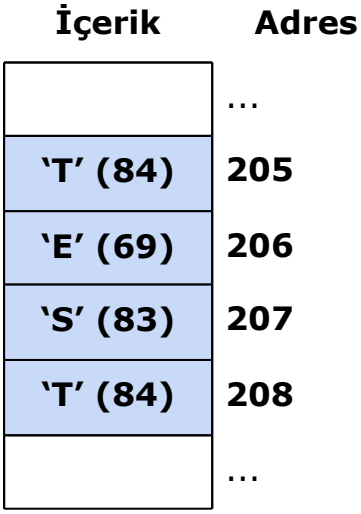
Pointer - Dizi İlişkisi

char tipinde diziler

Diziler aynı tipte birden fazla değişkeni peş peşe saklayan veri yapılarıdır. Örneğin aşağıdaki değişken tanımlandığında

```
char dizi_a[4] = {'T', 'E', 'S', 'T'}
```

Bellekte şöyle bir yerleşim olur



"char" değişken tipi 1 bellek gözü (1 byte) yer kaplar. Başka bir deyişle
"sizeof(char) = 1" dir.

Yukarıdaki d dizisinin boyutu:
4 * sizeof(char) = 4 * 1 = 4 byte'dır

dizi_a isimli değişken (dizi), 205 adresinden başlamaktadır.

- Dizinin ilk elemaninin adresi 205,
- ikinci elemaninin adresi 206'dır.

dizi_a[2] şeklinde yazıldığında ulaşılacak olan adres bilgisayar tarafından şöyle hespalanır:

adres = 205 + 2 * sizeof(char)
= 205 + 2*1
= **207**

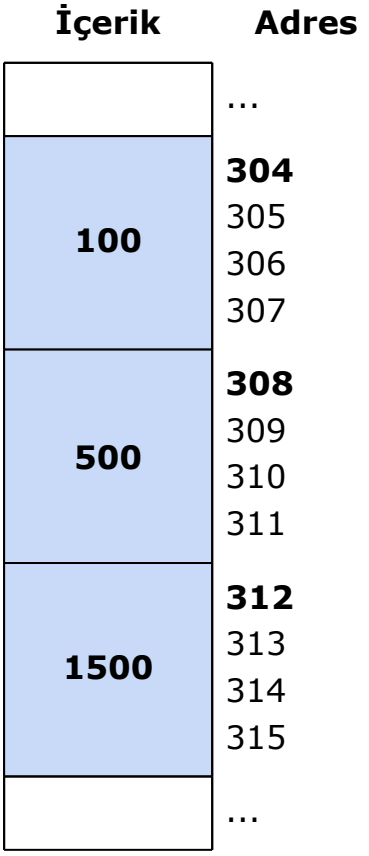
örnek:
printf("%c", dizi_a[2]); yazıldığında ekranda **'S'** çıktısını üretir.
printf("%d", &dizi_a[2]); yazıldığında ekranda **207** çıktısını üretir.

int tipinde diziler

Önceki örnektekine benzer şekilde int tipinde dizi tanımlanırsa

```
int dizi_b[3] = {100, 500, 1500};
```

Bellekte şöyle bir tablo ortaya çıkar:



'int' değişken tipi 4 bellek gözü (4 byte) yer kaplar. Yani
"sizeof(int) = 4"tür.

dizi_b'nin her elemani 4'er bellek gözü yer kapladığı için dizinin tamamı:
3 * sizeof(int) = 3 * 4 = 12 byte yer kaplar.

dizi_b isimli değişken (dizi), 304 adresinden başlamaktadır.

- Dizinin ilk elemaninin adresi 304,
- ikinci elemaninin adresi 308'dir.

dizi_b[2] şeklinde yazıldığında ulaşılacak olan adres bilgisayar tarafından şöyle hesaplanır:

adres = 304 + 2 * sizeof(int)
= 304 + 2 * 4
= **312**

Örnek:

printf(“%d”, dizi_b[2]); yazıldığında ekranda **1500** çıktısını üretir.
printf(“%d”, &dizi_b[2]); yazıldığında ekranda **312** çıktısını üretir.

Dizinin pointer olarak kullanılması

dizi bir sayı ile işleme sokulduğunda, adres şu şekilde hesaplanır:

```
dizi_b + 1  = 304 + 1 * sizeof(int)
            = 304 + 1 * 4
            = 308
```

printf(“%d\n”, dizi_b+1); yazılırsa çıktı **308** olur.

Not: 304 önceki başlıktaki şekle göre dizinin başlangıç adresi. sizeof(int) dizinin tipi int olduğu için

(dizi_b+1) adresinin içeriğine erişmek için: ***(dizi_b+1)** yazılır.
örnek:

printf(“%d\n”, *(dizi_b+1) yazılırsa çıktı **500** olur.

Burada pointer hesabı ile dizi hesabının aynı olduğunu fark etmişsinizdir. Diziler pointer, pointerlar dizi gibi kullanılabilir. Aşağıdaki eşitlikler geçerlidir.

dizi[**x**] == *(dizi + **x**)

ve

&dizi[**x**] == (dizi + **x**)

x: tamsayı

Örnek Kod

Bu dokümanda anlatılanlarla ilgili örnek kod sonraki sayfadadır.

Not: Gerçek programlarda göreceğiniz bellek adresleri 300 gibi küçük sayılar olmayacaktır. 16’lık tabanı kafadan dönüştüremiyorsanız, adresleri printf’te %u, %d gibi tiplerle yazdırabilirsiniz. Fakat adresler normalde %p tipiyle yazdırılır.

```c
int main() {

	/*-----------------------------------------
	 * char tipinde diziler
	 *-----------------------------------------*/
	printf("char tipinde diziler\n");
	printf("====================\n");

	int dizi_a[4] = {'T', 'E', 'S', 'T'};

	printf("sizeof(char) : %d\n", sizeof(char));
	printf("sizeof(dizi_a) : %d\n\n", sizeof(dizi_a));

	printf("ilk elemanin adresi : %u\n", &dizi_a[0]);
	printf("ikinci elemanin adresi : %u\n\n", &dizi_a[1]);

	printf("%c\n", dizi_a[2]);
	printf("%u\n\n", &dizi_a[2]);

	/*-----------------------------------------
	 * Int tipinde diziler
	 *-----------------------------------------*/
	printf("int tipinde diziler\n");
	printf("====================\n");

	int dizi_b[3] = {100, 500, 1500};

	printf("sizeof(int) : %d\n", sizeof(int));
	printf("sizeof(dizi_b) : %d\n\n", sizeof(dizi_b));

	printf("ilk elemanin adresi : %u\n", &dizi_b[0]);
	printf("ikinci elemanin adresi : %u\n\n", &dizi_b[1]);

	printf("%d\n", dizi_b[2]);
	printf("%u\n\n", &dizi_b[2]);

	/*-----------------------------------------
	 * Dizinin pointer olarak kullanilmasi
	 *-----------------------------------------*/
	printf("dizinin pointer olarak kullanilmasi\n");
	printf("===================================\n");

	printf("dizi_b[2] : %d\n", dizi_b[2]);
	printf("*(dizi_b + 2) : %d\n\n", *(dizi_b + 2));

	printf("&dizi_b[2] : %u\n", &dizi_b[2]);
	printf("(dizi_b + 2) : %u\n", (dizi_b + 2));

	return 0;
}
```