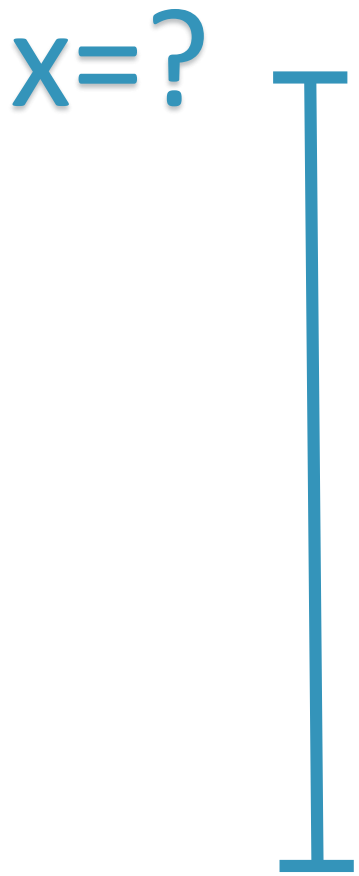
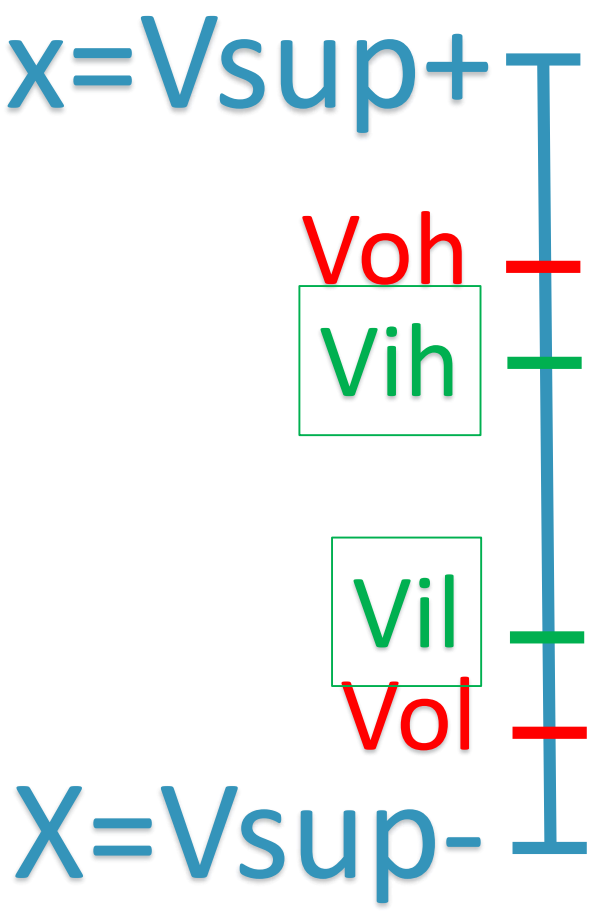
پیش‌نیاز

- آشنایی با مفهوم 0 و 1 دیجیتال
- نمایش اعداد به صورت باینری
- در سیستم باینری هر بیت با یک عدد مدل می‌شود
- مدل سازی فیزیکی با چه پارامتری!؟



پیش‌نیاز



- نمایش 0 و 1 با چه سطح ولتاژی؟!
- منطق TTL
- منطق CMOS
- تفاوت‌ها و مزایا
- دینامیک رنج ورودی و خروجی؟!
- در اکثر چیپ‌ها تغذیه منفی زمین یا 0 ولت می‌باشد اما در برخی از تقویت کننده‌های تفاضلی، تغذیه منفی، قرینه تغذیه مثبت است.

پیش‌نیاز

- نمایش‌های متداول اعداد
  - دودویی (باینری)، مبنای دو

0 1

  - اکتال، مبنای هشت

0 1 2 3 4 5 6 7

  - دسیمال، مبنای ده (سیستم اعداد رایج)

0 1 2 3 4 5 6 7 8 9

  - هگزادسیمال، مبنای شانزده (محبوب‌ترین سیستم نمایش)

0 1 2 3 4 5 6 7 8 9 A B C D E F

پیش‌نیاز

- تبدیل مبناها به یکدیگر
- تبدیل باینری (دودویی) به دسیمال (دهدهی)

5 4 3 2 1 0

1 0 1 1 0 0

$1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 44$

- تبدیل دسیمال (دهدهی) به باینری (دودویی)

44→12→4→0

$2^5 = 64$ $2^5 = 32$ $2^4 = 16$ $2^3 = 8$ $2^2 = 4$ $2^1 = 2$ $2^0 = 1$

64 32 16 8 4 2 1

0 1 0 1 1 0 0

پیش‌نیاز

- تبدیل مبناها به یکدیگر
- تبدیل باینری (دودویی) به اکتال
- روش اول : تبدیل غیرمستقیم (تبدیل باینری به دسیمال و تبدیل دسیمال به اکتال)

5 4 3 2 1 0

1 0 1 1 0 0

$1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 44$

44→4→0

$8^2 = 64 \quad 8^1 = 8 \quad 8^0 = 1$

64 8 1

0 5 4

- روش مستقیم

1 0 1 1 0 0

5 4

پیش‌نیاز

- تبدیل مبناها به یکدیگر
  - تبدیل اکتال به باینری
    - روش اول : تبدیل غیرمستقیم (تبدیل باینری به دسیمال و تبدیل دسیمال به اکتال)

0	1	2	3	4	5	6	7
000	001	010	011	100	101	110	111

5	4
101	100

    - روش مستقیم : هر عدد اکتال نمایشگر ۳ عدد باینری می‌باشد.

پیش‌نیاز

- تبدیل مبناها به یکدیگر
- تبدیل باینری (دودویی) به هگزادسیمال
- روش اول : تبدیل غیرمستقیم (تبدیل باینری به دسیمال و تبدیل دسیمال به اکتال)

5 4 3 2 1 0

1 0 1 1 0 0

$1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 44$

44→12→0

$16^2 = 256 \quad 16^1 = 16 \quad 16^0 = 1$

256 16 1

0 2 C

- روش مستقیم

1 0 1 1 0 0

2 C

پیش‌نیاز

- تبدیل مبناها به یکدیگر
- تبدیل هگزادسیمال به باینری
  - روش اول : تبدیل غیرمستقیم (تبدیل هگزادسیمال به دسیمال و تبدیل دسیمال به باینری)
  - روش مستقیم : هر عدد هگزادسیمال نمایشگر ۴ عدد باینری می‌باشد.

