Veri Yapıları

Lab Notları – 1

C++ Programlama Dili

C++ dili, hızlı ve düşük seviye özelliklere erişmek isteyen uygulamaların yazılması için popüler bir dildir. C programlama diline birçok ekstra özellik katmıştır. En önemlisi nesne yönelimli bir programlama dilidir. Sınıfları destekler. Çok biçimlilik ve kalıtımı destekler. Bütün C kodlarını derleyebilir. C++ programlama dilini kullanacak bir programcı bellek yönetimini çok iyi bilmelidir. Kaynak dosyaları ".cpp", ".cxx", ".C" gibi uzantılara sahip olabilir. Bu derste kodlar derlenirken bir GNU derleyicisi olan MinGW kullanılacaktır. MinGW kurulumunu yapmak için https://dosya.sakarya.edu.tr/Dokumanlar/2015/BSM208/553679379_mingw_kurulumu.pdf dosyası incelenebilir.

İlk Program

Girilen 4 basamaklı bir sayıyı basamaklarına ayırmak.

```
#include <iostream>
using namespace std;
int main()
{
  int sayi;
 setlocale(LC_ALL,"Turkish"); // Türkçe karakter desteği için
 do{
        cout<<"4 Basamaklı bir sayı girin:";
        cin>>sayi;
 }while(sayi<1000 || sayi>10000); //4 basamaklı sayı kontorlü
 short birler,onlar,yuzler,binler;
 binler=sayi/1000;
 yuzler=(sayi%1000)/100;
 onlar=(sayi%100)/10;
 birler=sayi%10;
 cout<<endl<<"Binler:"<<binler<<endl;
 cout<<endl<<"Yüzler:"<<yuzler<<endl;
 cout<<endl<<"Onlar:"<<onlar<<endl;
 cout<<endl<<"Birler:"<<birler<<endl;
 cin.get();
 cin.ignore();
 return 0;
}
```

Nasıl Derlenir: Komut satırından kodun bulunduğu klasöre gelerek "g++ -o basamak basamak.cpp" girilip enter tuşuna basılırsa derleme gerçekleşip basamak.exe dosyası oluşacaktır.

Kod incelendiğinde yorum satırlarının // ifadesiyle başladığı görülecektir. Birden çok satırda yorum yapmak için /* yorumlar… */ şeklinde bir kullanım yapılmalıdır. Yorum satırları derleyiciler tarafından görülmez ve o satırlar atlanır. Bu satırlar programcılar için açıklama mahiyetindedirler.

karakteri ile başlayan ifadeler, ön işlemci komutlarıdır. Derleme işleminin hemen öncesinde çalıştırılırlar. Bir başka dosya veya kütüphane ekleme işlemleri bu şekilde yapılmaktadır.

{ } ifadeleri C++ dilinde kod bloklarının açılma ve kapanma ifadeleridir. Fonksiyonlar, sınıf tanımlamaları döngüler, kontrol ifadelerinin hepsi birer kod bloğu olduğu için bu ifadeler ile başlayıp

biterler. Fakat bu ifadeleri kullanmak için illaki bu bahsi geçen yapıların olmasına gerek yoktur. Aşağıdaki örnek kod C++ dilinde derlenip çalışacaktır.

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

int main(){
	int a=10;
	{
		cout<<a<<endl;
		{
			cout<<"Merhaba";
		}
	}
	return 0;
}
```

iostream kütüphanesinin eklenmesinin nedeni cout ve cin gibi girdi ve çıktı fonksiyonlarının tanınması içindir. cin girdi alma ifadesidir ve bir dosyadan girdi alınabileceği gibi ekrandan da girdi almak için kullanılabilir. cout ise çıktı verme için kullanılır.

Çalıştırılabilir bir C++ programı olabilmesi için bir main yani giriş fonksiyonuna ihtiyaç vardır. Bu derleyicinin kodun hangi satırından başlayacağını bilmesi içindir. main fonksiyonu tamamlandığında program kapanmış demektir. main fonksiyonun sonudaki return 0; ifadesi programın, işletim sistemine "her şey yolunda gitti ve işlem başarıyla tamamlandı" demesidir.

C++ ta çalıştırılacak tüm ifadeler ; ile biter. C++ programlama dilinde kaynak kodda bırakılan boşluk miktarının bir önemi yoktur örneğin aşağıdaki program başarıyla çalışır. Sadece okunabilirliği düşüktür.

