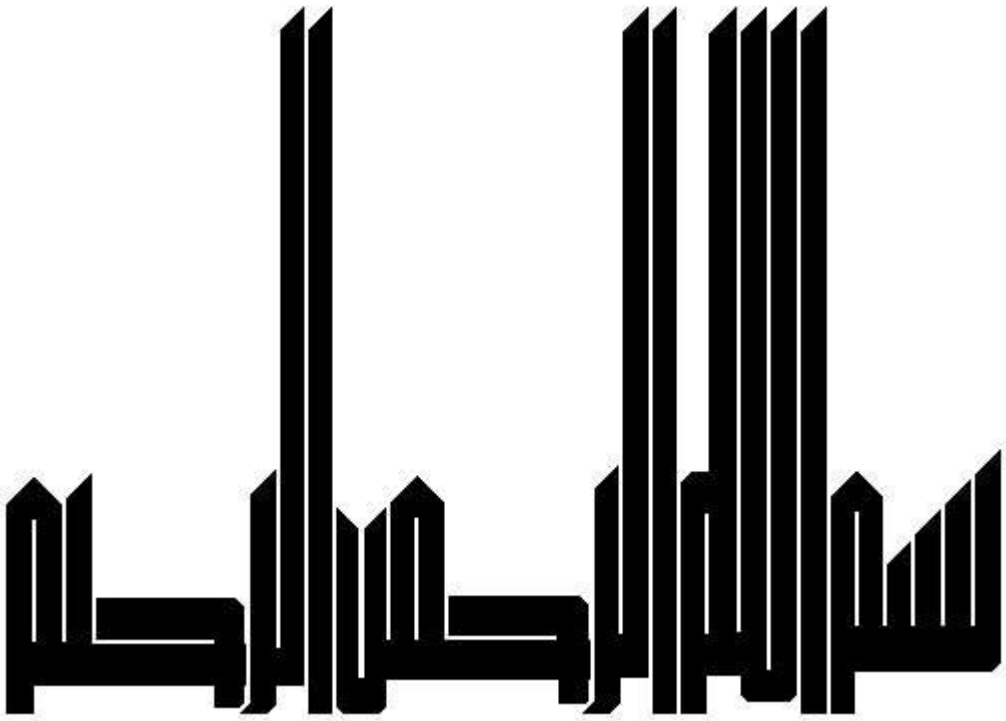

بسم الله الرحمن الرحیم

آرایه ها و رشته ها در زبان C

(کلیه مثالهای برنامه نویسی با کامپایلر codevisionAVR می باشد)

تهیه وتدوین

کاوه کیانمجد

kiyanmajd@gmail.com

WWW.Iseee.Ir

فهرست

آرایه ها:

آرایه اسمی برای چند متغیر هم نوع می باشد یا به عبارت دیگر آرایه از چندین کمیت درست شده است که همگی دارای یک نام می باشد و در خانه های متوالی حافظه ذخیره می گردند. هر یک از این کمیت ها را یک عنصر می گویند، برای دسترسی به عناصر آرایه باید اسم آرایه و شماره ی اندیس آرایه را ذکر کنیم. آرایه ها در زبان c از جایگاه ویژه ای برخوردارند، به طوری که در پروژه های خود به طور مکرر به آن برخورد خواهید کرد زیرا ارسال و دریافت داده به صورت رشته(آرایه ای از کاراکترها)و سریال انجام می شود.

نحو ه ی تعریف آرایه(تک بُعدی):

پیش از آنکه بتوان از یک آرایه یک بعدی استفاده کرد، باید آن را اعلان کرد. اعلان آرایه ها بصورت زیر انجام می گردد:

[تعداد عناصر آرایه] <اسم آرایه> <نوع آرایه>

```
Int a[10];        char str[20];        float[50];

Int A[10];
```

خط بالا یک آرایه ۱۰ تایی از اعداد صحیح بنام A ایجاد می نماید. هر کدام از عناصر این آرایه می توانند بعنوان یک متغیر مستقل مورد استفاده قرار گیرد. برای دسترسی به عناصر این آرایه باید از اندیس استفاده نمود. در زبان C اندیسها در داخل کروشه [] قرار می گیرند. نکته بسیار مهمی که باید بدان توجه کرد آنستکه در C اندیس یک عدد صحیح است که از ۰ آغاز می گردد. به مثال زیر توجه نمایید:

```
int A[10] ;
A[2] = 8;
```

و یا چنانچه بخواهیم مقدار خانه سوم را بر ۲ تقسیم و در متغیر x بریزیم، داریم:

```
x = A[2] / 2;
```

چند نکته مهم راجع به آرایه در C وجود دارد که حتما باید به آنها دقت کنید:

۱- اسم آرایه از قوانین نام گذاری متغیرها تبعیت می کند.
۲- در میکروکنترلر آرایه میتواند مانند دیگر متغیرها از نوع eeprom,flash,sram باشد.
۳- نوع آرایه از انواع اصلی در زبان C می باشد (char,double,float,int)
۴- اندازه آرایه ها در C ثابت بوده و حتما باید توسط یک مقدار ثابت صحیح تعیین گردد. بعنوان مثال اعلان زیر خطای نحوی محسوب می گردد:

```
int n ;
n=100 ;
int A[n];
```

اما می توان با استفاده از متغیر های ثابت (ثابتهای دارای نام)، اندازه آرایه را تعیین کرد، که در قسمتهای بعدی به آن اشاره خواهد شد.

۵- اندیس آرایه ها در C عدد صحیح بوده و همیشه از ۰ شروع می شود. لذا به تفاوت "عنصر چهار آرایه" یعنی A[4] و "چهارمین عنصر آرایه" یعنی A[3] دقت کنید. این مسئله معمولا باعث بروز خطاهای منطقی می گردد.

پس در زبان C اندیس آرایه از صفر شروع می شود.

Int kav[4];

kav[0]	kav[1]	kav[2]	kav[3]

۶- در C مرز آرایه ها بررسی نمی گردد. بدین معنا که چنانچه اندیسی خارج از محدوده مجاز یک آرایه استفاده شود، باعث ایجاد خطا توسط کامپایلر نمی گردد، اما مسلما برنامه را دچار یک خطای منطقی خواهد کرد. بعنوان مثال:

```
int A[10] ;
A[12] = 20 ; //this is not a syntax error but a logical error
```

لذا بررسی مرزهای آرایه بعهده خود برنامه نویس است و باید از درستی برنامه خود و خارج نشدن از محدوده مجاز مطمئن گردد.