0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

2 C

0010 1100

پیش‌نیاز

- عملیات ریاضی در مبناهای مختلف
  - جمع دو عدد باینری

$(11011)_2 + (00001)_2 = (11100)_2$

$$\begin{array}{r} {\scriptstyle 1\ 1} \\ 11011 \\ +\ 00001 \\ \hline 11100 \end{array}$$

  - جمع دو عدد اکتال

$(457)_8 + (203)_8 = (662)_8$

$$\begin{array}{r} {\scriptstyle 1} \\ 457 \\ +203 \\ \hline 662 \end{array}$$

  - جمع دو عدد هگزادسیمال

$(7EC6)_{16} + (340A)_{16} = (B2D0)_{16}$

$$\begin{array}{r} {\scriptstyle 1\ 1} \\ 7EC6 \\ +340A \\ \hline B2D0 \end{array}$$

$6+A=6+10=16 \Longrightarrow (10)_h$

$C+0+1=1201=13 \Longrightarrow (D)_h$

$E+4=14+4=18 \Longrightarrow (12)_h$

پیش‌نیاز

- عملیات ریاضی در مبناهای مختلف
  - تفریق دو عدد باینری

$(1101)_2 - (0110)_2 = (0111)_2$

$$\begin{array}{r} {\scriptstyle 2}\\ {\scriptstyle 0\ 0\ 2}\\ 1101\\ -\ 0110\\ \hline 0111 \end{array}$$

  - تفریق دو عدد اکتال

$(720)_8 - (311)_8 = (407)_8$

$$\begin{array}{r} {\scriptstyle 1\ 8}\\ 720\\ -\ 311\\ \hline 407 \end{array}$$

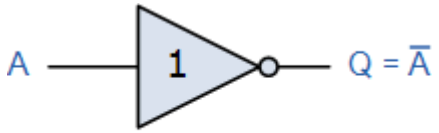
  - تفریق دو عدد هگزادسیمال

$(A20)_{16} - (331)_{16} = (6EF)_{16}$

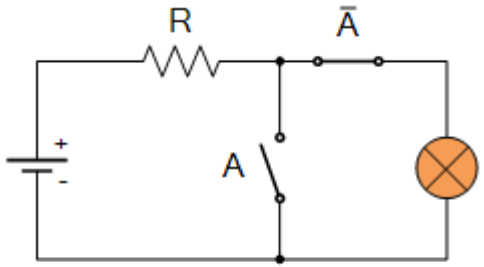
$$\begin{array}{r} {\scriptstyle 17}\\ {\scriptstyle 9\ 1\ 16}\\ A20\\ -\ 3\ 3\ 1\\ \hline 6\,E\,F \end{array}$$

پیش‌نیاز

- گیت‌های منطقی
  - گیت NOT



اینورتر یا گیت NOT

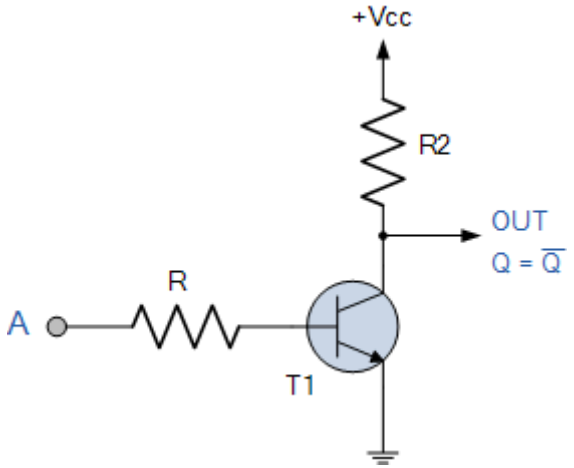


"1" = لامپ روشن، "0" = کلید باز
"0" = لامپ خاموش، "1" = کلید بسته

- 74LS04 (TTL)
- CD4049 (CMOS)

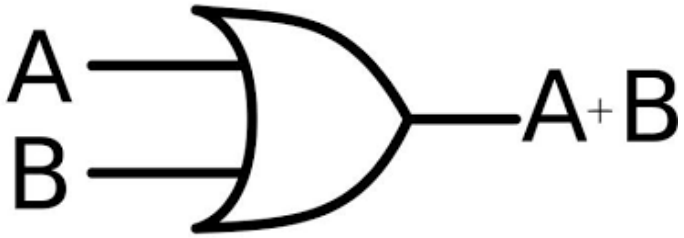
A	Y = ~A
0	1
1	0

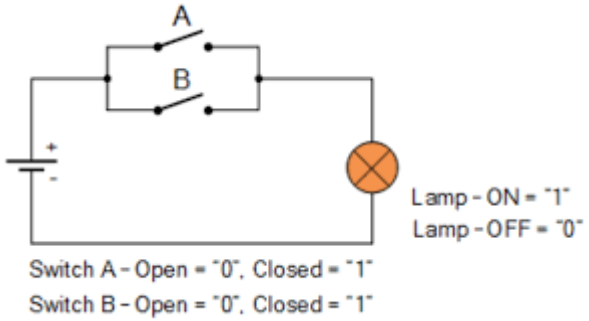
- کاربرد!؟



پیش‌نیاز

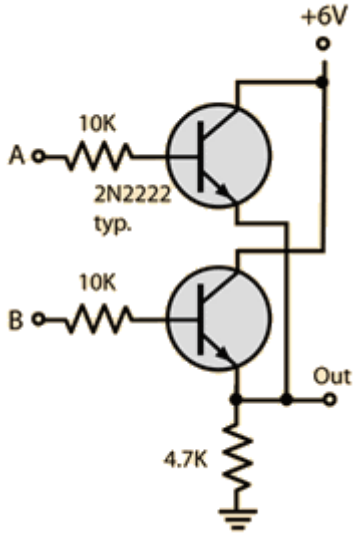
- گیت‌های منطقی
- گیت OR





A	B	Y = A+B
0	0	0
0	1	1
1	0	1
1	1	1

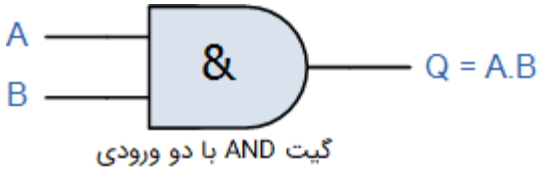
- 74LS32 (TTL)
- CD4071 (CMOS)