```
#include <iostream>
using namespace std; int main(){  for(int i=0;
i<10;
i++)cout<<i<<endl; }
```

Komut Satırı Parametreleri

Komut satırı parametreleri, daha program çalışmaya başlamadan önce programa belli parametreleri girmeyi sağlar. Bu C++'ta main metodunun parametreleri ile sağlanır. Örneğin aşağıdaki kod C++'ta komut satırından aldığı parametreyi ekrana yazar. 1. İndekstekini almamızın sebebi 0. İndekste programın kendisini tutmaktadır. Bunun dışında C++'ta argc isminde bir başka parametrede bulunmaktadır. Bu argc, komut satırından kaç adet parametre girildiğini gösterir. Birden çok parametre girmek için parametreler arası boşluk bırakılmalıdır.

```
// C++
int main(int argc,char *argv[]){
	cout<<argv[1]<<endl;
}
```

Veri Türleri

Karakter Tipi (char): C/C++ dillerinde karakter tek bir byte ile ifade edilir. Bir byte 8 bit olduğu için, en fazla 256 karakter ifade edilebilir. Bunlar ASCII kodu olarak ta bilinirler.

C++'ta \ karateri çıkış karakteri olarak kullanılır. Özel bir karakterdir.

\n Yeni satır
\b Bir karakter geri
\t Tab
\' Tek karakter koymak için
\" Çift karakter koymak için

```
int main(){
        char c='\b';
        cout<<"Merhaba"<<c<<c;
        cin.get();
        cin.ignore();
        return 0;
}
```

Yukarıdaki kod bloğunda c içerisinde bir karakter geri ifadesi tutulduğu için ekrana Merhaba yazdığında imleç b'nin üzerinde yanıp sönecektir.

Boolean Tipi: C++'ta 0 değeri yanlış kabul edilip bu değer dışındaki bütün değerler doğru olarak kabul edilir. Örneğin aşağıdaki kod parçasında ekrana Sakarya yazacaktır.

```
int main(){
        if(200) cout<<"Sakarya";
        else cout<<"Ankara";
        return 0;
}
```

Aynı kodu aşağıdaki gibi değiştirirsek, yapı olarak bir şey değişmeyecek yine ekrana Sakarya yazacaktır.

```
int main(){
   bool x=200;
   if(x) cout<<"Sakarya";
   else cout<<"Ankara";
   return 0;
}
```

C++'ta mimariden mimariye ilkel türlerin kaplamış oldukları alanda farklılıklar olabilir. Aşağıdaki kod çalıştırıldığında ekrana 4 yazacaktır. Bu int ilkel türünün bellekte 4 byte kapladığı anlamına gelir. X değişkenine atanan sayının büyüklüğü ile bellekte kapladığı yer arasında bir bağlantı yoktur. Örneğin kodun ikinci kısmında ekrana tekrar 4 byte yazacaktır. Eğer 4 byte'a sığmayacak bir sayı kullanılmak isteniyorsa örneğin double türü düşünülebilir. Aşağıda sizeof'un neden bir fonksiyon değil de operatör olarak ifade edildiği sorulursa aşağıdaki yazılış şeklinden anlaşılabilir.

```
int main(){
   int x=100;
   cout<<sizeof x;
```

```
    return 0;
}
```

```
int main(){
    int x=1000000000;
    cout<<sizeof x;
    return 0;
}
```

C++'ta ilkel türler kategori olarak ikiye ayrılırlar, kayan noktalı (ondalık) ve tamsayı olan türler. char, short, int ve long tamsayı türlerine girer. float, double ve long double ise kayan noktalı türlere girer.

Tür dönüşümlere bakıldığında, C++'ta küçük veri türünden büyük veri türüne dönüştürüldüğünde bir sıkıntı oluşmamaktadır.

```
int main(){
    int x=100;
   double a=x;
   cout<<a;
   return 0;
}
```

Sıkıntı **büyük veri türünden küçüğüne** dönüştürüldüğünde ortaya çıkmaktadır. C++ her hangi bir derlenme hatası vermez. Fakat dönüştürülen değer boyutu daha küçük olan veri türüne sığmayacaksa veri kaybı olur. Örneğin aşağıdaki C++ kodunda ondalık değer tamsayıya dönüştürülmüş ve ondalık kısmı kaybolmuştur.

