معرفی کوتاه زبان C

زبان برنامه نویسی c ، زبانی همه منظوره ، ساخت یافته و روند گرا می باشد. که در سال ۱۹۷۲ توسط دنیس ریچی در آزمایشگا های بل ساخته شد.

به طور کلی زبان های برنامه نویسی را می توان در سه سطح دسته بندی کرد:

- زبان های سطح بالا
- زبانهای سطح میانی
- زبانهای سطح پایین. زبان

زبان C یک زبان سطح میانی است که در آن هم می توان به سطح بیت و بایت و آدرس دسترسی داشت و هم از مفاهیم سطح بالا که به زبان محاوره ای انسان نزدیکتر است بهره گرفت. درزبان C هیچ محدودیتی برای برنامه نویس وجود ندارد و هر انچه را که فکر میکنید ، می توانید پیاده سازی کنید .

ارتباط تنگاتنگی بین زبان C و اسمبلی وجود دارد. به این صورت که می توان از برنامه نویسی اسمبلی در هر کجای برنامه زبان C استفاده کرد.

کلمات کلیدی در زبان C

زبان C زبانی کوچک است چرا که در آن 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 C به مراتب بالاتر است.

زبان C به حروف بزرگ و کوچک حساس است . (Case sensitive) در این زبان بین حروف بزرگ تفاوت است و تمام کلمات کلیدی باید با حروف کوچک نوشته شوند . در شکل زیر تمام کلمات کلیدی زبان C را مشاهده می کنید .

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

ویژگیهای یک برنامه به زبان C

- هر دستوردر زبان C با ; به پایان می رسد.
- حداکثر طول در دستور ۲۵۵ کاراکتر است.
- هر دستور می تواند در یک یا چند خط نوشته شود .
- توضیحات برنامه را می توان بین /* و */ (توضیحات چند خطی) یا بعد از // (توضیحات تک خطی) قرار داد.

ساختار یک برنامه به زبان C

هر زبان برنامه نویسی دارای یک ساختار کلی است. این ساختار یک قالب را برای برنامه نویسی فراهم می کند. ساختار کلی یک برنامه در زبان C را در زیر مشاهده می کنید.

```
#include <headerFile.h>
محل معرفی متغیرهای عمومی , ثوابت و توابع
Void main()
{
 محل مربوط به کدهای برنامه
}
```

بنابراین همانطور که مشاهده میشود :

۱. خطوط ابتدایی برنامه ، دستور فراخوانی فایلهای سرآمد (header file) می باشد ، فایل های سرآمد فایل هایی با پسوند h هستند که حاوی پیش تعریف ها و الگوهای توابع می باشند .
۲. قالب اصلی برنامه بر مبنای تابعی به نام main بنا شده است. تابعی که اصولا ورودی و خروجی ندارد و کدهای اصلی برنامه را در خود دارد.

ساختار برنامه میکروکنترلر به زبان C :

برنامه ای که کاربر می نویسد باید طوری نوشته شود که وقتی روی IC پروگرام شد دائما اجرا شود . راه حل این مسئله قرار دادن کدهای برنامه در حلقه ای نا متناهی است. این عمل باعث می شود تا میکروکنترلر هیچ وقت متوقف نشود و به طور مداوم عملکرد طراحی شده توسط کاربر را اجرا کند .

بنابراین ساختار یک برنامه به صورت زیر در می اید.

```
#include <HeaderFile.h>
محل معرفی متغیر های عمومی، ثابت ها و توابع
Void main (void)
{
 کدهایی که در این قسمت قرار می گیرند فقط یکبار اجرا می شوند
معمولا مقداردهی  اولیه رجیسترها در این ناحیه انجام می شود.
While(1)
{
کدهایی که باید به طور مداوم در میکروکنترلر اجرا شوند
}
}
```

متغیرها درزبان C

یک متغیر محدوده ای از فضای حافظه است که با یک نام مشخص می شود . یک متغیر بسته به نوع آن می توان حامل یک مقدار عددی باشد.

یک متغیر می تواند در محاسبات شرکت کند یا نتیجه محاسبات را در خود حفظ کند .

در کل می توان گفت که نتایج بخش های مختلف یک برنامه ، در متغیر ها ذخیره می شود .

در جدول زیر انواع متغیر ها ، فضایی که در حافظه اشغال می کنند و بازه مقدار پذیری آنها را در کامپایلر کدویژن مشاهده می کنید.

نوع داده	اندازه برحسب بیت	محدوده قابل قبول
bit	۱	۰ و ۱
char	۸	۱۲۸- تا ۱۲۷
unsigned char	۸	۰ تا ۲۵۵
signed car	۸	۱۲۸- تا ۱۲۷
int	۱۶	۳۲۷۶۸- تا ۳۲۷۶۷
short int	۱۶	۳۲۷۶۸- تا ۳۲۷۶۷
unsigned int	۱۶	۰ تا ۶۵۵۳۵
sniged int	۱۶	۳۲۷۶۸- تا ۳۲۷۶۷
long int	۳۲	۲۱۷۴۸۳۶۴۸- تا ۲۱۷۴۸۳۶۴۷
unsigned longint	۳۲	۰ تا ۴۲۹۴۶۷۲۹۵
signed longint	۳۲	۲۱۷۴۸۳۶۴۸- تا ۲۱۷۴۸۳۶۴۷
float	۳۲	±۳/۴۰۲e۳۸ تا ±۱/۱۷۵e۳۸
double	۳۲	±۳/۴۰۲e۳۸ تا ±۱/۱۷۵e۳۸

