به نام کسي که آفريد از عدم به انسان عطا کرد فکر و قلم

موضوع مقاله: رشته ها در زبان برنامه نويسي C

تاريخ: 89/6/14

ارائه دهنده: فريبا رضائي

نام استاد: مهندس محمد سليمي

www.salimiteach.com

رشته ها در برنامه نويسي C

رشته ها يكي از مهمترين انواع داده ها در زبانهاي برنامه سازي هستند. يك رشته به يك توالي از صفر يا چند كاراكتر گفته مي شود. مثلا Ali يك رشته كاراكتري است. از رشته ها در موارد بسياري مانند ذخيره اسم و آدرس استفاده مي شود.
در زبان C نوع داده مشخصي براي رشته وجود ندارد، بلكه رشته به صورت آرايه اي از كاراكترها تعريف مي گردد. به عنوان مثال:

```
char name[10];
```

در مثال فوق متغير name به عنوان يك آرايه 10 عضوي از كاراكترها تعريف شده است و مي تواند يك رشته با حداكثر 10 كاراكتر را در خود نگه دارد. اما فرض كنيد قصد داريم رشته اي مانند Ali را در اين متغير دخيره نماييم كه كمتر از 10 كاراكتر دارد. در اين صورت زبان C چگونه در مي يابد كه در هنگام انجام عمليات مختلف بر روي اين رشته مثلا در هنگام چاپ آن فقط بايد 3 حرف اول رشته را چاپ نمايد؟ براي حل اين مشكل، طراحان زبان C از كاراكتر خاصي به نام null استفاده كرده اند. كليه رشته ها در زبان C بايد به كاراكتر null ختم گردند . در حقيقت در زبان C يك رشته هنگامي خاتمه مي يابد كه به null برسد و نه زماني كه به انتهاي آرايه برسد. كاراكتر null داراي كد اسكي 0 است و با '\0' نشان داده مي شود. بنابراين دقت كنيد كه در هنگام تعريف يك رشته، يك عنصر اضافه براي كاراكتر null در نظر بگيريد.
در زبان C امكان تعريف ثابت رشته اي نيز وجود دارد. يك ثابت رشته اي دنباله اي از كاراكترهاست كه در داخل " " قرار مي گيرد. به عنوان مثال "Ali"نشان دهنده يك ثابت رشته اي است . توجه كنيد كه هنگامي كه از " " براي يك ثابت رشته اي استفاده مي كنيد، كامپايلر يك علامت null در انتهاي رشته اضافه مي كند . از ثوابت رشته اي براي مقدار دهي اوليه به متغيرهاي رشته اي مي توان استفاده كرد. به عنوان مثال:

```
char name[10]="Ahmad";
```

آرايه name به اين صورت مقدار مي گيرد:

name	A	h	m	a	d	\0				

نكته ديگر اين كه به جز در هنگام مقداردهي اوليه، نمي توان در برنامه از عملگر = براي مقداردهي به يك رشته استفاده كرد. به عنوان مثال دستور زير خطاي نحوي محسوب مي گردد:

```
Char name[10];
name="Ali";
```

فراموش نكنيد كه رشته ها در حقيقت يك آرايه هستند و نمي توان كل يك آرايه را بايك دستور مقداردهي كرد . براي مقدار دهي به يك رشته، بايد به تك تك عناصر آن جداگانه مقدار داد و يا از توابع كتابخانه اي C استفاده كرد.

خواندن و نوشتن رشته ها

براي خواندن و نوشتن رشته ها مي توان از توابع scanf و printf استفاده كرد. تنها نكته اين است كه در داخل رشته بايد از مشخصه تبديل %s استفاده نمود. به عنوان مثال به نمونه زير دقت كنيد:

```
#include <stdio.h>
#include <conio.h>
void  main()
{
char name[20];
printf("what is your name?");
scanf("%s",name);
```

```
printf("Hello %s!",name);
}
```

اجرا:

```
what is your name? Ali
Hello Ali!
```

همان طور كه مي بينيد در هنگام ارسال متغير name به توابع scanf و printf تنها نام بدون [] استفاده شده است . نكته ديگر اين كه در هنگام ارسال name به تابع scanf از علامت & استفاده نشده است چون نام يك آرايه به تنهايي برابر با آدرس اولين عنصر آن آرايه است. در نتيجه name به تنهايي خود يك آدرس است.
اما برنامه فوق مشكلي دارد. به اجراي ديگري از همين برنامه توجه كنيد:

```
what is your name? Mohammad Reza
Hello Mohammad!
```

