C
Temel Veri Tipleri ve Değişkenler

Dr. Öğr. Üyesi M. Ozan AKI

Rev 1.0

Veri Tipleri ve Değişken Tanımlama

[tip] [saklama] [işaret] [uzunluk] <veri tipi> <değişken adı> [=<ilk değer>];

[...] : Seçimlik alanlar
< ... > : Zorunlu alanlar

unsigned long int sayac; const float pi=3.14; char c; extern double ort;

Tip	Saklama	İşaret	Uzunluk	Veri Tipi
const	auto	signed	short	char
volatile	register	unsigned	long	int
	static			float
	extern			double

Değişkenlerin Tanım Kapsamları

- **Global Değişkenler**

main ya da herhangi fonksiyon bloğu dışında tanımlanan değişkenlerdir. Program çalışması boyunca tanımlıdır ve değerini korur. DATA Segment içerisinden tahsis edilir.

- **Lokal (Yerel) Değişkenler**

{ Blok } içerisinde tanımlanan değişkenlerdir. Sadece blok çalıştığı sürece geçerlidir. Blok çalışması sonlandığında tüm değişkenler yok edilir. Yerel değişkenler STACK bellekten tahsis edilir.

Değişkenlerin Tanım Kapsamları

```
int global;
int toplam = 15;

void main(void)
{
    int lokal;
    int toplam;
    toplam = 70;
    printf("Toplam: %d\n", toplam); // ???
}
```

Global bir değiken ile aynı isime sahip yerel değişken var ise; yerel değişken global değişkeni örter, yani blok içerisinde yapılan bütün referanslar yerel değişkene aittir.

Tip Bildirimleri : **const**

Derleyiciye, program tarafından değişkenin değerinin değiştirilemeyeceğini bildirir. Sadece ilk değer verilir.

const float pi = 3.14;

Genellikle fonksiyonlara geçilen işaretçi (pointer) tipindeki parametlerde, işaretçi tarafından gösterilen nesnenin değişmemesini garanti altına almak için kullanılır.

void function(const char *str) { ... };

Tip Bildirimleri : **volatile**

Derleyiciye, değişkenin değerinin, açıkça belirtilmeyen yollarla (örn, = atama) değiştirilebileceğini söyler.

Örneğin, sistem saati donanımına doğrudan bir işaretçi tanımlandığında, program içerisinde değiştirilmese dahi bu değişkenin içeriği donanım tarafından güncellenerek değiştirilir.

Ayrıca, derleyicinin optimizasyon sırasında hesaplama sırasını değiştirmesini engelleyerek bu değişken her seferinde güncel halini hesaplaması sağlanır.

Const ve volatile aynı anda kullanılabilir;

const volatile char *port = (const volatile char*) 0x30;

Saklama Bildirimleri : **auto**

Tanımlanan bütün yerel değişkenler, açıkça belirtilmese de varsayılan olarak **auto** bildirimine sahiptir.

Bu nedenle özel olarak auto deyimi genellikle hiç kullanılmaz.

auto tanımı, değişkenin tanımlandığı blok içerisine girildiğinde tüm yerel değişkenlerin otomatik olarak tanımlanması ve bloğun çalışması bittiğinde bunların otomatik olarak yok edilmesini ifade eder.

Saklama Bildirimleri : **static**

Yerel static değişkenler:

static olarak tanımlanan yerel değişkenler, tanım bloğundan çıkılsa dahi yok edilmeyerek değerini koruması sağlanır. Böylece, bloğun farklı zamanlardaki çağrısında önceki değişken değeri korunmuş olur.

Global static değişkenler:

static olarak tanımlanan global değişkenler ise, yanlızca içerisinde bulundukları dosyadan erişilebilir olurlar. Harici dosyalardan bu değişkene erişim derleyici tarafından engellenir.

Saklama Bildirimleri : **extern**

Deklarasyon: bir değişkenin adını ve tipini, bir fonksiyonun adını, parametrelerini ve dönüş değerini deklare eder.

Tanım: Bir değişken için bellekten yer ayrılmasını sağlayan bildiridir. Fonksiyon için, fonksiyonun kod bloğunu ifade eder.

Bir değişkenin birden fazla deklarasyonu olabilir, ancak sadece bir tanımı olabilir.

Saklama Bildirimleri : **extern**

Eğer başka bir dosyada tanımlanmış olan değişkeni kullanmamız gerekirse, aynı global değişkeni ikinci kez tanımlayamayız.

Ancak başka bir dosyada tanımlanmış olan değişkeni deklare ederek kullanabiliriz.

```
// sayac adlı değişkenin deklarasyonu
extern int sayac;
sayac++;
...

// sayac adlı değişken başka bir dosyada tanımlanmış
int sayac=0;
```

Saklama Bildirimleri : **register**

register olarak tanımlanan değişkenler, CPU (mikroişlemci) kaydedicilerinde (registers) ya da derleyicinin sürümüne göre ön bellekte (cache) saklanır.

