Algoritmalar ve Programlama II – BLM 102

Hafta 4: Sınıflar I

Fenerbahçe Üniversitesi

Öğretim Elemanları

Öğretim Üyesi: Dr. Vecdi Emre Levent

Ofis: 311

Email: emre.levent@fbu.edu.tr

Asistan: Arş. Gör. Uğur Özbalkan

Ofis: 311

Email: ugur.ozbalkan@fbu.edu.tr

Ders Planı

- Preprocessor'ler ile Aynı Header Eklenmesinin Engellenmesi
- Sınıf'larda Pointer Kullanımı
- Constructor'lar
- Destructor'lar
- Tasarım Örnekleri

Preprocessor İşlemleri

- Bir sınıf'a header dosyası eklenirken, dosyanın birden çok kez eklenmesini engellemek için preprocessor tanımları kullanılabilir.

Preprocessor İşlemleri

- Derleme işlemi esnasında yapılan kontrol ile tekrar eklenmenin önüne geçilir:

```
#ifndef Test_H
#define Test_H
  ...
#endif
```

  - Eğer dosya daha önceden eklenmemiş ise, define test_h satırı da tanımlanmamış olacağı için, dosya ilk eklenmesinde
    - ifndef test_h
  - Doğru olacaktır ve dosya include edilecektir.
  - Daha önceden eklenmiş ise, tekrar eklenmesinin önüne geçilecektir.

Preprocessor İşlemleri

```
#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
```

Preprocessor İşlemleri

- Her header dosyasında;
  - ifndef, define, endif
- Tanımları ile dosyanın birden çok kez eklenmesi engellenebilir.

Preprocessor İşlemleri

```
#include <iostream>
#include <iomanip>
#include <stdexcept> // for invalid_argument exception class
#include "Time.h" // include definition of class Time from Time.h

using namespace std;

// Time constructor initializes each data member to zero.
Time::Time()
{
   hour = minute = second = 0;
} // end Time constructor
void Time::setTime( int h, int m, int s )
{
   // validate hour, minute and second
   if ( ( h >= 0 && h < 24 ) && ( m >= 0 && m < 60 ) &&
      ( s >= 0 && s < 60 ) )
   {
      hour = h;
      minute = m;
      second = s;
   } // end if
   else
      throw invalid_argument(
         "hour, minute and/or second was out of range" );
} // end function setTime
```

```
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
```

Preprocessor İşlemleri

```
#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
   cout << "\n\nUniversal time after setTime is ";
   t.printUniversal(); // 13:27:06
   cout << "\nStandard time after setTime is ";
   t.printStandard(); // 1:27:06 PM

   // attempt to set the time with invalid values
   try
   {
      t.setTime( 99, 99, 99 ); // all values out of range
   } // end try
   catch ( invalid_argument &e )
   {
      cout << "Exception: " << e.what() << endl << endl;
   } // end catch

   // 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
```

Preprocessor İşlemleri

```
The initial universal time is 00:00:00
The initial standard time is 12:00:00 AM

Universal time after setTime is 13:27:06
Standard time after setTime is 1:27:06 PM

Exception: hour, minute and/or second was out of range

After attempting invalid settings:
Universal time: 13:27:06
Standard time: 1:27:06 PM
```

Preprocessor İşlemleri

- Her header dosyasında;
  - ifndef, define, endif
- Tanımları ile dosyanın birden çok kez eklenmesi engellenebilir.

Sınıfların Bellekte Tutulması

- Bir sınıftan bir obje yaratıldığında, o sınıftaki değişkenler için bellekte yer tanımlanır.
- Fonksiyonlar ise ortak kullanım için tek bir yerde tutulurlar.
- Aynı sınıftan başka bir obje yaratıldığında, değişkenlerin kopyaları yaratılır ancak fonksiyonlar, tek bir yerden kullanılırlar.

Sınıflarda Pointer Kullanımı

- Bir sınıf obje, dizi, referans ve pointer olacak kullanılabilir;

- Time sınıfı için;

```
Time sunset; // Obje
Time arrayOfTimes[ 5 ]; // Dizi
Time &dinnerTime = sunset; // Referans
Time *timePtr = &dinnerTime; // Pointer
```

Sınıflarda Pointer Kullanımı

- Eğer bir sınıftaki fonksiyonlardan birinde, sınıfın içinde tanımlanmış olan bir değişkenin ismi ile aynı bir değişken ismi ile tanım yapılırsa, değişken ismi ile sınıftaki tanımlanmış değişkene erişilemez.
    - Bunun için, this operatörü ile erişilmesi gerekmektedir.

