**C ile Programlamada Yazı İşlemleri**

Öncelikle bilinmesi gereken en önemli kısım C dilinde yazılara yaklaşımın diğer dillerden biraz farklı olmasıdır. Örneğin string Javada bir sınıftır. C++’ta ise bir tür eş ismidir. Doğrudan bir primitive type olmasa da yazı işini doğrudan yapan bir type-like bir şey bulunmaktadır. C’de ise string diye bir şey doğrudan yoktur. Bunun yerine char diziler ve char pointerlar bulunmaktadır. Yaklaşım böyle olduğundan dolayı C dilinde yazı işlemleri biraz daha zahmetli bir şekilde yapılmaktadır. Bu yazı işlemlerine yardımcı olan çok sayıda standart kütüphane fonksiyonu bulunsa da C++ dilinin yanında buz dağının görünen kısmıdır diyebiliriz. C dilinde bilinmesi gereken önemli durumlardan bir tanesi de her yazının sonunda null karakter(\0) olduğudur. Örneğin “Mustafa” yazısı her ne kadar 7 elemanlı bir char dizi gibi görülse de bu aslında 8 elemanlıdır. Elbette buna 7 elemanlı bir char dizi kullanılarak da ilk değer verilebilir ancak bu durumda sonunda null karakter olmaz. Peki yazıların sonunda null karakter olmaması nasıl bir soruna yol açabilir?

1-) Standart kütüphanede bulunan tüm fonksiyonlar yazıların sonunda null karakter olmasına güvenilerek yazılmıştır. Bu yüzden sonunda null karakter olmayan bir yazıyla bu fonksiyonları kullanmak doğru sonuçlar vermeyecektir.

2-) Programcı yazılarla işlem yaparken yazının sonuna gelinip gelinmediği anlamak isteyebilir. Bunu yapmanın en iyi yöntemi şüphesiz sonda null karakter olacağından karakterlerin null karaktere eşit mi kontrolü yapılmasıyla mümkün olur. Bu yüzden sonunda null karakter olmayan bir yazı ile işlem yapmak oldukça zahmetlidir.

Örneğin aşağıdaki gibi bir tanımlama geçerli olsa bile sonda NULL karakter olmadığından dolayı standart kütüphanenin fonksiyonları kullanılırken yanlış sonuçlar elde edilir:

int main(void)

{

char str[4] = "Mert"; //Dikkat sonda \0 karakter yok.

}

Şüphesiz bu sağlıklı bir tanımlama değildir. Ancak kod her şeye rağmen legaldir.

Bir yazı tanımlaması sadece dizi olarak değil aynı zamanda char pointer olarak da tanımlanır. Ancak bu durumda decay kuralı devreye girer. Biri her elemanı char türden olan bir dizidir ancak diğeri bu diziyi gösteren bir pointerdır. Pointerlarla ilgili detaylar ileride ele alınacaktır.

Aşağıdaki örnekte char diziyi gösteren bir pointer belirtilmiş:

int main(void)

{

char\*str = "Mert";

}

**Anahtar Notlar :** bir char diziyle bir char pointer arasında bilinmesi gereken en önemli fark eğer char dizinin elemanı değiştiriliyorsa bir sorun oluşmaz. Ancak bir char pointer üzerinden yazının elemanlarının değiştirilme girişimi sentaks hatasına neden olur örneğin aşağıdaki örnekte bir undefined behaviour oluşur:

#include <stdio.h>

int main(void)

{

char\*str = "Mustafa";

str[1] = 'e';

printf("%s",str);//ub(undefined behaviour)

}

Ayrıca burada gösterildiği gibi printf fonksiyonu kullanılarak %s kullanılarak ekrana basma yapılabilir. Fakat bu örnekte bir tanımsız davranış söz konusu olacağından muhtemel hiçbir şey basılmayacak. Ancak aynı yazı eğer bir char[] türünden bir değişken ile tutulması durumunda herhangi bir karakterin değiştirilmesi bir tanımsız davranışa sebep olmaz. Aynı aşağıdaki örnekte gösterildiği gibi:

#include <stdio.h>

int main(void)

{

char str[] = "Mustafa";

str[1] = 'e';

printf("%s",str);//Tanımsız davranış yok!