۷- میزان حافظه ای که به آرایه اختصاص داده می شود،به این شکل استفاده می شود:

طول ارایه ضرب در (طول نوع آرایه)=میزان حافظه آرایه (برحسب بایت)

آرایه ۲ بُعدی:

در واقعه همان ماتریس است و موارد استفاده ان بسیار زیاد می باشد به طور مثال در میکرو برای ال سی دی گرافیکی، میکروموس ها، تعریف آرایه ای از رشته ها و.....

<نوع آرایه> <اسم آرایه> [طول بُعد اول] [طول بُعد دوم]

تعداد سطرها ↓ تعداد ستون ها ↓

Int [4][5]; این آرایه از نوع صحیح می باشد.

آرایه ۳ بُعدی:

Int [4][5][7]; این آرایه از نوع صحیح می باشد.

نحوه ی تعریف آرایه های n بُعدی:

[طول بُعد n ام] .....[طول بُعد دوم] [طول بُعد اول] <اسم آرایه> <نوع آرایه>

روشهای مقدار دهی اولیه به آرایه ها:

نکته مهم در مقداردهی اولیه به آرایه ها این است که فقط در زمان تعریف یک آرایه می توان آن را مقدار دهی اولیه کرد.

مقداردهی اولیه به آرایه های یک بعدی بصورت زیر انجام می پذیرد:

۱- روش اول

```
int A[3] = {5, 2, 8};
```

که در اینجا A[0] برابر ۵ ، A[1] برابر ۲ و A[2] برابر ۸ خواهد شد.

5	2	8

۲- روش دوم

می توان فقط به تعدادی از عناصر آرایه مقدار داد، دراینصورت مقدار عناصر باقیمانده آرایه اتوماتیک ۰ خواهد شد.

```
int B[10] = {5, 8} ;
```

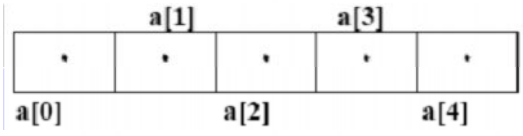
در اینجا عناصر B[2] به بعد مقدار ۰ خواهند گرفت.

5	8	0	0	0	0	0	0	0	0

۳- روش سوم

می توان برای ۰ کردن کلیه عناصر یک آرایه به شکل زیر عمل کرد :

```
int C[10] = {0};
int a[5] = {0};
```



۴- روش چهارم

چنانچه به آرایه مقدار دهی اولیه کرده باشیم، می توان تعداد عناصر آرایه را نیز ذکر نکرد، دراینصورت اندازه آرایه بطور اتوماتیک برابر تعداد مقادیر مشخص شده خواهد شد.

```
int C[] = {10, 15, 20};
```

در مثال فوق آرایه C با ۳ عضو درنظر گرفته می شود.

در واقع در این حالت کامپایلر تعداد عناصر را شمرده و عدد مناسب را در بجای بُعد آرایه(اول) قرار می دهد.

متغیرهای ثابت

گرچه اندازه یک آرایه باید ثابت صحیح باشد؛ اما می توان از متغیرهای ثابت نیز استفاده کرد. یک متغیر ثابت، متغیری است که فقط می تواند در هنگام اعلان مقدار اولیه بگیرد و این مقدار دیگر قابل تغییر نیست. برای اعلان متغیرهای ثابت، از کلمه کلیدی const قبل از نوع متغیر استفاده می گردد. بعنوان مثال:

```
const int k = 10;
```

اکنون هرگونه تلاش برای تغییر مقدار k، باعث ایجاد یک خطای نحوی توسط کامپایلر خواهد شد. به این نوع متغیرها، ثابتهای نام دار نیز گفته می شود.

این متغیرها در تعریف مقادیر ثابتی که مقدار آنها در طول برنامه تغییر نمی کند، بکار می روند. بعنوان مثال :

```
const float pi = 3.14;
```

این کار نه تنها خوانایی برنامه را بالا می برد (بدلیل استفاده از کلمه pi که برای همه شناخته شده است)، بلکه باعث می شود تغییر پذیری برنامه نیز بالا برود. بدین معنا که در صورتیکه برنامه نویس تصمیم گرفت مقدار ثابت را عوض کند، نیازی به تغییر کل برنامه نیست و فقط کافی است مقدار اولیه متغیر را عوض نماید. بعنوان مثال اگر برنامه نویس بخواهد عدد pi را با ۴ رقم اعشار در محاسبات شرکت دهد، فقط باید در تعریف اولیه آن، مقدار را عوض کرده و از ۴ رقم اعشار استفاده نماید.

از این مسئله می توان در تعریف آرایه ها نیز استفاده کرد. بدین صورت که بجای آنکه اندازه آرایه را با یک ثابت صحیح مشخص نماییم، آن را با یک متغیر ثابت تعریف می کنیم. با اینکار، درصورتیکه نیازی به تغییر اندازه آرایه (یا آرایه ها) گردد، فقط کافی است مقدار اولیه متغیر ثابت خود را تغییر دهیم. برای نمونه به مثال زیر دقت کنید:

برنامه ۱) برنامه ای بنویسید که سال ورود تعدادی دانشجو را دریافت و سپس تعداد ورودی های سالهای ۷۵ تا ۸۴ را محاسبه و چاپ نماید.

```
void main() {
const int startYear = 75;
const int yearNo = 10;
int count[yearNo] = {0};
int i, n, year;
printf("enter student no :");
scanf("%d",&n);
for (i= 0;i<n; i++) {
```

```
    printf("enter entrance year :");
    scanf("%d",&year);
    count [year – startYear] ++;
}
for (i= 0; i<yearNo ; i++) {
    printf("year = %d count = %d \n",startYear + i , count[i]);
}
```

مقداردهی اولیه به آرایه های دو بُعدی

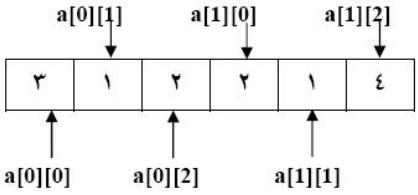
۱- روش اول

```
Int a[2][3]={ {3,1,2} , {7,4,6} }
```

سطر دوم سطر اول

۳	۱	۲
۷	۴	۶

```
Int a[2][3]={ {3,1,2} , {2,1,4} }
```



۲- روش دوم

```
Int a[2][3]={1,3}
```

مابقی عناصر با صفر پر می شوند

۱	۳	۰	۰	۰	۰

۳- روش سوم

ننوشتن بُعد اول، در زبان c در هنگام مقداردهی اولیه به آرایه ها ما فقط می توانیم بُعد اول را ننویسیم.

```
Int a[][3]={ {3,1,2} , {2,1,5} };
```

