42 C03 Çözümleri

ex00:

Strcmp fonksiyonu, iki karakter dizisini karşılaştırmak için kullanılır. Karakterler sırasıyla karşılaştırılır ve ilk farklılık bulunduğunda ASCII farkı döndürülür. Eğer diziler aynıysa, 0 döner. Farklıysa, iki karakterin ASCII değeri arasındaki farkı döndürür.

```
#include <stdio.h>
int     ft_strcmp(char *s1, char *s2)
{
    int i = 0;
    while (s1[i] == s2[i])
    {
        i++;
    }
    return s1[i] - s2[i];
}

int main ()
{
    printf("Ascii Değer Farkı : %d \n",ft_strcmp("code", "coder"));
    return (0);
}
```

\0 ascii değeri = 0
r ascii değeri = 114
0 - 114 = -114

ft_strcmp, iki karakter dizisini (s1 ve s2) karşılaştırır.

- while (s1[i] == s2[i]) ile iki dizinin her karakteri karşılaştırılır. Eşit oldukları sürece döngü devam eder.
- İlk farklılık bulunduğunda, döngü sona erer ve return s1[i] - s2[i]; ile ASCII farkı hesaplanır.
- Eğer s1[i] ve s2[i] farklıysa, bu fark döndürülür.
- Eğer aynıysa ve dizilerin sonuna kadar eşitse, 0 döndürülür.

ex01:

Bu fonksiyon, iki karakter dizisini (s1 ve s2) karşılaştırır. Ancak, yalnızca ilk **n** karakteri dikkate alır. Eğer ilk n karakter eşitse, 0 döndürür. Eşit değilse, ilk farklı karakterin ASCII farkını döndürür.

```
#include <stdio.h>
int     ft_strncmp(char *s1, char *s2, unsigned int n)
{
    unsigned int i = 0;
    while (i < n && s1[i] == s2[i])
    {
        i++;
    }
    if(i == n)
    {
        return (0);
    }
    return s1[i] - s2[i];
}

int main ()
{
    printf("Ascii Değer Farkı : %d \n",ft_strncmp("code","coder",5));
    return (0);
}
```

i < n;
Bu koşul, döngünün yalnızca n kadar karakteri karşılaştırmasını sağlar. Yani, ilk **n** karakteri karşılaştıracak şekilde sınır koyar. Eğer i değeri n'yi geçerse, döngü durur. Yani, **n**'den fazla karakter karşılaştırmak mümkün olmaz.

s1[i] == s2[i];
Bu kısımda, dizilerin (s1 ve s2) aynı pozisyondaki karakterleri karşılaştırılır. Eğer **eşitlerse**, döngü devam eder, i değeri bir artırılır ve bir sonraki karaktere geçilir.

i == n

Eğer döngü **n** karakteri tamamlayana kadar bitmeden çıkmazsa (yani diziler eşittir ve n karakterin tamamı aynıysa), fonksiyon **0** döndürür.

- Eğer bir farklılık bulunursa, fonksiyon **s1[i] - s2[i]** ile bu iki karakterin ASCII farkını döndürür.
- Yani, ilk farklı karakterin ASCII değeri arasında fark hesaplanır.

ex02:

```
#include <stdio.h>
char *ft_strcat(char *dest, char *src)
{
    int i = 0;
    int j = 0;
    while (dest[i] != '\0')
    {
        i++;
    }
    while (src[j] != '\0')
    {
        dest[i] = src[j];
        i++;
        j++;
    }
    dest[i] = '\0';

    return dest;
}

int main ()
{
    char src [10] = "Dünya";
    char dest[15] = "Merhaba ";
    printf("Birleştirilmemiş Metin (SRC) : %s \n",src);
    printf("Birleştirilmemiş Metin (DEST) : %s \n",dest);
    ft_strcat(dest, src);
    printf("Birleştirilmiş   Metin : %s \n",dest);
    return (0);
}
```

ft_strcat Fonksiyonu:

Parametreler:

- char *dest: Birleştirilecek olan hedef string (sonuç burada saklanacak).
- char *src: Hedef string'e eklenecek kaynak string.

Değişkenler:

- int i = 0; dest string'indeki son karakterin bulunduğu indexi tutacak.
- int j = 0; src string'indeki karakterlerin yerini tutacak.

İlk while Döngüsü (Dest'in sonuna kadar gitmek):

- while (dest[i] != '\0'): Bu döngü, dest dizisinin sonuna kadar gidiyor. Çünkü '\0' (null terminator) string'in sonunu belirtir. Bu döngüde, i indexi artarak dest string'inin sonuna ulaşır.

İkinci while Döngüsü (Src'yi dest'e eklemek):

- while (src[j] != '\0'): Bu döngüde, src dizisinin her bir karakteri, dest dizisinin sonuna eklenir. i burada dest dizisinin sonuna işaret ederken, j ise src dizisindeki karakterleri işaret eder.
- dest[i] = src[j]; Bu satır, src dizisindeki j'uncu karakteri dest dizisinin i'nci pozisyonuna kopyalar.
- i++ ve j++: i ve j değişkenlerini artırarak bir sonraki karaktere geçilir.

Son dest[i] = '\0';

- '\0', string'in sonunu belirten null karakteridir. Bu satır, dest string'inin sonuna eklenmiş olan yeni stringi tamamlar.