پیشوندهای مشخص کننده فرمت اعداد

- پیشوند 0b نشان دهنده اعداد باینری می باشد.(0b01010101)
- پیشوند 0x یا نشان دهنده اعداد هگزادسیمال می باشد (0x8F)
- اگر از پیشوند استفاده نکنیم ، در مبنا ۱۰ می باشد. (24)

نحوه تعریف متغیر ها

متغیرها به صورت زیر تعریف می شوند:

نوع متغیر نام متغیر = مقدار اولیه;

مثال :

```
unsigned char A=12;

int a,X,j=0x88;

int c=0b10101010;
```

توضیح : در خط اول متغیر ۸ بیتی بدون علامت با نام A که تنها می تواند مقادیر ۰ تا ۲۵۵ بگیرد با مقدار اولیه ۱۲ تعریف شده است . در خط دوم نیز ۳ متغیر علامت دار با نام های a و X و j که هر یک مقدار اولیه صفردارند تعریف شده است.

نکته : در صورت عدم مقدار اولیه در هنگام تعریف یک متغیر ، مقدار اولیه در حالت default برابر ۰ است.

ویژگیهای نام متغیر

- اولین کاراکتر نام متغیر نمی تواند عدد باشد.
- نام متغیر حداکثر ۳۱ کاراکتر باشد .
- نام متغیر ترکیبی از حروف ، اعداد و کاراکتر _ می تواند باشد.

انواع متغیرها از نظر محل تعریف در برنامه

متغیر ها از نظر مکانی که در برنامه تعریف می شوند ، به دو دسته کلی تقسیم می شوند :

۱- متغیرهای عمومی (Global)
۲- متغیرهای محلی (Local)

متغیرهایی که قبل از تابع main تعریف می شوند را متغیرهای عمومی گویند و در همه جای برنامه می توان به آن دسترسی داشت. اما متغیرهای محلی در بدنه توابع تعریف میشوند و در بیرون از ان تابع دسترسی به آن ممکن نیست. در واقع با تعریف یک متغیر عمومی در ابتدای برنامه ، مقدار مشخصی از حافظه برای همیشه به

آن متغیر اختصاص میابد اما متغیر های محلی تنها در زمان احتیاج تعریف شده و در حافظه می نشینند و بعد اتمام تابع از حافظه پاک می شوند.

محل تعریف متغیرها در حافظه میکروکنترلر

زمانی که یک متغیر به صورتی که در بالاگفته شد تعریف می شود آن متغیر در اولین مکان خالی در حافظه SRAM ذخیره می شود . برای تعریف متغیر در حافظه EEPROM ، از کلمه کلیدی eeprom و برای تعریف متغیر در حافظه Flash از کلمه کلیدی flash قبل از متغیر استفاده می شود .

بنابراین باید توجه داشت که با قطع منبع تغذیه ، کلیه حافظه SRAM پاک می شود و متغیرهایی که باید ذخیره دائمی شوند می بایست در حافظه EEPROM یا Flash تعریف و ذخیره شوند.

مثال :

```
int a;

eeprom char b;

flash float c;
```

نکته : همانطور که قبلاگفتیم ، حافظه flash ، حافظه برنامه کاربر است یعنی برنامه به زبان C بعد از کامپایل و ساخته شدن توسط کدویژن و پروگرام شده توسط پروگرامر روی میکروکنترلر در حافظه Flash ذخیره می شود.

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 بعدازانجام عملیات موردنظر روی آنها خروجی مطلوب را تحویل می دهد .

توابع در زبان C یا توابع کتابخانه ای هستند یا توابعی هستند که کاربر برحسب نیاز برنامه خود اضافه می کند.

زبان C دارای توابعی است که از قبل نوشته شده اند ، و توابع کتابخانه ای نامیده می شوند.

تابع اصلی برنامه نویسی به زبان سی تابع main است 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 کتابخانه ای تعریف کرد.

انواع توابع در زبان C

هر تابع مجموعه ای از دستورات است که بر روی داده ها پردازش انجام می دهد. ورودی و خروجی یک تابع مقادیری هستند که تابع در برنامه دریافت می کند و به برنامه باز می گرداند.

توابع برحسب ورودی و خروجی به ۴ دسته تقسیم می شوند :

۱- تابع با ورودی ، با خروجی

مثال: تابع با دو ورودی از جنس کاراکتر و یک خروجی از جنس کاراکتر

```
char F1(char x, char y);
```

۲- تابع با ورودی ، بدون خروجی

مثال : تابع دارای یک ورودی int و بدون خروجی

```
void F2(int x);
```

۳- تابع بدون ورودی ، با خروجی

مثال: تابع بدون ورودی با خروجی int

```
int F3(void)
```

۴- تابع بدون ورودی بدون خروجی

```
void F4(void)
```

بنابراین در اولین قدم باید مشخص کنیم که این تابع چه خروجی را به ما می دهد (در اصطلاح برنامه نویسی بر می گرداند) و فقط به ذکر نوع خروجی بسنده می کنیم ، یعنی اگر عدد صحیح برگرداند از int ، اگر کاراکتر برگرداند از char و به همین ترتیب برای انواع دیگر . و اگر هیچ مقداری را برنگرداند از void استفاده می کنیم.

