAR SİNAN FINE ARTS UNIVERSITY

Time.h

```
#ifndef TIME_H
#define TIME_H

// Time class definition
class Time
{
public:
   Time(); // constructor
   void setTime( int, int, int ); // set hour, minute and second
   void printUniversal(); // print time in universal-time format
   void printStandard(); // print time in standard-time format
private:
   int hour; // 0 - 23 (24-hour clock format)
   int minute; // 0 - 59
   int second; // 0 - 59
}; // end class Time

#endif
```

```
#include <iostream>
#include <iomanip>
using namespace std;
// Time constructor initializes each data member to zero.
// Ensures all Time objects start in a consistent state.
Time::Time()
{
   hour = minute = second = 0;
} // end Time constructor

// set new Time value using universal time; ensure that
// the data remains consistent by setting invalid values to zero
void Time::setTime( int h, int m, int s )
{
   hour = ( h >= 0 && h < 24 ) ? h : 0; // validate hour
   minute = ( m >= 0 && m < 60 ) ? m : 0; // validate minute
   second = ( s >= 0 && s < 60 ) ? s : 0; // validate second
} // end function setTime

// print Time in universal-time format (HH:MM:SS)
void Time::printUniversal()
{
   cout << setfill( '0' ) << setw( 2 ) << hour << ":"
      << setw( 2 ) << minute << ":" << setw( 2 ) << second;
} // end function printUniversal

// print Time in standard-time format (HH:MM:SS AM or PM)
void Time::printStandard()
{
   cout << ( ( hour == 0 || hour == 12 ) ? 12 : hour % 12 ) << ":"
      << setfill( '0' ) << setw( 2 ) << minute << ":" << setw( 2 )
      << second << ( hour < 12 ? " AM" : " PM" );
} // end function printStandard
```

MİMAR SİNAN FINE ARTS UNIVERSITY

Bilgisayar Programlama II
Ders 2
MSGSU Fizik Bölümü
Ferhat ÖZOK

```
#include <iostream>
#include "Time.h" // include definition of class Time from Time.h
using namespace std;

int main()
{
   Time t; // instantiate object t of class Time

   // output Time object t's initial values
   cout << "The initial universal time is ";
   t.printUniversal(); // 00:00:00
   cout << "\nThe initial standard time is ";
   t.printStandard(); // 12:00:00 AM

   t.setTime( 13, 27, 6 ); // change time

   // output Time object t's new values
   cout << "\n\nUniversal time after setTime is ";
   t.printUniversal(); // 13:27:06
   cout << "\nStandard time after setTime is ";
   t.printStandard(); // 1:27:06 PM

   t.setTime( 99, 99, 99 ); // attempt invalid settings

   // output t's values after specifying invalid values
   cout << "\n\nAfter attempting invalid settings:"
      << "\nUniversal time: ";
   t.printUniversal(); // 00:00:00
   cout << "\nStandard time: ";
   t.printStandard(); // 12:00:00 AM
   cout << endl;
} // end main
```