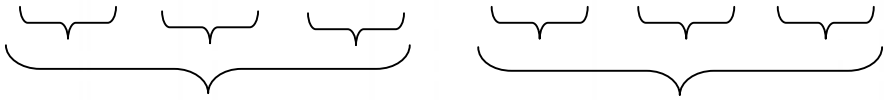
۴- روش چهارم

در این روش تمامی عناصر آرایه صفر می شوند

```
Int a[2][4]={0};
```

مقداردهی اولیه به آرایه های سه بُعدی

Int a[2][3][4]={ { {1,2,3,4} ,{5,4,3,2} , {2,3,4,5} } , { {7,6,5,4} , {9,0,8,7} , {6,7,4,1} } }



ارسال آرایه های یک بعدی به توابع در برنامه نویسی C

آرایه ها را نیز همچون سایر نوع داده ها می توان به یک تابع ارسال کرد. برای اینکار ابتدا باید تابع را بگونه ای تعریف کنیم که یک پارامتر از نوع آرایه را دریافت کند. فرض کنید تابعی بنام sumArray داریم که یک آرایه یک بعدی از اعداد صحیح را بعنوان ورودی دریافت می نماید و مجموع عناصر آن را باز می گرداند. تعریف این تابع بصورت زیر است:

```
int sumArray(int A[], int size) {
int i , sum = 0;
for (i=0; i< size; i++)
sum += A[i];
return(sum) ;
}
```

در تابع فوق، پارامتر A بعنوان یک آرایه از اعداد صحیح معرفی شده است. همانطور که می بینید، اندازه آرایه مشخص نشده است و این یک نکته مثبت است؛ چرا که تابع sumArray می تواند هر آرایه صحیحی را با هر اندازه ای دریافت نماید. درواقع حتی اگر اندازه آرایه را نیز مشخص نمایید، کامپایلر از آن صرفنظر خواهد کرد. دومین پارامتر، اندازه واقعی آرایه A را مشخص می نماید. معمولا توابع بگونه ای نوشته می شوند که هنگام ارسال یک آرایه به یک تابع، اندازه آن نیز بعنوان یک پارامتر ارسال گردد. درغیراینصورت مجبوریم در تابع اندازه مشخصی را برای آرایه در نظر بگیریم که باعث ایجاد محدودیت در ارسال آرایه های با اندازه دلخواه می گردد.

در هنگام فراخوانی تابع sumArray، برای ارسال آرایه موردنظر کافی است که تنها نام آرایه را بدون کروشه استفاده نماییم. البته اندازه واقعی آرایه نیز باید بعنوان دومین آرگومان به تابع ارسال شود.

```
void main() {
int data1[3] = {5, 10, 15};
int data2[5] = {1, 6, 4, 12, 5} ;
int sum1, sum2;
sum1 = sumArray(data1, 3);
sum2 = sumArray(data2, 5);
printf("sum1 = %d\n",sum1);
```

```
printf(“sum2 = %d\n”,sum2);
}
```

```
sum1 = 30
sum2 = 28
```

همانطور که در مثال فوق دیده می شود، تابع sumArray دوبار فراخوانی شده است. در بار اول یک آرایه با اندازه ۳، و در دفعه دوم یک آرایه با اندازه ۵ به آن ارسال شده است و تابع در هر دو مورد بدون هیچ مشکلی مجموع عناصر آرایه را باز گردنانده است.

نکته بسیار مهم، نحوه ارسال آرایه ها به توابع است. زبانC آرایه ها را توسط ارجاع به تابع ارسال می نماید (برخلاف انواع دیگر داده ها که درحالت عادی توسط مقدار به توابع ارسال می شدند). بدین معنا که در هنگام ارسال یک آرایه به تابع، بجای یک کپی از آرایه، خود آرایه ارسال می شود. در حقیقت در فصلهای بعدی خواهید دید که برای ارسال یک آرایه، آدرس اولین عنصر آن ارسال می گردد. لذا تابع می تواند از طریق این آدرس، به کلیه داده های آرایه اصلی دسترسی پیدا کند. اما چرا C در مورد آرایه ها به روش متفاوتی عمل می نماید؟ دلیل این مسئله آن است که معمولا یک آرایه حافظه بسیار زیادی را اشغال می کند، لذا تهیه یک کپی کردن از آن، نه تنها باعث اشغال حافظه می شود بلکه زمان زیادی را نیز صرف خواهد کرد. با ارسال آرایه ها توسط ارجاع در زمان و حافظه صرفه جویی زیادی صورت می گیرد.

اما آیا می توان یک آرایه را توسط مقدار به یک تابع ارسال کرد؟ متاسفانه خیر. اما اگر نگران تغییر سهوی آرایه ارسالی به یک تابع هستید می توانید آن را بگونه ای به تابع ارسال نمایید که تغییر آن در تابع ممکن نباشد. زبان C یک نحوه دیگر ارسال داده ها به توابع بنام ارسال توسط ارجاع ثابت می باشد. چنانچه در هنگام تعریف یک پارامتر از یک تابع، از کلمه کلیدی const استفاده شود، کامپایلر اجازه تغییر مقادیر آن پارامتر را در حین اجرای تابع نخواهد داد. با این ارسال آرایه ها بصورت ارجاع ثابت، می توانیم مانع از انجام تغییرات ناخواسته در آرایه شویم. مثال زیر نحوه انجام این کار را نشان می دهد.

برنامه) برنامه ای بنویسید که با استفاده از یک تابع، اشتراک دو مجموعه را محاسبه و چاپ نماید.

```
void intersection(const int A[], int na, const int B[], int nb, int C[], int &nc) {
k = 0;
for (i=0; i< na; i++) {
sw = 1;
for (j=0; j< nb && sw; j++)
if (A[i] == B[j]) {
C[k] = A[i] ;
k ++;
sw = 0;
}
}
nc = k;
}
void printSet(int set[], int size) {
```

```
int i;
printf(“{ “) ;
for (i=0; i< size; i++)
printf(”%d ”,set[i]) ;
printf(“}\n”);
}
```

```
void main() {
int set1[5] = {5, 8, 3, 12, 20};
int set2[3] = {12, 16, 8} ;
int result[3] , resultSize ;
intersection(set1, 5, set2, 3, result, resultSize);
printf(“set 1 = “);
printSet(set1) ;
printf(“set 2 = “);
printSet(set2) ;
printf(“intersection = “);
printSet(result) ;
}
```

```
set1 = { 5 8 3 12 20 }
set2 = { 12 16 8 }
intersection = { 8 12 }
```