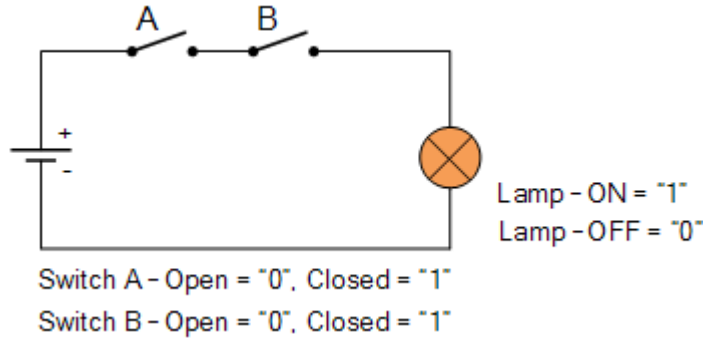
- کاربرد!؟

پیش‌نیاز

- گیت‌های منطقی
- گیت AND



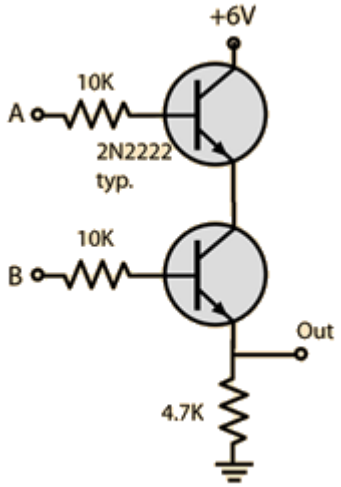
گیت AND با دو ورودی



- 74LS08 (TTL)
- CD4081 (CMOS)

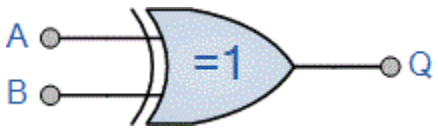
A	B	Y = A.B
0	0	0
0	1	0
1	0	0
1	1	1

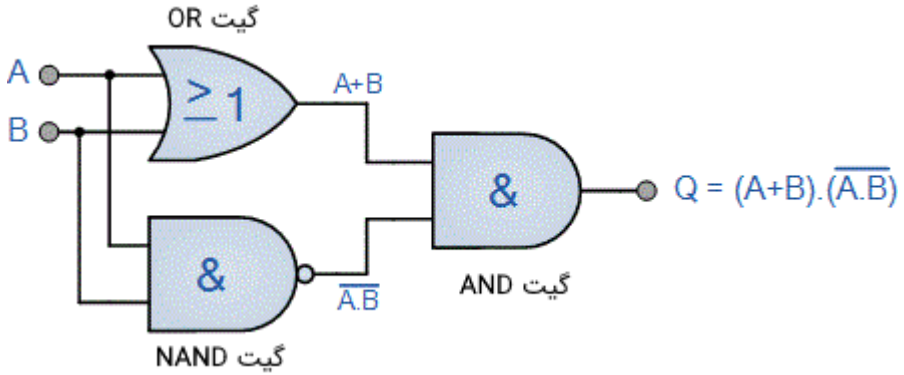
- کاربرد!؟



پیش‌نیاز

- گیت‌های منطقی
- گیت XOR





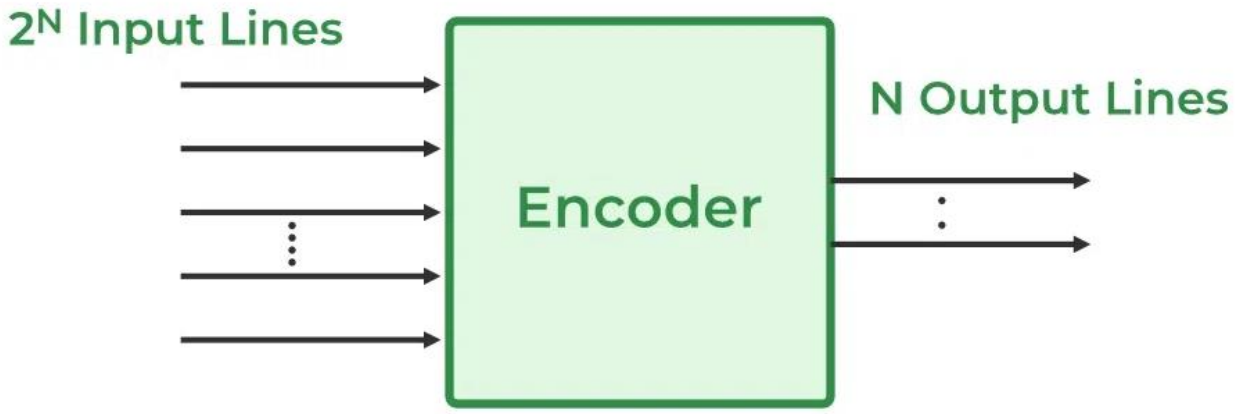
A	B	Y = A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

- 74LS86 (TTL)
- CD4030 (CMOS)

- کاربرد!؟

پیش‌نیاز

- انکودر (Encoder)
  - تبدیل 2^N ورودی به N خروجی
  - تعدادی ورودی را به یک کد منحصر به فرد نگاشت میکند
  - کاهش تعداد پورت‌های اشغالی



پیش‌نیاز

- انکودر (Encoder)
  - کاربرد!؟
  - تبدیل دسیمال به باینری
  - تشخیص وقفه در میکروکنترولرها

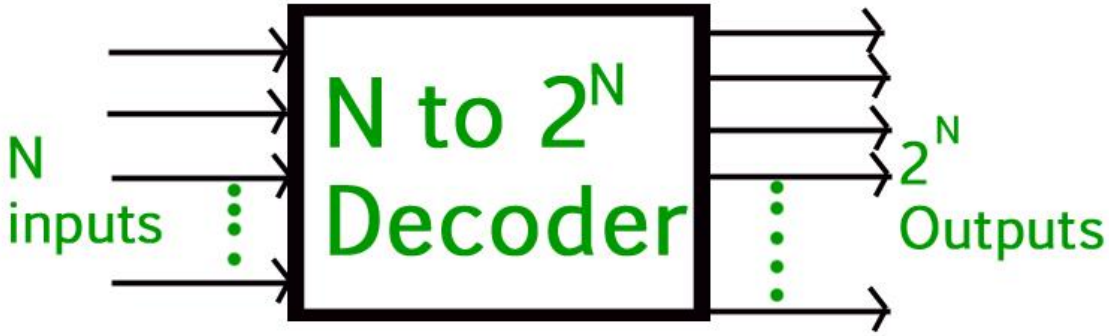
- مزایا :
  - کاهش پورتها
  - افزایش قابلیت اطمینان (کدگذاری)
  - افزایش راندمان (کاهش حجم دیتا)
- معایب :
  - افزایش پیچیدگی مدار
  - کاربرد در مدارهای ورودی موازی

INPUTS				OUTPUTS	
Y3	Y2	Y1	Y0	A1	A0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