Sınıflarda Pointer Kullanımı

```
#include <iostream>
using namespace std;

class sinif {
public:
  int a;
  int function() {
  int a = 5;
  a = 7;
  this->a = 8;
  cout << "local variable a: " << a << endl;
  cout << "class a object variable a: " << this->a << endl;
  return 0;
  }
};
```

```
int main() {
sinif sinifTest;
sinifTest.function();
cout << "A's variable a: " << sinifTest.a << endl;
return 0;
}
```

Çıktı:

```
local variable a: 7
class a object variable a: 8
A's variable a: 8
```

Sınıflarda Pointer Kullanımı

- Bir sınıftaki public tanımlanmış değişken veya fonksiyonlara ulaşırken,
  - (.) operatörü ile objenin elemanlarına ulaşılır
  - (->) operatörü ile obje pointer olarak tanımlandığında kullanılır.

Sınıflarda Pointer Kullanımı

```
#include <iostream>
using namespace std;
class Count
{
public: // public data is dangerous
   // sets the value of private data member x
   void setX( int value )
   {
      x = value;
   } // end function setX

   // prints the value of private data member x
   void print()
   {
      cout << x << endl;
   } // end function print

private:
   int x;
}; // end class Count
```

```
int main()
{
   Count counter; // create counter object
   Count *counterPtr = &counter; // create pointer to counter
   Count &counterRef = counter; // create reference to counter

   cout << "Set x to 1 and print using the object's name: ";
   counter.setX( 1 ); // set data member x to 1
   counter.print(); // call member function print

   cout << "Set x to 2 and print using a reference to an object: ";
   counterRef.setX( 2 ); // set data member x to 2
   counterRef.print(); // call member function print

   cout << "Set x to 3 and print using a pointer to an object: ";
   counterPtr->setX( 3 ); // set data member x to 3
   counterPtr->print(); // call member function print
} // end main
```

Sınıflarda Pointer Kullanımı

```
Set x to 1 and print using the object's name: 1
Set x to 2 and print using a reference to an object: 2
Set x to 3 and print using a pointer to an object: 3
```

Constructor’larda Default Argümanlar

- Constructorlarda tüm argümanlar beslenmese bile, default değerler ile çalışabilirler.

- Bunun için constructor tanımında argümanların başlangıç değerlerinin verilmesi gerekmektedir.

Constructor'larda Default Argümanlar

```
// prevent multiple inclusions of header
#ifndef TIME_H
#define TIME_H

// Time abstract data type definition
class Time
{
public:
   Time( int = 0, int = 0, int = 0 ); // default constructor

   // set functions
   void setTime( int, int, int ); // set hour, minute, second
   void setHour( int ); // set hour (after validation)
   void setMinute( int ); // set minute (after validation)
   void setSecond( int ); // set second (after validation)
    // get functions
    int getHour(); // return hour
    int getMinute(); // return minute
    int getSecond(); // return second

    void printUniversal(); // output time in universal-time format
    void printStandard(); // output 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
```

Constructor'larda Default Argümanlar

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

// Time constructor initializes each data member to zero
Time::Time( int hour, int minute, int second )
{
   setTime( hour, minute, second ); // validate and set time
} // end Time constructor

// set new Time value using universal time
void Time::setTime( int h, int m, int s )
{
   setHour( h ); // set private field hour
   setMinute( m ); // set private field minute
   setSecond( s ); // set private field second
} // end function setTime
```

```
void Time::setHour( int h )
{
   if ( h >= 0 && h < 24 )
      hour = h;
   else
      throw invalid_argument( "hour must be 0-23" );
} // end function setHour

// set minute value
void Time::setMinute( int m )
{
   if ( m >= 0 && m < 60 )
      minute = m;
   else
      throw invalid_argument( "minute must be 0-59" );
} // end function setMinute
```

Constructor'larda Default Argümanlar

```
void Time::setSecond( int s )
{
   if ( s >= 0 && s < 60 )
      second = s;
   else
      throw invalid_argument( "second must be 0-59" );
} // end function setSecond

// return hour value
int Time::getHour()
{
   return hour;
} // end function getHour

// return minute value
int Time::getMinute()
{
   return minute;
} // end function getMinute
```