همانگونه که در مثال بالا دیده می شود، تابع intersection، دو مجموعه را بعنوان ورودی دریافت و اشتراک آنها را بعنوان خروجی باز می گرداند. آرایه های A و B بعنوان پارامترهای ورودی هستند که نماینده دو مجموعه اولیه هستند. از آنجا که لزومی ندارد مقادیر این دو آرایه در تابع تغییر نماید، بعنوان پارامتر ثابت (const) به تابع ارسال شده اند. اندازه این دو آرایه نیز به ترتیب در قالب پارامترهای na و nb ارسال شده است. اما آرایه C پارامتر خروجی است که اشتراک دو مجموعه را باز می گرداند، به همین دلیل بصورت ثابت تعریف نشده است. تابع intersection، اشتراک دو مجموعه را محاسبه و مجموعه حاصل را در پارامتر C و اندازه آن را در پارامتر nc قرار می دهد. از آنجا که هم پارامتر C (بدلیل اینکه یک آرایه است) و هم پارامتر nc (بدلیل استفاده از عملگر &) توسط ارجاع به تابع ارسال شده اند، تغییرات انجام شده در آنها (یعنی حاصل نهایی) به تابع فراخواننده منتقل خواهد شد.

در تابع اصلی ابتدا دو مجموعه بنامهای set1 و set2 تعریف شده و مقدار اولیه گرفته اند. سپس با استفاده از تابع intersection، اشتراک آنها محاسبه و حاصل در آرایه result و اندازه آن نیز در متغیر resultSize قرار گرفته است. سرانجام دو مجموعه اولیه و اشتراک آنها با فراخوانی تابع printSet چاپ شده اند.

نکته مهم دیگر آنکه خروجی یک تابع نمی تواند یک آرایه باشد. برای بازگرداندن یک آرایه از تابع، باید آن را بصورت یک پارامتر خروجی به تابع ارسال نمود.

رشته ها:

در زبانهای برنامه سازی مختلف رشته ها به عنوان نوع داده (data type) می باشند که برای نگهداری اسامی و متن ها بکار می روند. در زبان c رشته ها نوع داده نیستند بلکه آرایه ای از کاراکترها می باشند که به NULL که دارای ارزش عددی صفر است ختم می شود. برای نمایش NULL از کاراکتر '\0' استفاده می شود.

```
char name[۱۰];
```

در مثال فوق متغیر name بعنوان یک آرایه ۱۰ عضوی از کارکترها تعریف شده است، بنابراین می تواند یک رشته با حداکثر ۱۰ کاراکتر را در خود نگاه دارد. اما فرض کنید قصد داریم رشته ای مانند Ali را در این متغیر ذخیره نماییم که کمتر از ۱۰ کارکتر دارد .دراینصورت زبان c چگونه دریابد که در هنگام انجام عملیات مختلف برروی این رشته، مثلا در هنگام چاپ آن، فقط باید ۳ حرف اول رشته را چاپ نماید؟ برای حل این مشکل، طراحان زبان c از یک کارکتر خاص بنام null استفاده کردند .کلیه رشته ها در زبان c باید به کاراکتر null ختم گردند .در حقیقت در زبان c یک رشته هنگامی که به null برسد، خاتمه می یابد و نه زمانیکه به انتهای آرایه برسد.

در زبان c باید طول رشته یک واحد بیشتر از طول واقعی آن باشد تا بتوانیم کاراکتر null را در آخر آن قرار دهیم.

به عنوان مثال به نمونه زیر دقت کنید.

```
char s [10] = {'A', 'l', 'i', '\0' };
```

a	l	i	\0	?	?	?	?	?	?
s[0]	s[1]	s[2]	s[3]	s[4]	s[5]	s[6]	s[7]	s[8]	s[9]

در مثال فوق، متغیر s بصورت آرایه ای ۱۰ عضوی ازکاراکتر تعریف شده و مقدار اولیه ali به آن نسبت داده شده است .دقت آنید که در حقیقت ۴ عضو از آرایه پر شده است و عضو آخر به null تخصص داده شده است .مقادیر عناصر بعدی آرایه مهم نیست، چرا که در هنگام انجام عملیات برروی رشته s از آنها صرفنظر می گردد .بنابراین دقت آنید آه در هنگام تعریف یک رشته، یک عنصر اضافه برای کاراکتر null در نظر بگیرید.

توجه درباره ی لیترال‌ها و کاراکترها

یک «لیترال» رشته‌ای از حروف، ارقام یا علایم چاپی است که میان دو علامت نقل قول " " محصور شده باشد.
یک «کاراکتر» یک حرف، رقم یا علامت قابل چاپ است که میان دونشانهٔ ' ' محصور شده باشد. پس 'w' و '!' و '1' هر کدام یک کاراکتر است.
به تفاوت سه موجودیت «عدد» و «کاراکتر» و «لیترال رشته‌ای» دقت کنید: 6 یک عدد است، '6' یک کاراکتر است و "6" یک لیترال رشته‌ای است.

مقداردهی اولیه به رشته ها:

هنگام تعریف رشته ها می توان به آنها مقدار اولیه داد. هنگام مقدار اولیه دادن می توان طول رشته را مشخص نکرد. در اینصورت، اندازه رشته یک واحد بیش از تعداد کاراکترهایی است که به آن نسبت داده می شود . دو روش برای مقدار اولیه دادن به رشته ها وجود دارد.

۱- رشته در داخل کوتیشن قرار گرفته و به متغیر رشته ای نسبت داده شود. در این روش کاراکتر '\0' به طور خودکار در انتهای رشته قرار می گیرد.

مثال:

char Str1[]=”kiyanmajd” // بدون تعیین بعد

char Str2[15]=”kiyanmajd” // با تعیین بعد

Str1

k	i	y	a	n	m	a	j	d	\0	?	?	?	?	?

مقادیر موجود در ۵ عضو آخر str1 هر مقداری می تواند باشد و از نظر برنامه نویس مهم نیستند، چرا که توابع رشته ای آنها را پردازش نمی کنند.

۲- هر یک از کاراکترهای رشته ای به عنوان یک عنصر رشته به ارایه نسبت داده شوند. در این روش کاراکتر بر خلاف روش اول، '\0' باید توسط برنامه نویس در انتهای رشته قرار داده شود.

مثال:

char Str3[]={‘k’,’l’,’y’,’a’,’n’,’m’,’a’,’j’,’d’,’\0’} // بدون تعیین بعد

char Str4[15]= {‘k’,’l’,’y’,’a’,’n’,’m’,’a’,’j’,’d’,’\0’} // با تعیین بعد

نکته آخر اینکه به جز در هنگام مقداردهی اولیه، نمی توانید در برنامه از عملگر = برای مقداردهی به یک رشته استفاده نمایید .بعنوان مثال دستور زیر خطای نحوی محسوب می گردد

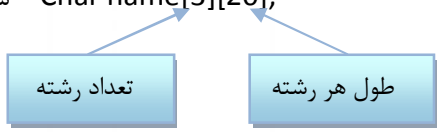
```
char str[10];
str = "Ali" ;
```

فراموش نکنید که رشته ها در حقیقت یک آرایه هستند و نمی توان کل یک آرایه را با یک دستور مقداردهی کرد. برای مقداردهی به یک رشته، باید به تک تک عناصر آن را جداگانه مقدار داد و یا از توابع کتابخانه ای c استفاده کرد.

