42 C01 Çözümleri

Bir pointer (işaretçi), bir değişkenin hafızadaki adresini tutan bir değişkendir.
Yani, bir pointer aslında şunu söyler:
"Ben bir kutuyum ve başka bir kutunun (değişkenin) nerede olduğunu biliyorum."

Betimlemeli Örnek Verecek Olursak Eğer;

Düşün ki evde bir oyuncak kutun var. Oyuncak kutusu 'değer', kutunun adresi ise 'pointer' gibi çalışır. Mesela:

- Kutunun içinde bir araba var (Değeri 5)
- Kutunun üstünde yazan "Raf No: 3, Sıra: 2" ise pointer'ın gösterdiği adres.

Bu Kutucukların Her biri 1 byte alandır

Adres	Ram Odacıkları
114	
113	
112	
111	
110	
109	
108	
107	***ptr**
106	
105	
104	
103	**int x = 25;**
102	
101	
100	

NOT: Bir değişken atandığı zaman, RAM üzerinde veri tipine bağlı olarak belirli miktarda bir alan işgal eder. Bu alan, programın bellek düzenine göre belirlenen bir adreste konumlanır.

1) Mesela int x = 25; dedik Ayrıldığı Alan ise 100. Adres olsun fakat 100’ den sonraki 4 byte lık alanı işgal etti. Çünkü integer 4 byte yer kaplar

2) Pointer Atanırken hangi veri türünün adresini tutacağıda önemli bir konudur. İnteger bir veri tipini mi tutuyor yoksa char mı ? float mı ? Bunları belirlemen gerekiyor.

İnteger bir pointer tutacağımdan;
int * ptr = &x;
(1)(2)(3) (4)(5)

1) Tutulacak olan pointer değişken türü.
2) Pointer tanımlama işaretidir.
3) Tanımlanan pointer değişken ismidir.
4) Önüne geldiği değişkenin ram'deki adresini verir.
5) Yukarıda tanımlı olan integer x değeri

Yani burada kısaca şunu yaptık ptr'nin içerisine integer x 'in ramdeki adres alanını atamış olduk.

4 Soruda Aslında Bunu Cevaplandıralım Yukarıdaki Örneğe Göre

ptr	**100**
***ptr**	**25**
&ptr	**107**
&x	**100**

& operatörü, bir değişkenin bellekteki adresini dönerken,
***ptr operatörü pointer'ın işaret ettiği adreste saklanan değeri gösterir.**

Pointer'a bir değer atanmamışsa tanımsız bir adres tutar. Sadece pointer adı (ör. ptr) kullanıldığında, bellekte tuttuğu adres döner.

ex00:

- Bu kod, bir sayının değerini değiştirmeyi gösterir. ft_ft fonksiyonu, bir sayının değerini 42 yapar.

```
#include <stdio.h>

void ft_ft(int *nbr)
{

*nbr = 42;  Pointer'ın işaret ettiği bellekteki adrese 42 değerini yazıyoruz.

}

int main()
{
 int number = 10; Bir integer değişken tanımladık ve ona 10 değerini atadık.
    printf("Pointer kullanmadan önce değer: %d \n", number); number'ın mevcut değerini yazdırıyoruz.
    ft_ft(&number); Fonksiyon, number'ın adresini alarak o adresin içindeki değeri değiştiriyor.
    printf("Pointer ile değiştirildikten sonra değer: %d \n", number); Değerin değiştiğini yazdırıyoruz.
    return (0);
}
```

Soru: bellek adresini veriyor & işareti nasıl bunu yazarak bu değeri nasıl değiştirebiliyoruz ?

ft_ft(int *nbr) fonksiyonu, bir adres (pointer) alıyor.
ft_ft(&number) çağrısında

- &number → number değişkeninin adresi fonksiyona gönderiliyor.
- Fonksiyon içinde *nbr = 42; yazıldığında, bu adresin içindeki değer 42 oluyor.

ex01:

Bir işaretleyiciyi diğer işaretleyicilere bağlayıp sonunda tam sayıya 42 atayan bir fonksiyon oluşturmamız isteniyor.

```
#include <stdio.h>
void ft_ultimate_ft(int *********nbr)
{
*********nbr = 42;
}
```

9 seviyeli bir pointer var. Bunun mantığı şu;
Her seviyedeki pointer, bir üst seviyedeki pointer'ın adresini tutuyor. En son içteki pointer'a ulaşıldığında, o pointer gösterdiği adresin içeriğini değiştiriyor, yani pointer'a atanan değeri değiştirmiş oluyor.

```
int main()
{
int number = 10;

int *ptr1 = &number;
int **ptr2 = &ptr1;
int ***ptr3 = &ptr2;
int ****ptr4 = &ptr3;
int *****ptr5 = &ptr4;
int ******ptr6 = &ptr5;
int *******ptr7 = &ptr6;
int ********ptr8 = &ptr7;
int *********ptr9 = &ptr8;

printf(“Pointerdan Önceki Değer : %d, \n”,number);
ft_ultimate_ft(ptr9);
printf(“Pointerdan Sonraki Değer : %d, \n”,number);
return (0);
}
```

Burada:

- ptr1: number'ın adresini tutar.
- ptr2: ptr1'in adresini tutar.
- ptr3: ptr2'nin adresini tutar.
- Bu şekilde, her pointer bir öncekini işaret eder.

