BTEP243 - Ders 1

C ve C++ arasındaki farklar:

C	C++
#include <stdio.h>	#include <iostream>
printf("… ");	cout<<"…..";
scanf("%...",&…);	cin>>
Yeni satır→ \n	\n veya endl
Varsayılan geri dönüş türü → "void"	"integer"

#include <iostream> → C++ ve C, giriş/çıkış sistemlerinde ortak özellik olan **kanal**ları kullanırlar. C'de kullanılan tüm kanallar C++'ta da aynen kullanılabilir. C++ kanal sınıfları <iostream> adındaki bir başlık dosyası içinde toplanmıştır ve C ++ standart giriş/çıkış kütüphanesi bildirimlerini içerir.

using namespace std → Aduzayları (namespace) bir kapsamı belirli bir etkinlik alanını tanımlar. Bir aduzayı içinde tanımlanmış tüm değişkenler aynı alana ait olurlar. "std" aduzayı, tüm sınıfları, nesneleri ve standart C++ kütüphanesinin fonksiyonlarını içerir.

Modüler Programlama: Programı modül denen küçük parçalara ayırma ve bu parçacıkları birleştirme.

BÖL ve Yönet yöntemi: Fonksiyonlar

Kendi içinde bağımsız olarak çalışabilen ve belli bir işlevi yerine getiren program modülüne fonksiyon denir. C programları fonksiyonlardan oluşur.
Fonksiyonların yazılmasındaki temel amaç, büyük boyutlardaki programların daha kolay yazılabilen ve test edilebilen küçük parçalar halinde oluşturulabilmesidir.

Fonksiyonların Özellikleri:

- Her fonksiyonun bir adı vardır. Fonksiyon isimlerinin verilmesinde değişken isimlerinde uygulanan kurallar geçerlidir.
- Foksiyonlar programın diğer parçalarından etkilenmeden bağımsız bir işlem yapabilirler.
- Belli bir işlevi yerine getirirler. Örneğin, ortalama hesaplamak, ekrana bir veri yazmak, bir dizideki en büyük elemanı bulmak gibi.
- Kendilerini çağıran programdan parametre olarak veri alabilirler.
- Gerektiği durumlarda ürettikleri sonuçları kendilerini çağıran programa parametre olarak geri gönderirler.

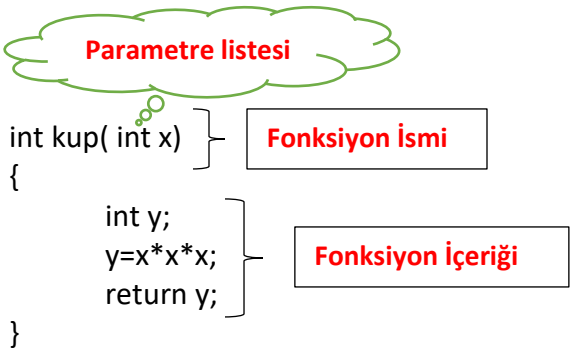
Fonksiyonların sağladığı faydalar:

- Program yazma süreci kısalır ve kolaylaşır.
- Programların sürdürülebilirliği ve yaşatılabiliriği artar.
- Karmaşık problemler daha küçük parçalara bölünebilir. Her parça ayrı ayrı fonksiyonlar şeklinde çözülerek sonradan ana programda birleştirilebilir.
- Grup çalışmaları için uygun bir ortam hazırlar. Grup elemanları bağımsız fonksiyonları ayrı ayrı tasarlarlar. Son aşamada bu fonksiyonlar ana programda birleştirilir.
- Daha önceden yazılmış fonksiyonlar arşivlerinden alınarak kullanılabilir. Aynı program parçasının tekrar yazılmasına gerek kalmaz.
- Programın içinde sık sık tekrar edilen bölümler fonksiyon olarak yazılabilir. Böylece aynı program parçasının defalarca tekrar edilmesine gerek kalmaz.