```
int Time::getSecond()
{
   return second;
} // end function getSecond

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

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

Constructor'larda Default Argümanlar

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

int main()
{
   Time t1; // all arguments defaulted
   Time t2( 2 ); // hour specified; minute and second defaulted
   Time t3( 21, 34 ); // hour and minute specified; second defaulted
   Time t4( 12, 25, 42 ); // hour, minute and second specified

   cout << "Constructed with:\n\nt1: all arguments defaulted\n  ";
   t1.printUniversal(); // 00:00:00
   cout << "\n  ";
   t1.printStandard(); // 12:00:00 AM

   cout << "\n\nt2: hour specified; minute and second defaulted\n  ";
   t2.printUniversal(); // 02:00:00
   cout << "\n  ";
   t2.printStandard(); // 2:00:00 AM

   cout << "\n\nt3: hour and minute specified; second defaulted\n  ";
   t3.printUniversal(); // 21:34:00
   cout << "\n  ";
   t3.printStandard(); // 9:34:00 PM

   cout << "\n\nt4: hour, minute and second specified\n  ";
   t4.printUniversal(); // 12:25:42
   cout << "\n  ";
   t4.printStandard(); // 12:25:42 PM

   try
   {
      Time t5( 27, 74, 99 ); // all bad values specified
   } // end try
   catch ( invalid_argument &e )
   {
      cout << "\n\nException while initializing t5: " << e.what() << endl;
   } // end catch
} // end main
```

Constructor'larda Default Argümanlar

```
Constructed with:

t1: all arguments defaulted
  00:00:00
  12:00:00 AM

t2: hour specified; minute and second defaulted
  02:00:00
  2:00:00 AM

t3: hour and minute specified; second defaulted
  21:34:00
  9:34:00 PM

t4: hour, minute and second specified
  12:25:42
  12:25:42 PM

Exception while initializing t5: hour must be 0-23
```

Destructor

- Obje silineceğinde dolaylı olarak çağrılmaktadır.
- (~) sınıf adı olarak tanımlanırlar
- Destructor kendisi bellekten objeleri silmez, obje silinirken yapılması istenen işlemler burada tanımlanır.

Destructor

- Bir parametre almaz veya döndürmez.
- Bir sınıfta bir destructor bulunabilir ve public olmalıdır.
- Eğer bir sınıfta destructor tanımlanmadıysa, arka planda boş bir destructor tanımlanır.

Destructor

```
#include <iostream>

using namespace std;
class Line {
public:
void setLength(double len);
double getLength(void);
Line();
~Line();
private:
double length;
};
Line::Line(void) {
cout << "Object is being created" << endl;
}
Line::~Line(void) {
cout << "Object is being deleted" << endl;
}
void Line::setLength(double len) {
length = len;
}
double Line::getLength(void) {
return length;
}
```

```
int main() {
Line line;

// set line length
line.setLength(6.0);
cout << "Length of line : " << line.getLength() << endl;

return 0;
}
```

Çıktı:

```
Object is being created
Length of line : 6
Object is being deleted
```

Destructor

```
#include <iostream>
using namespace std;
class apple {
public:
int roll;
apple() {          //Constructor
roll = 5;
cout << "Roll is " << roll << endl;
};
~apple() {         //Destructor
roll = -5;
cout << "Roll is " << roll << endl;
};
};
int main() {
apple obj1;
return 0;
}
```

```
Roll is 5
Roll is -5
```

Sınıfların Atanmaları

- = operatörü ile bir sınıftan yaratılmış bir obje, aynı sınıftan yaratılmış başka bir objeye eşitlenebilir.

- Tüm değişkenler birbirine atanmış olur.

Sınıfların Atanmaları

```
#ifndef DATE_H
#define DATE_H

// class Date definition
class Date
{
public:
   Date( int = 1, int = 1, int = 2000 ); // default constructor
   void print();
private:
   int month;
   int day;
   int year;
}; // end class Date

#endif
```

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

// Date constructor (should do range checking)
Date::Date( int m, int d, int y )
{
   month = m;
   day = d;
   year = y;
} // end constructor Date

// print Date in the format mm/dd/yyyy
void Date::print()
{
   cout << month << '/' << day << '/' << year;
} // end function print
```

Sınıfların Atanmaları

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

int main()
{
   Date date1( 7, 4, 2004 );
   Date date2; // date2 defaults to 1/1/2000

   cout << "date1 = ";
   date1.print();
   cout << "\ndate2 = ";
   date2.print();

   date2 = date1; // default memberwise assignment

   cout << "\n\nAfter default memberwise assignment, date2 = ";
   date2.print();
   cout << endl;
} // end main
```

Sınıfların Atanmaları

```
date1 = 7/4/2004
date2 = 1/1/2000

After default memberwise assignment, date2 = 7/4/2004
```