* Array :- Array is one of the derived secondary data type which is a collection of similar elements which are in adjacent location.

Syntax for declaring an Array

datatype variablename [element] ;

Ex :- 1) int a[5] ; // a is an array of 5-integer elements.
2) char a[10] ; // ch is an array of 10-character elements.
3) float f[5] ; // f is an array of 5-float elements.

Note :- If we want to access the elements of an array we need to use index operator ([]).
- Array index starts with 0.

```
#include <stdio.h>
void main()
{
 int a[5] ; // declaration
 int a[5] = {10,20,30,40,50} ; // intialization
 int ele, i ;
 ele = sizeof(a) / sizeof(a[0]) ; // calculating the number of elements
 for (i=0 ; i<ele ; i++)
   printf("%d", a[i]);
   printf ("\n");
 for (i=ele-1 ; i>=0 ; i--)
   printf ("%d", a[i]);
 printf("\n");
}
```

} for normal printing

} for reverse printing.

↳ 10 20 30 40 50
50 40 30 20 10

* Derived Datatype
  - Pointer
  - Array

* int a[5] = {10 20} ;
↓
In this case it is called partial initialization, remaining elements are filled with 0.

* int a[]; // error, because providing number of element is must

int a[5]; // no error, declaration

int a[5] = {10,20,30,40,50}; // no error as proper initialization

int a[] = {10,20,30,40,50}; // no error as at initialization number of element providing is option.

int a[5] = {10,20,30,40,50,60,70}; // warning as excess element and int will initialize only 5 elements.

* These are two method of declaring an array :

datatype variablename[ele];

Ex:- int a[5];
Constant

int n;
scanf("%d",&n); // dynamic declaration
int a[n];
Variable

* How to scan the array element & how to print it.

⇒
```
#include<stdio.h>
void main()
{
 int a[5],i,ele;
 ele = sizeof(a)/sizeof(a[0]);
 printf("Enter the ele...\n");
 for(i=0;i<ele;i++)
   scanf("%d",&a[i]);
 for(i=0;i<ele,i++)
   printf("%d",a[i]);
 printf("\n");
}
```

↳
```
10
20
30
40
50
10 20 30 40 50
```

* Write a program to reverse the element of an array which contains 10 integers.

Note :- Not reverse printing.

```
⇒ #include <stdio.h>
  void main()
  {
    int a[10], i, ele, j, t;
    ele = sizeof(a) / sizeof(a[0]);
    printf("Enter the element \n");
    for(i=0; i<ele; i++)
      scanf("%d", &a[i]);
    for(i=0; i<ele; i++)
      printf("%d", a[i]);
    printf("\n");
    for(i=0, j=ele-1; i<j; i++, j--)
      {
        t = a[i];
        a[i] = a[j];
        a[j] = t;
      }
    printf("After swapping...\n");
    for(i=0; i<ele; i++)
      printf("%d", a[i]);
    printf("\n");
  }.
```

I/P

swapping

O/P

	0	1	2	3	4	5	6	7	8	9
I/P →	10	20	30	40	50	60	70	80	90	100
O/P →	100	90	80	70	60	50	40	30	20	10

i	j = ele-1
0	9
1	8
2	7
3	6
4	5

* WAP to reverse first half and second half of an array elements.

⇒
```
#include <stdio.h>
void main()
{ int a[10], i, j, t, ele;
  // size of elements
  // printing and scanning of elements
  // Before swapping print
  for(i=0, j=ele/2; j<ele; i++, j++)
    { t = a[i];
      a[i] = a[j];
      a[j] = t;
    }
  // After swapping print.
```

i	j
0	5
1	6
2	7
3	8
4	9

I/P → | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |

O/P → | 60 | 70 | 80 | 90 | 100 | 10 | 20 | 30 | 40 | 50 |

* WAP to swap adjacent elements in an array of 10 integers.

