فصل دوم

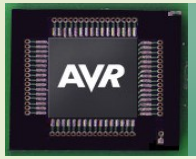
اصول طراحی و آموزش زبان C

دکتر حسین غلامعلی نژاد

زمستان ۱۴۰۱

اهداف

1. آشنایی با اصول طراحی و ترسیم الگوریتم
2. آموزش دستورات برنامه نویسی
3. معرفی توابع کتابخانه ای و کاربرد آنها در برنامه نویسی

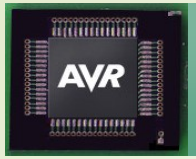


آموزش زبان C

زبان C در سال ۱۹۷۲ توسط دنیس ریچی طراحی شد.
زبا نهای برنامه نویسی به لحاظ ساختاری به سه دسته تقسیم می شوند:

- زبانهای سطح بالا : که به زبانهای انسان نزدیکترند مانند پاسکال و بیسیک
- زبانهای سطح میانی : که مستقیما به حافظه دستیابی دارند و از طرفی قابلیت انعطاف پذیری همچون زبانهای سطح بالا را برخوردارند.
- زبانهای سطح پایین : زبانهایی که با سخت افزار در ارتباط هستند مثل اسمبلی.

با توجه با کارایی بالای زبان C در برنامه نویسی میکروکنترلرها آن را می توان یک زبان قدرتمند و سطح بالا دانست.

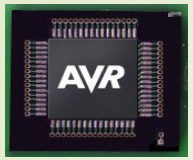


آموزش زبان C

مفاهیم اولیه در زبان C

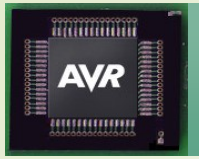
در زبان C بین حروف بزرگ و کوچک تفاوت وجود دارد . در این زبان دستورات و کلمات کلیدی با حروف کوچک نوشته می شوند. به عنوان مثال : کلمه کلیدی **void** با کلمه VOID تفاوت دارد. که اگر دستورات و کلمات کلیدی را با حروف بزرگ بنویسیم کامپایلر پیغام خطا می دهد.

- هر دستور به ; ختم می شود که نشان دهنده آن است که خط دستوری پایان یافته است.
- حداکثر طول یک دستور ،۲۵۵ کاراکتر است.
- هر دستور را می توان در یک یا چندین سطر نوشت.
- در هر سطر می توان چند دستور نوشت.
- توضیحات در یک سطر می توانند بعد از // قرار بگیرند و همچنین توضیحات گروهی با /* شروع می شوند و به */ ختم می شوند.



ساختار کلی زبان C

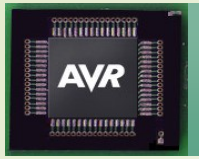
```
#include<نام تراشه.h>                    // معرفی فایل سرآمد یا الحاقی
#include<نام کتابخانه.h>
#define out1 PORTC.0                    // معرفی شناسه ها

Unsigned char x,y;                      // معرفی متغیر های ۸ بیتی کلی
Unsigned int N,count;                   // معرفی متغیر های ۱۶ بیتی کلی

Void function1() {                      // توابع فرعی
Unsigned char k,j;                      // معرفی متغیر های ۸ بیتی کلی
دستورالعمل ها;
}
Void main () {                          // تابع اصلی برنامه
دستورالعمل ها;
Function1();                            // فراخوانی توابع فرعی
While (1) {                             // حلقه بی نهایت برنامه
};
}
```



پیش پردازنده ها

فایل های سرآمد توسط دستورات پیش پردازنده در ابتدای برنامه معرفی (گنجانده) می شوند.
در زبان C می بایست میکروکنترلری که استفاده می کنید را معرفی نمائید و بعد از آن کتابخانه هایی را که به آنها در برنامه نویسی نیاز دارید ، باید معرفی کنید.

فایل های سرآمد را با دستور پیش پردازنده #include در بین دو علامت <> قرار می دهیم و پسوند اینگونه فایل ها *.h می باشد.



تعریف شناسه ها یا ثوابت

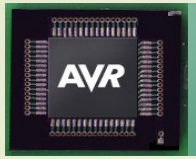
شناسه (ماکرو) برای تعریف برچسب هایی معادل یک نام رشته ای یا هر مقداری می باشند و ضمناً می توانند برای تعریف ماکرو های شبه تابع استفاده شوند که در واقع شبه دستور EQU در زبان اسمبلی عمل می کند. برای درک بیشتر به مثالهای زیر توجه کنید.

```
#define  name    “Micro”
#define  number  125.456
#define  out1   PORTA.5
```

در این تعاریف ،کلمه name با مقدار رشته ای ”Micro“ برابر است و number نیز با عدد 125.456 برابر است دقت داشته باشید که شما ثوابت را تعریف می کنید بنابراین مقدار های nameیاnumber نمی توانند در برنامه تغییر کنند.

مثالی دیگر :

```
#define sum (x+y)
        در اینجا حاصل ۴و۶ یعنی عدد ۱۰ در متغیر از قبل تعریف شده یعنی i قرار می گیرند.//
i=sum(4,6);
```



متغیرها

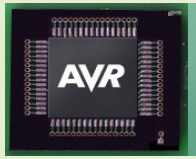
متغیرنامی است که برای حافظه موقت یا دائمی میکروکنترلر تعریف می شود.
متغیر یا داده می تواند در توابع دارای مقدار باشد و اعمال منطقی و ریاضیاتی بر روی آن در قالب یک تابع انجام پذیرد.
برای نام گذاری متغیرها از حروف کوچک و بزرگ لاتین (A...Z or a...z) و همچنین از علائمی نظیر آندرلاین (_) استفاده می شود.
برای نام گذاری متغیر نمی توان از کلمات کلیدی استفاده کرد.
متغیر نمی تواند با عدد شروع شود ولی می تواند به عدد منتهی شود.

آموزش زبان C

انواع داده ها در زبان C

نوع متغیر	اندازه بر حسب بیت	محدوده تغییرات
bit	1	0 or 1
char یا signed char	8	-128 to 127
signed char	8	0 to 255
int یا signed int	16	-32768 to 32767
signed int	16	0 to 65535
signed short int	16	-32768 to 32767
unsigned short int	16	0 to 65535
long int	32	-2147483648 to 2147483647
unsigned long int	32	0 to 4294967295
float	32	±1.175e-38 to ±3.402e38
double	32	±1.175e-38 to ±3.402e38

جدول ۲-۱ انواع داده‌های استاندارد در زبان C



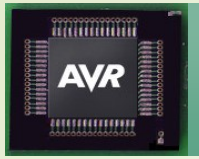
انواع متغیر در زبان C

1. **متغیر های کلی :** این متغیر ها بعد از معرفی پیش پردازنده ها و شناسه در ابتدای برنامه و خارج از بدنه های توابع می آیند و مقدار آنها در تمامی توابع قابل دسترسی می باشند و مقدار خود را در تمامی توابع حفظ می کنند .
2. **متغیر های محلی :** این متغیرها در داخل بدنه توابع تعریف می شوند و مقدار آنها با خارج شدن از بدنه توابع از بین می رود یعنی صفر می شوند. معرفی متغیر به شکل زیر است:

نوع داده نام متغیر ;

مثال : دو متغیر ۸ بیتی بی علامت zgX تعریف کنید و به متغیر x مقدار اولیه ۱۲ بدهید و یک متغیر دیگر با نام k از همین نوع در آدرس 200 هگزاد SRAM و همچنین یک متغیر از نوع اعشاری با نام fv تعریف کنید.

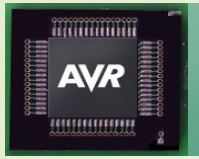
```
Unsigned char  x=12,z;
Unsigned char  k@0x200;
Float  fv=0.0;
```



متغیر های ZgX چون از یک نوع داده بودند در یک خط تعریف شده اند و با علامت کاما از هم جدا شده اند. این کار برای تعریف سریعتر متغیر در یک خط به کاربر کمک می کند .

می توانستیم متغیر های ZgX را به شکل زیر تعریف کنیم و مقدار X را در ابتدا ندهیم.

```
unigned char  x;
unsigned char  z;
X=12;
```



آموزش زبان C

فرمت یا مبنای متغیر ها

❖ ( ) : اگر شما قبل از عدد علامتی قرار ندهید به معنای دسیمال است.