```
int main(){
    double x=100.35;
    int a=x;
    cout<<a;
    return 0;
}
```

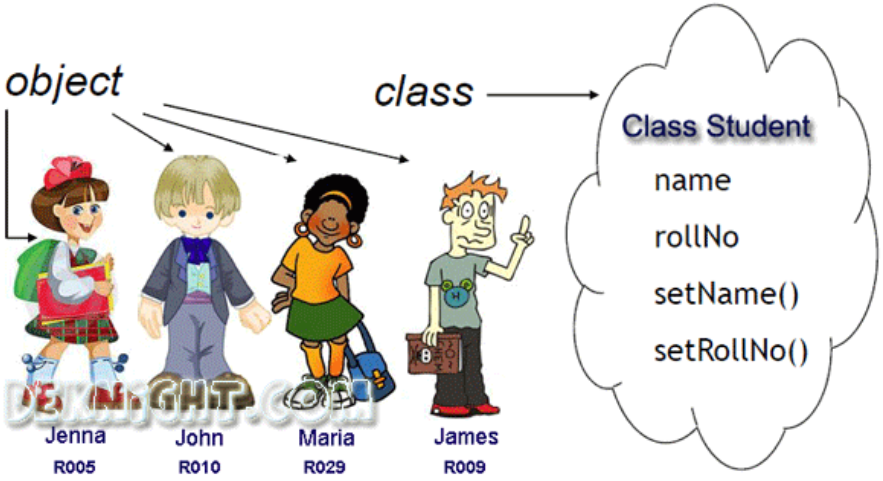
Sabitler

Bazen program yazılırken bazı değerlerin programın sonuna kadar sabit kalması istenebilir. Örneğin pi sayısı veya kat sayılar gibi. Aşağıdaki kod incelendiğinde 3.14'ün aslında orada bir sabit olduğu ve değişmemesi gerektiği görülecektir.

```
int main(){
    const double pi=3.14;
    double yariCap=5;
    cout<<"Cevre="<<2*pi*yariCap*yariCap;
    return 0;
}
```

Sınıf Tasarımı

C++'ta niteliklere veri üyeleri (data members), davranışlara ise üyelik fonksiyonları (member functions) karşılık gelir. Nesne ile sınıf arasındaki fark nesneler, sınıftan türetilen elemanlardır.



Yapıcı metot sınıf ile aynı adı taşır ve dönüş değeri tanımlanmaz. Bir sınıftan nesne birçok yolla oluşturulabilir. Bunun anlamı birden fazla yapıcı metot olabilir.

```
class Arac{
        private:
                float hiz; //km/saat
                int yil; // Model yılı
        public:
                Arac(int yl){
                        hiz=0;
                        yil=yl;
                }
                void Hizlan(float artiHiz){
                        hiz+=artiHiz;
                }
                void Yavasla(float eksiHiz){
                        hiz-=eksiHiz;
                }
};
```

C++'ta sınıf ile dosya adının aynı olma zorunluluğu yoktur. Ayrıca bir dosyada birden çok sınıf tanımı yapılabilir. C++'ta bir blok olarak public ya da private tanımlanır. C++ sınıf tanımı bittiğinde **;** **konmalıdır**.

this Terimi

this terimi C++'ta o anda oluşturulan nesneyi ifade etmek için kullanılır. Mesela aşağıdaki örneğe bakıldığında yapıcı metodun parametresi ile kişi sınıfının alt alanı aynı adı taşımakta. Bu derleyici için bir karmaşıklığa sebep olmaktadır. isim=isim; ifadesinde yapılan şey parametre olan isim'in **kendi üzerine atanmasıdır**. Dolayısıyla Kisi sınıfından bir nesne türetip ismini yazmaya kalktığımızda String olduğu için ve değeri verilmemiş olduğu için ekrana **null** yazacaktır. Bunu düzeltmek için this terimi kullanılmalıdır.