در قدم بعدی نام تابع را مشخص می کنیم . یک تابع باید دارای نام باشد تا در طول برنامه مورد استفاده قرار گیرد. هر نامی را می توان برای توابع انتخاب نمود که ازقانون نامگذاری متغیرها تبعیت می کند . ، اما سعی کنید از نام های مرتبط با عمل تابع استفاده نمایید .

همینطور باید ورودی های تابع را مشخص کنیم. که در اصطلاح برنامه نویسی به این ورودیها ، پارامترها یا آرگومانهای تابع نیز گفته می شود . اگر تابع بیش از یک پارامتر داشته باشد ، باید آنها را با استفاده از کاما از یکدیگر جدا نماییم. و اگر تابع پارامتری نداشت از کلمه void استفاده می کنیم.

نام پارامتر هم می تواند هر نام دلخواهی باشد . به یاد داشته باشید که قبل ازنام هر پارامتر باید نوع آن را مشخص نماییم. و بدانیم که کامپایلر هیچ متغیری بدون نوع را قبول نکرده و در صورت برخورد با این مورد از برنامه خطا میگیرد و در نتیجه برنامه را اجرا نخواهد کرد.

بنابراین در زبان C ، توابع حداکثر دارای یک خروجی می باشند و اگر تابعی خروجی داشته باشد آن را با دستور return به خروجی تابع و برنامه اصلی بر می گردانیم .

تعریف توابع در زبان C

برای نوشتن و اضافه کردن یک تابع باید دقت کرد که هر تابع سه بخش دارد :

اعلان تابع ، بدنه تابع و فراخوانی تابع.

نحوه تعریف تابع به یکی از سه صورت زیر است :

- اعلان تابع قبل از تابع main باشد و بدنه تابع بعد از تابع main باشد
- اعلان تابع و بدنه تابع هر دو قبل از تابع main باشد.
- اعلان تابع و بدنه تابع درون فایل کتابخانه ای باشد.

فراخونی تابع نیز درون تابع main صورت می گیرد. در صورتی که اعلان و فراخوانی تابع درون فایل کتابخانه ای باشد فقط یاز به فراخوانی آن در main است.

نکته : هیچ تابعی را نمی توان درون تابعی دیگرر تعریف نمود و فقط به صورت های گفتهشده صحیح است . اما می توان از فراخوانی توابع در داخل یکدیگر استفاده کرد. بدین صورت که تابعی که داخل تابع دیگر فراخوانی می شود باید در برنامه زودتر تعریف شود.

اعلان و بدنه تابع

(...,نام ورودی دوم نوع ورودی دوم , نام ورودی اول نوع ورودی اول) نام تابع نوع داده

{

بدنه تابع

}

مثال:

```
void sample(int x, int y )
{
.
.
}
```

در صورتی که بخواهیم قبل از تابع main اعلان و بدنه تابع باشد ، به همان صورت فوق این کار را انجام می دهیم با این تفاوت که یکجا ، هم اعلان و هم بدنه قبل از main تعریف می شود .

فراخوانی تابع

صدا زدن تابع درون برنامه را فراخوانی تابع می گویند. در فراخوانی تابع باید نام تابع و مقدار ورودی ها را بیان کنیم که به آن آرگومان های تابع نیز گفته می شود . در مقدار دهی آرگومان تابع در هنگام فرخوانی دیگر نباید نوع آرگومانها را ذکر کنیم.

مثال :

```
Void main()
{
...
        Sample(a,b);
...
}
```

در مورد نوع خروجی تابع دو حالت وجود دارد . اول اینکه اگر تابع بدون خروجی باشد باید آن را برابر مقدار متغیر از همان نوع قرار دهیم تا مقدار بازگشتی را در متغیر مذکور ریخته و در جای مناسب از آن استفاده نماییم.

مثال:

```
#include <mega32.h>
#include <delay.h>
unsigned char i=0;
unsigned char count (void){
i++;
delay_ms(300);
return i;
}
void main (void)
{
DDRA=0xff;
PORTA=0x00;
while(1)
{
        PORTA=count();
}
}
```

توضیح :

در برنامه فوق ابتدا هدر فایل مربوط به میکروکنترلر ATmega32 به برنامه اضافه می شود . با اضافه شدن این فایل برنامه تمام رجیسترهای میکروکنترلر را می شناسد. سپس هدر فایل delay برای اینکه بتوان از تابع delay_ms() استفاده کرد به برنامه اضافه شده است.

سپس یک نوع متغیر ۸ بیتی از نوع unsigned char با مقداراولیه صفر تعریف می شود . یک تابع دارای خروجی و بدون ورودی اعلان و بدنه آن تعریف شده است. در تابع main ابتدا رجیستر DDRA به منظور اینکه تمام ۸ بیت موجود در پورت A خروجی شود به صورت 0xff مقداردهی شده است. مقدار اولیه منطق پایه های خروجی پورت A در خط بعدی همگی برابر ۰ شده است . در اخر در حلقه بینهایت while ، تابع فراخوانی شده است و مقدار خروجی که تابع بر میگرداند در درون پورت A ریخته می شود. در صورتی که روی هر پایه از پورت LED قرار دهیم ، برنامه به صورت شمارنده عمل خواهد کرد. و در هر مرحله تعدادی LED روشن می گردد.

نکته: نوع متغیر رجیسترها در هدر فایل mega32.h همگی به علت ۸ بیتی بودن رجیسترها در میکروکنترلر های AVR ، از نوع unsigned char می باشد.