همان طور كه مي بينيد با وجود اين كه كلمه Mohammad Reza وارد شده است، اما تابع scanf فقط قسمت اول يعني Mohammad را در متغير name قرار داده است. دليل اين مسئله آن است كه تابع scanf به محض رسيدن به كاراكتر space گمان مي كند كه رشته خاتمه يافته است و از بقيه آن صرفنظر مي كند . براي رفع اين مشكل مي توان از تابع ديگري به نام gets استفاده كرد . اين تابع يك متغير رشته اي را به عنوان پارامتر ورودي دريافت و پس از خواندن يك رشته از صفحه كليد آن را در پارامتر ورودي قرار داده و باز مي گرداند. نكته مهم اين است كه تابع gets تا زماني كه كليد enter فشرده نشده است، به خواندن داده- ها از صفحه كليد ادامه مي دهد . بنابراين رشته مي تواند داراي space نيز باشد. به بازنويسي مثال قبل با تابع gets دقت كنيد:

```
#include <stdio.h>
#include <conio.h>
void  main()
{
char name[20];
printf("what is your name?");
gets(name);
printf ("Hello %s!",name);
}
```

اجرا:

```
what is your name? Mohammad Reza
Hello Mohammad Reza!
```

براي چاپ رشته ها نيز تابعي به نام puts وجود دارد. مثلا به جاي آخرين تابع printf در مثال بالا مي توان به شكل زير عمل كرد:

```
puts("Hello");
puts(name);
```

توابع كتابخانه اي رشته اي

زبان C داراي يك كتابخانه غني از توابع كار با رشته هاست. پيش تعريف اين توابع در فايل سرآمد string.h آمده است. در اين قسمت چند تابع از كتابخانه C به همراه الگوريتم آنها بررسي مي گردند. دقت كنيد كه ممكن است تعريف دقيق تابع كتابخانه اي در C با آنچه در اينجا آمده است كمي متفاوت باشد.

تابع strlen

نام اين تابع مخفف string length است. اين تابع يك رشته را دريافت كرده و طول آن را باز مي گرداند. منظور از طول رشته، تعداد كاراكترهاي آن تا رسيدن به null است.

تعريف اين تابع در زير آمده است:

```
int strlen(const char string[])
{
int i;
for(i=0;string[i];i++);
return(i);
}
```

متغير string به صورت ثابت ارسال شده است تا درداخل تابع به طور ناخواسته تغيير داده نشود. دقت كنيد كه ; پس از حلقه for نشان مي دهد كه بدنه اين حلقه خالي است. به اين معنا كه در هر بار اجراي حلقه، فقط عمليات مربوط به خود حلقه يعني افزايش شمارنده و بررسي شرط صورت مي پذيرد . علاوه بر اين شرط ادامه حلقه فقط string[i]قرارداده شده است كه توسط C اين گونه تفسير مي شود:" تا زماني كه string[i] درست است". مي دانيد كه هر عدد به جز 0 درست محسوب مي گردد، بنابراين تا هنگامي كه string[i] برابر null (كه كد اسكي آن 0 است) نباشد، درست محسوب مي گردد. به محض اين كه string[i] به null برسد، نادرست ارزيا بي شده و حلقه خاتمه خواهد يافت. مسلما" در اين حالت مقدار i، طول آرايه را نشان مي دهد . حلقه for را به صورت زير نيز مي توان نوشت:

```
for(i=0;stirng[i]!='\0';i++);
```

برنامه زير نحوه استفاده از strlen را نشان مي دهد:

```
void main()
{
char text[100];
printf("enter a text:");
gets(text);
int len=strlen(text);
printf("length of your text is %d",len);
}
```

اجرا:

```
enter a text:Hello
length  of your text is 5
```

تابع strcpy

نام اين تابع مخفف string copy است. اين تابع دو رشته را دريافت و رشته دوم را در رشته اول كپي مي كن د. تعريف اين تابع در زير آمده است:

```
void strcpy(char dest[],const char source[])
{
int i;
for(i=0;source[i];i++)
dest[i]=source[i];
dest[i]=\'0';
}
```

پارامتر dest رشته مقصد را نشان مي دهد كه رشته source در آن كپي خواهد شد. از آنجا كه پارامتر source نبايد تغيير كند، به صورت ثابت ارسال شده است . حلقه for تا زماني كه source[i] به null نرسيده تكرار مي شود و

عناصر رشته source را تك به تك در رشته dest كپي مي نمايد. به محض اين كه sourse[i] به null برسد، حلقه خاتمه مي يابد، در نتيجه خود كاراكتر '\0' در destكپي نخواهد شد. به همين دليل در انتها، آخرين عنصر آرايه dest برابر '\0' قرارداده شده است. برنامه زير نحوه استفاده از اين تابع را نشان مي دهد:

```
void main()
{
char string1[20],string2[20];
printf("please enter string1:");
gets(string1);
strcpy(string2,string1);
printf("copy string1 into string2\n");
printf("now string1 =%s and string2=%s\n",string1,string2);
strcpy(string1,"new");
printf("copy new into string1\n");
printf("now string1=%s",string1):
}
```

اجرا:

```
please enter string1:Hello
copy string 1 into string 2
now string1=Hello and string2=Hello
copy new into string1
now string1=new
```

نكته جالبي كه در مثال فوق ديده مي شود اين است كه چنانچه بخواهيم يك ثابت رشته اي را در يك متغير رشته اي قرار دهيم، مي توانيم از تابع strcpy استفاده كنيم.