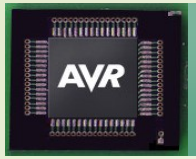
```
Unsigned  char  x=15;      // به معنی مقدار دادن ۱۵ دسیمال می باشد
Unsigned char  x=15u;     //.قرار دادن حرف Uبعد از دسیمال اختیاری است
Float                     //.قرار دادن حرف F بعد از یک داده اعشاری اختیاری است
x=3.14F;
```

❖ (0x) :اگر قبل از یک عدد بیاید به معنی هگزاد می باشد.

```
Unsigned char x=0x12;  به معنی قرار دادن مقدار ۱۲ هگزاد می
باشند.//
```

❖ (0b) : اگر قبل از عدد بیاید به معنای باینری است.

```
Unsigned char x=0b11001111;  به معنی قرار دادن مقدار بر حسب باینری می
باشد//
```

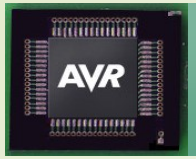


- (“ ”) : اگر کلمه ای در بین دو علامت دابل کوتیشن قرار می گیرد یعنی مقدار رشته ای است..

```
Char  x[]=“alvandi”]
```

- (‘ ’) : اگر یک کاراکتر در بین دو علامت کوتیشن قرار گیرد مقدار بر حسب اسکی است.

```
Unsigned  char  x=‘s’;
```



کلاس ذخیره سازی متغیر ها

در معرفی یک متغیر می توان نحوی ذخیره سازی آن را تعریف کرد . برای تعیین کلاس ذخیره سازی می بایست قبل از تعریف نوع داده متغیر ، کلاس آن را تعیین کنیم.

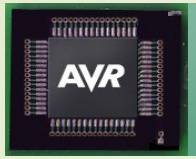
فرم کلی :

< کلاس ذخیره سازی حافظه > < نوع داده > نام متغیر ;

انواع کلاس ذخیره سازی :

- **Auto:** متغیر های که در داخل بدنه یک تابع تعریف می شوند ، متغیر های محلی نامیده می شوند.

این متغیرها با فراخوانی یک تابع ،در درون آن شکل می گیرند و با برگشتن از تابع از بین می روند به این نوع کلاس ذخیره سازی ، کلاس حافظه اتوماتیک گفته می شود.



آموزش زبان C

- **Static:** متغیر های محلی که از نوع ثابت تعریف می شوند با خارج شدن از یک تابع از بین نمی روند و فقط در بدنه همان تابعی که تعریف شده اند قابل دسترسی هستند.

- **extern:** این نوع کلاس برای متغیرهایی است که در یک فایل دیگر (مثل فایل سرآمد)معرفی و مقدار دهی اولیه شده است در فایل جاری برنامه ،می توان از آنها استفاده کرد.

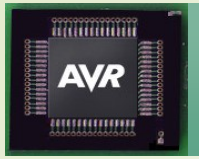
- **Register:** این نوع کلاس ،نحوه ذخیره شدن یک متغیر را در یکی از رجیسترهای میکروکنترلر تعیین می کند و فقط در قالب متغیرهای محلی کاربرد دارند.

عملگرها

در زبان C به علائم منطقی و ریاضیاتی ،عملگر گفته می شود. عملگر ها را نمی توان به عنوان دستور نامید زیرا آنها فقط یک عمل خاصی را انجام می دهند.

تقدم عملگر	عملگر	مفهوم	مثال
اولویت اول	++	افزایش به اندازه یک واحد	X++
	--	کاهش به اندازه یک واحد	x--
اولویت دوم	-	علامت منفی	-x
اولویت سوم	*	ضرب	x * y
	/	تقسیم (خارج قسمت)	X/10
	%	تقسیم (باقی مانده)	X%10
اولویت چهارم	-	تفریق	x-y
	+	جمع	X+y

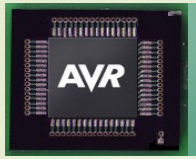
جدول ۲-۲ عملگرهای محاسباتی



مثال ۲-٦ : مقدار متغیر a=14 را بعد از یک واحد افزایش در متغیر b قرار دهید.

```
unsigned  char  a=14,b;                    (قسمت اول)
b=a++;     //b=14

unsigned  char  a=14,b;                    (قسمت دوم)
b=++a;     //b=15
```

در مثال ۲-۶ قسمت اول اشتباه می باشد . چون عملگر ++ در اولی بعد از متغیر a آمده است بنابراین ابتدا مقدار قبلی a در متغیر b قرار می گیرد و بعد یک واحد به متغیر a اضافه می شود.

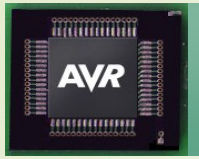


مثال ۲-۷ : چه مقداری در متغیر Z بعد از انجام عبارت زیر قرار قرار می گیرد؟

```
unsigned int x=7,y=6,z;
z=x+y*6/2;
```

برای مثال ۲-۷ ممکن است شما جواب های متفاوتی همچون اعداد ۳۹ و ۲۱.۵ و ۲۵ داشته باشید اما جواب صحیح عدد ۲۵ می باشد چرا که تقدم عملگر نشان دهنده حاصل عبارت خواهد بود.
در این عبارت ابتدا y در عدد ۶ ضرب شده و حاصل آن بر عدد ۲ تقسیم شده و نتیجه با مقدار متغیر x جمع می شود.

نکته : هر گاه از دو عملگر با تقدم یکسان استفاده شود اولویت با عملگری است که اول بیاید . در این مثال چون عملگر ضرب ،زودتر از عملگر تقسیم آمده است ،ابتدا عمل ضرب انجام می گیرد.



مثال ۲-۸: چه مقداری در متغیر Z بعد از انجام عبارت زیر قرار قرار می گیرد؟

```
unsigned int x=7,y=6,z;
z=(x+y)*(6/2) ;
```

در این مثال حاصل عبارت ۳۹ خواهد بود.

نکته : هر گاه یک عبارت در بین علامت پرانتز () قرار گیرد ،اولویت بالاتری خواهد داشت.

تقدم عملگر	عملگر	مفهوم	مثال
اولویت اول	>	بزرگتر	a>b
	>=	بزرگتر یا مساوی	a>=b
	<	کوچکتر	a<b
	<=	کوچکتر یا مساوی	a<=b
اولویت دوم	==	متساوی	a==b
	!=	نامساوی	a!=b

جدول ۲-۳ عملگرهای مقایسه‌ای (رابطه‌ای)

آموزش زبان C

تقدم عملگر	عملگر	مفهوم	مثال
اولویت اول	!	نقیض (not)	!x
اولویت دوم	&&	وَ (and)	a && b
اولویت سوم	||	یا (or)	a || b

جدول ۲-۴ عملگرهای منطقی

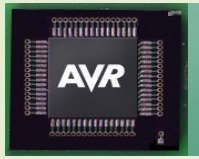
آموزش زبان C

عملگر	مفهوم	مثال	معادل
=	مساوی یا mov	x=12	
*=	انتساب ضرب	x*=y	x=x*y
/=	انتساب تقسیم	x/=10	x=x/10
%=	انتساب باقی مانده تقسیم	x%=10	x=x%10
+=	انتساب جمع	x+=y	x=x+y
-=	انتساب تفریق	x-=y	x=x-y
&=	انتساب AND بیتی	x&=z	x=x&z
^=	انتساب XOR بیتی	x^=z	x=x^z
|=	انتساب OR بیتی	x|=z	x=x|z
<<=	انتساب شیفت به چپ	x<<=5	x=x<<5
>>=	انتساب شیفت به راست	x>>=3	x=x>>3

جدول ۲-۵ عملگرهای بیتی و منطقی

تقدم عملگر	عملگر	مفهوم	مثال	نتیجه
اولویت اول	~	مکمل ۱	~ (0x66)	0x9A
اولویت دوم	>>	شیفت به راست	0b11001011 >> 4	0b00001100
	<<	شیفت به چپ	0b11001011 << 4	0b10110000
اولویت سوم	&	AND بیتی	0x66 & 0xff	0x66
اولویت چهارم	^	XOR بیتی	0x12 ^ 0x12	0
اولویت پنجم	|	OR بیتی	0x05 | 0x30	0x35

جدول ۲-۶ عملگرهای انتسابی یا ترکیبی



عملگرشرطی

فرم کلی استفاده از این عملگر بصورت زیر است:

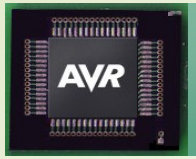
<عبارت ۳> : <عبارت ۲> ؟ <عبارت ۱> = متغیر

در این فرم دستوری عبارت شرطی اول مورد ارزیابی قرار می گیرد که اگر نتیجه آن درست باشد عبارت دوم در متغیر قرار می گیرد و در غیر اینصورت عبارت سوم در متغیر قرار خواهد گرفت.