Fonksiyonun Tanımı

Örnek:

// Bir sayının küpünü bulup geri döndüren fonsiyon




```
void main ( )
{
        int sayi;
        cout<<"Bir sayı giriniz:";
        cin>>sayi;

        cout<<"Girdiğiniz sayının (“<<sayi<<”) küpü:"<<kup(sayi)<<endl;
}
```

Fonksiyon tanımı aşağıdakileri içerir:

İsim: Fonksiyonun ismi. Değişken isimlerinde kullanılan kurallar, fonksiyon isimleri için de geçerlidir.
Parametre listesi: Fonksiyona gönderilen değerleri tutan değişkenlerdir.
Fonksiyon içeriği : Fonksiyonun gerçekleştirdiği işlemler topluluğudur.
Dödürdüğü değerin türü: Fonksiyonun geri döndürdüğü değerin türü. (örnek: void, int ...)

- Eğer fonksiyon herhangi bir değer döndürürse, türü mutlaka belirtilmelidir.
  int kup(int x)
- Ancak fonksiyon geriye değer döndürmeyecekse geri dönüş değerinin tipi void olarak tanımlanır. Bu durumda fonksiyondan çıkışı sağlayan return sözcüğünün fonksiyonda kullanılmasına gerek yoktur.
  void kup(int x)

Fonksiyonların çağrılması

- Fonksiyonlar, isimleri yazılarak ve parantez içinde gerekli sayıda argüman gönderilerek çağırılır.
- Eğer fonsiyon geriye bir değer döndürüyorsa, bu değer bir değişkene atanabilir , başka bir fonksiyona argüman olarak verilebilir, bir ifadenin içinde kullanılabilir.
- Fonksiyonu çağırırken, ismini, **()** (parantez) ve ; (noktalı virgül) kullanırız.
- Fonksiyon çağırıldığında, program fonksiyonun içeriğini çalıştırır.
- Fonksiyon sonlandığında ise program kaldığı yerden çalışmaya devam eder.

- Program çalışmaya başladığında, **main( )** otomatik olarak çağırılır.
- **main( )** birçok fonksiyon çağırabilir.
- Fonksiyonlar diğer fonksiyonları çağırabilir.

Örnek:

```
int kup( int x)
{
        int y;
        y=x*x*x;
        return y;
}
void main()
{
int sayi, sonuc;
...
sonuc=kup(sayi);
```

veya

```
cout<<"Girdiğiniz sayının (“<<sayi<<”) küpü:"<<kup(sayi)<<endl;
}
```

Fonksiyon Prototipleri

Derleyici (compiler) fonksiyon çağırılmadan önce aşağıdakileri bilmek zorundadır.

- İsmini
- Geri döndürdüğü değerin türünü
- Parametre sayısını
- Her paramtrenin veri türünü

Fonksiyon prototiplerinin kullanımı:

- Prototipleri programın en başına listeleyiniz.
- Program, fonksiyonların prototiplerini veya full tanımlarını, çağırmadan önce (main’den önce) mutlaka belirtmeliyiz, aksi takdirde derleyici hata verecektir.

Başlık: int kup(int x)
Prototip: int kup(int);

Yerel Değişken / Global Değişken

- Fonksiyon bloklarının (gövde) içerisinde tanımlanan değişkenler sadece o fonksiyonda kullanılabilen yerel değişkenlerdir. İlgili fonksiyon sonlandıktan sonra yerel değişkenler bellekten kaldırılır.
- Fonksiyon bloklarının dışında tanımlanan değişkenler ise global değişkenlerdir. Bu tür değişkenler bütün fonksiyonlar tarafından yazılıp okunabilirler ve programın çalışması süresince geçerlidirler.

Bazı olumsuz yönlerinden dolayı global değişken kullanımından kaçınmak gerekmektedir.

Örneğin;

- Global değişkenler bütün fonksiyonlar tarafından değiştirilebildiği için programdaki hataların ayıklanması zorlaşır.
- Grup elemanları arasındaki bağımlılık artar. Hangi global değişkenin ne işlevi olacağına, ismine ve kimin tarafından ne şekilde değiştirileceğine önceden karar vermek gerekir.

Örnek:

```
#include<iostream>
using namespace std;
int a; //global değişken
int kup(int x)
{
	int y; //yerel değişken
	y=x*x*x;
	return y;
}

void main()
{
	a=kup(5);
	cout<<a<<endl;
}
```

INLINE (SATIR İÇİ) FONKSİYONLAR

Programlarımızı yazarken fonksiyon oluşturmak yararlıdır. Ancak fonksiyonlar ne kadar çok olursa program çalıştırıldığında, fonksiyon çağrıları da o kadar çok olacaktır. İşte bu noktada satır içi (inline) fonksiyonlar devreye giriyor ve fonksiyon çağrılarının programa ek olarak getirdiği yükü azaltıyor.Tek dezavantajı, çalışma zamanını azaltırken programın boyutunu arttırmasıdır.

