ریزپردازنده‌ها و زبان اسمبلی

فصل چهارم
اصول برنامه‌نویسی به زبان C

محمدعلی شفیعیان

http://shafieian-education.ir/

مقدمه‌ای بر زبان C

- زبان C زبانی همه منظوره، ساخت یافته و روندگرا می باشد که در سال ۱۹۷۲ توسط دنیس ریچی طراحی شد.
- قدرتمندترین زبان برنامه نویسی میکروکنترلرها زبان C می باشد.
- برنامه نویسی برای یک ماشین بر مبنای پردازنده های RISC با برنامه‌نویسی برای یک ماشین بر مبنای پردازنده های CISC تفاوت اساسی دارد و آن هم حساسیت بیشتر RISC نسبت به CISC می‌باشد که برنامه نویس را مجبور می کند تا با دقت بیشتر و درک بیشتر سخت‌افزار برنامه‌نویسی کند.
- به طور کلی زبان های برنامه نویسی را می توان در سه سطح دسته‌بندی کرد: زبان های سطح بالا، زبان های سطح میانی و زبان های سطح پایین.

کلمات کلیدی در زبان C

- زبان C زبان کوچکی است چرا که در آن تعداد کلمات کلیدی ۳۲ کلمه است.
- کم بودن کلمات کلیدی در یک زبان مبنی بر ضعف آن نیست.
- زبان بیسیک حاوی ۱۵۰ کلمه کلیدی است اما قدرت زبان C به مراتب بالاتر است.

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

+

interrupt	flash
defined	sfrw
Inline	sfrb
bit	
eeprom	

ویژگی‌های یک برنامه به زبان C

- زبان C به حروف کوچک و بزرگ حساس است. در این زبان بین حروف کوچک و بزرگ تفاوت است و تمام کلمات کلیدی باید با حروف کوچک نوشته شوند.
- هر دستور در زبان C با ; به پایان می رسد.
- حداکثر طول هر دستور ۲۵۵ کاراکتر است.
- هر دستور می تواند در یک یا چند سطر نوشته شود.
- برای توضیحات تک خطی از // در ابتدای خط استفاده می شود و یا توضیحات چند خطی در بین /* و */ قرار می گیرد .

ساختار کلی یک برنامه به زبان C

```
#include <Headrfile.h>

                                  #define out1 PORTC.0
محل معرفی متغیرهای عمومی، ثوابت و توابع  unsigned char x,y;
                                  unsigned int N, count

void function1(){
      دستورالعمل‌ها   unsigned char k,j;
}

void main(){
      محل مربوط به کدهای برنامه
       function1();
       while(1){

       };
}
```

تفاوت برنامه نویسی برای کامپیوتر و میکروکنترلر

- برنامه نویسی میکروکنترلرها (ماشین RISC) با هدف اجرا در میکروکنترلر ، با برنامه‌نویسی در کامپیوتر کمی متفاوت است.
- تفاوت اصلی در کامپیوتر این است که عامل اجرای برنامه ، در زمان نیاز به عملکرد آن ، یک کاربر است. هر زمان که کاربر نیاز به عملکرد برنامه داشته باشد آن را اجرا می کند و نتیجه را بررسی می‌کند.
- اما در میکروکنترلر رفتار یک آی سی است که عامل اجرای برنامه منبع تغذیه است. با وصل منبع تغذیه برنامه شروع به کار می کند و تا زمانی که منبع تغذیه وصل است باید کارهای مورد نیاز را دائما اجرا نماید.

تعریف شناسه‌ها و ثابت‌ها

شناسه (ماکرو) برای تعریف برچسب‌هایی معادل یک نام رشته‌ای یا هر مقداری می‌باشند و ضمناً می‌توانند برای تعریف ماکروهای شبه تابع استفاده شوند که در واقع شبیه دستور EQU در زبان اسمبلی عمل می‌کند.

```
#define name "Micro"
#define number 125.456
#define out1 PORTA.5


#define sum(x,y) x+y
i = sum(4,6);
```

متغیرها در زبان C

- یک متغیر محدوده‌ای از فضای حافظه است که با یک نام مشخص می شود.
- یک متغیر بسته به نوع آن می‌تواند حامل یک مقدار عددی باشد. یک متغیر می تواند در محاسبات شرکت کند و یا نتیجه محاسبات را در خود حفظ کند.