INPUTS				OUTPUTS		
Y3	Y2	Y1	Y0	A1	A0	V
0	0	0	0	X	X	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1

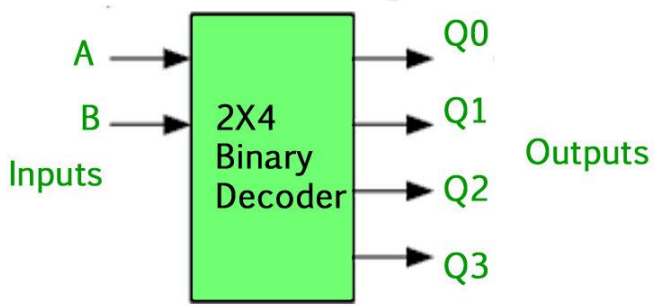
پیش‌نیاز

- دیکودر (Decoder)
  - تبدیل N ورودی به 2^N خروجی
  - با هر کد، خروجی مطلوب فعال می‌گردد.
  - اکثر دیکودرها فعال پایین می‌باشند.



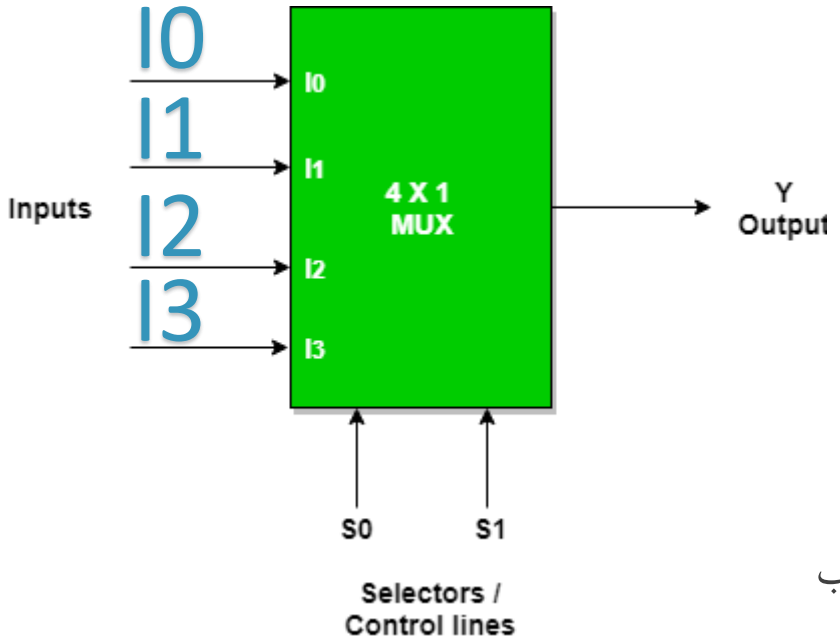
پیش‌نیاز

- دیکودر (Decoder)
  - کاربرد :
    - تعیین محدوده‌های حافظه و Chip Select
    - درایو ماژول‌ها، نمایشگرها و المان‌های مالتی پلکس
  - مزایا
    - بهبود راندمان با کاهش زمان انتخاب خروجی دلخواه به کمک کد
    - افزایش قابلیت اطمینان
  - معایب
    - افزایش پیچیدگی
    - کاربرد در مدارهای خروجی موازی



A	B	Q0	Q1	Q2	Q3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

پیش‌نیاز



Truth Table

S0	S1	Y
0	0	I0
0	1	I1
1	0	I2
1	1	I3

- مالتی پلکسر (Multiplexer)
  - 2^N ورودی و یک خروجی
  - N پایه انتخاب
  - کاربرد :
    - کلید چند وضعیت
    - ساخت گیت‌های منطقی

- مزایا
  - کاهش فضا و هزینه مصرفی در مدارات کامپیوتری با قابلیت انتخاب مسیر مناسب
- معایب
  - افزایش توان مصرفی
  - تاخیر در سیگنال

پیش‌نیاز

- مفهوم رجیستر
- مفهوم ترانزیستور

پیش‌نیاز

- آشنایی با زبان C
  - هدر یا کتابخانه‌ها
  - بدنه اصلی برنامه
  - تعریف متغییرها
  - ورودی و خروجی با توابع printf و scanf
  - عملیات محاسباتی
  - عملیات منطقی
  - عملیات شرطی
  - مفهوم حلقه
  - اشاره گر
  - توابع