Fonksiyon Sonu:

- return dest; Son olarak, birleştirilmiş dest string'ini geri döndürür.

ex03:

```
#include <stdio.h>
char *ft_strncat(char *dest, char *src, unsigned int nb)
{
    unsigned int i = 0;
    unsigned int j = 0;

    while (dest[i] != '\0')
    {
        i++;
    }

    while (src[j] != '\0' && j < nb)
    {
        dest[i] = src[j];
        i++;
        j++;
    }

    dest[i] = '\0';

    return dest;
}

int main ()
{
    char src [10] = "Dünya";
    char dest[15] = "Merhaba ";
    printf("Birleştirilmemiş Metin (SRC) : %s \n",src);
    printf("Birleştirilmemiş Metin (DEST): %s \n",dest);
    ft_strncat(dest,src,4);
    printf("Birleştirilmiş   Metin : %s \n",dest);
    return (0);
}
```

Parametreler:

1) char *dest: Hedef string, yani birleştirilmiş sonucu tutacak olan string.
2) char *src: Kaynak string, yani dest'e eklenecek olan string.
3) unsigned int nb: src string'inden kaç karakterin alınacağını belirten sayısal değer.

Değişkenler:

1) unsigned int i = 0; dest string'indeki son karakterin bulunduğu indexi tutar. Bu, dest'in sonuna yeni karakterleri eklemek için kullanılır.
2) unsigned int j = 0; src string'indeki karakterleri tutan değişken.

İlk while Döngüsü (Dest'in sonuna kadar gitmek):

1) while (dest[i] != '\0'): Bu döngü, dest dizisinin sonuna kadar gider.
2) Yani, dest string'indeki son karakterin bulunduğu indexi bulur.

İkinci while Döngüsü (Src'den nb kadar karakter almak ve eklemek):

1) while (src[j] != '\0' && j < nb): Bu döngüde, src dizisindeki ilk nb karakteri dest dizisinin sonuna ekler.
2) dest[i] = src[j];: src dizisindeki j'uncu karakteri dest dizisinin i'nci pozisyonuna kopyalar.
3) i++ ve j++: i ve j değişkenlerini artırarak, bir sonraki karaktere geçilir.

Son olarak '\0' eklenir:

1) dest[i] = '\0'; Bu satır, dest dizisinin sonuna null karakter ekler, böylece yeni string doğru bir şekilde sonlandırılmış olur.

Sonuç:

1) return dest; Birleştirilmiş string'i geri döndürür.

ex04 :
ft_strstr fonksiyonunun implementasyonudur ve bir dizedeki (string) belirli bir alt diziyi aramak için kullanılır. Fonksiyon, aranan alt dizeyi bulduğunda, alt dizinin başladığı yeri gösteren bir işaretçi döndürür. Eğer alt dize bulunamazsa, NULL döndürülür.

Başlangıç Kontrolü (if (to_find[0] == '\0'))

1) Eğer to_find dizisinin ilk karakteri null karakter ('\0') ise, yani aranan dize boş bir dizeyse, fonksiyon hemen str dizisini geri döndürür. Çünkü her dizenin içinde boş dize (null karakter) bulunabilir.

Ana Döngü (while (str[i] != '\0')):

1) str dizisi üzerinde gezinmek için bir döngü başlatılır. i değişkeni, str dizisinin her bir karakterini kontrol etmek için kullanılır. Bu döngü, str dizisinin sonuna kadar ('\0' karakterine kadar) devam eder.

Alt Dizeyi Karşılaştırma Döngüsü

(while (to_find[j] != '\0' && str[i + j] == to_find[j])):

1) Bu döngüde, aranan alt dize (to_find) ile str dizisinin ilgili kısmı karşılaştırılır. j değişkeni, to_find dizisinin her bir karakterine karşılık gelir ve her iki dizinin karşılık gelen karakterleri eşit olduğunda döngü devam eder.
2) Eğer karakterler eşleşirse, j++ ile to_find dizisinin bir sonraki karakterine geçilir.

Alt Dizeyi Bulma:

1) Eğer to_find[j + 1] == '\0' kontrolü sağlanırsa, bu, tüm to_find dizisinin başarıyla eşleştiği anlamına gelir. Bu durumda, str + i ifadesi döndürülür; bu, str dizisindeki aranan alt dizinin başını işaret eder.

Ana Döngüde Artış:

1) Eğer alt dize henüz bulunmamışsa, ana döngüde i++ işlemi yapılır ve str dizisinin bir sonraki karakterine geçilir.

Alt Dize Bulunamazsa (return (0)):

1) Eğer tüm str dizisi tarandıysa ve alt dize bulunamadıysa, 0 (NULL işaretçisi) döndürülür.

```
#include <stdio.h>

char *ft_strstr(char *str, char *to_find)
{
    int i, j;

    if (to_find[0] == '\0')
        return (str);

    i = 0;
    while (str[i] != '\0')
    {
        j = 0;
        while (to_find[j] != '\0' && str[i + j] == to_find[j])
        {
            if (to_find[j + 1] == '\0')
                return (str + i);
            j++;
        }
        i++;
    }
    return (0);
}

int main()
{
    char str[] = "Merhaba dünya";
    char to_find[] = "dünya";
    printf("%s\n", ft_strstr(str, to_find));
    return (0);
}
```