

1D - Array

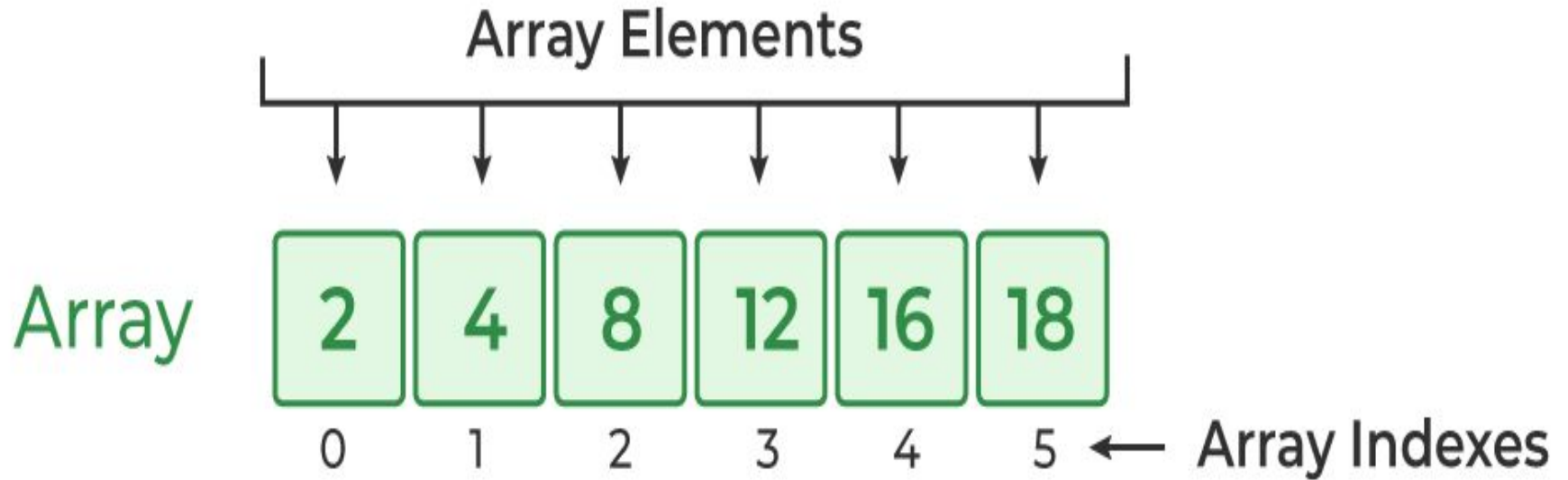
BY:MOAYAD ALMASRY

CPMP133

1D - Array

| | | | | |
|---|---|---|---|---|
| a | r | r | a | y |
|---|---|---|---|---|

Array in C



1D - Array

```
DataType arrayName[ ] ;
```



طريقة تعريف الـ Array

1D - Array

```
DataType arrayName[ ] ;
```

طريقة تعريف ال Array, هكذا تم إنشاء Array فارغة

```
arrayName[ 5 ]={1,2,3,4,5 };
```

ثم نعطي قيمة لل Array عبر هذه الطريقة

```
arrayName[ 5 ]={0};
```

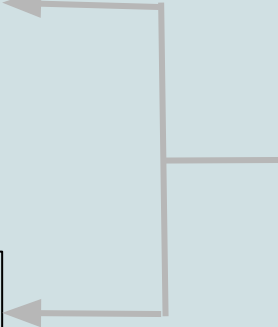
وهكذا تم إنشاء Array فارغة

1D - Array

```
DataType arrayName [length] = { };
```

```
DataType arrayName[ ] = {DT0 , DT1 , DT2 , DT3 ...,};
```

يمكن ايضا تعريف
الarray
وإعطائها قيمة
في خطوة واحدة
بهذه الطريقة



1D - Array

نقوم بتحديد نوع الـ Array

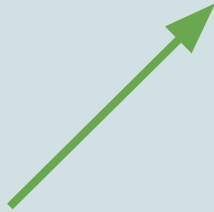
ويمكننا استخدام **primitive DataType** مثل : `int` , `char` , `double` ,



```
DataType arrayName[ArrayLength] ;
```

1D - Array

نقوم بتحديد طول المصفوفة بعد اختيار نوعها, ويجب ان نتذكر ان العنصر الأول من المصفوفة يرمز له بـ `array[0]` , والعنصر الأخير `array[length-1]`



```
DataType arrayName[ ] = { }
```