⇒
```
#include <stdio.h>
void main()
{
  int a[10], i, t, ele;
  // size of elements
  // printing and scanning of elements
  // Before swap printing
  for(i=0; i<ele; i=i+2)
    { t = a[i];
      a[i] = a[i+1];
      a[i+1] = t;
    }
  // After swapping print.
```

I/P → | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |

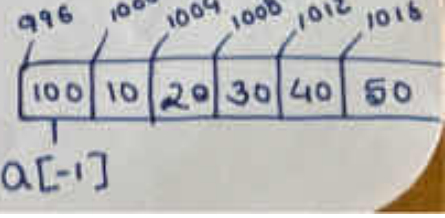
O/P → | 20 | 10 | 40 | 30 | 60 | 50 | 80 | 70 | 100 | 90 |

* WAP to count how many prime numbers are present in an array of 5 integers.

⇒
```
#include <stdio.h>
void main()
{ int a[5], i, j, ele, c;
  ele = sizeof(a) / size(a[0]);
  printf("Enter the elements \n");
  for(i=0; i<ele; i++)
    printf("%d", a[i]);
  printf("\n");
  for(i=0, c=0; i<ele; i++)
  {
    for(j=2; j<a[i]; j++)
    { if(a[i] % j == 0)
      break;
    }
    if(a[i] == j)
    c++;
    printf("c = %d \n", c);
  }
}
```

* WAP to prove that within array all the elements are adjacent locations

⇒
```
#include <stdio.h>
void main()
{
  int a[i] = {10, 20, 30, 40, 50}, i = 100;
  for(i=0; i<5; i++)
    printf("%p %d \n", &a[i], a[i]);
    printf("%p \n", &i);
printf("i = %d a[-1] = %d \n", i, a[-1]);
a[-1] = 200;
printf("i = %d a[-1] = %d \n", i, a[-1]);
```



Note :- Array name represents address

```
#include <stdio.h>
void main()
{
 int a[5] = {10,20,30,40,50};
 printf("a= %p &a= %p \n", a, &a);
 printf("a+1= %p &a+1= %p \n", a+1, &a+1);
}
```

a = 1000, &a = 1000

a+1 = 1004, &a+1 = 1020

In this example when we print the a, &a we getting same result.

a means base address of array - 1000
&a means - 1000

* Index operator can be represented in the form of de-reference operator.

a[i] == i[a] == *(a+i).

```
int a[5] = {10,20,30,40,50}
for(i=0; i<ele; i++)
  printf("%d %d %d \n", a[i], i[a], *(a+i));
```

O/P :-

a[i]	i[a]	*(a+i)
10	10	10
20	20	20
30	30	30
40	40	40
50	50	50

* for scanning the array elements we are used

scanf("%d", &a[i]);

De-reference &a[i] (∵ a[i] = *(a+i))

Reference ← &*(a+i)

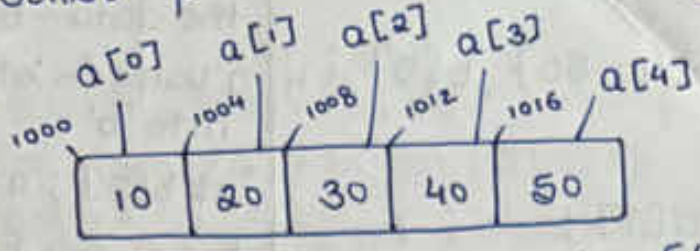
so instead of &a[i], we can replace it by (a+i).

→ Array is treated as constant pointer.