نوع متغیر	اندازه بر حسب بیت	محدوده تغییرات
bit	1	0 or 1
char یا signed char	8	-128 to 127
unsigned char	8	0 to 255
int یا signed int	16	-32768 to 32767
unsigned 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

متغیرها در زبان C

انواع متغیر در زبان C

۱– متغیرهای عمومی یا کلی (Global): این متغیرها بعد از معرفی پیش‌پردازنده‌ها و شناسه‌ها در ابتدای برنامه و خارج از بدنه‌های توابع می‌آیند و مقدار آن‌ها در تمامی توابع قابل دسترسی می‌باشد و مقدار خود را در تمامی توابع حفظ می‌کنند.

۲– متغیرهای محلی (Local): این متغیرها در داخل بدنه توابع تعریف می‌شوند و مقدار آن‌ها با خارج شدن از بدنه توابع از بین می‌رود یعنی صفر می‌شوند.

نوع متغیر نام متغیر = مقدار اولیه;

```
unsigned char A=12;
int a,X,j;
```

```
unsigned char x=12, z;
unsigned char k@0x200;
float fv=0.0;
```

متغیرها در زبان C

فرمت یا مبنای متغیرها

```
unsigned char X=15;
unsigned char X=15U;
float X=3.14F;
unsigned char X=0x12;
unsigned char X=0b11001111;
char X[]="Hello";
unsigned char X='S';
```

متغیرها در زبان C

ویژگی نام متغیرها

- اولین کاراکتر نام متغیر عدد نمی‌تواند باشد.
- نام متغیر بیشتر از ۳۱ کاراکتر مورد استفاده نیست.
- نام متغیر تنها ترکیبی از حروف a تا z و A تا Z و اعداد و کاراکتر _ می‌تواند باشد.

تعریف متغیرها در زبان C

کلاس ذخیره‌سازی متغیرها

در معرفی یک متغیر می‌توان نحوه‌ی ذخیره‌سازی آن را تعریف کرد. برای تعیین کلاس ذخیره‌سازی می‌بایست قبل از تعریف نوع داده متغیر، کلاس آن را تعیین کنیم. فرم کلی:

< کلاس ذخیره‌سازی حافظه > < نوع داده > نام متغیر ;

انواع کلاس ذخیره‌سازی :

auto extern

static register

تعریف متغیرها در زبان C

محل تعریف متغیرها در حافظه میکروکنترلر

- زمانی که یک متغیر تعریف می شود آن متغیر در اولین مکان خالی در حافظه SRAM ذخیره می‌شود.
- برای تعریف متغیر در حافظه EEPROM از کلمه کلیدی eeprom استفاده می‌شود و برای تعریف متغیر در حافظه FLASH از کلمه کلیدی flash قبل از تعریف متغیر استفاده می‌شود.
- باید توجه داشت که با قطع منبع تغذیه کلیه حافظه SRAM پاک خواهد شد و متغیرهایی که باید ذخیره دائمی شوند می بایست در حافظه EEPROM یا FLASH تعریف و ذخیره شوند.

```
int a;
eeprom char b;
flash float c;
```

تعریف متغیرها در زبان C

محل تعریف متغیرها در حافظه میکروکنترلر

نکته ۱: حافظه Flash حافظه برنامه کاربر است یعنی برنامه به زبان C بعد از کامپایل و پروگرام شدن روی میکروکنترلر در حافظه Flash ذخیره می‌شود. بنابراین برای تعریف متغیر در حافظه Flash تنها در صورت خالی بودن قسمتی از آن امکان پذیر است.

نکته ۲: حافظه SRAM تنها با قطع تغذیه پاک می شود و می‌توان کاری کرد که با ریست شدن میکرو متغیرهای عمومی موجود در SRAM ریست نشده و مقدار قبلی خود را حفظ نمایند.

عملگرها در زبان C

- در زبان C به علائم منطقی و ریاضیاتی، عملگر گفته می‌شود.
- عملگرها را نمی‌توان به‌عنوان دستور نامید زیرا فقط یک عمل خاص را انجام می‌دهند.

عملگرهای محاسباتی

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

عملگرها در زبان C

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

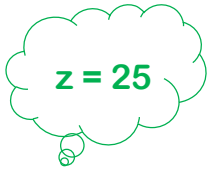
✗
```
unsigned char a=14,b;
b = a++;
```

✓
```
unsigned char a=14,b;
b = ++a;
```

عملگرها در زبان C

مثال: چه مقداری در متغیر z بعد از انجام عبارت زیر قرار می‌گیرد.