کلاس های حافظه متغیرها

کلاس حافظه هر متغیر دو چیز اساسی را برای آن متغیر ، تعیین می کند:

- مدت حضور یا طول عمر (life time) آن متغیر
- محدوده قابل دسترسی بودن متغیر در برنامه (Scope)

۴ نوع کلاس حافظه به صورت زیر تعریف شده است:

- اتوماتیک (Automatic)
- خارجی (External)
- استاتیک (Static)
- ثبات (Register)

نحوه تعریف کلاس حافظه یک متغیر

برای تعیین نوع کلاس حافظه برای متغیر ها کافی است نام کلاس مورد نظر را در هنگام تعریف متغیر به ابتدا آن اضافه کنیم :

```
نوع کلاس حافظه  نوع متغیر  نام متغیر = مقدار اولیه;
```

که نوع کلاس با استفاده از کلمات کلیدیauto (برای کلاس حافظه اتوماتیک) ، static (برای کلاس حافظه استاتیک) ، register (برای کلاس حافظه ثبات) و extern (برای کلاس حافظه خارجی) تعیین می گردد. به عنوان مثال در کد زیر دو متغیر a و b (با مقدار دهی اولیه برای b) ، از نوع عدد صحیح تعریف شده اند که کلاس حافظه آنها static می باشد.

```
Static int a,b=10;
```

کلاس حافظه اتوماتیک

این کلاس که پرکاربرد ترین کلاس حافظه است با کلمه کلیدی auto مشخص میشود . اگر نوع کلاس حافظه متغیری را ذکر نکنیم ، کامپایلر خود به خود auto در نظر میگیرد. متغیرهایی که در داخل توابع تعریف می شوند از این نوع هستند که بافراخوانی تابع به طور اتوماتیک ایجاد می شوند و با پایان رفتن تابع به طور اتوماتیک از بین می روند. این نوع متغیر ها دارای خواص زیر هستند .

۱. به صورت محلی هستند (Local). یعنی فقط در داخل بلاکیکه تعریف شده اند ، قابل دسترسی اند.
۲. هنگام ورود یک متغیر به یک تابع یا بلاک ، به آن حافظه اختصاص داده می شود و این حافظه هنگام خروج از تابع یا بلاک , پس گرفته می شود.
۳. چندین بار می توانند مقدار اولیه بگیرند.

کلاس حافظه ثبات

متغیرهای کلاس حافظه ثبات (register) در صورت امکان در یکی از ثبات های CPU (در AVR یکی از ۳۲ رجیستر همه منظوره) قرار می گیرند ؛ لذا سرعت انجام عملیات با آنها به علت نزدیک بودن و دسترسی مستقیم به CPU به آن بسیار بالاست و در نتیجه موجب افزایش سرعت افزایش برنامه می شود. معمولا متغیرهای شمارنده حلقه های تکرار را از این نوع تعریف میکنند. این کلاس دارای ویژگیها و محدودیت های زیر می باشد:

۱. همانطور که در بالا ذکر شد ، متغیر از نوع ثبات در صورت امکان در یکی از ثبات های CPU قرار می گیرد. زیرا به دلیل کم بودن تعداد ثبات های CPU ، تعداد محدودی متغیر می توانند در ثبات ها قرار بگیرند. پس اگر تعداد متغیرهایی که از نوع کلاس حافظه ثبات تعریف شده اند زیاد باشد ، کامپایلر کلاس حافظه ثبات را از متغیر ها حذف می کند.
۲. کلاس حافظه ثبات تنها می تواند برای متغیرهای محلی و همچنینی پارامترهای تابع به کار گرفته شود.
۳. آدرس در مفهوم کلاس حافظه ثبات بی معنی است زیرا متغیرها در ثبات های CPU قرار می گیرند و نه در RAM. پس در مورد آن کلاس حافظه ، نمی توان از عملگر & برای اشاره به آدرس متغیرها استفاده کرد.

کلاس حافظه خارجی

اگر برنامه هایی که می نویسی طولانی باشند ف می توانیم آنها را به قسمتهای کوچکتری تقسیم کنیم که به هر قیمت آنها واحد (Unit) گفته میشود . اگر بخواهیم متغیرهایی که در واحداصلی تعریف شده اند را در واحد های فرعی استفاده کنیم و دیگر آنهارادر واحدهای فرعی تعریف نکنیم ، می توان آن متغیرها را با استفاده از کلاس حافظه خارجی تعریف کنیم. بدین منظور باید این متغیرها در واحد اصلی به صورت عمومی تعریف شده باشند و در واحد فرعی از کلمه extern قبل از تعریف این متغیرها استفاده کنیم.

طول عمر متغیرهایی که از کلاس حافظه extern هستند ، از هنگام شروع برنامه تا پایان آن است و همچنین این متغیرها در سراسر برنامه قابل دسترس هستند . دستور extern به کامپایلر اعلام می کند که برای این متغیر ها , حافظه جدیدی در نظر نگیرد ،بلکه از همان حافظه ای که در جای دیگر برنامه به آن اختصاص یافته استفاده کند.