آرایه ای از رشته ها

اگر بخواهیم متغیری را تعریف کنیم که چندین رشته را در خود نگه دارد باید آرایه ای از رشته ها(رشته ها که در بالا آورده شده خود آرایه ای از کاراکترها هستند.) تعریف کنیم. برای تعریف آرایه ای از رشته ها به صورت زیر عمل می کنیم.

Char name[3][20]; سه رشته بیست حرفی



برای بازیابی هر یک از رشته ها باید از یک اندیس استفاده کنیم. به طور مثال name[0] اولین رشته، name[1] دومین رشته، name[2] سومین رشته می باشد. اگر از دو اندیس استفاده کنیم، اندیس اول مشخص کننده ی رشته و اندیس دوم شماره ی کاراکتری از رشته را مشخص می کند. به عنوان مثال name[2][5] کاراکتر ۵ از رشته سوم را تعیین می کند.

مقداردهی اولیه به رشته های دو بُعدی:

Char my_friend[3][25]={“ali salehi”,”ali abedini”,”mahmood hassani paju ”,”esmaeil hamzei vash”};

my_friend[0]

a	l	i		s	a	l	e	h	i	\0	?	?	?	?	?	?	?	?	?	?	?	?	?	?

my_friend[1]

a	l	i		a	b	e	d	i	n	i	\0	?	?	?	?	?	?	?	?	?	?	?	?	?

my friend[2]

m	a	h	m	o	o	d		h	a	s	s	a	n	i		p	a	j	u	\0	?	?	?	?

my_friend[3]

e	s	m	a	e	i	l		h	a	m	z	e	i		v	a	s	h	\0	?	?	?	?	?

my_friend[3][5]=’e’

my_friend[3] =”esmaeil hamzei vash”

توابع کتابخانه ای رشته ها

زبان c دارای یک کتابخانه غنی از توابع کار با رشته ها است. پیش تعریف این توابع در فایل سرآمد string.h آمده است .در این قسمت چندین تابع از کتابخانه c به همراه الگوریتم آنها بررسی می گردند .هدف از بررسی الگوریتم این توابع، آشنایی بیشتر شما با نحوه کار با رشته ها می باشد. دقت کنید که ممکن است تعریف دقیق تابع کتابخانه ای در c با آنچه که ما در اینجا آورده ایم، کمی متفاوت باشد.

نکته بسیار مهمی که در هنگام بررسی این توابع باید بدان توجه داشته باشید، این است که هیچیک از آنها حدود آرایه (اندازه آرایه) را بررسی نمی کنند. بنابراین این وظیفه خود برنامه نویس است که آرایه هایی با اندازه مناسب را به توابع ارسال کند.

توابع پرکاربرد موجود در #include <string.h>

در کامپایلر codvision این تابع می تواند فقط بر روی متغیرهایی از نوع flash و sram عملیات انجام دهد.

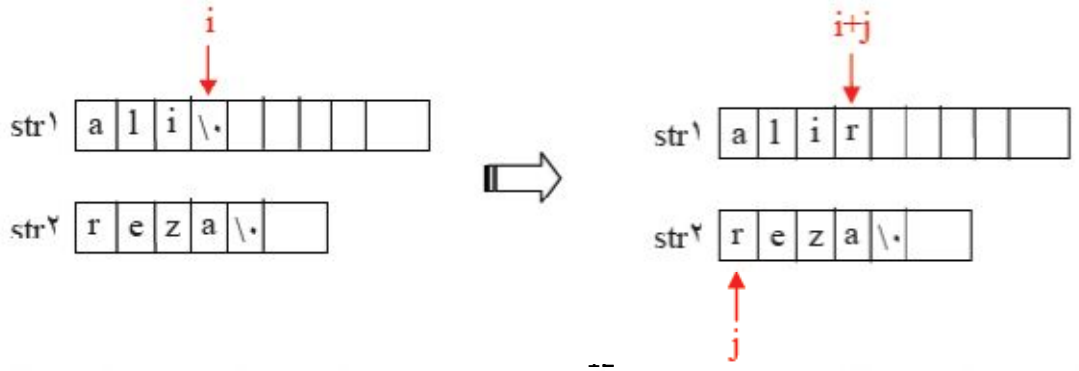
char *strcat(char *str1, char *str2)

