1 - Pointer tipinde değişken tanımlama

```c
int main() {
	int a = 5;

	printf("a'nin degeri: %d\n\n", a);

	printf("a'nin adresi: %p\n", &a);  // adresin 16'lik sayi tabaninda degeri
	printf("10'luk tabanda: %u\n\n", &a); // adresin 10'luk sayi tabaninda degeri

	int * a_ptr;
	// a'nin adresini a_ptr (pointera) atiyoruz
	a_ptr = &a;

	// pointer'in degerini yazdiriyoruz (a_ptr = a'nin adresi)
	printf("a_ptr : %p\n", a_ptr);

	// pointer'in gosterdigi yerin degerini yazdiriyoruz (a'nin degeri)
	printf("*a_ptr : %d\n\n", *a_ptr);

	// pointer'in gosterdigi yerin degerini degistiriyoruz
	*a_ptr = 32;

	// a'nin degerini yazdiriyoruz
	printf("a'nin degeri: %d\n", a);

	return 0;
}
```

2.1 - fonksiyona adres gönderme

```c
void atama_yap_1(int x) {
	// x -> fonksiyona gelen degeri saklayan degisken

	x = 20; // fonksiyondaki x degiskenini degistiriyoruz
}

void atama_yap_2(int *x) {
	// x -> fonksiyona gelen adresi saklayan degisken (pointer)

	*x = 30; // pointerin gosterdigi yerin degerini degistiriyoruz
}

int main() {
	int a = 5;
	printf("a: %d\n", a);

	atama_yap_1(a); // fonksiyona a'nin degeri gonderiyoruz
	printf("a: %d\n", a); // a'nin degerini yazdiriyoruz

	atama_yap_2(&a); // fonksiyona a'nin adresini gonderiyoruz
	printf("a: %d\n", a); // a'nin degerini yazdiriyoruz

	return 0;
}
```

2.2 - fonksiyona adres gönderme

```c
void takas_1(int x, int y) {
	// x ve y gelen degerleri saklayan degiskenler

	// bu degiskenlerde yer degistirme yapiyoruz
	int tmp = x;
	x = y;
	y = tmp;
}

void takas_2(int *x, int *y) {
	// x ve y fonksiyona gelen adresleri sakliyor

	// adreslerin gosterdigi yerlerin degerlerini degistiriyoruz
	int tmp = *x;
	*x = *y;
	*y = tmp;
}

int main() {
	int a = 1;
	int b = 2;

	printf("a: %d - b: %d\n", a, b);
	takas_1(a, b);
	printf("a: %d - b: %d\n\n", a, b);

	printf("a: %d - b: %d\n", a, b);
	takas_2(&a, &b);
	printf("a: %d - b: %d\n", a, b);

	return 0;
}
```

2.3 - fonksiyona adres gönderme

```c
// fonksiyondan birden fazla degeri dondurmek istersek pointer kullanmaliyiz

/**
 * girilen degeri saat, dakika ve saniye olarak parcalar
 */
void saat_cevir(int toplam_saniye, int *saat, int *dk, int *sn) {
	*sn = toplam_saniye % 60;

	*dk = (toplam_saniye / 60) % 60;

	*saat = (toplam_saniye / 3600);
}

int main() {

	while (1) {
		int saat, dakika, saniye;
		int girilen_sayi;

		// kullanicidan bir sayi istiyoruz
		printf("sayi girin: ");
```

```
            scanf("%d", &girilen_sayi);

            // fonksiyona girilen_sayi'yi ve
            // saat, dakika, saniye degiskenlerinin adreslerini gonderiyoruz
            saat_cevir(girilen_sayi, &saat, &dakika, &saniye);

            printf("%d saat %d dakika %d saniye\n\n", saat, dakika, saniye);
        }

        return 0;
}
```