کلاس حافظه استاتیک

- متغیرهای استاتیک محلی
- متغیرهای استاتیک عمومی

متغیرهای استاتیک محلی:

در توابعی کاربرد دارند که نمی خواهیم مقداری که متغیر موجود در تابع به خود گرفته با خاتمه عملیات تابع پاک شود . بنابراین در توابعی که متغیرها با کلاس حافظه استاتیک معرفی شده باشند ، بعد از خاتمه عملیات تابع ، مقدار نهایی خود را حفظ کرده و با فراخوانی بعدی تابع آن مقدار را به خود میگیرد. متغیرهای تعریف شده در این کلاس دارای خواص زیر می باشد:

۱. فقط در همان تابعی که تعریف شده اند ، قابل دسترسی اند.
۲. می توانند مقدار اولیه بگیرند.
۳. در هنگام خروج از تابع ، مقادیر متغیر ها ، آخرین مقداری خواهدبود که در تابع به آنها اختصاص یافته است و هنگام اجرای دوباره تابع ، مقدار اولیه نمی گیرند.

متغیرهای استاتیک عمومی :

فقط در یک واحد از برنامه ، از جایی که تعریف می شوند ، به بعد قابل دسترس اند. استفاده از متغیرهای استاتیک عمومی از یک طرف موجب میشود تا متغیر ها در جایی که به آنها نیاز است تعریف شوند و از طرف دیگر ، فقط توابعی که به آنها نیاز دارند می توانند از انها استفاده کنند.

ثابتها در زبان C

ثابت یک مقدار مشخص است که در ابتدای برنامه قبل از main تعریف می شود و یک نام به آن تعلق میگیرد. این مقدارهیچ گاه قابل تغییر توسط کدهای برنامه نمی باشد. معمولا مقادیر ثابت عددی که در طول برنامه زیاد تعریف می شود را یک ثابت با نامی مشخص تبدیل می کنند.برای تعریف ثابت می توان از دو روش زیر استفاده کرد.

۱- استفاده از دستور const

```
Const  flaot  p=3.14;
```

۲- استفاده از دستور#define

```
#define        p       3.14
```

تعریف ثابت ها معمولا در ابتدا برنامه با استفاده از دستور #define صورت می گیرد زیرا این دستور پیش پردازنده بوده و به بهبود برنامه کمک می کند. به طورکلی در زبان C دستوراتی که با # آغاز می شوند پیش پردازنده هستند یعنی کامپایلر ابتدا آنها را پردازش و سپس بقیه برنامه را کامپایل می کند.

دستورات شرطی

دستور شرطی if

If (شرط) دستور;

If (شرط) {دستورات;}

If (شرط) {دستورات;} else {دستورات;}

دستور شرطی سوئیچ

Switch(عامل مورد شرط)

{

Case مقدار اول:

کدهایی که در صورت برابر بودن عامل شرط با مقداراول باید اجرا شود

Break;

Case مقداردوم:

کدهایی که در صورت برابر بودن عامل شرط با مقدار دوم باید اجرا شود

Break;

.

.

.

Default:

کدهایی که در صورت برابر نبودن عامل شرط با هیچ یک از مقادیر باید اجرا شود.

}

حلقه های تکرار

حلقه while

در ابتدا شرط حلقه مورد بررسی قرار می گیرد اگر شرطبرقرار باشد یکبار کدهای درون حلقه اجرا می شود و دوباره شرط حلقه چک می شود و این روند تا زمانی که شرط برقرار است ادامه می یابد.

While(شرط حلقه)

{;کدهایی که تا زمان برقراری شرط حلقه تکرار می شود}

برای اینکه شرط در آخر بررسی شود از حلقه do...while استفاده می شود.

do{

;کدهایی که حداقل یکبار انجام می شوند

}while(شرط حلقه)

حلقه for

for(مقدار اولیه شمارنده;شرط حلقه;گام حرکت حلقه)