مثال ۲-۹:

```
Y=(x > z) ? X : z
```

در این مثال مقدار X و Z با هم مقایسه می شوند که اگر X بزرگتر باشد عبارت دوم یعنی X در متغیر Y قرار می گیرد وگرنه مقدار Z در متغیر Y قرار خواهد گرفت.

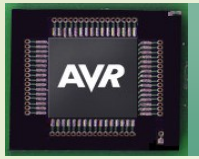


عملگرهای & و *

با استفاده از عملگر & می توانیم به آدرس یک متغیر و با استفاده از عملگر * می توانیم به محتوای آدرس یک متغیر دسترسی داشته باشیم.

مثال ۲-۱۰:

```
unsigned   char      //SRAM در هگزاد ۴۰۰ مکان در متغیر در معرفی
y@0x400;
unsigned   char      // i دلخواه و متغیر x گر اشاره متغیر معرفی
*x,i;
// مقدار دهی متغیر با عدد ۱۵ بر حسب دسیمال
y=15;
//متتغیر x برابر آدرس متغیرy یعنی ۴۰۰ هگزاد می باشد
x=&y;
//محتوای مکانی که xبه آن اشاره می کند را داخل i کپی می کنند.
i=*x;
```



عملگر کاما (,)

این عملگر برای جدا سازی چند متغیر ،عبارت و عمل دستوری می باشد.

مثال ۲-۱۱ :

```
unsigned char a,b ;
b=(a=10 , a/2 );
```

عملگر کاما در سطر اول ،برای جدا سازی دو متغیر در یک سطر است و در سطر دوم برای جداسازی دو عبارت بکار رفته است که حاصل متغیر b برابر 5 خواهد بود.

عملگر sizeof ()

این عملگر طول یک متغیر یا نوع داده را بر حسب بایت بر می گرداند به طور مثال داریم :

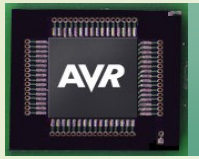
```
Unsigned int y,x ;
x=sizeof(y); //طول نوع داده int بر حسب بایت ۲ می باشد پس مقدار x=2 می شود.
```

عملگر ( )

پرانتزها عملگرهایی هستند که تقدم عملگر های داخل خود را بالا می برند و در تعریف توابع نیز کاربرد دارند. جدول ۲-۷ این مسئله را بیان می کند.

تقدم	عملگر	تقدم	عملگر
۸	&	۱	( )
۹	^	۲	! ~ ++ -- sizeof
۱۰	|	۳	* / %
۱۱	&&	۴	+ -
۱۲	||	۵	<< >>
۱۳	?	۶	< <= > >=
۱۴	= += -= *= /= %=	۷	== !=

جدول ۲-۷ تقدم عملگرها در حالت کلی



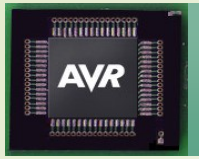
دستورات زبان C

دستورات زبان C در قالب بدنه یک تابع می آیند و یک عمل خاصی را بر روی متغیر های کلی و محلی انجام می دهند . هر دستور در زبان C با علامت } باز می شود و با علامت { بسته می شود.

1. **دستور شرطی if-else:** از این دستور برای ارزیابی یک شرط کنترلی در برنامه استفاده می شود. ساختار کلی این دستور به صورت زیر است :

```
If (شرط) {
دستورالعمل های ۱ ;
}
Else {
 دستورالعمل های ۲
}
```

در این دستور ابتدا شرط if بررسی می شود اگر درست باشد دستور العمل ۱ انجام و در غیر اینصورت دستورالعمل ۲ اجرا می شود.



مثال ۲-۱۲ :

```
Unsigned  char  a,b ;
If (a > b) {
a ++ ;
b+=10 ; }
Else {
a -- ;
b-=10 ;
}
```

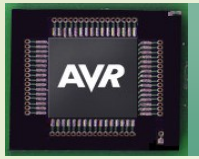
در این مثال اگر متغیر a از متغیر b بزرگتر باشد a یک واحد افزایش پیدا کرده و به متغیر b عدد 10 اضافه می شود و در غیر اینصورت متغیر a یک واحد کم شده و از متغیر b عدد 10 کم می شود.

مثال ۲-۱۳ :

```
Unsigned  char  zx,sd ;
If (zx==sd)  zx++ ;
Else sd++ ;
```

در این مثال اگر هر دو متغیر با هم برابر باشند ،به متغیر ZX یک واحد اضافه می شود و در غیر اینصورت یک واحد به متغیر sd اضافه خواهد شد .

همچنین می بینید که علامت های {} وجود ندارند . زمانی که یک دستورالعمل وجود داشته باشد نیازی به علامت } و { نمی باشد.



مثال ۲-۱٤ :

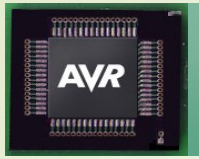
```
Unsigned  char  data ,i ;
Bit  x,y ;
If ((x==1)&&(y==0)
{
Data*=10 ;
i=data ;
}
```

در این مثال متغیر های تک بیتی x وy با هم مقایسه می شوند که اگر x=1 باشد و y=0 باشد data در عدد10 ضرب می شود و در متغیر i کپی می شود .
همچنین علامت } در پایین if آمده است و هیچ فرقی ندارد.

مثال ۲-۱۵ :

```
Unsigned  int  x,y ;
If ((x==y) && (z==15)) x++;
Else if ((x>=y) && (z==15)) y++;
Else if ((x<=y) && (z==15)) z++;
```

همانطور که می بینید از دستور else if برای شرط های متوالی استفاده شده است . اگر شرط سطر دوم درست باشد متغیر X یک واحد افزایش پیدا می کند . در غیر اینصورت شرط سطر سوم مورد ارزیابی قرار می گیرد و همینطور تمامی دستورات به طور متوالی تست می شوند.

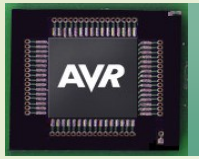


۲. دستور حلقه for : از این دستور برای ساختار حلقه شرطی در برنامه و زمانی که نیاز به کانتر حلقه باشد،استفاده می شود.

فرم کلی این دستور بصورت زیر است:

```
for      ( شمارنده حلقه  ; شرط حلقه  ; مقدار دهی اولیه ; )
{
دستورالعمل ها;
}
```

در دستور for ابتدا متغیر حلقه ، مقدار دهی اولیه می شود و در هر مرحله از اجرای حلقه ، یک واحد شمارنده حلقه افزایش پیدا می کند و تا زمانی که شرط حلقه درست باشد ،این حلقه تکرا می شود.



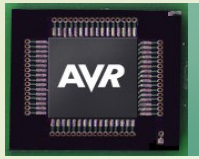
مثال ۲-۱۶ :

```
Unsigned  char  i ;
Unsigned  int  w;
For  ( i=0 ; i<25  ;i++ ) {
W+=5 ;
}
```

در این مثال متغیر حلقه ، یعنی i در ابتدای حلقه صفر است و در هر مرحله که حلقه تکرار می شود یک واحد افزایش پیدا می کند تا زمانی که مساوی عدد 25 شده و شرط حلقه نقض می شود و خط برنامه از حلقه خارج می گردد.

بنابراین حلقه 25 بار تکرار می گردد و مقدار 5 در هر مرحله از تکرار به متغیر W اضافه می شود.

توجه کنید که مقدار دهی اولیه یک بار در ابتدای ورود به حلقه انجام می گیرد و در هر بار تکرار حلقه مقدار دهی اولیه انجام نمی شود.

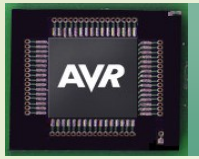


حلقه در حلقه با دستور for:

در بسیاری از موارد نظیر تاخیر های طولانی ،برگرداندن کد های آرایه مانند برنامه تابلو روان و ... ما نیاز به حلقه های تو در تو داریم که آنها را با استفاده از دستور for بوجود می آوریم.