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

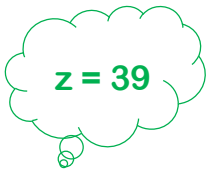


نکته: هرگاه از دو عملگر با تقدم یکسان استفاده شود، اولویت با عملگری است که اول بیاید.

عملگرها در زبان C

مثال: چه مقداری در متغیر z بعد از انجام عبارت زیر قرار می‌گیرد.

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



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

عملگرها در زبان C

عملگرهای مقایسه‌ای (رابطه‌ای)

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

عملگرها در زبان C

عملگرهای منطقی

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

عملگرها در زبان C

عملگرهای بیتی و منطقی

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

عملگرها در زبان C

عملگرهای انتسابی یا ترکیبی

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

عملگرها در زبان C

عملگر شرطی ؟

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

در این فرم دستوری، عبارت شرطی اول مورد ارزیابی قرار می‌گیرد که اگر نتیجه آن درست باشد، عبارت دوم در متغیر قرار می‌گیرد و در غیر این‌صورت عبارت سوم در متغیر قرار می‌گیرد.

```
y = (x>z) ? x:z
```

عملگرها در زبان C

عملگرهای & و *

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

```
unsigned char y@0x400;
unsigned char *x,i;
y = 15;
x = &y;
i = *x;
```

نکته: اگر از اشاره‌گر * استفاده می‌کنید، باید توجه داشته باشید که نوع داده اشاره‌گر باید با نوع داده آرایه یا رشته مورد نظر، یکسان باشد.

عملگرها در زبان C

عملگر کاما (,)

این عملگر برای جداسازی چند متغیر، عبارت و عمل دستوری می‌باشد.

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

عملگر sizeof()

این عملگر طول یک متغیر یا داده را بر حسب بایت برمی‌گرداند.

```
unsigned int y,x;
x = sizeof(y);
```

عملگر پرانتز ()

پرانتزها عملگرهایی هستند که تقدم عملگرهای داخل خود را بالا می‌برند و در تعریف توابع نیز بکار می‌روند.

عملگرها در زبان C

تقدم عملگرها در حالت کلی

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

دستورات زبان C

دستورات زبان C در قالب بدنه یک تابع می‌آیند و یک عمل خاص را بر روی متغیرهای عمومی و محلی انجام می‌دهند.

دستور شرطی if-else

از این دستور برای ارزیابی یک شرط کنترلی در برنامه استفاده می‌شود.

```
if (شرط) {
      دستورالعملهای ۱;
}
else{
      دستورالعملهای ۲;
}
```

دستورات زبان C

دستور شرطی if-else

```
unsigned char a,b;
if (a>b){
     a++;
     b+=10;}
else{
     a--;
     b-=10;
}
```

```
unsigned char data,i;
bit x,y;
if ((x==1)&&(y==0))
{
     data*=10;
     i=data;
}
```

```
unsigned char zx,sd;
if (zx==sd) zx++;
else sd++
```

```
unsigned int x,y,z;
if ((x==y)&&(z==15)) x++;
else if ((x>=y)&&(z==15)) y++;
else if ((x<=y)&&(z==15)) z++;
```

دستورات زبان C

دستور حلقه for

از این دستور برای ساختار حلقه شرطی در برنامه و زمانی که نیاز به کانتر حلقه باشد، استفاده می‌شود.

```
for (مقداردهی اولیه ; شرط حلقه ; شماره حلقه) {
        دستورالعمل‌ها;
}
```

دستورات زبان C

دستور حلقه for

```
unsigned char i;
unsigned int w;
for (i=0 ; i<25 ; i++){
     w+=5;
}
```

```
unsigned char i,j,z=2;
for (i=10 ; i>0 ; i--){
     for (j=0 ; j<=50 ; j++){
     z*=10;
     if (z==140) z=2;
     }
}
```

دستورات زبان C

دستور حلقه for

نکته: ایجاد کردن حلقه بینهایت با استفاده از دستور for به‌صورت زیر است:

```
for (;;) {
      دستورالعمل‌ها;
}
```

دستورات زبان C

دستور حلقه شرطی while

این دستور نیز برای اجرای دستورات در یک حلقه برنامه می‌باشد با این تفاوت که فاقد شمارنده حلقه است. این دستور ابتدا شرط حلقه را تست می‌کند و اگر شرط درست باشد، خط برنامه وارد حلقه می‌شود و در غیر این‌صورت این حلقه هرگز اجرا نمی‌شود.