because array is constant

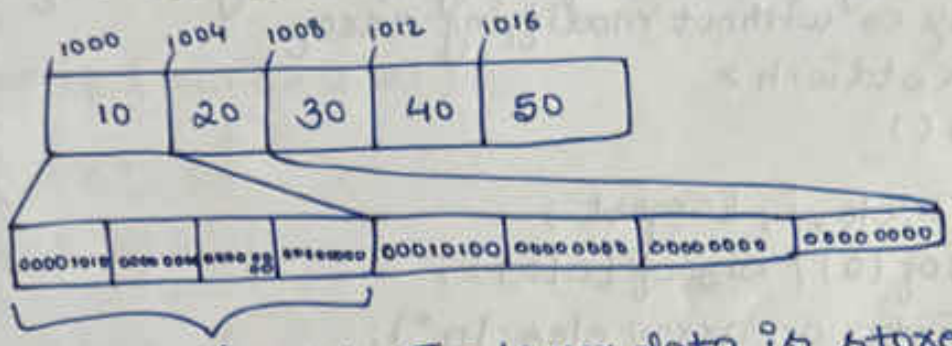
int const *p

a++ → Error
a = a+1 → Error
a+1 → No Error



int a[5] = {10, 20, 30, 40, 50}

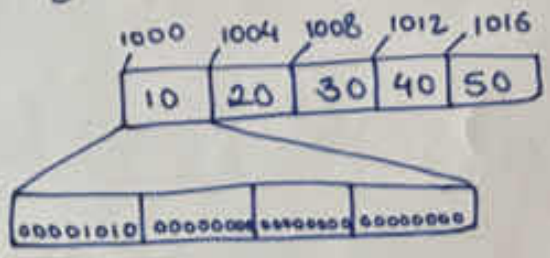
int a[5] = {10, 20, 30, 40, 50}



Little endianess. The lower data is stored in given lower address.

* WAP to print the elements of an array using integer pointer :-

```
=> #include <stdio.h>
   void main()
   {
     int a[5]={10,20,30,40,50},*p,i;
     p=a;
     for(i=0;i<5;i++)
       printf("%d",p[i]);
     printf("\n");
   }
```



array is represented as a pointer so pointer represents in an array

a[i] = *(a+i)

o/p :- 10
20
30
40
50

* WAP to copy one array element into another array.

```
#include <stdio.h>
void main()
{
int a[5] = {10,20,30,40,50}, b[5], i;
for(i=0; i<5; i++)
*(b+i)=*(a+i)  (or)  b[i]=a[i];
for(i=0; i<5; i++)
  printf("%d", b[i]);
printf("\n");
}
```

In integer to copy the data b = a
a value is stored into 'b'.
int b[5]; array
b = a → Error

```
for(i=0; i<ele; i++)
  b[i] = a[i];
```

* WAP to find largest element from given integer array without sorting array or without modifying array.

```
#include <stdio.h>
void main()
{
 int a[5], ele, i, largest;
 ele = sizeof(a) / sizeof(a[0]);
 printf("Enter an array eles:\n");
 for(i=0; i<ele; i++)
  scanf("%d", &a[i]);
  largest = a[i];          // largest = a[0];
  for(i=1; i<ele; i++)
    if(a[i] > largest)
      largest = a[i];
   printf("largest ele: %d \n", largest);
}
```

* WAP to find largest and second largest element from given integer array without using any sorting technique.

=>
```
#include <stdio.h>
void main()
{
 int a[5], ele, i, L, SL;
 ele = sizeof(a) / sizeof(a[0]);
 printf("enter an array eles : \n");
 for (i=0; i<ele ; i++)
  scanf("%d", &a[i]);
  if (a[0] > a[1])
  {
    L = a[0];
    SL = a[1];
  }
  else if (a[1] > a[0])
  {
    L = a[1];
   SL = a[0];
  }
  for (i=2 ; i<ele ; i++)
  {
    if (a[i] > L)
    {
      SL = L;
       L = a[i];
    }
    else if (a[i] > SL && a[i] != L)
      SL = a[i];
  }
  printf ("L = %d  SL = %d \n ", L, SL);
}
```

* Bubble Sort

0-1	0-1	0-1	0-1
1-2	1-2	1-2	
2-3	2-3		
3-4			

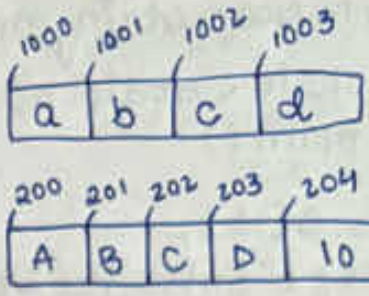
* Selection Sort

0-1	1-2	2-3	3-4
0-2	1-3	2-4	
0-3	1-4		
0-4			

* Bubble sort technique to print array of element in a ascending order.