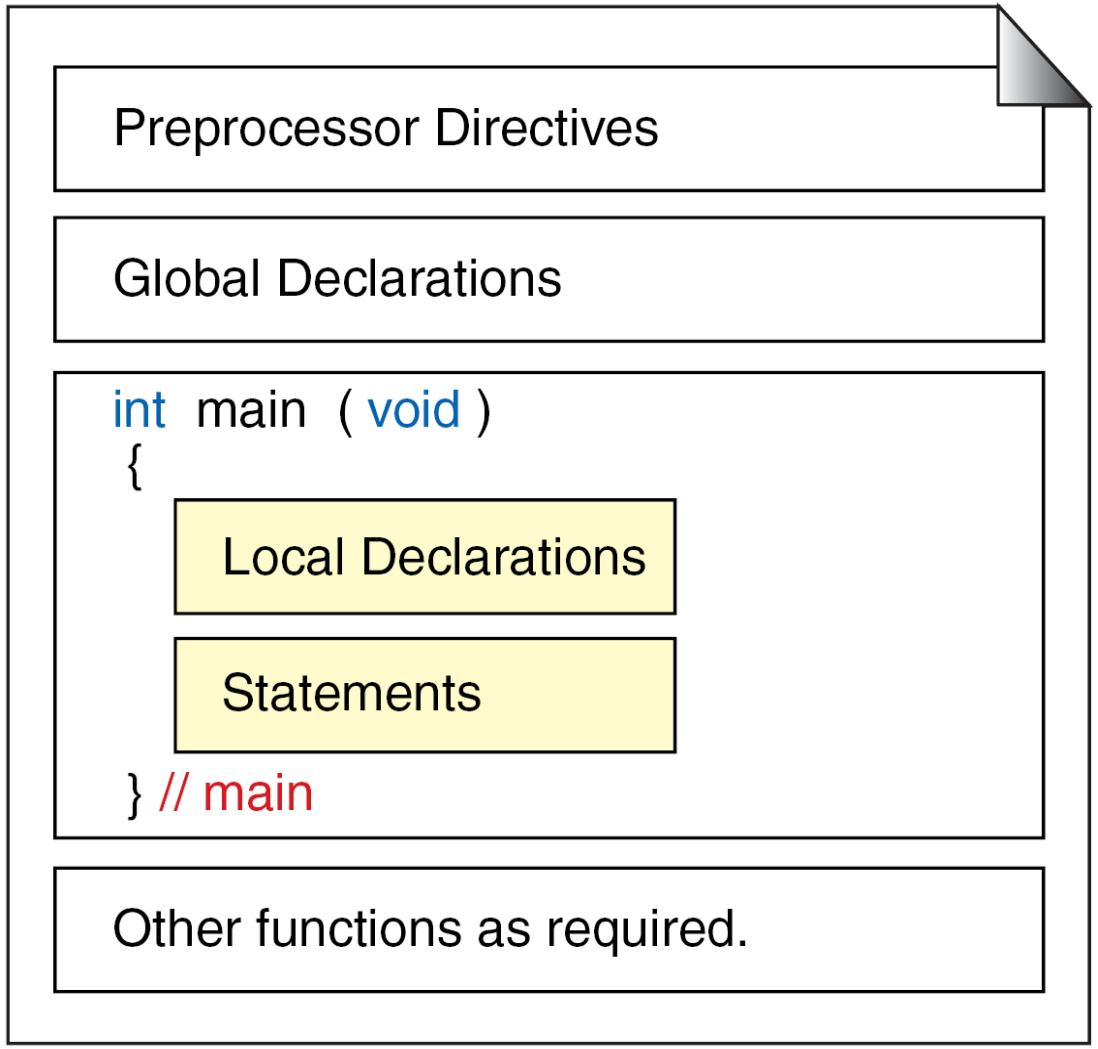
آشنایی با زبان C

- زبان‌های برنامه نویسی
  - سطح بالا : نزدیک به زبان محاوره مثل : Basic, Pascal, Python
  - سطح میانی : هم ویژگی زبان‌های سطح بالا را دارد و هم سطح پایین مثل C
  - سطح پایین : به زبان ماشین نزدیک است مثل Assembly