تابع strcat

نام اين تابع مخفف string concat است. اين تابع دو رشته را دريافت و رشته دوم را به انتهاي رشته اول الحاق مي كند. به تعريف اين تابع دقت كنيد:

```
void strcat(char str1[],const char str2[])
{
int i,j;
for(i=0;str1[i];i++);
for(j=0;str2[j];j++)
str1[i+j]=str2[j];
str1[i+j]='\0';
}
```

در تابع فوق، قصد داريم رشته str2 را به انتهاي رشته str1 اضافه كنيم. حلقه for اول، دقيقا" همانند تابع strlenاست و طول رشته str1 را محاسبه و در متغير i قرار مي دهد. سپس در حلقه for دوم، از ابتداي رشته str2 شروع كرده و هر كاراكتر را به انتهاي رشته str1 اضافهمي كند. در پايان كاراكتر '\0' نيز به انتهاي رشته str1 الحاق مي گردد. براي آشنايي با نحوه فراخواني اين تابغ به برنامه زير دقت كنيد:

```
void main()
{
char string1[20],string2[20];
printf("please enter string1:");
gets (string1);
printf("please enter string2:");
gets (string2);
strcat(string1,string2);
```

```
printf("concatenate of string1 and string 2 is:%s",string1);
}
```

اجرا:

```
please enter string1:Hello
please enter string 2:everybody!
concatenate of string 1 and string2 is:Hello everybody!
```

تابع strcmp

نام اين تابع مخفف string compare است. اين تابع دو رشته را دريافت و پس از مقايسه آنها يك از 3 مقدار زير را برمي گرداند:

- در صورتي كه مساوي باشند: صفر
- در صورتي كه رشته اول بزرگتو باشد:1+
- در صورتي كه رشته دوم بزرگتر باشد:1-

نحوه مقايسه دو رشته، به همان ترتيبي است كه يك لغت نامه كلمات را مرتب مي كنند. يعني ابتدا حروف اول دو رشته مقايسه مي شود. اگر يكي از آنها بزرگتر بود كه نتيجه بازگردانده مي شود . اما در صورتي كه حروف اول دو رشته يكسان بود، حروف دوم با يكديگر مقايسه مي شود و اين عمل تا زماني كه يك اختلاف بين دو رشته پيدا شود ادامه مي يابد. در صورتي كه هيچ اختلافي بين دو رشته پيدا نشود، مقدار صفر بازگردانده مي شود. پياده سازي اين تابع در زير آمده است:

```
int strcmp(const char str1[],const char str2[])
{
int i=-1;
do
{
i++;
if(str1[i]>str2[i])
return(1);
if(str1[i]<str2[i])
return(-1);
}while(str1[i]);
return(0);
}
```

تابع فوق دو رشته str1 وstr2 را با يكديگر مقايسه مي كند. از آنجا كه هيچ يك از اين دو نبايد تغيير داده شوند، هر دو به صورت ثابت به تابع ارسال شده اند. حلقه do while عناصر اين دو رشته را با يكديگر مقايسه مي كند . توجه كنيد كه عمل مقايسه براي كاراكترها در زبان تعريف شده و در حقيقت كد اسكي آنها را با يكديگر مقايسه مي كند. در هر بار اجراي حلقه، چنان چه يكي از كاراكترهابرزگتر بود، بلافاصله حاصل بازگردانده مي شود. اما در صورتي كه هر دو مساوي باشند، حلقه دور زده و عمليات تكرار مي شود. به محض اين كه str1[i] به null برسد (كه در اين صورت str2[i] هم قطعا" به null رسيده است چون هر دو مساوي بوده اند) از حلقه خارج شده و مقدار 0 را به نشانه تساوي دو رشته باز مي گرداند.
براي آشنايي بيشتر به برنامه زير و اجراهاي مختلف آن توجه كنيد:

```
void main()
{
char string1[20],string2[20];
int result;
printf("please enter string1:");
gets(string1);
printf("please enter string2:");
```

```
gets(string2);
result=strcmp(string1,string2);
if(result= =0)
printf("%s equals %s\n",string1,string2);
else if(result= =1)
printf("%s is greater than %s\n",string1,string2);
else
printf("%s is less than %s\n",string1,string2);
}
```