Örnek:

```
#include<iostream>
using namespace std;
inline int dikdortgen(int sayi1, int sayi2)
{
	return sayi1*sayi2;
}
void main()
{
	int sayi1, sayi2;
	cout<< "Kisa kenar:";
	cin>>sayi1;
	cout<< "Uzun kenar:";
	cin>>sayi2;
	cout<< "Dikdortgenin Alani:"<<dikdortgen(sayi1,sayi2)<<endl;

	system("pause");
}
```

Fonksiyonların çağırlamasında kullanılabilecek yöntemler:

1. **Değer-ile-çağırma:** Fonksiyon çağırılırken, argüman olarak ilgili değer fonksiyona gönderilir. Orjinal değer herhangi bir değişikliğe uğramaz.
2. **Referans-ile-çağırma:** Fonksiyon çağırılırken, argüman olarak ilgili değer fonksiyona gönderilir. Ancak burada gönerilen argüman, fonksiyon içerisinde yapılan tüm değişikliklerin ilgili adresteki değerin değişmesini sağlar. Fonksiyon tanımlanırken, (&) referans operatörü kullanılır.
3. **Adres-ile-çağırma:** Fonksiyon çağırılırken, argüman değişkeni yerine argüman değişkenin adresinin gönderilmesini sağlar. Argümanı bir adres olduğundan, fonksiyonda kullanılan parametre bir gösterici (pointer) olmalıdır. Fonksiyon çağrılırken ise işaretçi yerine referenas (&) operatörü kullanılmalıdır ki ilgili adresteki değer değişimi sağlanabilsin.

Örnekler

1)

```
#include<iostream>
using namespace std;
int toplam(int, int); //fonksiyonun prototipi
int carp(int, int);
void fonksiyon1();
void main()
{
        fonksiyon1();
        int a,b;
        cout<<"Iki sayi giriniz:";
        cin>>a>>b;
        cout<<a<<"+"<<b<<"="<<toplam(a,b)<<endl;
        cout<<a<<"*"<<b<<"="<<carp(a,b)<<endl;

        system("pause");
}
```

```
int toplam(int s1, int s2)
{
        return s1+s2;
}
int carp(int s1, int s2)
{
        return s1*s2;
}
void fonksiyon1()
{
        cout<<"Deneme Programi..."<<endl;
}
```

2)

C/C++	C/C++	sadece C++
//deger ile cagirma (call-by-value) *- Fonksiyondaki değişiklik orjinal değeri etkilemez.*	//referans ile cagirma (call-by-reference) *- Fonksiyondaki değişiklik orjinal değeri değiştirir.*	*//adres ile cagirma (call-by-address)* *- Fonksiyondaki değişiklik orjinal değeri değiştirir.*
#include<iostream> using namespace std; void degistir(int, int); void main() { int a,b; cout<<"Iki sayi giriniz:"; cin>>a>>b; cout<<"Degisiklikten once:"; cout<<a<<" "<<b<<endl; degistir(a,b); cout<<"Degisiklikten sonra:"; cout<<a<<" "<<b<<endl; system("pause"); } void degistir(int s1, int s2) { int g; g=s1; s1=s2; s2=g; }	#include<iostream> using namespace std; void degistir(int&,int&); void main() { int a,b; cout<<"Iki sayi giriniz:"; cin>>a>>b; cout<<"Degisiklikten once:"; cout<<a<<" "<<b<<endl; degistir(a,b); cout<<"Degisiklikten sonra:"; cout<<a<<" "<<b<<endl; system("pause"); } void degistir(int&s1,int&s2) { int g; g=s1; s1=s2; s2=g; }	*#include<iostream>* *using namespace std;* *void degistir(int*,int*);* *void main()* *{* int a,b; cout<<"Iki sayi giriniz:"; cin>>a>>b; cout<<"Degisiklikten once:"; cout<<a<<" "<<b<<endl; degistir(&a,&b); cout<<"Degisiklikten sonra:"; cout<<a<<" "<<b<<endl; system("pause"); } void degistir(int*s1,int*s2) { int g; g=*s1; *s1=*s2; *s2=g; }
Ekran Çıktısı: İki sayi giriniz: 5 3 Degisiklikten once: 5 3 Degisiklikten sonra 5 3 Press any key to continue . . .	Ekran Çıktısı: İki sayi giriniz: 5 3 Degisiklikten once 5 3 Degisiklikten sonra 3 5 Press any key to continue . . .	Ekran Çıktısı: İki sayi giriniz: 5 3 Degisiklikten once 5 3 Degisiklikten sonra 3 5 Press any key to continue . . .