این دستور بعد از هر بار تکرار حلقه، شرط را بررسی می‌کند. اگر نتیجه شرط حلقه درست (غیرصفر) باشد، حلقه را ادامه می‌دهد و در غیر این‌صورت از حلقه خارج می‌شود.

```
while (شرط) {
      دستورالعمل‌ها;
}
```

```
unsigned char a,b;
while (a>b){
      a=(a/b*2);
}
```

دستورات زبان C

دستور حلقه شرطی while

نکته: ایجاد کردن حلقه بینهایت با استفاده از دستور while به‌صورت زیر است:

```
while (1) {
      دستورالعمل‌ها;
}
```

دستورات زبان C

دستور حلقه شرطی do-while

این دستور نیز مانند دستور حلقه شرطی while است با این تفاوت که ابتدا اجازه تکرار یک بار حلقه را می‌دهد و در پایان حلقه شرط را بررسی می‌کند.

```
do {
      دستورالعمل‌ها;
} while (شرط);
```

```
unsigned char i,sw,y;
do (a>b){
     i++;
     y=sw*i;
} while(i<10)
```

دستورات زبان C

دستور حلقه شرطی do-while

نکته: ایجاد کردن حلقه بینهایت با استفاده از دستور do-while به‌صورت زیر است:

```
do {
      دستورالعمل‌ها;
} while (1);
```

دستورات زبان C

دستور break

- این دستور برای شکستن و خارج شدن بدون شرط از حلقه می‌باشد. هرگاه برنامه به این دستور برسد از ساختار حلقه (منظور علامت { }) خارج شده و برنامه از اولین دستور بعد از ساختار حلقه ادامه می‌یابد.
- معمولاً دستور break به‌همراه دستور switch استفاده می‌شود.

دستورات زبان C

دستور switch

- یکی از دستورات مهم زبان C دستور مقایسه‌ای switch می‌باشد. این دستور یک عبارت (متغیر) را با یک سری از اعداد ثابت مقایسه می‌کند و عمل خاصی را مطابق مقایسه انجام‌شده صورت می‌دهد.

- در این دستور عبارت (متغیر) با مقدارهای ثابت قرار داده‌شده مقایسه می‌شود و اگر با مقداری که جلوی دستور case قرار می‌گیرد برابر بود، بعد از انجام دستورالعمل آن، با استفاده از دستور break سریعاً از ساختار دستور switch خارج می‌گردد.

دستورات زبان C

دستور switch

```
switch (عبارت) {
        case مقدار ۱ :
            دستورالعمل‌های اول;
            break;
        case مقدار ۲ :
            دستورالعمل‌های دوم;
            break;
        default :
            دستورالعملهای پیش‌فرض;
}
```

```
unsigned char x,z;
switch (x){
     case '1' : z=10;
     break;
     case 26 : z=20;
     break;
     default : z=50;
}
```

دستورات زبان C

دستور goto

این دستور برای پرش به یک برچسب محلی در داخل یک تابع می‌باشد.

```
goto برچسب ;
```

```
Loop:
دستورالعمل‌ها;
goto Loop;
```

دستورات زبان C

دستور continue

هرگاه خط برنامه به این دستور برسد، برنامه به ابتدای حلقه تکرار پرش می‌کند و شرط بررسی می‌شود که اگر نادرست باشد حلقه خاتمه می‌یابد.

```
continue;
```

```
unsigned char x=1,i,n;
while (x){
      i++;
      if (n==10){
            x=0;
            continue;
      }
}
```

دستورات زبان C

دستور typedef

با استفاده از این دستور می‌توانیم برای داده‌ها در زبان C یک نام جدید تعریف کنیم و نوع متغیرها را با نام جدید تعریف کنیم.

typedef نام قدیمی نوع داده نام جدید نوع داده ;

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

توابع در زبان C

تابع یکی از مهمترین بخش های زبان C می باشد. یک تابع همانند دستگاهی است که مواد اولیه را دریافت می‌کند و بعد از انجام عملیات مورد نظر روی آنها خروجی مطلوب را تحویل می‌دهد.

توابع در زبان C
- توابع کتابخانه‌ای
- توابع تعریف‌شده توسط کاربر (توابعی که کاربر بر حسب نیاز برنامه خود اضافه می‌کند)

توابع در زبان C

زبان C دارای توابعی است که از قبل نوشته شده اند، و توابع کتابخانه ای نامیده می شوند. در واقع فرایندهایی که پر کاربرد هستند و در اغلب برنامه ها مورد استفاده قرار می‌گیرند به صورت توابع مستقل قبلاً نوشته شده‌اند و درون فایل‌هایی قرار داده شده‌اند . با اضافه کردن فایل های سرآمد که تعریف آن توابع در آنها قرار دارد می توان از آن توابع استفاده کرد .