{;کدهایی که تا زمان برقراری شرط حلقه تکرار می شود}

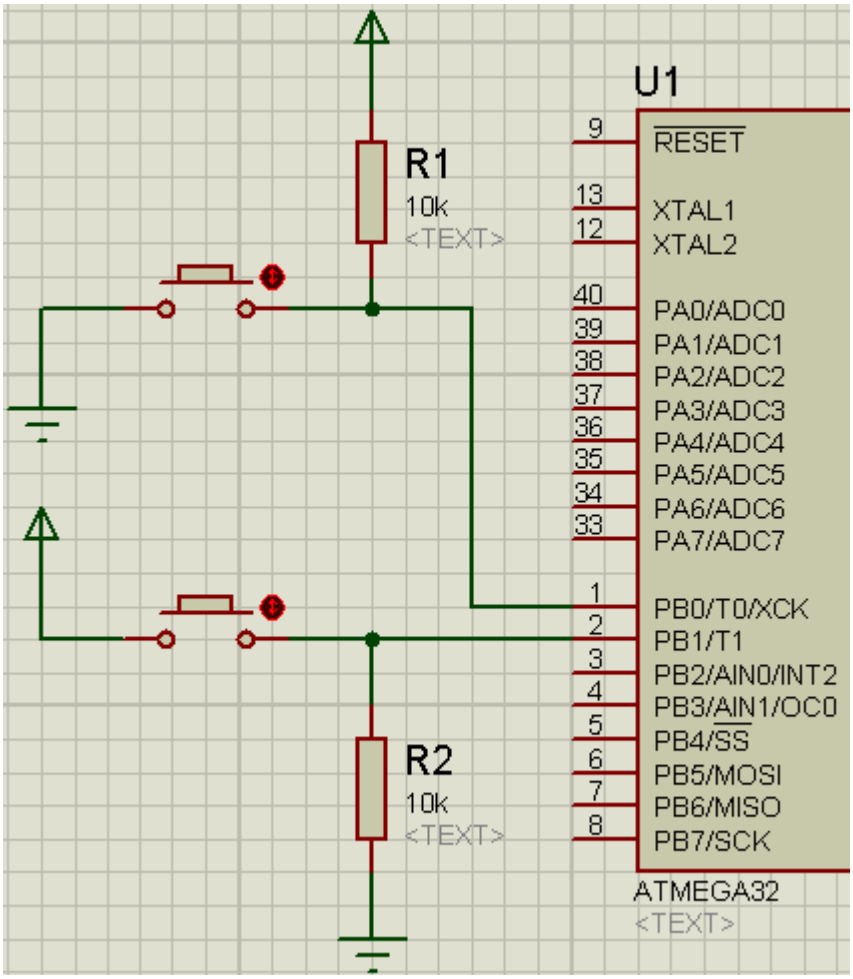
دستور Break و continue

برنامه با دیدن دستور break در هر نقطه از حلقه ، در همان نقطه از حلقه خارج شده و کدهای زیر حلقه را اجرا می کند. دستور Break در حلقه خروج بدون شرط از حلقه است.

برنامه با دیدن دستور continue در هر نقطه از حلقه ، کدهای زیر دستور continue را رها کرده و به ابتدای حلقه باز می گردد.

اتصال کلید به میکرو

یکی از ساده ترین و پرکاربردترین دستگاه های ورودی ، کلید است. توسط کلید میتوان منطق ۰ یا ۱ را به میکرو وارد کرد و براسا آن پردازش را انجام داد. برای خواندن منطق کلید از رجیستر PIN استفاده می شود. نحوه استفاده از رجیستر PIN به این صورت است که : " اگر کلید زده شد انگاه کار مورد نظر انجام شود ". عبارت اخیر معادل استفاده از دستور شرطی if به صورتی است که اگر کلید زده شده بود یک منطق و در غیر اینصورت منطق دیگری وارد پایه میکرو شود. بنابراین اتصال کلید را به دو صورت pullUp و PullDown انجام می شود که در شکل زیر مشاهده می کنید. در کلید pullUp در حالتی که کلید زده نشده منطق ۱ و در حالت فشردن کلید منطق ۰ وارد میکرو می شود. معمولا از مقادمت 10K برای اینکار استفاده شود.



بنابراین برای فهمانده کلید به میکرو به صورتی که اگر کلید زده شد آنگاه کار مورد نظر انجام شود " برای کلیدهای فوق به صورت زیر می شود :

```
If(PINB.0==0{...}//pullup
If(PINB.1==1{...}//pulldown
```

اما مشکلی که در کلیدوجود دارد ، بوجود آمدن لزرش در هنگام قطع و وصل کلید است. در زمانی که کاربر کلید را فشار می دهد تقریبا ۲۰ میلی ثانیه طول می کشد تا منطق کلید از صفر به ۱ یابالعکس ثابت شود .تا

قبل از این به علت وجود جرقه کلید منطق ثابتی ندارد. در زمانی که کاربر دست خود را از روی کلید بر می دارد نیز تقریبا ۲۰ میلی ثانیه طول می کشد تا ۰ و ۱ شدن کلید تمام شده وکلید به منطق اولیه خود باز گردد. در واقع این مشکل زمانی برای ما مسئله ساز می شود که زمانی که کاربر کلید را یکبار فشار میدهد و انتظار دارد تا میکرو متوجه یکبار فشاردادن آن شود اما به علت قرار گرفتن if در حلقه نامتناهی while چندین بار PINB.0==0 برقرار شده و کار مورد نظر چندین بار انجام می شود. راه حل این مشکل ساده است و آن هم قرار دادن مقداری delay در حلقه if است. بنابراین برای حل این مشکل بسته به نوع برنامه یکی از سه روش زیر قابل استفاده است:

کلید نوع ۱:

```
If(PINB.0==0){
دستورات مربوط به بعد از زدن کلید
Delay_ms(200);
}
```

توضیح :

به محض فشار دادن کلید توسط کاربر شرط if برقرار شده و دستورات مورد نظر اجرا می شود سپس به علت ایجاد تاخیر زیاد توسط تابع delay_ms (در اینجا ۲۰۰ میلی ثانیه) با اینکار احتمال اینکه زمانی که برنامه در حلقه while به if می رسد و شرط برقرار باشد , کاهش می یابد. مزیت این کلید این است که در صورتی که کاربر کلید را فشار داده و نگه دارد تقریبا هر ۲۰۰ میلی ثانیه یکبار کار مورد نظر صورت میگیرد.

عیب این روش نیز این است که هنوز احتمال دارد که زمانی که کلید یکبار زده شود ، دو بار کار مورد نظر انجام شود.

کلید نوع ۲:

```
If(PINB.0==0){
Delay_ms(20);
While(PINB.0==0);
دستورات مربوط به بعد از زدن کلید
}
```

توضیح : به محض فشار دادن کلید توسط کاربر شرط if برقرار شده و برنامه به مدت ۲۰ میلی ثانیه صبرمی کند تا منطق کلید ثابت شود. سپس توسط حلقه while با همان شرط برقرای کلید در این مرحله برنامه تا زمانی که کلید توسط کاربر فشرده شده است در حلقه گیر می کند و هیچ کاری انجام نمی دهد. به محض اینکه کاربر دست خودرا بر می دارد ، شرط برقرار نبوده و خط بعدی یعنی دستورات مربوطه اجرا می شود . مزیت این روش

این است که در هر بار فشردن کلید برنامه تنها یکبار اجرا می شود. معایب این روش این است که تا زمانی که کاربر کلید را مگه داشته باشد اتفاقی نمی افتد و به محض رها کردن کلید کار مورد نظر انجام می شود.