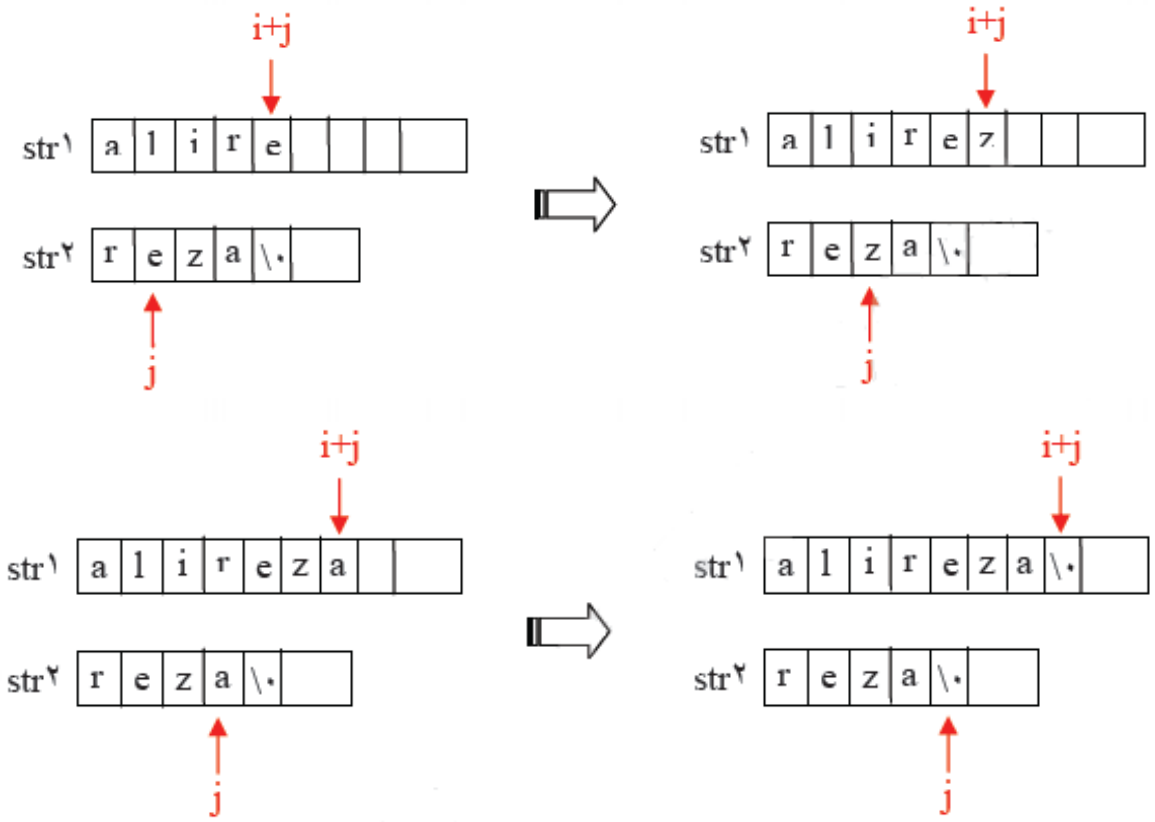
نام این تابع مخفف string concat است. این تابع دو رشته را دریافت و رشته دوم را به انتهای رشته اول الحاق می کند. به تعریف این تابع دقت کنید:

```
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 داریم رشته str2 را به انتهای رشته str1 اضافه کنیم .حلقه for اول، دقیقا همانند تابع strlen است و طول رشته str1 را محاسبه و در i قرار می دهد. سپس در حلقه for دوم، از ابتدای رشته str2 شروع کرده و هر کاراکتر را به انتهای رشته str1 اضافه می کند. در پایان کاراکتر ’\0‘ نیز به انتهای رشته str1 الحاق می گردد.

بعنوان مثال فرض کنید str1=”ali” و str2=”reza” باشد.شکل زیر مراحل حلقه دوم را نشان می دهد.





مثال:

```
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 string2 is : %s", string1) ;
}
```

خروجیش

```
please enter string1 : Hello
please enter string2 : everybody!
concatenate of string1 and string2 is : Helo everybody !
```

char *strcatf(char *str1, char flash *str2)

این تابع رشته str2که در حافظه ی flash قرار دارد را به انتهای رشته str1 الحاق می کند.

Strcatf(s1,s2);

char *strncat(char *str1, char *str2, unsigned char n)

این تابع به تعداد حداکثر n کاراکتر از رشته str2 را به انتهای str1 متصل می کند.

Strncatf(s1,s2,4);

char *strncatf(char *str1, char flash *str2, unsigned char n)

همان تابع بالا با این تفاوت که str2 از نوع متغیر flash می باشد.

char *strchr(char *str, char c)

این تابع کاراکتر(c) را در درون یک رشته جستجو کرده و زیر رشته ای که از محل اولین وقوع کاراکتر مورد نظر تا آخر آن بر می گرداند.

خروجی یک رشته = strchr (رشته , کاراکتر)

```
S[]=”in the name of GOD”;
Char c=’m’;
Char *p=strchr(s,c);    =>    p=”me of GOD”
```

char *strrchr(char *str, char c)

این تابع کاراکتر(c) را در درون یک رشته جستجو کرده و زیر رشته ای که از محل اخرین وقوع کاراکتر مورد نظر تا آخر آن بر می گرداند.

```
S[]=”in the name of GOD”;
Char c=’o’;
Char *p=strchr(s,c);    =>    p=”OD”
```

signed char strpos(char *str, char c)

این تابع اندیس(شماره ی) اولین خانه از آرایه رشته ای str که شامل کاراکتر c می باشد را برمی گرداند. و در صورتی که رشته ی شامل کاراکتر مورد نظر نباشد تابع مقدار ۱-(کد اسکی ۲۵۵) را برمی گرداند.

مثال:

```
#include <mega16.h>
#asm
  .equ __lcd_port=0x18 ;PORTD
#endasm
#include <lcd.h>
#include <delay.h>
#include <string.h>
#include <stdlib.h>
char s1[]="kaveh kiyanmajd" ,s3[5],j,c='a';
void main(void)
{
    lcd_init(16);
    while (1){
    j=strpos(s1,c);
    lcd_clear();
    itoa(j,s3);
    lcd_puts(s3);
    delay_ms(500);
 }
}
```

در این مثال کاراکتر مورد جستجو 'a' می باشد که اولین وقوع آن در خانه اول (اندیس شماره ۱ (اندیسها از صفر شروع می شوند....این صد بار.)) می باشد پس عدد ۱ درون متغیر j برگردانده می شود.

یه مثال دیه:

```
#include <mega16.h>
#asm
  .equ __lcd_port=0x18 ;PORTD
#endasm
#include <lcd.h>
#include <delay.h>
#include <string.h>
#include <stdlib.h>
char s1[]="kaveh kiyanmajd" ,s3[5],j,c='s';
void main(void)
```

```
{
    lcd_init(16);
    while (1){
    j=strpos(s1,c);
    lcd_clear();
    itoa(j,s3);
    lcd_puts(s3);
    delay_ms(500);
 }
}
```

در این مثال کاراکتر مورد جستجو 's' می باشد که در این رشته موجود نمی باشد، پس عدد ۱- درون j برگردانده می شود.

signed char strrpos(char *str, char c)

این تابع اندیس(شماره ی) آخرین خانه از آرایه رشته ای str که شامل کاراکتر c می باشد را برمی گرداند. و در صورتی که رشته ی شامل کاراکتر مورد نظر نباشد تابع مقدار ۱- را برمی گرداند.

مثال:

```
#include <mega16.h>
#asm
  .equ __lcd_port=0x18 ;PORTD
#endasm
#include <lcd.h>
#include <delay.h>
#include <string.h>
#include <stdlib.h>
char s1[]="kaveh kiyanmajd" ,s3[5],j,c='a';
void main(void)
{
    lcd_init(16);
    while (1){
    j=strpos(s1,c);
    lcd_clear();
    itoa(j,s3);
    lcd_puts(s3);
    delay_ms(500);
 }
}
```

در این مثال کاراکتر مورد جستجو 'a' می باشد که اخرین وقوع آن در خانه دوازدهم(اندیس شماره ۱۲) می باشد پس عدد ۱۲ درون متغیر j برگردانده می شود.

signed char strcmp(char *str1, char *str2)

نام این تابع مخفف string compare است .این تابع دو رشته را دریافت و پس از مقایسه آنها یکی از ۳ مقدار زیر را باز می گرداند:

۱- در صورتیکه مساوی باشند : صفر را برمی گرداند
۲- در صورتیکه رشته اول بزرگتر باشد: عدد مثبتی را برمی گرداند (اختلاف عددی کد اسکی اولین کاراکتر نابرابر را برمیگرداند)
۳- درصورتیکه رشته اول کوچکتر باشد: عدد منفی را برمی گرداند (اختلاف عددی کد اسکی اولین کاراکتر نابرابر را با علامت منفی برمیگرداند)

نحوه مقایسه دو رشته، به همان ترتیبی است که یک لغتنامه کلمات را مرتب می کند .یعنی ابتدا حروف اول دو رشته مقایسه می شود، اگر یکی از آنها بزرگتر بودکه نتیجه بازگردانده می شود .اما در صورتیکه حروف اول دو رشته یکسان بود، حروف دوم با یکدیگر مقایسه می شوند و این عمل تا زمانیکه یک اختلاف بین دو رشته پیدا شود ادامه می یابد .در صورتیکه هیچ اختلافی بین دو رشته پیدا نشد، مقدار ۰ باز گردانده می شود.
پیاده سازی این تابع در زیر آمده است:

```
int strcmp(const char str1[], const char str2[]) {
int i;
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 و str2 را با یکدیگر مقایسه می کند .از آنجا که هیچیک از این دو نباید تغییر داده شوند، هردو بصورت ثابت به تابع ارسال شده اند. حلقه ی do-while عناصر این دو رشته را با یکدیگر مقایسه میکند .توجه آنید آه عمل مقایسه برای کاراکترها در زبان تعریف شده و در حقیقت کد اسکی آنها را با یکدیگر مقایسه می کند .در هربار اجرای حلقه، چنانچه یکی از کاراکترها بزرگتر بود، بلافاصله حاصل بازگردانده می شود، اما درصورتیکه هردو مساوی باشند، حلقه دور زده و عملیات تکرار می شود .به محض اینکه str[i] به null برسد (که در اینصورت str2[i] هم قطعا به null رسیده است، چرا که هر دو مساوی بوده اند) از برای آشنایی بیشتر به برنامه زیر و اجراهای مختلف آن توجه کنید:

```
void main() {
char string١[٢٠], string٢[٢٠] ;
int result;
printf("please enter string١ : ");
gets(string١);
printf("please enter string٢ : ");
gets(string٢);
result = strcmp(string١,string٢);
if (result == ٠)
printf("%s equals %s\n", string١, string٢) ;
else if (result == ١)
printf("%s is grater than %s\n", string١, string٢) ;
else printf("%s is less than %s\n", string١, string٢) ;
}
```

```
please enter string١ : ali
please enter string٢ : ahmad
ali is grater than ahmad
```

```
please enter string١ : ali
please enter string٢ : alireza
ali is less than alireza
```

```
please enter string١ : ali
please enter string٢ : ali
ali equals ali
```

signed char strcmpf(char *str1, char flash *str2)

همان تابع بالایی با این تفاوت که str1 درون sram قرار دارد ولی str2 دورن حافظه فلش که با هم مقایسه می شوند

signed char strncmp(char *str1, char *str2, unsigned char n)

این تابع نیز همانند تابع strcmp() عمل می کند با این تفاوت که حداکثر n کاراکتر از str1 را با str2مقایسه می کند و نتیجه مربوطه را بر می گرداند.

signed char strncmpf(char *str1, char flash *str2, unsigned char n)

این تابع نیز همانند تابع strcmpf() عمل می کند با این تفاوت که حداکثر n کاراکتر از str1(که درون sram می باشد) را با str2 (که درون flash می باشد)مقایسه می کند و نتیجه مربوطه را بر می گرداند.

char *strcpy(char *dest, char *src)

این تابع رشته SRC را روی رشته dest کپی می کند.

char *strcpyf(char *dest, char flash *src)

این تابع رشته SRC را که بر روی حافظه flash قرار دارد، را روی رشته dest که بر روی حافظه sram قرار دارد،کپی می کند.

char *strncpy(char *dest, char *src, unsigned char n)

این تابع همان تابع strcpy() است با این تفاوت که حداکثر n کاراکتر از رشته src را بر روی dest کپی می کند.

char *strncpyf(char *dest, char flash *src, unsigned char n)

این تابع همان تابع strcpyf() است با این تفاوت که حداکثر n کاراکتر از رشته src را که بر روی حافظه flash قرار دارد، را بر روی dest که بر روی حافظه sram قرار دارد، کپی می کند.

unsigned char strspn(char *str, char *set)

این تابع کاراکترهای رشته str را از اول به صورت تک تک با کاراکترهای موجود در رشته set مقایسه می کند و اندیس اولین کاراکتر از رشته str که در رشته set موجود نباشد را برمی گرداند.

اگر تمام کاراکتر های str در درون رشته set وجود داشته باشد، تابع طول رشته str را بر می گرداند.

مثال:

```
#include <mega16.h>
#asm
  .equ __lcd_port=0x18 ;PORTD
#endasm
```

```
#include <lcd.h>
#include <delay.h>
#include <string.h>// for strspn(s1,s2)
#include <stdlib.h>//for itoa()
char s1[]="hamid",s2[]="mahmood",s3[2],x;
void main(void)
{
    lcd_init(16);
    while (1){
    x=strspn(s1,s2);
    lcd_clear();
    itoa(x,s3);
    lcd_puts(s3);
    delay_ms(500);

 }

}
```

در برنامه بالا به ترتیب کاراکترهای a، m و h در رشته s2 وجود دارد ولی کاراکتر i در این رشته وجود ندارد پس اندیس آن که عدد ۳ (اندیسها از صفر شروع می شوند) می باشد به داخل متغیر x برگردانده می شود.

unsigned char strspnf(char *str, char flash *set)

همان تابع بالا با این تفاوت که رشته str در حافظه SRAM، و رشته set در حافظه FLASH قرار دارد.

اگر تمام کاراکتر های str در درون رشته set وجود داشته باشد، تابع طول رشته str را بر می گرداند.

unsigned char strcspn(char *str, char *set)

این تابع کاراکترهای رشته str را از اول به صورت تک تک با کاراکترهای موجود در رشته set مقایسه می کند و اندیس اولین کاراکتر از رشته str که در رشته set موجود باشد را برمی گرداند.(برعکس تابع strspn())

اگر هیچ کاراکتری از STR درون رشته set نباشد، تابع طول رشته str را بر می گرداند.