تابع اصلی برنامه نویسی به زبان C تابع main نام دارد که در تمامی برنامه ها وجود داشته و بدون ورودی و خروجی است . توابع دیگر را می توان در بالای تابع main، در پایین تابع main و یا در فایل های کتابخانه‌ای تعریف کرد .

توابع در زبان C

فرم کلی تعریف تابع به‌صورت زیر است:

```
کلاس ذخیره‌سازی تابع   نوع برگشتن تابع   نام تابع(آرگومان‌های تابع) {
        دستورالعمل‌ها;
    return متغیر یا عدد;
}
```

نکته: در کامپایلر CodeVisionAVR مقدار برگشتی تابع در رجیسترهای R23,R22 و R31,R30 ذخیره می‌شوند.

نکته: درصورتی‌که نوع بازگشتی تابع void باشد، دیگر نیازی به دستور return نیست.

توابع در زبان C

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

```
void display( ){
      دستورالعمل‌ها;
}

void main( ){
      display( );
      دستورالعمل‌ها;
}
```

توابع در زبان C

مثال: برنامه یک شمارنده 0 تا 9 با Seven Segment کاتد مشترک را بنویسید.

```
#include <mega16.h>
#include <delay.h>
unsigned char i;
unsigned char mask(unsigned char num){
switch(num){
     case 0 : return 0x3f;
     case 1 : return 0x06;
     case 2 : return 0x5b;
     case 3 : return 0x4f;
     case 4 : return 0x66;
     case 5 : return 0x6d;
```

```
    case 6 : return 0x7d;
    case 7 : return 0x07;
    case 8 : return 0x7f;
    case 9 : return 0x6f;
    default : return 0x00;
        }
}
    void main(){
        PORTA = 0x00;
        DDRA = 0xff;
        while(1){
            for(i=0 ; i<=9 ; i++){
                PORTA = mask(i);
                delay_ms(1000);
            }
        }
    }
```

آرایه‌ها

- به متغیرهای به‌هم پیوسته که یک‌جا تعریف می‌شوند، آرایه گفته می‌شود و چگونگی تعریف کردن آن‌ها مشابه متغیرها می‌باشد.
- آرایه‌ها می‌توانند یک‌بعدی یا چندبعدی به‌صورت ماتریسی باشند.

نوع داده نام آرایه [اندازه متغیرها] ;

a = | a[0] | a[1] | a[2] | ... | a[n] |

- نوع داده می‌تواند یک از انواع داده باشد.
- نام آرایه نیز می‌تواند یک نام مجاز از حروف لاتین باشد.
- تعداد عضوهای موجود در آرایه نیز می‌تواند تعریف شود.
- برای دسترسی به اعضای آرایه می‌بایست نام آرایه به‌اضافه شماره عضو در بین دو علامت [] قرار بگیرد.

آرایه‌ها

مثال: یک آرایه تک‌بعدی با چهار عضو تعریف کنید و به آن‌ها مقدار بدهید.

```
unsigned char code[4];
code[0]=0x10;
code[1]=0xaf;
code[2]=0x19;
code[3]=0x66;
```

مثال: آرایه قبلی را تعریف کنید و همان لحظه اول مقداردهی را انجام دهید.

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

آرایه‌ها

آرایه دوبُعدی

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

تعداد سطرها تعداد ستون‌ها

B =

B[0][0]	B[0][1]	B[0][2]
B[1][0]	B[1][1]	B[1][2]

مثال: یک آرایه دوبعدی 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];
```

آرایه‌ها

مثال: برنامه یک شمارنده 0 تا 9 را این بار با استفاده از آرایه ذخیره‌شده در حافظه ثابت (flash) میکروکنترلر بنویسید.

```
#include <mega16.h>
#include <delay.h>
flash unsigned char display[]=
{0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0x7f, 0x6f};
void main(){
      unsigned char  i;
      PORTA = 0x00;
      DDRA = 0xff;
      while (1){
            for(i=0 ; i<=9 ; i++){
                  PORTA = display[i];
                  delay_ms(1000);
            }
      }
}
```

رشته‌ها

- در واقع رشته‌ها همان آرایه‌هایی هستند که از کاراکترهای به‌هم پیوسته تشکیل شده‌اند و بین دو علامت " " تعریف می‌گردند.
- رشته‌ها در برنامه‌نویسی فقط به‌صورت تک‌بُعدی تعریف می‌شوند.

مثال: یک رشته بدون تعیین اعضا تعریف کنید.