Bir değişkenin **register** olarak saklanması, o değişkene mümkün olan en hızlı erişimin sağlanması anlamına gelir.

```
register int t,i;
t=0;
for(i=0; i<1000; i++) t+=i;
```

Neredeyiz

[tip] [saklama] **[işaret] [uzunluk] <veri tipi> <değişken adı> [=<ilk değer>];**

Tip	Saklama	İşaret	Uzunluk	Veri Tipi
const	auto	signed	short	char
volatile	register	unsigned	long	int
	static			float
	extern			double

Veri Tipleri

İşaret	Uzunluk	Tip	Bit	Byte	Tanım Aralığı
signed unsigned		char	8	1	-128 ... +127 0 ... 255
signed unsigned	short	int	16	2	-32,768 ... 32767 0 ... 65,535
signed unsigned	long	int	32	4	-2,147,483,648 ... 2,147,483,647 0 ... 4,294,967,295
signed unsigned	long long	int	64	8	$-(2^{63})$ ... $2^{63}-1$ 0 ... $2^{64}-1$
-	-	float	32 (1+23+8)	4	Kesir: $-(2^{22})$ ... $2^{22}-1$ Üs : $-(10^{38})$ ... $10^{38}-1$
-	-	double	64 (1+52+11)	8	Kesir: $-(2^{51})$ ... $2^{51}-1$ Üs: $-(10^{308})$ ... $10^{308}-1$

unsigned char c = 250; c = c + 10; c == ??

Değişkenlere İlk Değerin Atanması

Değişkenlere ilk tanımlandıklarında isteğe bağlı olarak bir ilk değer atanabilir.

```
char harf = 'A';
int a = 5;
const double pi = 3.14;
int a,b,c=3;
int a=1,b=2,c=3;
char str[] = "Trakya Universitesi";
```

Değişkenlere İlk Değerin Atanması

- Global değişkenler ile static olarak tanımlanmış değişkenlere ilk değer ataması yapılmamış ise derleyici tarafından varsayılan olarak 0 değeri atanır.
- static olmayan yerel değişkenlere ilk değer ataması yapılmamış ise, değerinin ne olacağı konusunda bir garanti yoktur.
- static olarak tanımlanmamış yerel değişkenlere ilk değer ataması yapılmış ise, bloğa her girişte ilk değer ataması yeniden yapılır.

Sabit Değer Notasyonları

- Doğrudan yazılan sayılar **int** tipi olarak kabul edilir. 1 123 10500 -43 -1 0
- Sayı sonuna "**L**" harfi eklenirse **long int** tipi olarak kabul edilir. 1000L 23L 0L
- Sayı sonuna "**U**" harfi eklenirse, işaretsiz (**unsigned**) olarak kabul edilir. 123U, 32500U
- Sayı sonuna "**F**" harfi eklenirse **float** tipi kabul edilir. 1.0F -0,25f 3.45e-3f 1e3f
- Ondalıklı bir sayının sonuna "L" gelirse, double tipi kabul edilir. 1.0L -1e3L 123.456L

Sabit Değer Notasyonları

- Aksi belirtilmediği sürece, kod içerisine yazılan sabit sayılar 10'luk tabandadır.
- Eğer sayı "**0**" ile başlıyorsa 8'lik tabanda (**octal**) sayı olarak ele alınır.

int a = 12; //10 luk tabanda 12 sayısıdır.

int b = 012; //10 luk tabanda 10 sayısıdır.

- Eğer sabit sayı "**0x**" ile başlıyorsa 16'lık tabanda (**hexadecimal**) sayı olarak ele alınır.

int c = 80; // 10'luk tabanda 80 sayısıdır.

int d = 0x80; // 10'luk tabanda 128 sayısıdır.

Sabit Değer Notasyonları

- Karakter girişleri içi ise tek tırnak (‘) kullanılır. Örneğin A harfini karakter olarak bir değişkene atamak için;
- **char c = ‘A’;**

Bunun yanında, ters bölü (\) işareti kullanılarak doğrudan ASCII kodu ile bir karakter tanımlanabilir.

- **char c = ‘\x41’;**

(65 == 0x41)

Sabit Değer Notasyonları

- Ters bölü (\) ile kullanılabilecek ön tanımlı çeşitli karakterler mevcuttur.
- **\n** -> yeni satır
- **\r** -> satır başı
- **\t** -> tab
- **\v** -> bir sonraki satıra geçiş
- **\a** -> bip sesi
- **\b** -> back space
- **** -> ters bölü karakteri
- **\’** -> tek tırnak
- **\”** -> çift tırnak