3)

```
//deger-ile-cagirma
//referans-ile-cagirma
//adres-ile-cagirma
#include<iostream>
using namespace std;
int kare_deger(int);
void kare_ref(int&);
void kare_adres(int*);
void main()
{
	int a;
	cout<<"Bir tam sayi giriniz:";
	cin>>a;
	cout<<"Girdiginiz sayinin karesi(Deger-ile-
cagirma):"<<kare_deger(a)<<endl;

	kare_ref(a);
	cout<<"Girdiginiz sayinin karesi(Referans-ile-
cagirma):"<<a<<endl;
	int b;
	cout<<"Bir tam sayi giriniz:";
	cin>>b;
	kare_adres(&b);
	cout<<"Girdiginiz sayinin karesi(Adres-ile-
cagirma):"<<b<<endl;

	system("pause");
}
```

```
int kare_deger(int x)
{
	return x*x;
}
void kare_ref(int&x)
{
	x=x*x;
}
void kare_adres(int *x)
{
	*x=(*x)*(*x);
}
```

4)

```
#include<iostream>
using namespace std;
//varsayilan degeri olan fonksiyonlar
void fonk(int =10, int =20);
void main()
{
	int a=3, b=5;
	fonk(a,b);
	fonk(a);
	fonk(b);
	fonk();
	system("pause");
}
```

```
void fonk(int x, int y)
{
	cout<<x<<" "<<y<<endl;
}
```

5)

```
#include<iostream>
using namespace std;
int fonk(int=3);
void main()
{
	int a=5, b=7;
	cout<<fonk(a)<<endl;
	cout<<fonk(b)<<endl;
	cout<<fonk()<<endl;
	system("pause");
}
```

```
int fonk(int x)
{
	int k=10;
	k=k+x;
	return k;
}
```

6)

```
#include<iostream>
using namespace std;
int fonk(int=3);
void main()
{
	int a=5, b=7;
	cout<<fonk(a)<<endl;
	cout<<fonk(b)<<endl;
	cout<<fonk()<<endl;
	system("pause");
}
```

```
int fonk(int x)
{
	static int k=10;
	k=k+x;
	return k;
}
```

7)

```
#include<iostream>
using namespace std;
//varsayilan degeri olan fonksiyonlar
void fonk(int , int =20);
void main()
{
	int a=3, b=5;
	fonk(a,b);
	fonk(a);
	fonk(b);
	//fonk();HATA!!! En az bir tane deger
gondermemiz gerekiyor
	system("pause");
}
```

```
void fonk(int x, int y)
{
	cout<<x<<" "<<y<<endl;
}
```

8)

```
#include<iostream>
using namespace std;
//varsayilan degeriler mutlaka son parametrelere eklenir
void fonk(int =10 , int );
void main()
{
	int a=3, b=5;
	fonk(a,b);
/*	fonk(a);
	fonk(b);
	fonk();HATA!!! En az iki tane deger gondermemiz gerekiyor*/
	system("pause");
}
```

```
void fonk(int x, int y)
{
	cout<<x<<" "<<y<<endl;
}
```

9)

```
#include<iostream>
using namespace std;
void fonk();
void fonk(int,int);
void fonk(int);
float fonk(float,float);
void main()
{
	fonk();
	int a=4, b=5;
	fonk(a,b);
	fonk(b);
	float d=3.2, e=5.3;
	cout<<fonk(d,e)<<endl;
	system("pause");
}
```

```
void fonk()
{
	cout<<"TEST PROGRAMI"<<endl;
}
void fonk(int x,int y)
{
	x=y+5;
	y=x+2;
	cout<<x<<" "<<y<<endl;
}
void fonk(int z)
{
	static int s=5;
	s=s*z;
	cout<<s<<endl;
}
float fonk(float f,float g)
{
	cout<<f<<" "<<g<<endl;
	return f+g;
}
```