```
char name[] = "John";
```

اشاره‌گرها (Pointers)

اشاره‌گر متغیری است که به‌جای مقدار یک داده، محل یا آدرس آن را معرفی می‌کند. یعنی مقدار آن، آدرس یک محل از حافظه است. به‌عبارت دیگر، اشاره‌گر متغیری است که آدرس متغیر دیگری را در خود نگه می‌دارد.

نوع داده *نام اشاره‌گر ;

```
void main (){
     int *p,x,y;
     x = 50;
     p = &x;
     y = *p;
}
```

در این مثال، p اشاره‌گر از نوع int تعریف شده است. p حاوی آدرس متغیر x خواهد بود به‌طوری‌که اگر x در خانه 1000H از حافظه باشد، مقدار p برابر 1000H خواهد بود. متغیر y نیز برابر محتوای آدرس موجود در p یعنی محتوای آدرس 1000H است، بنابراین مقدار y همان مقدار x یعنی 50 است.

اشاره‌گرها (Pointers)

در کامپایلر CodeVisionAVR بر اساس معماری AVRها که حافظه به سه بخش مجزا برای داده (SRAM)، برنامه (FLASH) و EEPROM تقسیم شده است، دارای ۳ نوع اشاره‌گر می‌باشد.

نوع داده محل ذخیره اشاره‌گر *نام اشاره‌گر ;

یا

نوع داده محل ذخیره اشاره‌گر *نام اشاره‌گر ;

```
char *p;
eeprom char  *p1,x,y;
flash int *p2;
int eeprom *p3;
char flash *p4;
```

محل ذخیره اشاره‌گر می‌تواند یکی از کلمات falsh (یا const) و یا eeprom باشد و در صورت تعیین نکردن محل ذخیره اشاره‌گر، به‌صورت پیش‌فرض ناحیه SRAM انتخاب می‌شود.

اشاره‌گرها (Pointers)

آرایه‌ای از اشاره‌گرها

آر ایه‌ای از اشاره‌گرها، آرایه‌ای است که هر عنصر آن یک اشاره‌گر است.

```
float *a[6];
char  *b[3];
```

ساختمان (ساختارها) – Structures

- ساختارها مجموعه‌ای از اعضای متغیرها با نوع داده‌های مختلف، تحت عنوان یک نام می‌باشند.
- تا کنون هر متغیری را که تعریف می‌کردیم می‌بایست نوع داده آن نیز ذکر شود و اگر متغیری با نوع داده دیگری بخواهد تعریف شود، باید به‌طور جداگانه تعریف شود.
- اما توسط ساختارها می‌توانیم برای اعضای ساختمان خود یک نام کلی در نظر بگیریم.
- برای دسترسی به اعضای ساختار از علامت "." استفاده می‌کنیم.

```
struct نام ساختار{
        اعضای ساختار;
} اسامی متغیرهای ساختار ;
```

```
struct time {
      char contr1;
      unsigned int hour,minute,seconds;
}par;
...
par.contr1 = 0x12;
par.minute = 0x60;
```

ساختمان (ساختارها) – Structures

مثال:

```
struct abc{
      int a;
      char b;
      float c;
      char d[10];
} x,z;
```

```
struct abc{
      int a;
      char b;
      float c;
      char d[10];
};

stuct abc x,z;
```

```
x.a=9;
z.b = 'W';
z.c = 5.16;
x.d[1] = 'M';
```

ساختمان (ساختارها) – Structures

مقداردهی اولیه به ساختارها

```
struct abc{
      int a;
      char b;
      float c;
} x={6,'Q',3.5};
```

آرایه‌ای از ساختارها

```
struct abc{
      int a;
      char b;
      float c;
}x[20];
```

```
x[0].a = 9;
x[0].b = 'D';
x[0].c = 3.14;
x[1].a = 12;
```

ساختمان (ساختارها) – Structures

نکته: اگر اشاره‌گری از نوع ساختار تعریف شده باشد، با استفاده از علامت "->" قابل دسترسی است.

مثال:

```
struct time {
     char contr1;
     unsigned int hour,minute,seconds;
}par;
...
par.contr1 = 0x12;
par.minute = 0x60;

struct time *ptr;
ptr->seconds = 30;
```

ساختمان (ساختارها) – Structures

مثال:

```
struct abc{
     int a;
     char b;
     float c;
}x,z;

struct abc *p;


p->a = 2;
p->b = 'C';
p->c = 3.14;
```

ساختمان (ساختارها) – Structures

ساختار بیتی

اعضای یک ساختار را می‌توان به‌صورت بیتی نیز تعریف کرد که از یک بیت یا چند بیت تشکیل می‌شود.

نوع داده نام : طول بیت‌ها ;

```
struct bit{
      char a:3;
      int b:10;
      long c:30;
};
```

نکته ۱: در این روش حداکثر طول بیت‌ها با توجه به نوع داه مشخص می‌شود. به‌طور مثال اگر داده نوع char باشد، حداکثر طول بیت‌ها ۸ و اگر Long باشد، حداکثر طول بیت‌ها ۳۲ است.

نکته ۲: روش دستیابی به عناصر این ساختار مانند حالت‌های قبل است.

ساختمان (ساختارها) – Structures

نکته: در کامپایلر CodeVisionAVR این قابلیت وجود دارد که ساختارها را در ناحیه SRAM، FLASH یا EEPROM تعریف نمود.

محل ذخیره‌سازی ساختار struct نام ساختار {

اعضای ساختار ;

} اسامی متغیرهای ساختار ;

```
flash struct max{
      int a;
      char b;
};
```

```
eeprom struct min{
      int a;
      char b;
};
```

ساختمان (ساختارها) – Structures

نکته ۱: درصورتی‌که نام محل ذخیره‌سازی ساختار ذکر نشود، به‌طور پیش‌فرض فضای SRAM در نظر گرفته می‌شود. می‌توان از کلمه کلیدی const به‌جای flash نیز استفاده کرد.

نکته ۲: ساختارهایی که در حافظه FLASH تعریف می‌شوند، از نوع ساختار ثابت می‌باشند؛ یعنی فقط می‌توان عناصر آن را خواند و نمی‌توان در عناصر آن مقداری نوشت.

نکته ۳: هنگامی‌که ساختار در حافظه FLASH یا EEPROM تعریف می‌شود، به‌علت آن‌که اشاره‌گرها در حافظه SRAM ذخیره می‌شوند، لذا نمی‌توان آن‌ها را در ساختارهایی که در حافظه flash یا eeprom قرار دارند ذخیره نمود.

ساختمان (ساختارها) – Structures

```
eeprom struct abc{
      int a;
      char b;
      float c;
} *p;
```

در این حالت اشاره‌گر در همان حافظه EEPROM ذخیره می‌شود.

```
eeprom struct abc{
      int a;
      char b;
      float c;
};
struct abc *p;
```

در این حالت کامپایلر یک خطا گزارش می‌کند.

نکته ۴: توصیه می‌شود به‌علت حجم پایین SRAM در برخی از AVRها و همچنین به‌منظور کوچک نگه داشتن فضای پشته از ساختارها به‌عنوان پارامتر توابع استفاده نشود و به‌جای آن از اشاره‌گرها استفاده شود.

یونیون‌ها (اتحادها)

- اتحادها نیز مانند ساختارها یک نوع داده گروهی می‌باشند.
- تفاوت آن‌ها با ساختارها در این است که مکان‌های حافظه اختصاص یافته به اتحادها، بین اعضای گروه **به‌طور مشترک** (البته در زمان‌های متفاوت) استفاده می‌شود.

```
union نام اتحاد {
        اعضای اتحادها;
} اسامی متغیرهای اتحادها ;
```

```
union save_R {
        char x,y;
        int z;
}s_data;
```

دو متغیر ۸ بیتی x و y و یک متغیر ۱۶ بیتی z داریم و چون از یونیون استفاده کردیم فقط به ۱۶ بیت از حافظه نیاز داریم.
اگر از ساختار استفاده می‌کردیم به ۳۲ بیت از حافظه نیاز داشتیم.

نکته ۱: چگونگی دسترسی به عناصر یونیون مانند ساختارها می‌باشد.

نکته ۲: در کامپایلر CodeVisionAVR یونیون‌ها در حافظه FLASH قابل تعریف نیستند.

شمارش‌ها

نوع داده شمارشی، برای تعریف یک مجموعه متناهی جهت نام‌گذاری یا شماره‌گذاری است.