کلید نوع ۳:

```
If((PINB.0==0) &&(flag==0))      {
Flag=1; start=!start;}
Else if (PINB.0==1)flag=0;
If(start){
رستورات مربوط به بعد از زدن کلید
}
```

توضیح: این کلید به صورت start/stop عمل می کند یعنی بار اولی کاربر کلید را فشار می دهد دستورات مربوط به بعد از زدن کلید دائما اجرا میشود تا زمانی که کاربر دست خود را ازروی کلید رها کرده و دوباره کلید را فشار دهد ، در این صورت دستورات دیگر اجرا نمی شود. دو متغیر از نوع bit با نام های flag و start با مقدار اولیه ۰ برای این کلید باید تعریف شود. زمانی که کاربر برای اولین بار کلید را فشاردهد شرط if برقرار شده و flag=1 و strat=1 می شود. دراینصورت شرط if دوم برقرار بوده و دستورات مربوطه با هر بار چرخش برنامه درون حلقه نامتناهی while یکبار اجرا می شود. زمانی که کاربر دست خود را از روی کلید بر میدارد و منطق ۱ وارد میکرو می شود flag=0 شده و برنامه دوباره آماده این می شود که کاربر برای بار دوم کلید را فشار دهد . زمانی که کاربر بار دوم کلید را می فشارد start=0 شده و دستورات مربوطه اجرا نخواهد شد سپس با برداشته شدن دست کاربر از روی کلید ، همه چیز به حالت اول بر میگردد. این کلید طوری نوشته شده است .

آرایه ها در زبان C

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 اندیس آرایه را ذکر کنیم. آرایه ها در زبان سی از جایگاه ویژه ای برخوردارند ، به طوری که در پروژهای 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.

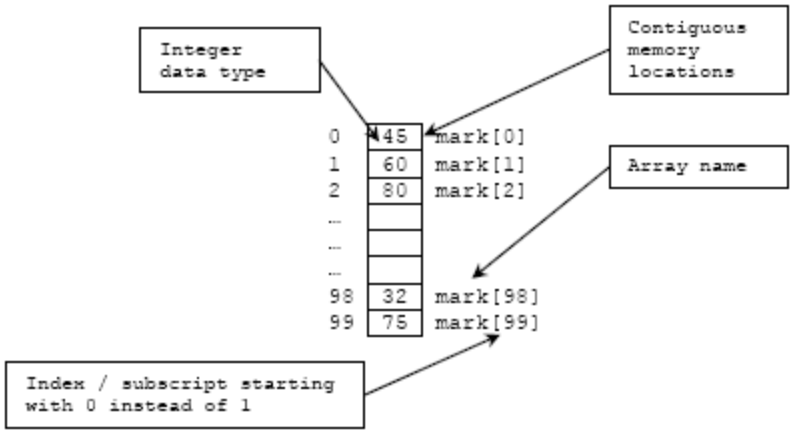
تعریف آرایه یک بعدی

نوع متغیر نام ارایه [تعداد خانه ها]={مقدار اول , مقدار دوم ,...};

```
int   str[10]={50,75,120,190,210,220,234,250,255,0};
```

با تعریف ارایه به همان مقدار خانه های حافظه بسته و به نوع متغیر و تعداد خانه های آرایه تخصیص می یابد که در شکل زیر مثالی از آن را مشاهده می کنید. میزان حافظه ای که به آرایه اختصاص داده می شود ، به این شکل استفاده می شود:

میزان حافظه آرایه (برحسب بایت) = (طول نوع آرایه) * (طول آرایه)



برای دسترسی به هر یک از خانه ها ادرس آنرا لازم داریم. آدرس هر خانه از ۰ تا n-1 است که در آن تعداد خانه های تعریف شده است. هر عضو آرایه به تنهایی می تواند در محاسبات شرکت کند.

مثال :

```
Unsigned chara[5]={7,12,0,99,1};
a[0]=a[4]*a[2];
```

نکته : اندیس آرایه می تواند متغیر باشد و این قابلیت می تواند در برنامه ها کاربرد زیادی داشته باشد.

نکته : در صورتی که می خواهید آرایه به صورت دائمی ذخیره شود باید به ابتدای تعریف ، کلمه کلیدی eeprom یا flash را اضافه کنید تا در حافظه های دائمی ذخیره گردد.

آرایه چند بعدی

در تعریف آرایه دو بعدی باید ۲ اندیس و در تعریف آرایه سه بعدی باید ۳ اندیس و در آرایه n بعدی باید n اندیس ذکر کرد. آرایه های چند بعدی در نمایشگرهای lcd کاربرد دارند. به عنوان مثال:

```
Int table[10][10];
```

یک آرایه دو بعدی به نام table را تعریف می کند که دارای ۱۰ سطر و ۱۰ ستون است و نوع عناصر آن int است.

```
int k[5][10][15];
```

آرایه سه بعدی به نام K را تعریف می کند که دارای ۵ سطر ، ۱۰ ستون و ۱۵ ارتفاع است و نوع عناصر آن int میباشد.