}

Bu örnekte bir tanımsız davranış söz konusu olmayacak ve ekrana başarıyla “Mestafa” yazısı basılacaktır. Char pointerlar ve char diziler arası farklar mülakatlarda en sık sorulardan biridir.

**Klavyeden Yazı Alma**

C dilinde klavyeden yazı alma iki şekilde yapılabilir. Bunlar:

1-) standart kütüphanenin gets fonksiyonu kullanılarak Örneğin:

#include <stdio.h>

int main(void)

{

char str[20];

gets(str);

}

2-) Klavyeden sayı almak içinde daha önceden de sıkça kullanmış olduğumuz stdio.h başlık dosyasında bulunan scanf fonksiyonu kullanılarak. Örneğin:

#include <stdio.h>

int main(void)

{

char str[20];

scanf("%s",str);

}

Ancak dikkat edilmesi gereken bir durum scanf fonksiyonunda arasında boşluk bırakılan bir yazı girildiği takdirde scanf bu boşluğa new line muamelesi yaptığından dolayı sadece boşluk öncesindeki sayı klavyeden alınmış olur. Bu yüzden gets fonksiyonu genellikle daha efektif bir yapıya sahiptir.

**Standart Kütüphanenin Yazı İşlemleri Yapan Fonksiyonları**

Standart kütüphanenin string.h başlık dosyasının yazı işlemleri yapan strxxx fonksiyonları bulunmaktadır. İşte bunlardan bazıları:

1-) size\_t strlen(const char\* str): Bu fonksiyonun geri dönüş değeri unsigned long long türünden(size\_t aslında bu türün tür eş ismidir) olan ve const char\* türünden tek parametreye sahip bir fonksiyondur. Bu fonksiyon kendisine gönderilen yazının uzunluğunu geri döndürür. Örneğin:

#include <stdio.h>

#include <string.h>

int main(void)

{

char str[20];

printf("Bir yazi giriniz?\n");

gets(str);

printf("Yazinin Uzunlugu : %llu\n",strlen(str));

}

2-)char\* strcpy(char\* dest,const char\* source) : Bu fonksiyon bildiriminden de anlaşılabileceği gibi ikinci parametre ile gönderilen yazıyı ilk parametre ile gönderilen diziye kopyalayan bir fonksiyondur. Kısaca adından anlaşılabileceği gibi copyden gelmektedir. İkinci parametre de adından da anlaşılabileceği gibi kaynak yazıdır. Bu yüzden const yapılmıştır. İşte bu fonksiyonun bir kullanımı :

#include <stdio.h>

#include <string.h>

int main(void)

{

char s1[20];

char s2[20];

printf("Bir yazi giriniz?\n");

gets(s1);

strcpy(s2,s1);

printf("Kopyalanan Yazi : %s\n",s2);

}

3-)char\* strrev(char \* str) : char\* türünden geri dönüş değerine sahip char\* türünden tek parametreye sahip olan bir standart kütüphane fonksiyonudur. Bu fonksiyon kendisine gönderilen yazıyı tersten yazdırır. Örneğin :

#include <stdio.h>

#include <string.h>

int main(void)

{

char s1[20];

printf("Bir yazi giriniz?\n");

gets(s1);

strrev(s1);

printf("Kopyalanan Yazi : %s\n",s1);

}

4-) char\* strncpy(char\* dest, const char\* source, size\_t count) : Bu fonksiyon ikinci parametredeki yazının üçüncü parametre kadarını birinci parametredeki yazıya kopyalayan strcpy fonksiyonuna oldukça benzeyen bir fonksiyondur. Örneğin:

#include <stdio.h>

#include <string.h>

int main(void)

{

char s1[20];

char s2[20];

printf("Bir yazi giriniz?\n");

gets(s1);

int n;

printf("Kac karakter kopyalansın?\n");

scanf("%d",&n);

strncpy(s2,s1,n);

printf("%s\n",s2);

}

5-) char\* strchr(const char\* str, int val) : Bu ilk parametreli yazı içerisindeki ikinci parametrede kendisine gönderilen karakteri arayan bir fonksiyondur. Bu fonksiyon karakter bulunduğu takdirde bulunan karakteri konumunu döndürür. Eğer karakter bulunamazsa bu durumda NULL pointer döndürür. Örneğin:

#include <stdio.h>

#include <string.h>

int main(void)

{

char str[20];

printf("Bir yazi giriniz?\n");

gets(str);

char ch;

printf("Aranacak Karakter?\n");

scanf("%c",&ch);

if(strchr(str,ch))

printf("Karakter var\n");

else

printf("Karakter yok\n");

}

6-) int strcmp(const char\* str1, const char\* str2) : Bu fonksiyon geri dönüş değeri 1, 0 veya -1 olabilen bir fonksiyondur. Eğer kendisine gönderilen birinci parametredeki yazı büyükse 1, ikinci parametredeki yazı büyükse -1, yazılar aynıysa 0 döndüren bir fonksiyondur. Örneğin:

#include <stdio.h>

#include <string.h>

int main(void)

{

char s1[20];

char s2[20];

printf("Birinci yaziyi giriniz?\n");

gets(s1);

printf("ikinci yaziyi giriniz?\n");

gets(s2);

if(!strcmp(s1,s2))

{

printf("Yazilar ayni\n");

}

else

{

printf("Yazilar farkli\n");

}

}

**Anahtar Notlar :** Bu fonksiyonun kullanılmasının önemli bir nedeni yazıların karşılaştırma operatörleriyle karşılaştırılmasının mümkün olmamasından kaynaklı karşılaştırma operatörleriyle yalnızca adresler karşılaştırılır. Örneğin aşağıdaki gibi bir karşılaştırma doğru bir sonuç vermez:

#include <stdio.h>

#include <string.h>

int main(void)

{

char s1[20];

char s2[20];

printf("Birinci yaziyi giriniz?\n");

gets(s1);

printf("ikinci yaziyi giriniz?\n");

gets(s2);

if(s1 == s2)

{

printf("Yazilar ayni\n");

}

else

{

printf("Yazilar farkli\n");

}

}

Standart strcmp fonksiyonu lexicographic compare işlemi yapar. Bu da karakter birbiriyle baştan başlayarak karşılaştırılması sonucu olur. Örneğin UNICODE tabloda bir yazının ilk karakteri diğer yazının ilk karakterinden büyük ise bu durumda ilk ikinci yazıdan büyüktür. Diğer karakterlere bakılmaz. Örneğin büyük masa(MASA) ile küçük masa(masa) yazıları karşılaştırıldığında ‘m’ karakterinin UNICODE olarak değeri ‘M’ karakterinden büyüktür. Bu durumda “masa” yazısı “MASA” yazısından büyüktür. Örneğin aşağıdaki örnekteki kod ile bu test edilebilir:

#include <stdio.h>

#include <string.h>

int main(void)

{

char s1[20];

char s2[20];

printf("Birinci yaziyi giriniz?\n");

gets(s1);

printf("ikinci yaziyi giriniz?\n");

gets(s2);

if(strcmp(s1,s2) > 0)

{

printf("s1 yazisi s2 yazisindan buyuk\n");

}

else

{

printf("s2 yazisi s1 yazisinden buyuk\n");

}

}

Yazılarla birlikte pointerların kullanımı ilerde ele alınacaktır.