```
class Kisi{
public:
 string isim;
…
Kisi(String isim){
      isim=isim;
```

```
    yas=0;
    boy=20;
    kilo=4;
  }
…
}
```

```
// Doğrusu
Kisi(string isim){
    This->isim=isim;
    yas=0;
    boy=20;
    kilo=4;
  }
```

Yıkıcı Metotlar

C++ programlama dilinde heap bellek bölgesinin kontrolü programcının elinde olduğu için o bölgede işi bittiğinde geri döndürmelidir. Geri döndürülecek alanda yine heap bellek bölgesini gösteren işaretçiler olabilir ve bunlar private olup dışarıdan erişilemiyor olabilir bu durumda onları da geri döndürecek bir metoda ihtiyaç vardır. İşte bu metot yıkıcı metottur. Aşağıdaki örneğe bakıldığında yıkıcı metot sınıf adı ile aynı parametre almayan ve dönüş değeri olmayan sadece başında ~ işareti olan bir metottur.

```
class Arac{
        private:
                float hiz; //km/saat
                int yil; // Model yılı
                Kisi *surucu;
         ….
                ~Arac(){
                    delete surucu;
                }
};
```

Yukarıdaki koda bakıldığında Arac sınıfından bir nesne türetildiğinde aynı anda kişi sınıfından, araç sınıfına bağlı bir sürücü nesnesi türetilmektedir. Bunun erişim niteleyicisi private olduğu için ve heap bellek bölgesinde oluştuğu için geri döndürülmesi gerekmektedir. Fakat sadece sınıf içerisinden geri döndürülmelidir. Bu durumda Yıkıcı metot kullanılmış ve araç nesnesi yıkıldığı an kişi nesnesi de geri döndürülmüştür.

Erişim Niteleyicileri

C++'ta public, protected ve private niteleyicileri bulunmaktadır. Bu niteleyiciler sınıfın elemanlarının görünürlüğünü ayarlamaktadır. Varsayılan olarak private kabul edilir ve private özelliklerini korur. private niteleyicisi sadece sınıf içerisinden ve arkadaş (friend) fonksiyonlar tarafından erişilebilir. protected ise private niteleyicisine çok benzer tek farkı kalıtım almış sınıflar da protected niteleyicisine erişebilir. public ise sınıf içi ve sınıf dışından erişilebilir.

Önemli: Şu ana kadar bahsedilen niteleyicilere sınıf dışından erişilebiliyorsa ve programcı bunlara erişmek istiyorsa mutlaka sınıftan bir nesne türetmelidir. Fakat bazı durumlarda sınıftan nesne

türetilmeden kullanılmak istenebilir veya zorunda kalınabilir bu durumda bir başka niteleyici olan static devreye girer.

Başlık Dosyaları

C++ derleyicisi kendi başına tanımlamaları arayıp bulma yeteneğinden yoksundur. Dolayısıyla programcının bu tanımlamaları derlenme ve link işlemlerinde derleyiciye göstermesi gerekmektedir. Bir programcının tasarlamış olduğu bir aracı (tool) bir başka programcı da kullanacaktır. Fakat burada tanımlamaları verme zorunluluğu bulunduğu için ama diğer tarafta da kod gizliliği olduğu için bunu ancak başlık dosyaları ile sağlayabilir. Başlık dosyaları tanımlamaları (imzaları) verirken gerçekleştirim (fonksiyon gövdelerini) vermez. Bu şekilde hem kod gizlenmiş hem de derleyiciye imzalar yardımıyla tanımlamalar verilmiş olur. C++'ta başlık dosyaları genellikle .hpp uzantılı olur. Kaynak dosyaları da genellikle .cpp uzantılı olur. Aşağıdaki örneği inceleyelim.

Arac.hpp

```
#ifndef ARAC_HPP
#define ARAC_HPP
class Arac{
	private:
		float hiz; //km/saat
		int yil; // Model yılı
	public:
		Arac(int);
		void Hizlan(float);
		void Yavasla(float);
		float Hiz();
		float Yil();
};
#endif
```

Arac.cpp

```
#include "Arac.hpp"

Arac::Arac(int yl){
	hiz=0;
	yil=yl;
}
void Arac::Hizlan(float artiHiz){
	hiz+=artiHiz;
}
void Arac::Yavasla(float eksiHiz){
	hiz-=eksiHiz;
}
float Arac::Hiz(){
	return hiz;
}
float Arac::Yil(){
	return yil;
}
```

Test.cpp (Geliştirilmiş olan aracı kullanan program)

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

```
#include "Arac.hpp"

int main(){
	Arac *a = new Arac(1995);
	Arac b(2007);
	cout<<a->Hiz()<<endl;
	cout<<b.Yil()<<endl;
	delete a;

	return 0;
}
```

Hazırlayan
Arş. Gör. M. Fatih ADAK