مثال ۲-۱۷ :

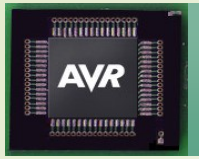
```
Unsigned char  i,j,z=2;
For  (i=10;  i>10;  i-- )  {
For (j=0;  j<=50;  j++ ) {
Z*=10;
If  (z==140)  z=2;
}
}
```



در مثال ۲-۱۷ برنامه ابتدا وارد حلقه اول شده و متغیرi=10 می شود و چون بزرگتر از عدد صف (شرط درست) است وارد حلقه دوم شده و متغیر حلقه دوم یعنی j=10 می شود و چون j کوچکتر مساوی عدد 50 می باشد ،این حلقه ادامه پیدا می کند و متغیر z در عدد 10 ضرب شده و با دستور if مورد ارزیابی قرار می گیرد که اگر مقدار آن برابر عدد 140 شود ،متغیر z برابر 2 می گردد .
بنابراین حلقه دوم ۵۱ مرتبه تکرار می شود و سپس برنامه از حلقه دوم خارج و یک واحد از شمارنده حلقه اول کم شده و چون شرایط همچنان برقرار است دوباره وارد حلقه دوم می شود.

- ایجاد حلقه بی نهایت با استفاده از دستور for بصورت زیر است:

```
For ( ;  ; ) {
دستورالعمل ها;
}
```



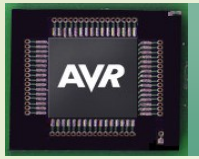
۳.دستور حلقه شرطی while:

این دستور نیز برای تکرار اجرای دستورات در یک حلقه برنامه می باشد با این تفاوت که فاقد شمارنده حلقه است .

این دستور ابتدا شرط حلقه را تست می کند اگر شرط درست باشد ،خط برنامه وارد حلقه می شود و در غیر اینصورت این حلقه هرگز اجرا نمی شود . این دستور بعد از هر بار تکرار حلقه شرط را بررسی می کند اگر نتیجه شرط حلقه درست باشد (غیر صفر باشد) حلقه را ادامه می دهد و در غیر اینصورت از حلقه خارج می شود .

فرم کلی این دستور به صورت زیر می باشد:

```
While (شرط)  {
دستورالعمل ها;
}
```



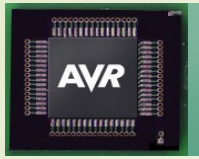
مثال ۲-۱۸ :

```
Unsigned char a,b;
While (a>b) {
A= (a/b*2) ;
}
```

در این مثال ابتدار متغیر a با b مقایسه شده که اگر a بزرگتر از b باشد ، برنامه وارد حلقه شده و در هر بار تکرار حلقه شرط مورد تست قرار می گیرد . همانطور که قبلا گفته شد در موقعیکه یک دستورالعمل وجود دارد می توانیم علامت های {} را قرار ندهیم اما گذاشتن آنها خطا محسوب نمی شود.

- ایجاد کردن حلقه بی نهایت با استفاده از دستور while بصورت زیر است:

```
While (1) {
دستورالعمل ها;
}
```



٤.دستور حلقه شرطی do-while:

این دستور نیز مانند دستور قبلی است با این تفاوت که،ابتدا اجازه تکرار یکبار حلقه را می دهد و در پایان حلقه ،شرط را بررسی می کند.

فرم این دستور به صورت زیر است :

```
Do {
دستورالعمل ها;
} while  (شرط) ;
```

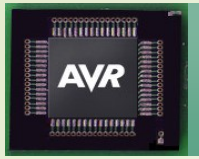
مثال ۲-۱۹:

```
Unsigned  char  i,sw,y;
Do {
i++ ;
Y=sw*i ;
} while (i < 10) ;
```

در این مثال ابتدا برنامه وارد حلقه شده و متغیر شرط حلقه یک واحد افزایش پیدا می کند و متغیر sw در i ضرب شده و حاصل آن در y قرار می گیرد و شرط در پایان حلقه تست می شود اگر شرط درست بود دوباره حلقه تکرار می شود و در غیر اینصورت از حلقه خارج می گردد.

❑ ایجاد حلقه بی نهایت با استفاده از دستور do-while به صورت زیر است:

```
Do {
دستورالعمل ها;
} while (1) ;
```



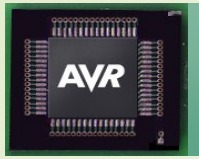
۵. دستور break:

این دستور برای شکستن و خارج شدن بدون شرط از حلقه می باشد. هر گاه برنامه به این دستور برسد از ساختار حلقه (منظور علامت {}) خارج شده و برنامه از اولین دستور بعد از ساختار حلقه ادامه می یابد . معمولا دستور break همراه با دستور switch استفاده می شود.

٦. دستور switch:

یکی از دستورات مهم زبان C دستور مقایسه ای switch می باشد. این دستور یک عبارت (متغیر) را با یکسری از اعداد ثابت مقایسه می کند و عمل خاصی را مطابق مقایسه انجام شده صورت می دهد .

در این دستور عبارت با مقدار های ثابت قرار داده شده مقایسه می شود و اگر با مقداری که جلوی دستور case قرار می گیرد برابر بود،بعد از انجام دستورالعمل آن ،با استفاده از دستور break سریعا از ساختار دستور switch خارج می گردد.



فرم کلی دستور switch یه صورت زیر است:

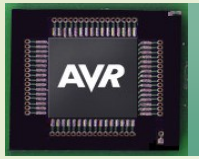
```
Switch (عبارت)  {
Case ۱ مقدار;
دستورالعمل های اول;
Break ;
Case ۲ مقدار;
دستورالعمل های دوم;
Break ;
Default:
دستورالعمل های پیش فرض;
}
```

در دستور switch ، عبارت فقط با مقادیر ثابت مقایسه می شود. بنابراین نمی توان در جلوی دستور case از متغیر استفاده کرد.

مثال ۲-۲۰:

```
Unsigned  char  x,z ;
Switch (x) {
Case  '1' : z=10 ;
Break ;
Case 26 :  z=20;
Break ;
Default  : z=50;
}
```

در این مثال مقدار x با عدد اسکی 1 مقایسه می شود که اگر درست باشد z=10 می شود و از دستور switch خارج می شود ولی اگر برابر نبود با عدد 26 دسیمال مقایسه می شود که اگر درست باشد z=20 خواهد شد و از دستور switch خارج می شود ام اگر با هیچ یک از مقادیر caseها برابر نبود Z به طور پیش فرض برابر50 می شود و از switch خارج می شود.



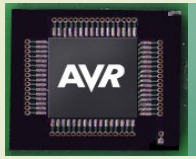
۷. دستور goto:

این دستور برای پرش به یک برچسب محلی داخل یک تابع می باشد . فرم آن به صورت زیر است:

goto برچسب ;

مثال ۲-۲۱:

```
Loop :
دستورالعمل ها;
Goto  loop ;
```



۸.دستور continue:

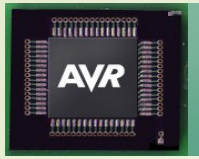
هرگاه خط برنامه به این دستور برسد برنامه به ابتدای حلقه تکرار ، پرش می کند و شرط بررسی می شود که اگر نادرست باشد حلقه خاتمه می یابد .

فرم این دستور به صورت زیر است:

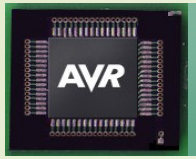
Continue ;

مثال ۲-۲۲ :

```
Unsigned  char  x=1 ,i,n;
While (x)  {
i++ ;
If ( n==10 ) {
X=0 ;
Continue ;
}
}
```



در مثال ۲-۲۲ برنامه وارد حلقه می شود که شرط آن صفر نبودن x است. متغیر i در هر مرحله تکرار حلقه ، یک واحد افزایش پیدا می کند و متغیر n نیز که در قسمت دیگری مثل وقفه ها مقدار دهی می شود مورد تست قرار می گیرد که اگر برابر با عدد 10 بود x=0 شده و به دستور continue رسیده و خط برنامه به ابتدای حلقه پرش کرده و شرط مورد ارزیابی قرار می گیرد و چون شرط برقرار نیست برنامه از حلقه خارج می گردد.



۹. دستور typedef:

با استفاده از این دستور می توانیم برای داده ها در زبان C یک نام جدید تعریف کنیم و نوع داده متغیر ها را با نام جدید تعریف کنیم .