برای مقداردهی به عناصر سه بعدی ابتدا سطرهای طبقه اول و سپس سطرهای بقیه طبقات را مقداردهی می کنیم.

مثال :

```
int a[2][3][4]={{{1,2,3,4},{2,3,4,5},{2,3,4,5}},{{1,1,1,1},{2,2,2,2},{3,3,3,3}}}
```

رشته ها

در زبان c برای نمایش کلمات و جملات ازرشته ها استفاده می شود. رشته همان آرایه ای از کاراکترها است که حاوی اطلاعاتی می باشد. تمام کاراکترها شامل اعداد وحروف و برخی کاراکترهای دیگر که روی صفحه کلید کامپیوتر وجود دارند ، دارای کد شناسایی ASCII میباشند.

Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char
32	[space]	48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	[backspace]

تعریف یک کاراکتر

تعریف یک کاراکتر توسط نوع متغیر char صورت میگیرد و مقدار اولیه آن داخل کوتیشین (‘ ’) قرار می گیرد. در زبان c وقتی یک حرف بین ‘ و ’ قرار میگیرد کد اسکی آن درون متغیر ذخیره می شود.

مثال :

```
char c='H';
```

عدد ۷۲ در متغیر c قرار میگیرد.

تعریف رشته (آرایه ای از کاراکترها)

برای تعریف رشته از آرایه ای از کاراکترها استفاده می شود و مقدار اولیه آنداخل دابل کوتیشن (“ ”) قرار می گیرد. مثال:

```
Char s[10]="Hello";
```

اگر تعداد خانه های آرایه ذکر شود ، آرایه سایز مشخصی دارد و در صورتی که تعداد عبارت یا جمله ای که درون آن میریزیم بیشتر از سایز آرایه باشد ، عبارت ناقص ذخیره خواهد شد و در صورتی که تعداد کاراکترهای مورد نظر کمتر باشد بقیه آرایه خالی خواهد ماند. اگر تعداد خانه های آرایه ذکر نشود یعنی سایز آرایه براساس مقداری که درون آن ریخته می شود محاسبه شود. مثال :

```
Char str[]="Hello!...";
```

عملگرها

با استفاده از عملگرها می توان روی اعداد ، متغیرها ، آرایه ها ، رشته ها و ... عملیات حسابی ، منطقی ، مقایسه ، بیتی ، بایتی و .... انجام داد.

عملگرهای محاسباتی

عملگر	نام	مثال
-	تفریق و علامت منفی	z = x - y یا - x
+	جمع	z = x + y
*	ضرب	z = x * y
/	تقسیم	z = x / y
%	باقیمانده تقسیم	z = x % y
--	یک واحد کاهش	x -- یا -- x
++	یک واحد افزایش	x ++ یا ++ x

تقدم عملگرهای محاسباتی

تقدم	عملگر
۱	++ و --
۲	- علامت منفی
۳	/ و * و %
۴	+ و -

عملگر	نام	مثال
+=	انتساب جمع	x += y
-=	انتساب تفریق	x -= y
*=	انتساب ضرب	x *= y
/=	انتساب تقسیم	x /= y
%=	انتساب باقیمانده تقسیم	x %= y

عملگرهای رابطه ای و منطقی

عملگر	نام	مثال
>	بزرگتر	x > y
>=	بزرگتر مساوی	x >= y
<	کوچکتر	x < y
<=	کوچکتر مساوی	x <= y
==	متساوی	x == y
!=	نا مساوی	x != y
!	نقیض	!(x > y)
&&	و	x > y && z > w
||	یا	x > y || z > w

تعریف عملگرهای بیتی

x	y	x & y	x | y	x ^ y	~x
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

جدول صحت عملگر های بیتی

عملگرهای بیتی

عملگر	نام	مثال
&	And	x = 11101101 , y = 00001111 → x & y = 00001101
|	Or	x = 11101101 , y = 00001111 → x | y = 11101111
^	Xor	x = 11101101 , y = 00001111 → x ^ y = 11100010
~	Not	x = 00001111 → x = ~ x → x = 11110000
<<	شیفت به چپ	x = 00000111 → x = x << 2 → x = 00011100
>>	شیفت به راست	x = 11000000 → x = x >> 3 → x = 00011000

تقدم عملگرهای بیتی

تقدم	عملگر
۱	~
۲	<< و >>
۳	&
۴	^
۵	|

تقدم کلی در عملگرها

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

همانطور که مشاهده می شود عملگر پرانتز دارای بیشترین اولویت است. با استفاده از این عملگر میتوان انجام محاسبلت را اولویت داد به طوری که اولویت اول همیشه با داخلی ترین پرانتز است و سپس به ترتیب تا پرانتز بیرونی اولویت دارند.

تبدیل نوع در محاسبات

برای درک بهتر در مورد تبدیل نوع محاسبات ، به مثال زیر توجه کنید.

```
Int a=9,b=2;

Float x,y,z;

X=a/b;

Y=(float)a/b;

Z=9.0/2.0;
```

در مثال بالا با اینکه متغیرهای x و y و z هر سه از نوع float هستند ولی مقدار x برابر ۴ و مقدار z و y برابر با ۴.۵ است. اگر متغیرهای a و b از نوع float بودند ، مقدار x برابر ۴.۵ میشد اما چون متغیر های a و b از نوع int هستند باید یک تبدیل نوع در محاسبات توسط عبارت (float) در ابتدای محاسبه انجام شود.