⇒
```
#include <stdio.h>
void main()
{
 int a[5],ele,i,j,t;
 ele = sizeof(a)/sizeof(a[0]);
 printf("Enter an array eles:\n");
 for(i=0;i<ele;i++)
  scanf("%d",&a[i]);
 printf("before an array eles:\n");
 for(i=0;i<ele;i++)
   printf("%d",a[i]);
 printf("\n");
for(i=0;i<ele-1;i++)
{
 for(j=0;j<ele-1-i;j++)
 {if(a[j]<a[j+1])
   {
    t=a[j];
    a[j]=a[j+1];
    a[j+1]=t;
   }
 }
}
 printf("After sorting array eles:\n");
  for(i=0;i<ele;i++)
  printf("%d",a[i]);
printf("\n");
}
```

* Character array and Strings :




```
#include <stdio.h>
void main()
{
 char a[] = {'a','b','c','d'};
 char s[] = "ABCD";
 printf("%ld %ld \n", sizeof(a), sizeof(s));
}   ↳ 4  5
```

1. What is character array ?
⇒ Collection of characters.
2. What is string ?
⇒ Collection of characters ended with '\0'.
3. What is '\0' ?
⇒ • '\0' is one of the escape sequence.
• '\0' indicates end of the valid data.
• '\0' acts as separator between valid and invalid data.
• To store '\0' we need 1 byte of memory.
• In that 1 byte all bits are zero's (0).
• '\0' is invisible character and also it is not printable character

`char ch = '0';` ✗

0	0	1	1	0	0	0	0
128	64	32	16	8	4	2	1

`char ch = 0;` ✓

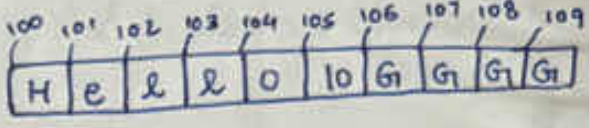
0	0	0	0	0	0	0	0

⇒ '\0'

```
#include <stdio.h>
void main()
{
 char a[10] = {'a','b','c','d'};
 char s[10] = "ABCD";
 printf("%ld %ld \n", sizeof(a), sizeof(s));
}   ↳ 10
    10
```

* How to scan string and how to print it.

⇒
```
#include <stdio.h>
void main()
{
char s[10];
printf("Enter the string \n");
scanf("%s", s); // array name
                   represent address
printf("s=%s \n", s);
printf("s+1=%s \n", s+1);
}
```



o/p → Hello
ello.

```
char ch;
sf("%c",&ch);
pf("ch=%c\n",ch);
```

```
int i;
sf("%d",&i);
pf("%d\n",
    i);
```

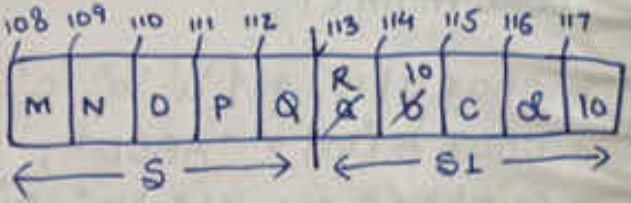
Note :- "%s" format specifies need address while scanning as well as while printing.

* Here array name represent address

*
```
#include <stdio.h>
void main()
{
char S[10];
printf("Enter the string \n");
scanf("%s", s);
printf("s=%s \n", s);
s[3] = '\0'      // explicity type casting
printf("s=%s \n", s);
}
```

↳ S=Hello
S= Hel

*
```
#include <stdio.h>
void main()
{
char S[5], S1[5];
printf("Enter S1 \n");
scanf("%s", S1);
printf("S1=%s \n" s1);
printf("Enter s \n");
Scanf("%s", s);
printf("s=%s S1=%s \n", s, s1);
}
```



o/p – abcd
S = MNOPQR
S1 = abcd
SL = R

So, please provide below size of string otherwise the adjacent values are disturbed.

* If we want to scan spaces also we need to write another format specifies which is given below.

Space ASCII = 32
\n ASCII = 10

```
{
 char s[10];
printf("Enter the string\n");
scanf("%[^\n]", s);   // to scan spaces
printf("s=%s\n", s);   while giving i/p.
}.
```

(or)

```
pf("Enter the string\n"
get(s);
put(s);
pf("s=%s\n", s);
```

* get(s) and put(s) are specially designed function for strings.