فرم استفاده از این دستور بصورت زیر است :

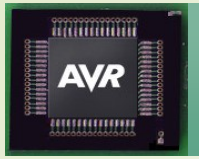
typedef نام قدیمی نوع داده نام جدید نوع داده ;

مثال ۲-۲۳ :

```
typedef  unsigned  int  2byte ;
```

اگر بصورت این مثال ابتدای برنامه ، نامی برای نوع داده در نظر گرفته شود می توان یک متغیر ۱۶ بیتی بدون علامت را به این شکل تعریف کرد:

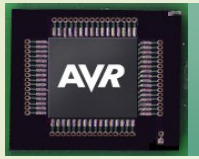
```
2byte  x ;
```



توابع در زبان C

در حقیقت یک برنامه به زبان C از یک تابع اصلی به نام main و چند تا تابع فرعی با هر نام مجازی تشکیل شده است .

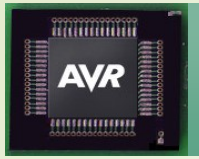
- تابع زیر برنامه ای است که دستورات و فرخوانی توابع دیگر در آن وجود داشته و عمل خاصی را انجام می دهد.
- هر برنامه در زبانC دارای یک تابع اصلی است. منظور از تابع اصلی ،تابع main یک برنامه می باشد. در هنگامی که یک برنامه آغاز می شود اولین تابعی که اجرا می شود تابع main می باشد.در صورتی که تابع اصلی با نام main نداشته باشیم نرم افزار کامپایلر اعلام خطا می کند.
- تابع اصلی مانند یک ریشه درخت می ماند ، بنابراین این تابع می تواند انشعاب داشته باشد و دیگر توابع را فراخوانی کند ولی نمی توان از توابع فرعی تابع اصلی را فراخوانی کرد.



فرم کلی تعریف تابع returnبصورت زیر است :

```
{ (آرگومانهای تابع)نام تابع    نوع برگشتن تابع    کلاس ذخیره سازی تابع
; دستورالعمل ها
return  متغیر یا عدد  ;
}
```

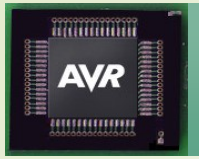
همانطور که قبلا بحث شد در تعریف توابع نیز می توان کلاس ذخیره سازی را تعریف کرد اما الزامی نیست. همچنین می توان نوع داده برگشتی توسط یک تابع را مشخص نمود.
برگشت یک تابع یعنی متغیر یا مقدار عددی که با دستور return تعیین می گردد.
در واقع توابع مانند یک فرمول ریاضیاتی می باشند که در نهایت پس از انجام اعمال خاصی در برنامه ، یک حاصلی را نیز به همراه خواهند داشت.



مثال ۲-۲٤: یک تابع فرعی با نام display تعریف کرده و آن را فراخوانی کنید.

```
Void   display ( )  {
دستورالعمل ها;
}
Void  main (  ) {
Display ( ) ;
دستورالعمل ها;
}
```

همانطورکه گفته شد توابع قبل از فراخوانی می بایست تعریف شوند حال اگر شما از فایل های کتابخانه ای استفاده کنید در واقع در داخل این فایل ها توابعب که عمل خاصی را انجام می دهند تعریف شده است و ما بعد از معرفی فایل کتابخانه ای در برنامه ،می توانیم توابع آن را نیز فراخوانی کنیم .

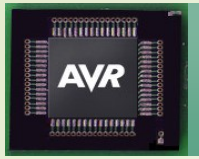


مثال ۲-۲۵: برنامه یک شمارنده 0 تا 9 با سون سگمنت کاتد مشترک را بنویسید.

```
#include <mega16.h>        // معرفی میکروکنترلر استفاده شده
#include <delay.h>         // معرفی کتابخانه تاخیر زمانی
unsigned                   // تعریف متغیر ۸ بیتی برای کانتر حلقه for
                                                    char i;
unsigned char mask(unsigned char num) { تابع برگشتی کد
//سون سگمنت
Switch (num) {             //مقایسه مقدار متغیر با اعداد زیر
Case 0 : return 0x3f;      //کد معادل عدد صفر برای سون سگمنت کاتد مشترک
Case 1 : return 0x06;      //کد معادل عدد یک برای سون سگمنت کاتد مشترک
Case 2 : return 0x5b;      //کد معادل عدد دو برای سون سگمنت کاتد مشترک
Case 3 : return 0x4f;      //کد معادل عدد سه برای سون سگمنت کاتد مشترک
Case 4 : return 0x66;      //کد معادل عدد چهار برای سون سگمنت کاتد مشترک
Case 5 : return 0x6d;      //کد معادل عدد پنج برای سون سگمنت کاتد مشترک
Case 6 : return 0x7d;      //کد معادل عدد شش برای سون سگمنت کاتد مشترک
```

ادامه مثال ۲-۲۵

```
Case  7: return  0x07;      //کد معادل عدد هفت برای سون سگمنت کاتد مشترک
Case  8: return  0x7f;      //کد معادل عدد هشت برای سون سگمنت کاتد مشترک
Case  9: return  0x6f;      //کد معادل عدد نه برای سون سگمنت کاتد مشترک
Default :return 0x00;       //کد پیش فرض
}
}
Void  main ( ) {        // تابع اصلی برنامه
PORT=0x00;          // مقدار اولیه پورت متصل به سون سگمنت را صفر قرار می دهیم
//      تعیین      پورت      A      به      عنوان      خروجی
                                                    DDRA=0xff;
While (1) {         // ایجاد کردن حلقه بی نهایت برای تکرار شمارش
For (i=0  ;  i<=9  ;  i++ ) {      //ایجاد یک حلقه که ۱۰ مرتبه تکرار می شود
PORTA=mask (i) ; //برگشت تابع کد معادل سون سگمنت کانتر حلقه بوده و به خروجی ارسال می شود.
Delay_ms (1000) ;     // تاخیر زمانی ۱ ثانیه برای شمارش
```

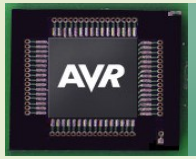


در مثال ۲–۲۵ ابتدا برنامه وارد تابع اصلی main می شود و پس از تنظیم پورت متصل سون سگمنت ، وارد یک حلقه بی نهایت می شود و یک حلقه for جهت شمارش و تغییر مقدار آرگومان تابع mask که فراخوانی می شود ، استفاده شده است.

در واقع مقدار i در موقع فراخوانی در متغیر num کپی می شود. متغیر num به عنوان پارامتر یا آرگومان تابع معرفی شده است . مقدار num در دستور switch قرار می گیرد و مطابق با هر یک از مقادیر ثابت که برابر بود یک کد معادل سون سگمنت کاتد مشترک را بر می گرداند.

این مقدار برگشتی در PORTA که به سون سگمنت کاتد مشترک وصل شده است فرستاده می شود.

مقدار داده برگشتی ۸ بیتی بدون علامت می باشد.



آرایه ها

به متغیر های به هم پیوسته که یکجا تعریف می شوند ،آرایه گفته می شود و نحوه تعریف کردن آنها مشابه متغیر ها می باشد.

آرایه ها می توانند یک بعدی یا چند بعدی بصورت ماتریسی باشند.

فرم کلی تعریف آرایه یک بعدی بصورت زیر است:

نوع داده نام آرایه[اندازه متغیرها];

نوع داده می تواند یکی از انواع داده باشد و نام آرایه نیز می تواند یک نام مجاز از حروف لاتین بوده و تعدا عضو های موجود در آرایه نیز می توانند تعریف شوند.

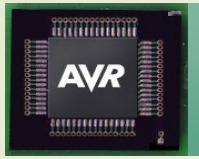
برای دسترسی به اعضای آرایه می بایست نام آرایه به اضافه شماره عضو در بین دو علامت [ ] قرار بگیرند.