- هدف : آشنایی با زبان C و مقدمه ای بر زبان Assembly

آشنایی با زبان C

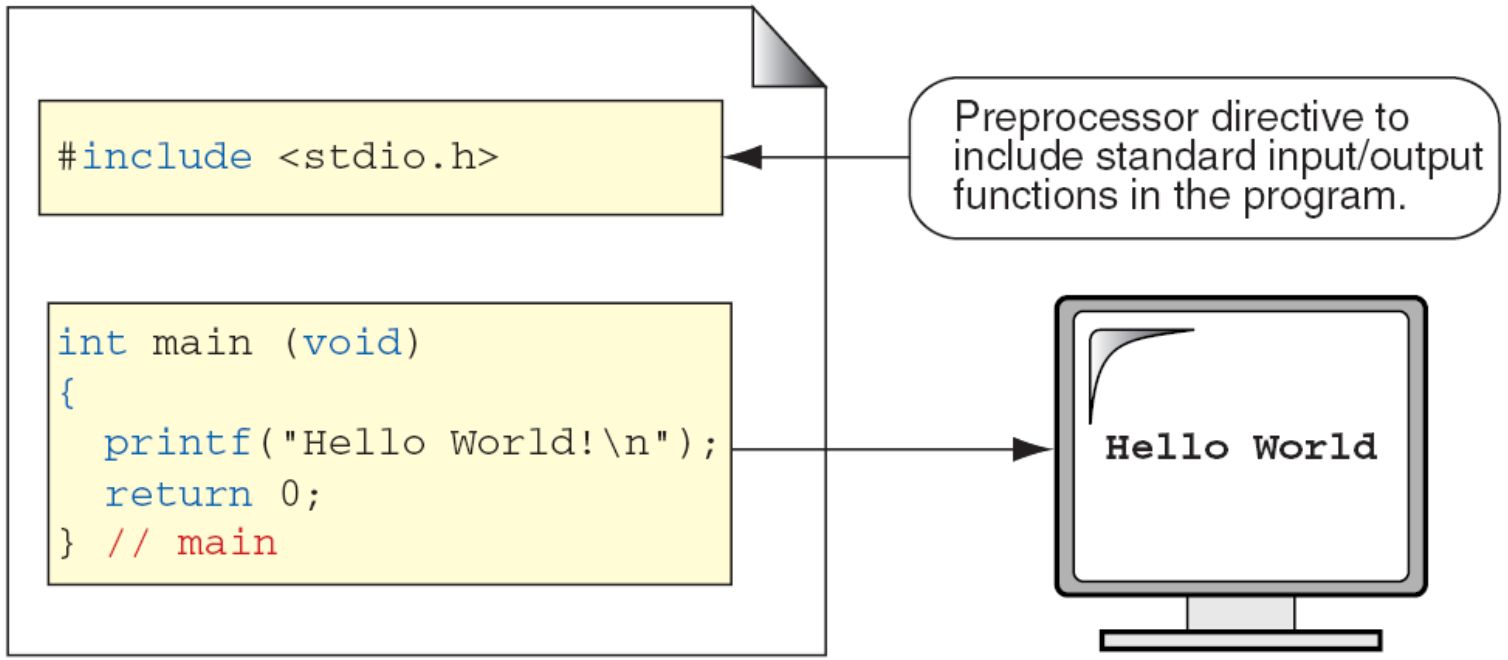
- بدنه زبان C



آشنایی با زبان C

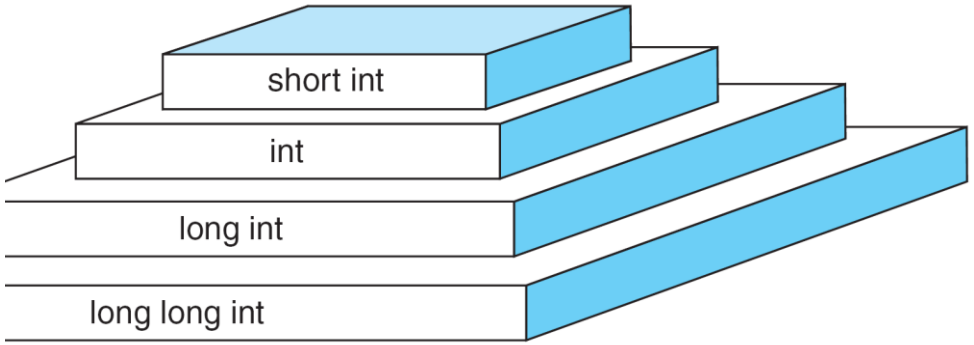
- فایل‌های سرآمد

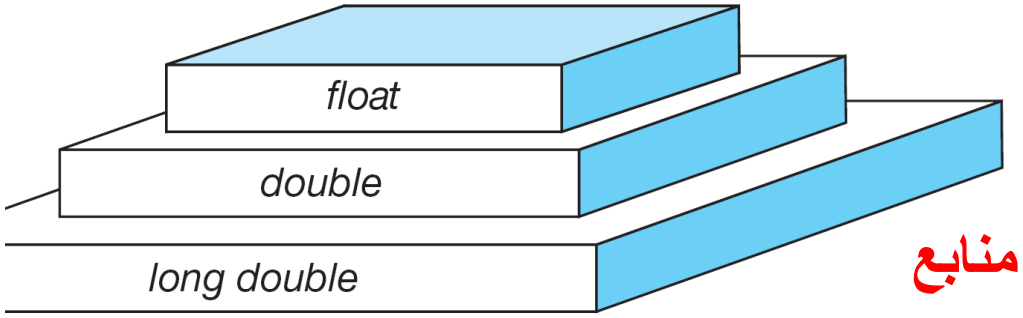
<owl.h>	<window.h>
<dos.h>	<stdlib.h>
<math.h>	<msystem.h>
<stdio.h>	<bios.h>
<aclock.h>	<bitmap.h>
<string.h>	<dloc.h>



آشنایی با زبان C

- انواع متغییر
  - Char
  - Short int
  - Int
  - Long int
  - Long long int
  - Float
  - Double
  - Long double





صرفه جویی در منابع

نوع متغییر		مقدار حافظه	دامنه
عدد صحیح	int	byte 2	-32000 – +32000
عدد اعشاری	float	byte 4	-72000 – +72000
کاراکتر	char	byte 1	- 127 – 127
	double float	64	$^{38-}10^{-}10^{38}$
	unsigned char	8	0 – 255
	singed char	8	- 127 – 127
	int	16 تا 32	-32767 – 32767
	unsinged int	16 تا 32	0 – 65535
	singed int	16 تا 32	-32767 – 32767
	short int	16	-32767 – 32767
	unsigned short int	16	0 – 65535
	long int	32	2447483647 تا منفی
	long double	80	10^{4932} – $^{4932-}10$

آشنایی با زبان C

- ثابت
  - متغییر ثابت
  - Define

نوع ثابت نام ثابت = مقدار ثابت ;

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

- تعریف متغییر و مقداردهی
  - زبان C به حروف کوچک و بزرگ حساس است.
  - نام متغییر را به نحوی انتخاب کنید که گویای محتوای آن باشد.
  - نام‌هایی که شامل تعریف‌های اولیه زبان C باشند قابل قبول نیست! مانند int، float و ...

```
Int   A;
A = 10;

Int   A=10;

Float  B, c, g;
```

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

مقداردهی اولیه فراموش نشود!

آشنایی با زبان C

- توابع یا دستورات ورودی و خروجی
  - Printf
  - Scanf

```
scanf (" فرمت خواندن " , متغییر های ورودی) ;

scanf  (" %i%i" , &a&b) ;
scanf  ("%d %f " , &a &b) ;

printf (" فرمت نوشتن یا عبارت ", یا متغییر ها  یا عبارت ریاضی) ;

printf (" %d " , a ) ;
printf (" the sum is : %d " , a + b  );
```

آشنایی با زبان C

- توابع یا دستورات ورودی و خروجی
  - مثال : برنامه‌ای بنویسید که شعاع دایره‌ای را از ورودی خوانده، سپس محیط و مساحت آن را محاسبه و در خروجی چاپ کند

```
# include < stdio.h >
main ()
{
   const   float    pi = 3.14 ;
   float    r,p,s ;
   printf (" please enter reduce :  \n ") ;
   scanf ( " %f " , & r ) ;
   p = 2*pi * r ;
   s = pi * r *r ;
   printf (" The S = %f , The P = % f " , s,p) ;
 }
```

- خروجی

```
please enter reduce :      2
The S = 12.56 , The P = 12.56
```

آشنایی با زبان C

- کاراکترهای کنترلی در دستور Printf

\f : موجب انتقال کنترل به صفحه جدید می شود
\n : موجب انتقال کنترل به خط جدید می شود
\t : انتقال به ۸ محل بعدی صفحه نمایش
\“ : چاپ دابل کوتیشن (“)
\‘ : چاپ کوتیشن (‘)
\V : انتقال کنترل به ۸ سطر بعدی
\N : ثابت های مبنای ۸ (Nعدد مبنای ۸ است)
\xN : ثابت های مبنای ۱۶ (N عدد مبنای ۱۶ است)

\r : موجب انتقال به ابتدای سطر می‌شود.

آشنایی با زبان C

- کاراکترهای فرمت در دستور Printf

%C : یک کاراکتر
%d : اعداد صحیح دهدهی مثبت و منفی
%i : اعداد صحیح دهدهی مثبت و منفی
%e : نمایش علمی عدد همراه با حرف e
%E : نمایش علمی عدد همراه با حرف E
%f : عدد اعشاری ممیز شناور
%g : اعداد اعشاری ممیز شناور
%G : اعداد اعشاری ممیز شناور
%O : اعداد مبنای ۸ مثبت
%S : رشته ای از کاراکترها (عبارت رشته ای)
%U : اعداد صحیح بدون علامت (مثبت)
%x : اعداد مبنای ۱۶ مثبت با حروف کوچک
%X : اعداد مبنای ۱۶ مثبت با حروف بزرگ
%p : اشاره گر
%n : موجب میشود تا تعداد کاراکترهایی که تا قبل ازاین کاراکتر به خروجی منتقل شده اند شمارش شده و در پارامتر متناظر با آن قرار گیرد .
%% : علامت %

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Assignment Operators

Assignment operators are used to assign values to variables.

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

آشنایی با زبان C

Arithmetic Operators

Arithmetic operators are used to perform common mathematical operations.

Operator	Name	Description	Example
+	Addition	Adds together two values	x + y
-	Subtraction	Subtracts one value from another	x - y
*	Multiplication	Multiplies two values	x * y
/	Division	Divides one value by another	x / y
%	Modulus	Returns the division remainder	x % y
++	Increment	Increases the value of a variable by 1	++x
--	Decrement	Decreases the value of a variable by 1	--x

Comparison Operators

Comparison operators are used to compare two values (or variables). This is important in programming, because it helps us to find answers and make decisions.

Operator	Name	Example
==	Equal to	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Comparison Operators

Comparison operators are used to compare two values (or variables). This is important in programming, because it helps us to find answers and make decisions.

Operator	Name	Example
==	Equal to	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

آشنایی با زبان C

- عملگرهای منطقی

Logical Operators

You can also test for true or false values with logical operators.