* WAP to print the given string character by character.

=>
```
#include <stdio.h>
void main()
{
 char s[10];
 int i;
 printf("Enter the string\n");
 scanf("%s", s);
 for(i=0; s[i]!='\0'; i++)
        (or)
 for(i=0; s[i]; i++)
   printf("%c", s[i]);
 printf("\n");
}
```

s[i]!='\0'

```
for(i=0; i<10; i++)
```
it will show junk data so it is not a correct condition

* WAP to find the length of the given string.

=> String length means number of valid characters excluding '\0'.

```
#include <stdio.h>
void main()
{ char s[10];
  int i;
  printf("Enter the string\n");
  Scanf("%s", s);
  for(i=0; s[i]; i++); // Dummy loop
  printf("The length of %s = %d\n", s, i);
}.
```

* Single line code for string length.

* WAP to reverse string content (not reverse printing)

⇒
```
#include <stdio.h>
void main()
{
 char s[10], t;
 int i, j, l;
 printf("Enter the string : \n");
 scanf("%s", s);
 printf("Before : %s\n", s);
 for (l=0; s[l]; l++);
 for (i=0, j=l-1; i<j; i++, j--)
 {
   t = s[i];
   s[i] = s[j];
   s[j] = t;
 }
 printf("after : %s\n", s);
}
```

I/P

a	b	c	d	e	\0

O/P

e	d	c	b	a	\0

* WAP to search specific character is present or not in a given string ?

⇒
```
#include <stdio.h>
void main()
{
 char s[10], ch;
 int i;
 printf("Enter the string : \n");
 scanf("%s", s);
 printf("Enter character : \n");
 scanf("%c", &ch);
 for (i=0; s[i]; i++)
 {
   if (s[i] == ch)
   {
     printf("char is present \n");
     return;
   }
 }
 printf("char is not present \n");
}
```

I/P

h	e	l	l	o	\0

char = 'e'

o/p → char is present

* WAP to search a character is present in how many times.

=>
```
#include <stdio.h>
void main()
{
 char s[10], c;
 int i, c=0;
 printf("Enter string :\n");
 Scanf("%s", s);
 printf("Enter character :\n");
 scanf("%c", &ch);
 for(i=0; s[i]; i++)
  { if (s[i]==ch)
      c++;
  }
  printf("char is present %d times\n", c);
}.
```

* WAP to convert lower case string to upper case string.

=>
```
#include <stdio.h>
void main()
{ char s[10];
  int i;
  printf("Enter string:\n");
  scanf("%s", s);
  printf("Before: %s\n", s);
  for (i=0; s[i]; i++)
   { if (s[i]>='a' && s[i]<='z')
       s[i] = s[i] - 32;
   }
   printf("after : %s\n", s);
}.
```

↳ aBcDe
ABCDE

// predefined function string operation

* WAP to copy one string into another string

(or)

WAP to copy source string into destination string.

⇒
```
#include <stdio.h>
void main()
{
 char s[10], d[10];
 int i;
 printf("Enter string: \n");
 scanf("%s", s);
 printf("src: %s dest: %s \n", s, d);
 for(i=0; s[i]; i++)
    d[i] = s[i];
 d[i]='\0';      (or)  d[i] = s[i];
 printf("src: %s  dest: %s \n", s, d);
}
```

s → a b c d \0
    ↓ ↓ ↓ ↓ ↓
d → a b c d \0

d[i] = s[i]