مثال ۲-۲۶: یک آرایه تک بعدی با چهار عضو تعریف کنید و به آنها مقدار دهید.

```
Unsigned  char  code [4];          //تعریف یک آرایه یک بعدی با ۴ عضو
Code [0] = 0x10;                   //مقدار دهی به عضو اول
Code [1] =0xaf;                    //مقدار دهی به عضو دوم
Code [2] =0x19;                    //مقدار دهی به عضو سوم
Code [3] =0x66;                    //مقدار دهی به عضو چهارم
```

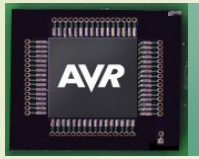
مثال ۲-۲۷: آرایه قبلی را تعریف کنید و همان لحظه اول مقدار دهی را انجام دهید.

```
Unsigned  char  code [4]= {0x10, 0xaf, 0x19, 0x66};
```



مثال ۲-۲۸: برنامه یک شمارنده 0 تا 9 را این بار با استفاده از آرایه ذخیره شده در حافظه ثابت میکروکنترلر بنویسید.

```
#include  <mega16.h>      // معرفی میکروکنترلر استفاده شده
#include  <delay.h>       // معرفی کتابخانه تاخیر زمانی
Flash  unsigned  char display []={  //کد های سون سگمنت کاتد مشترک در حافظه ثابت
0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x7f, 0x6f };
Void  main () {                    // تابع اصلی برنامه
Unsigned    //for  تعریف  متغیر  ۸  بیتی  برای  کانتر  حلقه
                                                    char  i;
PORTA=0x00;                  //مقدار اولیه پورت متصل به سون سگمنت را صفر قرار می دهیم
//          تعیین          پورت          A          به          عنوان          خروجی
                                                    DDRA=0xff;
While (1) {                                  // ایجاد کردن حلقه بی نهایت برای تکرار شمارش
For (i=0   ;   i<=9   ;   i++ )  {      //ایجاد یک حلقه که ۱۰ مرتبه تکرار می شود
PORTA=display [i] ; //کانتر حلقه اندیس عضو آرایه را تعیین کرده و به خروجی ارسال می کنیم..
Delay_ms (1000) ;      // تاخیر زمانی ۱ ثانیه برای شمارش
} };   }
```



در مثال ۲-۲۸ ابتدا در تابع main متغیر محلی i تعریف می شود و پورتA به عنوان خروجی تنظیم می گردد و برنامه وارد حلقه بی نهایت while می شود و حلقه for تشکیل می گردد در هر بار که حلقه for تکرار می شود شمارنده حلقه یعنی i افزایش می یابد و به ترتیب اعضای آرایه display را به پورت A ارسال می کند و در اینجا تاخیر حلقه یک ثانیه است هر وقت i=10 شود شرط حلقه for برقرار نخواهد بود و وارد حلقه بی نهایت می شود و مجددا وارد حلقه for می شود.

فرم کلی تعریف آرایه دو بعدی بصورت زیر است:

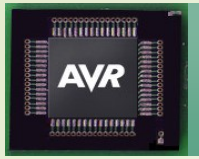
نوع داده نام آرایه[اندازه بعد ۱][اندازه بعد ۲];

اندازه بعُد اول تعداد سطرها و اندازه بُعددوم تعداد ستونهای آرایه را تعیین می کند که در واقع یک ماتریس را به وجود می آورد. ترتیب نوشتن مقدار اولیه آرایه های دو بعدی باید رعایت شود.

مثال ۲-۲۹: یک آرایه دو بعدی تعریف کنید و به اعضای آن دسترسی پیدا کنید.

```
Unsigned  char  x,j;
Unsigned  char matrix [2][3] = {{ '0' , '1' , '2' }
                                { '3' , '4' , '5' }};
x= matrix [1] [2];
J= matrix [0] [0];
```

در این مثال مقدار گرفته شده در متغیر x='5' خواهد بود چون سطر دوم وستون سوم انتخاب شده است و در متغیر j نیز عدد اسکی '0' قرار می گیرد.
اعداد می توانند در آرایه ها با هر مبنایی باشند و لزومی ندارد که حتما اعداد اسکی باشند و دقت کنید اندیس اعضا از صفر شروع می شود.



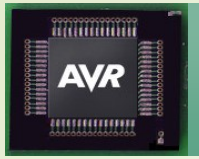
رشته ها

در واقع رشته ها همان آرایه هایی هستند که از کاراکتر هایی به هم پیوسته تشکیل شده اند و بین دو علامت " " تعریف می گردند . رشته ها در برنامه نویسی فقط به صورت تک بعدی تعریف می شوند.

مثال۲-۳۰: یک رشته بدون تعیین اعضا تعریف کنید.

```
Char  name []=" alvandi "
```

هر رشته به یک کاراکتر تهی null (برابر 0دسیمال یا '\0' اسکی) ختم می شود .کاراکتر تهی نشاندهنده پایان رشته می باشد . بنابراین اگر آرایه هایی داشته باشیم که اندازه آن ۸ باشد فقط از ۷ مکان آن استفاده می شود که آخری یک کاراکتر تهی می باشد.



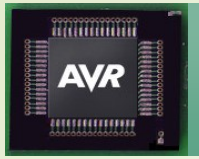
ساختمان (ساختارها)

ساختارها مجموعه ای از اعضای متغیرها با نوع داده های مختلف، تحت عنوان یک نام می باشند.تاکنون هر متغیری را که تعریف می کردیم می بایست نوع داده آن نیز ذکر شود و اگر متغیری با نوع داده دیگری بخواهد تعریف شود. اما توسط ساختارها ما می توانیم برای اعضای ساختمان خود یک نام کلی در نظر بگیریم.

فرم کلی تعریف ساختار بصورت زیر است:

```
Struct  <نام ساختمان>
اعضای ساختمان;
}اسامی متغیرهای ساختمان;
```

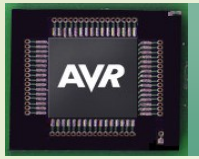
کلمه کلیدی struct نشان دهنده تعریف ساختمان می باشد ونام ساختمان می تواند از حروف لاتین باشد و اعضای ساختمان نیز می توانند متغیر هایی با هر نوع داده ای باشند و اسامی متغیر های ساختمان نیز می توانند با هر نامی از حروف لاتین تعریف شوند.



مثال ۲-۳۱:

```
Struct  time  {
Char  contrl ;
Unsigned  int  hour,minute,seconds;
}par;
...
Par.contrl=0x12;
Par.minute=0x60;
```

برای دسترسی به اعضای ساختار از علامت " . " استفاده می کنیم و همچنین اگر اشاره گری از نوع ساختمان تعریف شده باشد با استفاده از علامت "->"قابل دسترسی است.

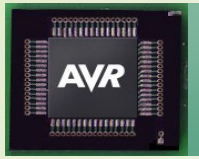


یونیون ها (اتحاد ها)

اتحاد ها نیز مانند ساختارها یک نوع داده گروهی می باشند. تفائت آنها با ساختار در این است که از مکانهای حافظه اختصاص یافته به اتحاد ها ،بین اعضای گروه به طور مشترک استفاده می شود.

فرم کلی تعریف اتحاد ها به صورت زیر است:

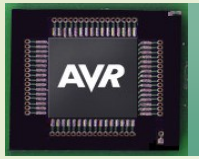
```
Union   نام اتحاد {
 اعضای اتحاد ها;
} اسامی متغیر های اتحاد ها;
```



مثال ۲-۳۳:

```
Union  save_R {
Char  x,y;
Int z;
} s_data;
```

نحوه دسترسی به اعضای اتحادها نیز مانند ساختار ها می باشد. در این مثال دو متغیر ۸ بیتی Xو Y و یک متغیر ۱۶ بیتی داریم.
پس اگر از نوع ساختمان باشد در مجموع ۳۲ بیت حافظه مورد نیاز است.ولی چون از اتخادها استفاده کردیم ،فقط به ۱۶ بیت نیاز داریم و از طرفی چون ،برنامه یک مکان ۱۶بیتی را برای اعضای گروه به اشتراک می گیرد ،در هر لحظه می توان فقط یکی از اعضای گروه استفاده کرد.

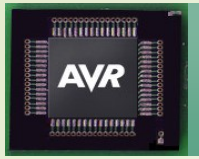


شمارش ها

نوع داده شمارشی برای تعریف یک مجموعه متناهی جهت نام گذاری یا شماره گذاری است.

فرم کلی تعریف نوع داده شمارشی بصورت زیر است:

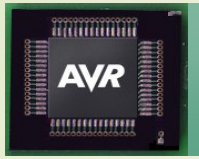
```
Enum نام شمارش {
اعضای شمارش;
اسامی متغیر های شمارش };
```



مثال ۲-۳٤:

```
Enum  color{
Red;
Blue;
Green;
}color1,color2;
```

در این مثال به عنصر اولی یعنی red عدد۱ و به دومی یعنی blue عدد ۲ و به سومی یعنی green عدد ۳ نسبت داده می شودو همین طور اگر ما کلی عنصر در یک شمارشگر قرار دهیم اعداد صحیح به آنها نسبت داده می شود،مگر اینکه کاربر شماره عنصر را تعیین کند.
مثلا green=6،آنگاه به عنصر سوم عدد ۶ نسبت داده می شود.