مثال:

```
#include <mega16.h>
#asm
  .equ __lcd_port=0x18 ;PORTD
#endasm
#include <lcd.h>
#include <delay.h>
#include <string.h>
#include <stdlib.h>
```

```
char s1[]="tiny",s2[]="mahmood",s3[3],x;
void main(void)
{
    lcd_init(16);
    while (1){
    x=strcspn(s1,s2);
    lcd_clear();
    itoa(x,s3);
    lcd_puts(s3);
    delay_ms(500);
    x++;
 }
}
```

در مثال بالا هیچ کدام از کاراکترهای s1 درون s2 وجود ندارد پس طول رشته s1 یعنی ۴ برگردانده می شود.

unsigned char strcspnf(char *str, char flash *set)

همان تابع بالا با این تفاوت که رشته str در حافظه SRAM، و رشته set در حافظه FLASH قرار دارد.

اگر هیچ کاراکتری از STR درون رشته set نباشد، تابع طول رشته str را بر می گرداند.

char *strpbrk(char *str, char *set)

این تابع رشته str را درون رشته set جستوجو کرده و اولین کاراکتری از رشته str که درون رشته set وجود دارد را پیدا و زیر رشته ای را که از محل اولین وقوع آن کاراکتر در رشته str وجود دارد را برمیگرداند.

در صورت نبود هیچ یک از کاراکترهای str در set کاراکتر NULL برگردانده می شود.

تذکر: خروجی این تابع خود یک رشته می باشد.

Exp:

```
#include <mega16.h>
#asm
  .equ __lcd_port=0x18 ;PORTD
#endasm
#include <lcd.h>
#include <delay.h>
#include <string.h>
char s1[]="saerofzz",s2[]="mahmood", s4[9]=" " ;
void main(void)
```

```
{
    lcd_init(16);
    while (1)
       {
        strcpy(s4,strpbrk(s1,s2));
        lcd_clear();
        lcd_puts(s4);
       delay_ms(500);
      }
}
```

خروجی

⇨ S4= " aerofzz " اولین حرف که در رشته s2 می باشد حرف a است.

char *strpbrkf(char *str, char flash *set)

همان تابع بالا با این تفاوت که رشته str در حافظه SRAM، و رشته set در حافظه FLASH قرار دارد.

char *strrpbrk(char *str, char *set)

این تابع رشته str را درون رشته set جستوجو کرده و اخرین کاراکتری از رشته str که درون رشته set وجود دارد را پیدا و زیر رشته ای را که از محل وقوع آن کاراکتر در رشته str وجود دارد را برمیگرداند.

در صورت نبود هیچ یک از کاراکترهای str در set کاراکتر NULL برگردانده می شود.

EXP:

```
#include <mega16.h>
#asm
  .equ __lcd_port=0x18 ;PORTD
#endasm
#include <lcd.h>
#include <delay.h>
#include <string.h>

char s1[]="saoerofzoz",s2[]="mahmood",s4[9]="";
void main(void)
{
    lcd_init(16);
    while (1){
    strcpy(s4,strrpbrk(s1,s2));
    lcd_clear();
    lcd_puts(s4);
    delay_ms(500);
 }
```

}

خروجی

⇨ s4= “oz “

char *strrpbrkf(char *str, char flash *set)

همان تابع بالا با این تفاوت که رشته str در حافظه SRAM، و رشته set در حافظه FLASH قرار دارد.

char *strstr(char *str1, char *str2)

این تابع زیر رشته str2 را درون رشته str1 جستوجو کرده و زیر رشته ای را از محل اولین وقوع زیر رشته مورد نظر برمیگرداند.

مثال:

```
Char  *p=strstr(str1,str2);

Char  str1[]=”in the name of god”;
Char  str2[]=”he”;
Char  *p=strstr(str1,str2);
```

⇨ P=”he name of god”;

char *strstrf(char *str1, char flash *str2)

همان تابع بالا با این تفاوت که رشته str1 در حافظه SRAM، و رشته str2 در حافظه FLASH قرار دارد.

unsigned char strlen(char *str)

طول یک رشته را برمی گرداند

برای حافظه های مدل tiny (طول رشته از ۰ تا ۲۵۵)

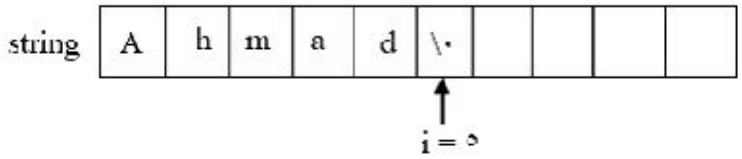
مسلما منظور از طول رشته، تعداد کاراکترهای آن تا رسیدن به null است .تعریف این تابع در زیر آمده است

```
int strlen(const char string[]) {
int i;
for (i=۰; string[i]; i++) ;
return(i);
}
```

دقت در تابع بالا نکات جالب را به ما نشان می دهد . متغیرstring بصورت ثابت ارسال شده است تا در داخل تابع بطور ناخواسته تغییر داده نشود . دقت کنید که ; پس از حلقه for نشان می دهد آه بدنه این حلقه، خالی است .به این معنا آه در هربار اجرای حلقه، فقط عملیات مربوط به خود حلقه یعنی افزایش شمارنده و بررسی شرط صورت می پذیرد . علاوه براین شرط ادامه حلقه فقط string[i] قرار داده شده است آه توسط c اینگونه تفسیر می شود" : تا زمانیکه string[i] درست است ."می دانید آه هر عدد بجز ۰ درست محسوب می گردد، بنابراین تا هنگامی که string[i] برابر null برسد، نادرست ارزیابی شده و حلقه خاتمه خواهد یافت .مسلما در این حالت مقدار متغیر i ،طول آرایه را نشان می دهد .می توانستیم حلقه for را بصورت زیر نیز بنویسیم:

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

برای روشن شدن موضوع به شکل زیر دقت کنید:



همانطور که می بینید، هنگامی که i برابر ۵ شود،string[i] برابرشده و شرط برقرار نخواهد بود. درنتیجه مقدار i یعنی ۵ نشاندهنده طول رشته است.

unsigned int strlen(char *str)

طول یک رشته را برمی گرداند

برای حافظه های مدل small (طول رشته از ۰ تا 65535)

unsigned int strlenf(char flash *str)

طول یک رشته را برمی گرداند(رشته ای که در حافظه flash قرار دارد.)

مثال: برنامه زیر نحوه استفاده از strlenرا نشان می دهد:

```
void main() {
```

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

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

منابع

http://www.prdev.com ابریشمی سعید

کتاب برنامه نویسی به زبان C (مهندس جعفر نژاد قمی)

جزوه درس برنامه سازی پیشرفته (c و c++) دانشگاه آزاد واحد دزفول