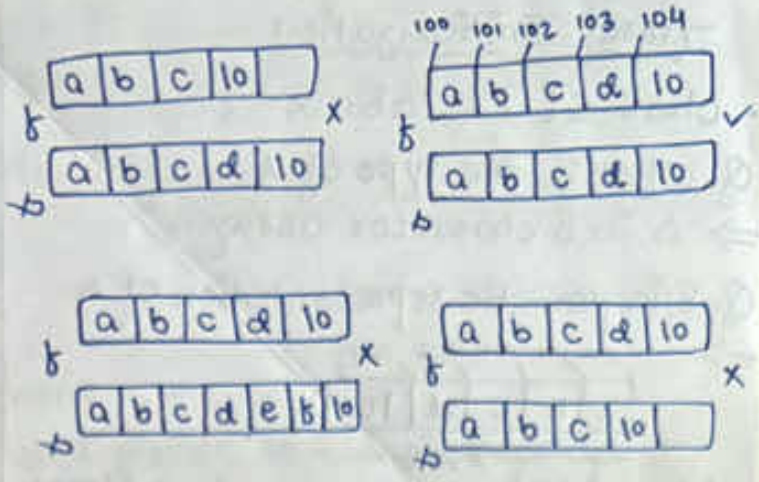
* Selection sort technique to print array of element :-

⇒
```
#include <stdio.h>
void main()
{ int a[5], ele, i, j, t;
  ele = sizeof(a) / sizeof(a[0]);
          ,  ,  ,,
          ,  ,  ,,        } i/p
          ,  ,  ,,
 for(i=0; i<ele; i++)
  { for(j=i+1; j<ele; j++)
    { if(a[i] > a[j]);
      {
        t = a[j];
        a[j] = a[i];
        a[i] = t;
      }
    }
  }
    ,,   ,,
    ,,   ,,      } o/p
    ,,   ,,
}
```

sorting

* WAP to compare the two strings. If both string are equal or not, display it.

```
#include <stdio.h>
void main()
{
char f[10], s[10], i;
printf("%s %s", f, s);
for(i=0; f[i]; i++)
{ if (f[i] != s[i]);
    break;
}
if (f[i] == s[i])
   printf("Equal\n");
else
   printf("Not Equal\n");
}.
```



* WAP to delete a given characters in a given string.

```
#include <stdio.h>
void main()
{ char s[10], ch;
  int i, j;
  printf("Enter string : \n");
  scanf("%c", &ch);
  printf("Before s = %s\n", s);
  for(i=0; s[i]; i++)
   { if (s[i] == ch)
     { for(j=i; s[j]; j++)
        s[j] = s[j+1];
       i--;
     }
   }
   printf("After s = %d\n", s);
}.
```

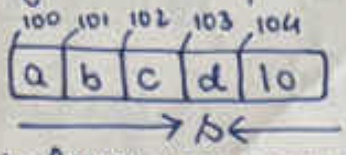
* Diff. between character array initialization and character pointer initialization.

Array Initialization

- char s[] = "abcd";

Q. What is the type of s
⇒ s is a character array.

Q. Diagramatic representation of s
⇒



Note :- Array name represents address.

Q. What is the size of s
⇒ 5 bytes (including \0).

Q. How much of memory involved in s
⇒ 5 bytes.

- printf("s=%s\n", s);
  ↳ abcd.
- s = 100
  &s = 100.
- s++ ⟶ error.
- s = p; ⟶ error.
- P.f("%c\n", s[i]); // s[1]
  ↳ b.
- P.f("s=%s\n", s);
  s[1] = 'm';
  P.f("s=%s\n", s);
  ↳ amcd.
- s = "mnop" ⟶ Error.
  |
  const ptr

Pointer Initialization

- char *p = "ABCD";

Q. What is the type of p
⇒ p is a character pointer

Q. Diagramatic representation of p
⇒

Note :- String constant represent address

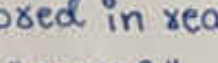
Q. What is the size of p
⇒ 8 bytes.

Q. How much of memory involved in p
⇒ 8 + 5 bytes (i.e., 8 byte for pointer & 5 byte for data).

- printf("p=%s\n", p);
  ↳ ABCD.
- p = 500
  &p = 200.
- p++ ⟶ No error.
- p = s ⟶ No error.
- P.f("%s", p); ⟶ ABCD
  p = s;
  P.f("%s", p); ⟶ abcd.
- Pf("%c\n", p[1]);
  ↳ B.
- Pf("p=%s\n", p);
  p[1] = 'm'; // modification happening
  Pf("s=%s\n", p);
  ↳ because of this modification segmentation fault is occuring.

Note :- When we initialize a character pointer with one string that string is stored in read only section.

- p = "MNOP"



⟶ No Error.