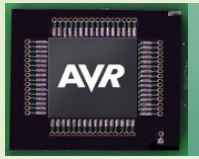
توابع کتابخانه های استاندارد

توابع کتابخانه های استاندارد

کتابخانه های ctype,stdlib,string,math جزء کتابخانه های استاندارد در زبان C می باشند و کامپایلر Code Vision AVR نیز دارای این کتابخانه ها می باشد.
توابع کتابخانه ای در مسیر نصب نرم افزار در پوشه INC وlib قرار دارند که ما در صورت نیاز ،می بایست آنها را در ابتدای برنامه معرفی کنیم و از توابع آماده ای آنها استفاده کنیم.

برخی از این توابع عبارتند از :

STDIO,STDARG,SPI,SLEEP,SETJMP,PCF8583,MEM,LM75,LCD4X40,LCD,io,GRAY,BCD,43USB355,86RF401,DS1307,DELAY,mega16,TINY13,90s8535



توابع کتابخانه های استاندارد

کتابخانه ctype.h

Unsigned char isalnum(char c)

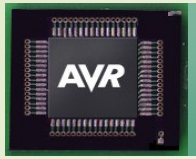
اگر c یک کاراکتر پارامتر ورودی ،یکی از حروف الفبایی –رقمی لاتین (شامل حروف و ارقام) باشد،مقدار 1 را برمی گرداند و در غیر اینصورت مقدار 0 را بر میگرداند.

Unsigned char isalpha(char c)

اگر c یک کاراکتر پارامتر ورودی ،یکی از حروف الفبای لاتین(فقط شامل حروف) باشد مقدار 1 را بر می گرداند و در غیر اینصورت مقدار 0 را بر می گرداند.

Unsigned char isascii(char c)

اگر c یک کاراکتر پارامتر ورودی باشد اگر این کاراکتر یکی کاراکتر های اسکی در محدوده اعداد (0 تا 127) باشد مقدار 1و در غیر اینصورت مقدار 0 را بر می گرداند.



توابع کتابخانه های استاندارد

Unsigned char iscntrl(char c)

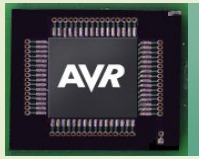
اگر c یک کاراکتر کنترلی به عنوان پارامتر ورودی تابع باشد،اگر این کاراکتر ورودی در محدوده اعداد (0 تا 31 یا 127)باشد،مقدار 1 را و در غیر اینصورت مقدار 0 را بر میگرداند.

Unsigned char isdigit(char c)

اگر c یک کاراکتر پارامتر ورودی یکی از اعداد دسیمال 0 تا 9 باشد این تابع مقدار 1 و در غیر اینصورت مقدار 0 را بر می گرداند.

Unsigned char islower(char c)

اگر c یک کاراکتر پارامتر ورودی یکی از حروف کوچک لاتین a تا z باشد، این تابع مقدار 1و در غیر اینصورت مقدار 0 را بر می گرداند.



توابع کتابخانه های استاندارد

Unsigned char isprint(char c)

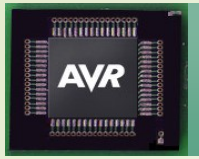
اگر c یک کاراکتر پارامتر ورودی یکی از کاراکترهای قابل چاپ در محدوده (32 تا 127) باشد این تابع مقدار 1 و در غیر اینصورت مقدار 0 را بر میگرداند.

Unsigned char ispunct(char c)

اگر c یک کاراکتر پارامتر ورودی یکی از کاراکترهای علائمی مانند (؟ ، ! و ...) باشد این تابع مقدار 1 و در غیر اینصورت مقدار 0 را بر می گرداند.

Unsigned char isspace(char c)

اگر c یک کاراکتر پارامتر ورودی یکی از کاراکترهای فضای خالی ' \0 ' و یا carrige return (" \r\n")باشد، این تابع مقدار 1و در غیر اینصورت مقدار 0 را بر می گرداند.



توابع کتابخانه های استاندارد

Unsigned char isupper(char c)

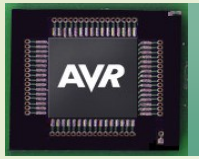
اگر c یک کاراکتر پارامتر ورودی یکی از حروف لاتین A تا Z باشد. این تابع مقدار 1 و در غیر اینصورت مقدار 0 را بر می گرداند.

Unsigned char isxdigit(char c)

اگر c یک کاراکتر پارامتر ورودی یکی از اعداد هگزا دسیمال 0 تاF باشد این تابع مقدار 1 و در غیر اینصورت مقدار 0 را بر می گرداند.

Unsigned char toint(char c)

این تابع کاراکتر c را به عنوان پارامتر ورودی بر حسب هگزاد دسیمال می گیرد و معادل رقمی بر حسب دسیمال اعداد 0 تا 15 را بر می گرداند.



توابع کتابخانه های استاندارد

char toascii (char c)

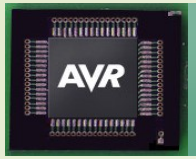
این تابع کاراکتر C را به عنوان پارامتر ورودی می گیرد و معادل اسکی آن را بر می گرداند.

char tolower (char c)

این تابع کاراکتر C را به عنوان پارامتر ورودی می گیرد و اگر این کاراکتر حرف بزرگ باشد آن را به حرف کوچک تبدیل می کند و آن را بر میگرداند در غیر اینصورت خود کاراکتر C را بر می گرداند.

char toupper (char c)

این تابع کاراکتر C را به عنوان پارامتر ورودی می گیرد و اگر این کاراکتر حرف کوچک باشد آن را به حرف بزرگ تبدیل می کند و آن را بر میگرداند در غیر اینصورت خود کاراکتر C را بر می گرداند.



توابع کتابخانه های استاندارد

کتابخانه stdlib.h

Int atoi (char *str)

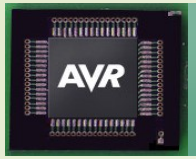
یک متغیر رشته ای به عنوان پارامتر ورودی می پذیرد و آن را به عدد صحیح تبدیل می کند.

Long int atol (char *str)

این تابع یک متغیر رشته ای به عنوان پارامتر ورودی می پذیرد و آن را به عدد صحیح بلند می کند.

Void itoa (int n, char *str)

این تابع یک متغیر عدد صحیح n و یک متغیر رشته ای str را به عنوان پارامتر ورودی می گیرد و عدد صحیح را تبدیل به کاراکتر های اسکی کرده و در متغیر رشته ای str دخیره می کند.



توابع کتابخانه های استاندارد

Void Itoa (long int n, char *str)

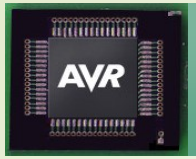
این تابع یک متغیر عدد صحیح بلند n و یک متغیر رشته ای str را به عنوان پارامتر ورودی می گیرد و عدد صحیح بلند را تبدیل به کاراکتر های اسکی کرده و در متغیر رشته ای str ذخیره می کند.

Void ftoa (float n, unsigned char desimals, char *str)

این تابع یک متغیر اعشاری n و یک متغیر عدد صحیح دسیمال و یک متغیر رشته ای str را به عنوان پارامتر ورودی می گیرد و متغیر اعشاری را تبدیل به کاراکترهای اسکی کرده و در متغیر رشته ای str ذخیره می کند. مقدار متغیر decimals دقت ارقام اعشار برای تبدیل به کاراکتر اسکی راتعیین می کند.

Float atof (char *str)

یک متغیر رشته عددی را به عنوان پارامتر ورودی می پذیرد و ان را به عدد اعشاری تبدیل می نماید.



توابع کتابخانه های استاندارد

Void ftoe (float n, unsigned char desimals, char *str)

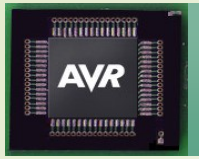
این تابع یک متغیر اعشاری n و یک متغیر عدد صحیح دسیمال و یک متغیر رشته ای str را به عنوان پارامتر ورودی می گیرد و متغیر اعشاری را تبدیل به کاراکترهای اسکی کرده و در متغیر رشته ای str ذخیره می کند. مقدار متغیر decimals دقت ارقام اعشار را تعیین می کند.
به طور مثال با ۲ رقم دقت اعشار عدد 12.3598×10^{-5} را تبدیل به رشته "12.35e-5" می نماید.