3 - sizeof operatörü

```
// pointer boyutu ve long kullanilan islemciye ve isletim sistemine gore degisiklik gosterebilir
int main() {
        short a;
        int b;
        long c;
        char d;

        int dizi[5];
        int *ptr;

        printf("sizeof(short) : %d\n", sizeof(short));
        printf("sizeof(int) : %d\n", sizeof(int));
        printf("sizeof(long) : %d\n", sizeof(long));
        printf("sizeof(char) : %d\n", sizeof(char));

        printf("\n");

        printf("sizeof(a) : %d\n", sizeof(a));
        printf("sizeof(b) : %d\n", sizeof(b));
        printf("sizeof(c) : %d\n", sizeof(c));
        printf("sizeof(d) : %d\n", sizeof(d));

        printf("\n");

        printf("sizeof(dizi) : %d\n", sizeof(dizi));
        printf("sizeof(ptr) : %d\n", sizeof(ptr));

        return 0;
}
```

4 - pointer aritmetiği

```
int main() {
        int a;
        int * a_ptr = &a;

        // printf("baslangictaki adres degeri: %p\n", a_ptr); // 16'lik taban
        printf("baslangictaki adres degeri: %u\n", a_ptr); // 10'luk taban

        a_ptr++;
        // printf("1 eklendigindeki adres degeri: %p\n", a_ptr);
        printf("1 eklendikten sonra adres degeri: %u\n", a_ptr);
```

```
        a_ptr -= 2;
        // printf("2 eklendigindeki adres degeri: %p\n", a_ptr);
        printf("2 cikarildiktan sonra adres degeri: %u\n", a_ptr);

        a_ptr--;
        // printf("2 eklendigindeki adres degeri: %p\n", a_ptr);
        printf("1 cikarildiktan sonra adres degeri: %u\n", a_ptr);

        printf("\n");
        printf("a'nin 2 sonraki adresi: %u\n", (&a) +2);
        printf("a'nin 1 onceki adresi: %u\n", (&a) -1);

        return 0;
}
```

5 - pointer dizi ilişkisi

```
int main() {
        int i;

        int a[5] = {10, 20, 30, 40, 50};
        int *aptr;

        aptr = a;
        // aptr = &a[0]

        // dizinin 2. elemanini yazdiriyoruz
        printf("a[2] : %d\n", a[2]);

        // pointer'in 2 sonraki adresinin degerini yazdiriyoruz
        printf("*(aptr+2) : %d\n", *(aptr+2));

        printf("\n");

        // pointer dizi yazim sekliyle kullanilabilir
        printf("aptr[2] : %d\n", aptr[2]);

        // dizi pointer yazim sekliyle kullanilabilir
        printf("*(a+2) : %d\n", *(a+2));

        printf("\n");

        // dizinin 2. indexteki elemaninin adresi
        printf("&a[2] : %d\n", &a[2]);
        printf("a+2 : %d\n", a+2);

        return 0;
}
```

6 - pointer dizi ilişkisi kullanım örneği

```
void ekrana_yaz(int d[], int N) {
        int i;
        for (i = 0 ; i < N ; i++)
                printf("%d\n", d[i]);
        printf("\n");
```

```
}

// yukaridaki fonksiyon ile tamamen ayni sekilde calisir
void ekrana_yaz_2(int *d, int N) {
	int i;
	for (i = 0 ; i < N ; i++)
		printf("%d\n", d[i]);
	printf("\n");
}


int main() {

	int dizi[10] = {0, 10, 20, 30, 40, 50, 60, 70, 80, 90};

	// ilk 2 elemani yazmak istersek
	ekrana_yaz(dizi, 2);
	// ekrana_yaz_2 fonksiyonunda da ayni sekilde kullanilabilir
	// ekrana_yaz_2(dizi, 2);


	// 2'ciden itibaren 3 tane eleman yazmak istersek
	ekrana_yaz(&dizi[2], 3);

	// usttekinin diger yazim sekli
	ekrana_yaz(dizi+2, 3);

	return 0;
}
```