Logical operators are used to determine the logic between variables or values:

Operator	Name	Description	Example
&&	Logical and	Returns true if both statements are true	x < 5 && x < 10
||	Logical or	Returns true if one of the statements is true	x < 5 || x < 4
!	Logical not	Reverse the result, returns false if the result is true	!(x < 5 && x < 10)

آشنایی با زبان C

- دستورات شرطی
- If … else
- If statement
- If else statement
- If else if ladder
- Nested if

```
if (condition) {
  // block of code to be executed if the
condition is true
}
```

```
if (condition) {
  // block of code to be executed if the
condition is true
} else {
  // block of code to be executed if the
condition is false
}
```

آشنایی با زبان C

- دستورات شرطی
  - If … else
    - If statement
    - If else statement
    - If else if ladder
    - Nested if

```
if (condition1) {
  // block of code to be executed if condition1
is true
} else if (condition2) {
  // block of code to be executed if the
condition1 is false and condition2 is true
} else {
  // block of code to be executed if the
condition1 is false and condition2 is false
}
```

آشنایی با زبان C

- دستورات شرطی
  - If … else
    - If statement
    - If else statement
    - If else if ladder
    - Nested if

```
if (condition) {
  if (condition) {

  }
}
else {
  if (condition) {
  }
}
```

آشنایی با زبان C

- دستورات شرطی
- Switch...case
- مفهوم !?break
- مفهوم !?default

```
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

آشنایی با زبان C

- دستورات شرطی
- Switch…case
- مثال برنامه ای که با دریافت شماره روز، نام آن را بیان میکند

```c
int day = 4;

switch (day) {
  case 1:
    printf("Monday");
    break;
  case 2:
    printf("Tuesday");
    break;
  case 3:
    printf("Wednesday");
    break;
  case 4:
    printf("Thursday");
    break;
  case 5:
    printf("Friday");
    break;
  case 6:
    printf("Saturday");
    break;
  case 7:
    printf("Sunday");
    break;
}

// Outputs "Thursday" (day 4)
```

آشنایی با زبان C

- حلقه‌ها
  - For
  - While
  - Do while

آشنایی با زبان C

- حلقه While

```
while (condition) {
  // code block to be executed
}
```

```
int i = 0;

while (i < 5) {
  printf("%d\n", i);
  i++;
}
```

آشنایی با زبان C

- حلقه do while

```
do {
  // code block to be executed
}
while (condition);
```

- فراموش نکنید پارامتر شرط را برآورده سازید، در غیر اینصورت حلقه پایانی ندارد.
- تفاوت while و do while!؟

```
int i = 0;

do {
  printf("%d\n", i);
  i++;
}
while (i < 5);
```

آشنایی با زبان C

- حلقه for
- تعریف پارامترهای حلقه for
- خروجی مثال زیر!؟

```
for (statement 1; statement 2; statement 3) {
  // code block to be executed
}
```

```
int i;

for (i = 0; i < 5; i++) {
  printf("%d\n", i);
}
```

آشنایی با زبان C

- حلقه for

- برنامه ای بنویسید که اعداد مضرب ۷ بین ۰ تا ۱۰۰ را پیدا کرده و چاپ نماید.

آشنایی با زبان C

Nested Loops

- تعریف پارامترهای حلقه for
- خروجی مثال زیر!؟

```
for (statement 1; statement 2; statement 3) {
  // code block to be executed
  for (statement 1; statement 2; statement 3) {
  // code block to be executed
  }
}
```

```
int i, j;

// Outer loop
for (i = 1; i <= 2; ++i) {
  printf("Outer: %d\n", i);  // Executes 2 times

  // Inner loop
  for (j = 1; j <= 3; ++j) {
    printf(" Inner: %d\n", j);  // Executes 6 times (2 * 3)
  }
}
```

آشنایی با زبان C

- Nested Loops
- برنامه ای بنویسید که شکل مقابل را در خروجی نمایش دهد

```
*
**
***
****
*****
```

آشنایی با زبان C

- دستور Break

```
int i;

for (i = 0; i < 10; i++) {
  if (i == 4) {
    break;
  }
  printf("%d\n", i);
}
```

- خروجی!؟

آشنایی با زبان C

- دستور continue

```c
int i;

for (i = 0; i < 10; i++) {
  if (i == 4) {
    continue;
  }
  printf("%d\n", i);
}
```

- خروجی!؟

آشنایی با زبان C

- آرایه‌ها
- آرایه یک بعدی

Arrays

Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.

To create an array, define the data type (like `int`) and specify the name of the array followed by **square brackets []**.

To insert values to it, use a comma-separated list, inside curly braces:

```
int myNumbers[] = {25, 50, 75, 100};
```

```
int myNumbers[] = {25, 50, 75, 100};
printf("%d", myNumbers[0]);
```

آشنایی با زبان C

- آرایه‌ها
- آرایه یک بعدی

- تفاوت آرایه و متغییر!؟
- حافظه یکپارچه ...

Loop Through an Array

You can loop through the array elements with the `for` loop.

The following example outputs all elements in the `myNumbers` array:

```
int myNumbers[] = {25, 50, 75, 100};
int i;

for (i = 0; i < 4; i++) {
  printf("%d\n", myNumbers[i]);
}
```

آشنایی با زبان C

- آرایه‌ها
- آرایه چند بعدی

Two-Dimensional Arrays

A 2D array is also known as a matrix (a table of rows and columns).

To create a 2D array of integers, take a look at the following example:

```
int matrix[2][3] = { {1, 4, 2}, {3, 6, 8} };
```

	COLUMN 0	COLUMN 1	COLUMN 2
ROW 0	1	4	2
ROW 1	3	6	8

آشنایی با زبان C

- آرایه‌ها
- آرایه چند بعدی
- مقداردهی به آرایه

```
int matrix[2][3] = { {1, 4, 2}, {3, 6, 8} };
matrix[0][0] = 9;

printf("%d", matrix[0][0]);  // Now outputs 9
instead of 1
```

- دسترسی به مقادیر آرایه به کمک حلقه for

```
int matrix[2][3] = { {1, 4, 2}, {3, 6, 8} };

int i, j;
for (i = 0; i < 2; i++) {
  for (j = 0; j < 3; j++) {
    printf("%d\n", matrix[i][j]);
  }
}
```

آشنایی با زبان C

- رشته‌ها
- رشته‌ها همان آرایه‌ها هستند که از نوع char تعریف شده‌اند و معمولا برای نمایش متن استفاده می‌شوند.
- دستور printf از کجا پایان رشته را تشخیص می‌دهد!؟

```c
char greetings[] = "Hello World!";
```

```c
char greetings[] = "Hello World!";
printf("%s", greetings);
```

```c
char greetings[] = "Hello World!";
printf("%c", greetings[0]);
```

```c
char carName[] = "Volvo";
int i;

for (i = 0; i < 5; ++i) {
  printf("%c\n", carName[i]);
}
```

آشنایی با زبان C

```
char greetings[] = "Hello World!";
```

- رشته‌ها
- روش دیگر مقداردهی به رشته‌ها

- اگر NULL (\0) را فراموش کنیم
  چه اتفاقی می‌افتد!؟

```
char greetings[] = {'H', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd', '!', '\0'};
printf("%s", greetings);
```

آشنایی با زبان C

- توابع رشته‌ها
- این توابع در کتابخانه <string.h> می‌باشند.