Int rand (void)

این تابع در لحظه فراخوانی یک عدد صحیح از بین ارقام 0 تا 32767 را به طور تصادفی بر می گرداند.

Void srand (int seed)

این تابع مانند تابع قبلی عمل می کند با این تفاوت که رقم شروع برای انتخاب رقم تصادفی را به عنوان پارامتر ورودی می گیرد و سرانجام یک رقم تصادفی بر می گرداند.



توابع کتابخانه های استاندارد

کتابخانه math.h

Unsigned int abs (int x)

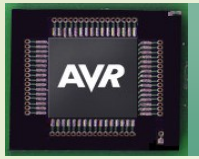
این تابع یک متغیر عدد صحیح به عنوان پارامتر ورودی می گیرد و قدر مطلق آن را بر می گرداند.

Unsigned long labs (long int x)

یک متغیر عدد صحیح بلند به عنوان پارامتر ورودی می گیرد و قدر مطلق آن را بر می گرداند.

Float fabs (float x)

یک متغیر عدد اعشاری بلند به عنوان پارامتر ورودی می گیرد و قدر مطلق آن را بر می گرداند.



توابع کتابخانه های استاندارد

Int max (int a ,int b)

تابع دو متغیر ۱۶ بیتی a وb را به عنوان پارامتر ورودی می گیرد و مقدار بزرگتر را بر می گرداند.

Int min (int a ,int b)

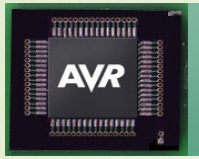
تابع دو متغیر ۱۶ بیتی a وb را به عنوان پارامتر ورودی می گیرد و مقدار کوچکتر را بر می گرداند.

Signed char sign (int x)

این تابع یک متغیر عدد صحیح را به عنوان پارامتر ورودی می گیرد و اگر متغیر منفی باشد 1-، صفر باشد 0 و مثبت باشد 1+ را بر می گرداند.

Signed char fsign (float x)

این تابع یک متغیر عدد اعشاری را به عنوان پارامتر ورودی می گیرد و اگر متغیر منفی باشد 1-، صفر باشد 0 و مثبت باشد 1+ را بر می گرداند.



توابع کتابخانه های استاندارد

Float sqrt (float x)

یک متغیر عدد اعشاری را به عنوان پارامتر ورودی می گیرد و مقدار $\sqrt{x}$ را بر می گرداند.

Float exp (float x)

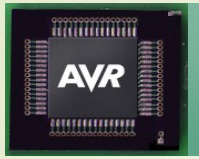
یک متغیر عدد اعشاری را به عنوان پارامتر ورودی می گیرد و مقدار e^xرا بر می گرداند.

Float log (float x)

یک متغیر عدد اعشاری را به عنوان پارامتر ورودی می گیرد و مقدار $Ln(x)$را بر می گرداند.

Float log 10 (float x)

یک متغیر عدد اعشاری را به عنوان پارامتر ورودی می گیرد و لگاریتم در مبنای ۱۰ را بر می گرداند.



توابع کتابخانه های استاندارد

Float pow (float x,float y)

این تابع دو متغیر اعشاری x و y را به عنوان پارامتر ورودی می گیرد و مقدار x^y را بر میگرداند.

Float sin (float x)

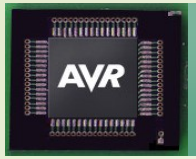
متغیر اعشاری را به عنوان پارامتر ورودی می گیرد و سینوس آن را بر حسب رادیان بر می گرداند.

Float cos (float x)

متغیر اعشاری را به عنوان پارامتر ورودی می گیرد و کسینوس آن را بر حسب رادیان بر می گرداند.

Float tan (float x)

متغیر اعشاری را به عنوان پارامتر ورودی می گیرد و تانژانت آن را بر حسب رادیان بر می گرداند.



توابع کتابخانه های استاندارد

Float asin (float x)

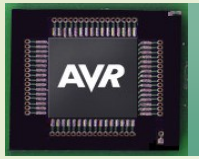
یک متغیر اعشاری را به عنوان پارامتر ورودی می گیرد و معکوس سینوس آن عدد را بر حسب رادیان بر می گرداند.

Float acos (float x)

یک متغیر اعشاری را به عنوان پارامتر ورودی می گیرد و معکوس کسینوس آن عدد را بر حسب رادیان بر می گرداند.

Float atan (float x)

یک متغیر اعشاری را به عنوان پارامتر ورودی می گیرد و معکوس تانژانت آن عدد را بر حسب رادیان بر می گرداند.



توابع کتابخانه های استاندارد

کتابخانه string.h

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

این تابع رشته str2 را به انتهای رشته str1 متصل می کند.

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

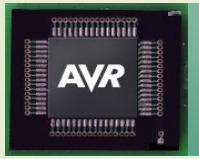
این تابع رشته str2 که در حافظه ثابت قرار دارد را به انتهای رشته str1 متصل می کند.

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

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

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

حداکثر n کاراکتر از رشته str2 که در حافظه ثابت قرار دارد را به انتهای رشته str1 متصل



توابع کتابخانه های استاندارد

Char *strchr (char *str, char c)

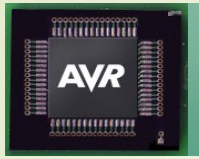
این تابع رشته str را برای محل اولین وقوع کاراکتر ورودی c جستجو می کند و آن را به یک اشاره گر نسبت می دهد و در غیر اینصورت اشارهگر برابر کاراکتر null (تهی) می شود.

Char*strrchr(char *str , char c)

این تابع رشته str را برای محل آخرین وقوع کاراکتر ورودی c جستجو می کند و آن را به یک اشاره گر نسبت می دهد و در غیر اینصورت اشاره گر برابر کاراکتر Null (تهی) می شود.

Signed char strcmp (char *stra1, char *str2)

این تابع رشته str را با رشته str2 مقایسه می کند. اگر str1>str2 مقدار بزرگتر از صفر و اگر str1=str2 مقدار صفر و اگر str1<str2 مقدار کوچکتر از صفر (عدد منفی) را برمی گرداند.



توابع کتابخانه های استاندارد

Signed char strcmpf (char *stra1, char flash *str2)

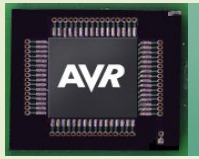
این تابع رشته str را واقع در حافظه SRAM با رشته str2 واقع در FLASHمقایسه می کند. اگر str1>str2 مقدار بزرگتر از صفر و اگر str1=str2 مقدار صفر و اگر str1<str2 مقدار کوچکتر از صفر (عدد منفی) را برمی گرداند.

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

این تابع رشته srcرا به رشته dest کپی می کند.

Char *strcpyf (char *dest, char falsh *src)

این تابع رشته src واقع در حافظه flashرا به رشته dest واقع در حافظه Sram کپی می کند.



توابع کتابخانه های استاندارد

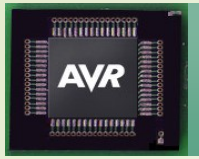
کتابخانه bcd.h

Unsigned char bcd2bin (unsigned char n)

این تابع یک پارامتر ورودی n بر حسب کد BCD می گیرد و کد باینری معادل آن را بر می گرداند.

Unsigned char bin2bcd (unsigned char n)

این تابع یک پارامتر ورودی n که می تواند در محدوده اعداد 0 تا 99 باشد را می گیرد و کدBCD معادل آن را بر می گرداند.



توابع کتابخانه های استاندارد

کتابخانه delay.h

Void delay_us (unsigned int n)

این تابع یک عدد ثابت به عنوان پارامتر ورودی در محدوده اعدا 0 تا 65535 می گیرد و بر حسب میکروثانیه تاخیر زمانی ایجاد می کند . به طور مثال برای تاخیر $100\mu s$ داریم :

```
Delay_us(100);
```

Void delay_ms (unsigned int n)

این تابع یک متغیر عدد صحیح به عنوان پارامتر ورودی در محدوده اعداد 0 تا65535 می گیرد و بر حسب میلی ثانیه تاخیر زمانی ایجاد می کند.

به طور مثال برای تاخیر یک ثانیه 1000ms داریم:

```
Delay_ms(1000);
```

همچنین فرض کنید یک متغیر مانند TD=250 مفروض باشد :

```
Delay_ms(TD);
```

در تابع فوق به مدت 250 میلی ثانیه تاخیر ایجاد می شود.