اجرا:

```
1.
please enter string1:ali
please enter string2:ahmad
ali is greater than ahmad

2.
please enter string1:ali
please enter string2:alireza
ali is less than alireza

3.
please enter string1:ali
please enter string2:ali
ali equals ali
```

تابع strstr

اين تابع دو رشته را دريافت و در رشته اول به دنبال رشته دوم جستجو مي كند و در صورت پيدا شدن، مكان اولين كاراكتر آن را باز مي گرداند. در صورتي كه در رشته اول چندين نمونه از رشته دوم وجود داشته باشد، مكان اولين نمونه را باز مي گرداند. پياده سازي اين تابع در زير آمده است:

```
void strstr(const char str1[],const char str2[])
{
int i,j;
for(i=0;str1[i];i++)
if(str1[i]= = str2[0])
{
for(j=1;str2[j]&&str1[i+j]= =str2[j];j++);
if(!str2[j])
return(i);
}
return(-1);
}
```

حلقه for اول تك تك عناصر str1 را با اولين عنصر str2 مقايسه مي كند. چنانچه يكي از اين عناصر با str2[0] برابر بود، آن گاه حلقه for دوم شروع به مقايسه عناصر بعدي str1 (از محل i به بعد) و str2 (از محل 0 به بعد) مي نمايد. به شرط حلقه for دوم دقت كنيد. اين حلقه تا زماني كه str2 به انتها نرسيده و همچنين عنصر بعدي str2 و str1 با يكديگر برابر باشند تكرار مي شود. دقت كنيد كه بدنه حلقه خالي است. بعد از پايان حلقه بررسي مي گردد كه چنانچه دليل خروج از حلقه به پايان رسيدن str2 بوده است (يعني str2[j]برابر null شده است)، بنابراين رشته مورد نظر پيدا شده و مكان شروع آن يعني i بازگردانده مي شود. در غير اين صورت، دليل خروج مساوي نبودن دو كاراكتر از

دو رشته بوده است بنابراين عمليات جستجو ادامه مي يابد. البته تابع فوق بهينه نيست. چرا كه هنگامي كه در رشته اول شمارنده به مكاني برسد كه تعداد كاراكترهاي باقيمانده تا پايان رشته كمتر از اندازه رشته دوم باشد، قطعا" ادامه جستجو لزومي ندارد، چون كه رشته دوم پيدا نخواهد شد.
براي روشن شدن موضوع به برنامه زير دقت كنيد:

```
void main()
{
char text[100],word[20];
int i,n,result;
printf("enter a text:");
gets(text);
printf("how many words do you have:");
scanf(%d",&n);
for(i=0;i<n;i++)
{
printf("enter a word to search:");
gets(word);
result=strstr(text,word);
if(result= = -1)
printf("(%s) not found\n",word);
else
printf("(%s) is found in position %d\n",word,result);
}
}
```

اجرا:

```
enter a text:this is a sample text!
how many words do you have:3
enter a word to search:sample
(sample) is found in position 10
enter a word to search:is
(is) is found in position 2
enter a word to search:test
(test) not found
```

تابع strrev

نام اين تابع مخفف string reverse است. اين تابع يك رشته را دريافت و آن را معكوس مي كند. پياده سازي اين تابع به صورت زير است:

```
void strrev(char string[])
{
int i,j,k;
char temp;
for(k=0;string[k];k++);
for(i=0,j=k-1;i<j;i++,j--)
{
temp=string[i];
string[i]=string[j];
string[j]=temp;
}
}
```

تابع فوق يك رشته را از طريق پارامتر string دريافت و آن را معكوس مي نمايد . حلقه for اول طول رشته را محاسبه و در متغير k قرار مي دهد . سپس در حلقه for دوم، متغير i از ابتداي رشته شروغ به حركت رو به جلو و متغير j از انتهاي رشته شروع به حركت رو به عقب مي نمايد و در حين حركت جاي عناصر متناظر خود را با يكديگر عوض مي كنند. اين كار تا زماني كه i و j به يكديگر برسند تكرار مي شود.
برنامه زير نحوه كار اين تابع را نشان مي دهد:

```
void main()
{
char word[20];
printf("enter a word:");
gets(word);
printf("reverse of  %s is",word);
strrev(word);
printf("%s",word);
}
```

اجرا:

```
enter a word:hello
reverse of hello is olleh
```

توابعي كه مورد بحث قرار گرفتند، از جمله مهمترين توابعي بودند كه براي كار با رشته ها مورد نياز هستند . اما در فايل سرآيند string.h توابع مفيد ديگري نيز وجود دارند كه براي آشنايي با آنها مي توانيد به مستندات كامپايلر خود رجوع كنيد.

منبع: www.prdev.com