```
char alphabet[] = "ABCDE";
printf("%d", strlen(alphabet));
printf("%d", sizeof(alphabet));
```

```
char alphabet[20] = "ABCDE";
printf("%d", strlen(alphabet));
printf("%d", sizeof(alphabet));
```

- تابع strlen
- تابع sizeof

آشنایی با زبان C

- آدرس حافظه
- استفاده از علامت & قبل از نام متغییر، آدرس آن را نشان می‌دهد.
- برای نمایش مقدار حافظه از %p استفاده میکنیم (اشاره‌گر)
- مقدار خروجی همواره در قالب هگزادسیمال می‌باشد.

```
int myAge = 43;
printf("%p", &myAge); // Outputs 0x7ffe5367e044
```

تفاوت حافظه در سیستم عامل وBareMetal؟!

Hint : Memory Controller

آشنایی با زبان C

- اشاره‌گر
- اشاره‌گر متغییری است که مقادیر حافظه را در خود نگه می‌دارد (به آدرس حافظه مربوطه اشاره می‌کند.)
- نوع اشاره‌گر بایستی با نوع دیتایی که به آن اشاره می‌کن یکسان باشد (خیلی مهم)
- فایده اشاره‌گر!؟

```c
int myAge = 43;      // An int variable
int* ptr = &myAge;  // A pointer variable, with
the name ptr, that stores the address of myAge

// Output the value of myAge (43)
printf("%d\n", myAge);

// Output the memory address of myAge
(0x7ffe5367e044)
printf("%p\n", &myAge);

// Output the memory address of myAge with the
pointer (0x7ffe5367e044)
printf("%p\n", ptr);
```

آشنایی با زبان C

- اشاره‌گر
  - نحوه دسترسی به مقادیر آرایه به کمک اشاره‌گر!؟

```c
int myNumbers[4] = {25, 50, 75, 100};
int i;

for (i = 0; i < 4; i++) {
  printf("%d\n", myNumbers[i]);
}
```

```c
int myNumbers[4] = {25, 50, 75, 100};
int i;

for (i = 0; i < 4; i++) {
  printf("%p\n", &myNumbers[i]);
}
```

```c
int myNumbers[4] = {25, 50, 75, 100};

// Get the memory address of the myNumbers array
printf("%p\n", myNumbers);

// Get the memory address of the first array element
printf("%p\n", &myNumbers[0]);
```

```c
int myNumbers[4] = {25, 50, 75, 100};
int *ptr = myNumbers;
int i;

for (i = 0; i < 4; i++) {
  printf("%d\n", *(ptr + i));
}
```

آشنایی با زبان C

```
returnType functionName(parameter1, parameter2, parameter3) {
  // code to be executed
}
```

- توابع
- توابع مجموعه ای کدها هستند که یکبار تعریف می‌شوند و می‌توان بارها از آن استفاده کرد.
- تابع main، اصلی‌ترین تابع برنامه‌نویسی C می‌باشد.
- توابع از دید پارامترهای ورودی و خروجی به 4 دسته تقسیم می‌شوند.
  - توابع بدون ورودی، بدون خروجی
  - توابع با ورودی و بدون خروجی
  - توابع بدون ورودی با خروجی
  - توابع با ورودی و خروجی

نوع خروجی نام تابع (نوع پارامترهای ورودی)

Void
Int
Float
char

Void
Int
Float
Char
Array
string

آشنایی با زبان C

- توابع
- مثال : تابع بدون ورودی، بدون خروجی

```
void myFunction() {
  printf(“Hello World!\r\n");
}
```

- استفاده از myFunction به صورت ۳ بار پشت سر هم

```
int main() {
  myFunction(); // call the function
  return 0;
}
```

آشنایی با زبان C

- توابع
- مثال : تابع با ورودی رشته و بدون خروجی

```c
void myFunction(char name[]) {
  printf("Hello %s\n", name);
}

int main() {
  myFunction("Liam");
  myFunction("Jenny");
  myFunction("Anja");
  return 0;
}
```

- خروجی!؟
- تابع با چند ورودی

```c
void myFunction(char name[], int age) {
  printf("Hello %s. You are %d years old.\n", name, age);
}

int main() {
  myFunction("Liam", 3);
  myFunction("Jenny", 14);
  myFunction("Anja", 30);
  return 0;
}
```

آشنایی با زبان C

- توابع
- مثال : تابع با ورودی آرایه و بدون خروجی

```
void myFunction(int myNumbers[5]) {
  for (int i = 0; i < 5; i++) {
    printf("%d\n", myNumbers[i]);
  }
}
```

- نحوه فراخوانی ورودی از جنس آرایه؟

```
int main() {
  int myNumbers[5] = {10, 20, 30, 40, 50};
  myFunction(?!?!?!?);
  return 0;
}
```

- روش دیگری نیز برای ورودی آرایه وجود دارد؟!
- کاربرد return 0!?

آشنایی با زبان C

- توابع
- Declaration & Definition

- در Declaration نام تابع، نوع مقداری که برمی‌گرداند و پارامترهای ورودی در صورت وجود بیان می‌شود.
- در Definition، دستوراتی که قرار است تابع انجام دهد تعریف می‌شود.

```c
// Function declaration
int myFunction(int, int);

// The main method
int main() {
  int result = myFunction(5, 3); // call the
function
  printf("Result is = %d", result);
  return 0;
}

// Function definition
int myFunction(int x, int y) {
  return x + y;
}
```

آشنایی با زبان C

- Enumeration
- تعریف متغییر جدید به منظور خوانایی کد
- توجه شود که در Enumeration بایستی متغییرها ثابت باشند.
- کاربرد مهم!؟

```c
enum Level {
   LOW,
   MEDIUM,
   HIGH
};
```

```c
int main() {
   enum Level myVar = MEDIUM;

   switch (myVar) {
     case 1:
       printf("Low Level");
       break;
     case 2:
       printf("Medium level");
       break;
     case 3:
       printf("High level");
       break;
   }
   return 0;
}
```