```
enum نام شمارش {
        اعضای شمارش;
} اسامی متغیرهای شمارش ;
```

```
enum color {
        red;
        blue;
        green;
}color1,color2;
```

به عنصر اول یعنی red عدد ۱ و به دومی یعنی blue عدد ۲ و به سومی یعنی green عدد ۳ نسبت داده می‌شود و این روند ادامه می‌یابد مگر اینکه کاربر شماره عنصر را تعیین کند؛ مثلاً green=6 آنگاه به عنصر سوم عدد ۶ نسبت داده می‌شود.

توابع کتابخانه‌ای استاندارد

- کتابخانه‌های ctype، stdlib، string و math جزء کتابخانه‌های استاندارد در زبان C می‌باشند و کامپایلر CodeVisionAVR نیز دارای این کتابخانه‌ها می‌باشد.
- توابع کتابخانه‌ای در مسیر نصب نرم‌افزار در پوشه INC و LIB قرار دارند که ما در صورت نیاز، می‌بایست آن‌ها را در ابتدای برنامه معرفی کنیم و از توابع آماده آن‌ها استفاده کنیم.

STDIO	SETJMP	LCD4X40	BCD	maga16
STDARG	PCF8583	LCD	43USB355	maga32
SPI	MEM	io	86RF401	DELAY
SLEEP	LM75	GRAY	DS1307	TINY13
				90S8535

توابع کتابخانه‌ای استاندارد

نکات مهم:

- قبل از استفاده از هر تابع کتابخانه‌ای، باید با دستور #include <نام کتابخانه.h> آن کتابخانه را در ابتدای برنامه معرفی کنیم.
- توابع کتابخانه‌ای که قبل از آن‌ها کلمه کلیدی void وجود دارد، هنگام فراخوانی نباید کلمه void را تایپ کنیم و همچنین برای توابع برگشتی نیز، نباید نوع داده برگشتی هنگام فراخوانی تابع تایپ گردد.
- توابعی که برگشت‌پذیر هستند، مقداری را که برمی‌گردانند ممکن است علامت دار یا بدون علامت باشد.

توابع کتابخانه‌ای استاندارد

نکات مهم:

- توابعی که پارامتر ورودی دریافت می‌کنند (خواه برگشت‌پذیر یا برگشت ناپذیر باشند)، باید به آن‌ها متناسب با پذیرش مقدار ورودی، مقداری را ارسال کنیم.
- توابعی که پارامتر ورودی آن‌ها رشته یا آرایه می‌باشد، هنگام ارسال رشته یا آرایه نباید علامت کروشه [ ] تایپ گردد و فقط نام رشته یا آرایه را می‌نویسیم.
- هنگام فراخوانی باید در انتهای هر تابع علامت ; قرار داده شود.
- تمامی توابع باید با حروف کوچک فراخوانی شوند.
- منظور از علامت * در برخی از توابع، اشاره‌گر رشته‌ای می‌باشد.

کتابخانه delay.h

از این کتابخانه برای ایجاد تأخیر در برنامه استفاده می‌شود.

نکته: مقدار تأخیر توابع این کتابخانه به مقدار فرکانس کاری برنامه در نرم‌افزار کامپایلر بستگی دارد که از مسیر Projects/Configure/C compiler تنظیم می‌شود.

```
void delay_us(unsigned int n)
```

این تابع یک عدد ثابت را به‌عنوان پارامتر ورودی در محدوده 0 تا 65535 می‌گیرد و برحسب میکروثانیه تأخیر زمانی ایجاد می‌کند.

```
delay_us(100);
```

کتابخانه delay.h

```
void delay_ms(unsigned int n)
```

این تابع یک عدد صحیح را به‌عنوان پارامتر ورودی در محدوده 0 تا 65535 می‌گیرد و برحسب میلی‌ثانیه تأخیر زمانی ایجاد می‌کند.

```
delay_ms(100);
```

```
unsigned int TD = 250;
delay_ms(TD);
```

دستورات اسمبلی در C

می‌توان در هر قسمت از برنامه به‌طور مستقیم و توسط عبارات #asm و #endasm دستورات اسمبلی را وارد نمود.

```
void delay (unsigned char i){
    while (i--){
        #asm
            nop
            nop
        #endasm
    }
}
```

همچنین می‌توان دستورات اسمبلی را در یک خط وارد نمود.

```
#asm ("sei")
```

برای کسب اطلاعات بیشتر در مورد این درس می‌توانید به وب سایت آموزشی در لینک زیر مراجعه نمایید

http://shafieian-education.ir