*ptr1'in adresi, *********ptr9'a kadar gidiyor. **Çünkü her seviyedeki pointer bir üst seviyedeki pointer'ın adresini tutuyor ve en sonunda ptr9 ile number'ın adresine ulaşmış oluyorsunuz.**

Ptr9'un içerisinde aslında number'ın bellekteki adresi var ve fonksiyon, pointer'ın **içindeki adresi dereference ederek number'ın değerini 42 yapıyor.**

- ÖNEMLİ

Number'ın ptr9 kadar geldiğinin kanıtı ise ptr8 bellek adresini verincede sonucun doğru çıkması yani fonksiyon içine **&ptr8 dersende çalışacaktır...**

Dereference ne demek ?
Bir işaretleyici bir bellek adresini tutar. Dereference işlemi, o adresin içindeki veriye ulaşmak ve o veriyi kullanmak anlamına gelir.

ex02:

Burada pointer kullanarak girilen iki tam sayının yerlerini değiştirmemiz isteniyor bunu şöyle hayal etmelisin;

3 Adet Bardak Düşün Bunların Rengi Kırmızı Beyaz Mavi Olsun biz Kırmızının değerini Maviye Mavinin Değerini ise Kırmızıya Aktarmamız gerek bunu direkt yaparsak karışacak bunun için

1) Kırmızı Bardaktakini Beyaz Bardağa bırak.
2) Mavinin Değerini Kırmızıya geçir.
3) Beyazın içindeki kırmızının değerlerlerini maviye geçir

Beyaz bardak burada ne oldu bir yardımcı bizde yardımcı bir int değişken oluşturacağız.

```
#include <stdio.h>
void ft_swap(int *a, int *b)
{

int temp = *a;  // Kırmızı beyazın içine aktarıldı
*a = *b;       // Mavi Kırmızının içine aktarıldı
*b = temp;    // Beyaz bardağın içindeki özellikler mavi bardağa aktarıldı

}

int main()
{
    int number_a = 10; int number_b = 20;
    printf("Swap Kullanıldıktan Önce : %d, %d \n",number_a, number_b);
    ft_swap(&number_a, &number_b);
    printf("Swap Kullanıldıktan Sonra : %d, %d, \n", number_a, number_b);
    return (0);
}
```

ex03:

Aslında Pek Bir şey Yok Burada bize verilen prototipleri kullanarak bunu yapıyoruz

*div içine a ile b ‘nin bölündükten sonra çıkan sonucu aktaracağız
*mod içine a ile b ‘nin modunu aldıktan sonra kalan sayıyı atacağız

NOT: Koşul koymamızın sebebi bir sayının 0’a bölümü tanımsızdır o yüzden b 0 olamaz.

```
#include <stdio.h>
void ft_div_mod(int a, int b, int *div, int *mod)
{
    if(b != 0)
    {
        *div = a / b ;
        *mod = a % b ;
    }
}

int main()
{
    int num1 = 10;
    int num2 = 2;
    int div; int mod;

printf("Sayıların İşleme Girmeden Önceki Hali : %d , %d \n", num1,num2);
ft_div_mod(num1, num2, &div, &mod);
printf("Sayılar İşleme Girdikten Sonraki Hali : %d , %d \n", div, mod);
return (0);

}
```

ex04:

Kodun amacı
2 tamsayıyı bölmek ve sonuçları pointer'lar aracılığıyla geri döndürmek

```
#include <stdio.h>
void ft_ultimate_div_mod(int *a, int *b)
{
      if(*b!=0 && b!= NULL)
      {
            int temp_a = *a;
            int temp_b = *b;
                  *a = temp_a / temp_b;
                  *b = temp_a % temp_b;
      }
}
```

1) ***b!=0 dediğimizde*** pointer içindeki değere erişip onun 0 eşit değilsen yap diyoruz çünkü bir sayının 0'a bölümü tanımsızdır.
2) **b!=NULL ise** bellekteki adresi kontrol eder eğer boş ise girmez.
3) **temp_a ve temp_b** değişkenlerini kullanarak ***a ile *b nin işlemden önceki değerlerini buraya saklıyoruz.**
4) Ardından *a 'nın içerisine bölümün sonucunu atarak güncelliyorum
5) Ardından *b 'nin içerisine kalanın sonucunu atarak güncelliyorum.

```
int main()
{
int a = 10; int b = 0;
printf("Öncesi : %d %d \n", a,b);
ft_ultimate_div_mod(&a,&b);
printf("Sonrası : %d %d \n", a,b);
return (0);
}
```

ex05:

Bir karakter dizisinin içeriğini standart çıktıya (ekrana) yazdıran bir fonksiyon yazmamız isteniyor.

```
void ft_putstr(char *str)
{
      int i = 0;
      while(str[i] != '\0')
      {
            write(1,&str[i],1);
            i++;
      }
}
```

1) i değişkeni bizim indisimizi kontrol ediyor aslında, indis 0 dan başlar;

E = 0. İndis Değeri
N = 1. İndis Değeri
E = 2. İndis Değeri
S = 3. İndis Değeri

2)

- while(str[i] != '\0') bu döngüde ise dizimizin değerlerini 0 dan başlatarak okunmasını sağlıyoruz ve bu döngü karakter sonunu görünceye kadar devam edecek.

- '\0' Bu, karakter dizisinin sonunu belirten özel bir karakterdir (null karakter). Her karakter dizisi, sonunda otomatik olarak '\0' ile biter.

3)

write(1, &str[i], 1) ve i++, şartlar sağlandığı sürece dizi içindeki karakterleri tek tek yazar. Çünkü char veri tipi tek bir karakter taşıdığı için birden fazla karakteri aynı anda yazamaz. Şart sağlandıkça i değeri 0, 1, 2 şeklinde artarak devam eder ve karakterler sırasıyla ekrana yazdırılır

```
int main()
{
      ft_putstr(“STDOUT”);
      return 0;
}
```

ex06:

Bir karakter dizisindeki elemanları sayan ve bu sayıyı döndüren bir fonksiyon oluşturmamız isteniyor.

```
int     ft_strlen(char *str)
{
    int i = 0;
    while (str[i] != '\0')
    {
        i++;
    }
    return (i);
}
```

```
int main()
{
printf(“Karakter Uzunluğu : %d”,ft_strlen(“STDOUT”));
return (0);
}
```

ex07:

Bu fonksiyon, verilen bir tamsayı dizisini tersine çevirir. Yani, ilk eleman sona, son eleman başa gelecek şekilde değiştirilir.

Örnek:
Başlangıç Dizisi: [1, 2, 3, 4, 5]
Tersine Çevrildikten Sonra: [5, 4, 3, 2, 1]

İşleyiş Mantığı:

- İlk ve son elemanı yer değiştir.
- İkinci ve sondan ikinci elemanı yer değiştir.
- Ortaya ulaşana kadar bu işlemi sürdür.

```
void ft_rev_int_tab(int *tab, int size)
{
    int i = 0;
    int temp;

    while (i < size / 2)
    {
        temp = tab[i];
        tab[i] = tab[size - 1 - i];
        tab[size - 1 - i] = temp;
        i++;
    }
}
```

1) Fonksiyon Parametreleri:

- *tab: Tersine çevrilecek diziyi ifade eder.
- size: Dizinin kaç elemanlı olduğunu belirtir.

2) Değişkenler:

- i: Dizinin elemanlarını takip etmek için kullanılır.
- temp: Yer değiştirme sırasında veri kaybını önlemek için geçici olarak elemanı saklar.

3) Döngü Mantığı (while döngüsü):

- Döngü, dizinin **yarısına kadar** çalışır (size / 2).
- Çünkü her adımda baştan ve sondan elemanlar değiştirildiği için, dizinin yarısına gelindiğinde tüm elemanlar tersine dönmüş olur.

Örnek Çalışma:
Başlangıç: [1, 2, 3, 4, 5]

1. Adım → [5, 2, 3, 4, 1] → (0. indis ile 4. indis değiştirildi)
2. Adım → [5, 4, 3, 2, 1] → (1. indis ile 3. indis değiştirildi)
   Dizinin ortasına ulaşıldığı için işlem tamamlandı.

4) Swap (Yer Değiştirme) Mantığı:

- temp = tab[i]; → İlk değeri geçici değişkende sakla.
- tab[i] = tab[size - 1 - i]; → Son elemanı, ilk elemana ata.
- tab[size - 1 - i] = temp; → İlk elemanı, son elemana ata.

5) size - 1 - i Mantığı:

- size - 1, dizinin son indisini temsil eder (tab[4]).
- -i, her döngü adımında bir önceki elemanı seçer.
- Döngü devam ettikçe baştan ve sondan elemanlar değiştirilmeye devam eder.

```
int main()
{
    int tab[5] = {1, 2, 3, 4, 5};
    int size = 5;

printf("İşlemden Önce Dizi: %d %d %d %d %d \n", tab[0], tab[1], tab[2], tab[3],
tab[4]);

    ft_rev_int_tab(tab, size);

printf("İşlemden Sonra Dizi: %d %d %d %d %d \n", tab[0], tab[1], tab[2], tab[3],
tab[4]);

    return (0);
}
```