1D - Array

```
int a[ 7] = { };
```

هكذا تم انشاء Array من نوع int ,
وإعطاء طول للـ Array ويساوي 7 ,
حيث أن :

أول عنصر بالـ Array سيكون array[0]
آخر عنصر بالـ Array سيكون array[6]

1D - Array

```
Int a[7]={ };
```

```
array [0] = 1 ;  
array [1] = 2 ;  
array [2] = 2 ;  
array [3] = 3 ;  
array [4] = 0 ;  
array [5] = 6 ;  
array [6] = 3 ;
```

قمنا بتعبئة الـ array كاملة ,
وهكذا سيكون شكل الـ Array

array

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 2 | 2 | 3 | 0 | 6 | 3 |
|---|---|---|---|---|---|---|

1D - Array

```
int a[7]={ };
```

```
array [0] = 1 ;
```

```
array [1] = 2 ;
```

```
array [2] = 2 ;
```

```
array [3] = 3 ;
```

```
array [4] = 0 ;
```

```
array [5] = 6 ;
```

```
array [6] = 3 ;
```



1D - Array

```
int array[ ] = {1 , 2 , 2 , 3 , 0 , 6 , 3 };
```

الطريقة الثانية :
نقوم بإعطاء قيمة لل array بعد
إنشائها بوضع { } وبداخلهما قيم
ال array من نوع
ال dataType التي تم انشاء
ال array منها , وطول ال array
هو عدد العناصر التي تم إدخالها
بها

EXAMPLE

1

```
int array[ ] = {1 , 2 , 2 , 3 , 0 , 6 , 3 };
```

| Code | output |
|----------|--------|
| length | |
| array[1] | |
| array[7] | |

EXAMPLE

1

```
int array[ ] = {1 , 2 , 2 , 3 , 0 , 6 , 3 };
```

| Code | output |
|----------|--------|
| length | 7 |
| array[1] | 2 |
| array[7] | random |

EXAMPLE 2

what is the output ?

```
double array[5]={ };  
array[0] = 3 ;  
array[2] = 8.2 ;  
array[3] = 3.2 ;  
  
for (int i = 0; i < 5; i++) {  
    printf("%d",array[i]);  
}
```

output

| |
|--|
| |
| |
| |
| |
| |
| |

EXAMPLE 2

what is the output ?

```
double array[5] = { };  
array[0] = 3 ;  
array[2] = 8.2 ;  
array[3] = 3.2 ;  
  
for (int i = 0; i < 5; i++) {  
    printf("%f",array[i]);  
}
```

output

3.0

0.0

8.2

3.2

0.0

1D - Array

array of char

```
char name[4] = { } ;  
name[0] = "a" ;  
name[1] = "b" ;  
name[3] = "c" ;  
for (int i = 0; i < 4; i++) {  
    printf("%c", name[i]);  
}
```

output

| |
|--------|
| output |
| |
| |
| |
| |
| |

1D - Array

array of char

```
Char name[4] = {} ;  
name[0] = "a" ;  
name[1] = "b" ;  
name[3] = "c" ;  
for (int i = 0; i < 4; i++) {  
    printf("%c", name[i]);  
}
```

output

a

b

No thing

c

1D - Array

copy arrays

```
int array1[ ]= { 4 , 2 , 11 , 2 , 0 } ;  
int *array2 = array1 ;  
  
for (int i = 0; i < 5; i++) {  
    printf("%d \n",array2[i]);  
}
```

output

| |
|--|
| |
| |
| |
| |
| |

1D - Array

copy arrays

```
int array1[ ] = {4 , 2 , 11 , 2 , 0} ;  
int array2 []={ };
```

```
for (int i = 0; i <5; i++) {  
    array2[i]=array1[i];  
}
```

```
for (int i = 0;i<5; i++) {  
    printf("%d \n",array2[i]);  
}
```

note:

Int array2[];

الصيغة خطأ لأنه لم يحدد

حجم ال array

اما هذه الصيغة صحيحة

int array2 []={}

output

4

2

11

2

0

1D - Array

copy arrays

```
int array1[] = { 4 , 2 , 11 , 2 , 0 }  
;
```

array1:int[]

عند إنشاء الarray
يتم حجز مكان لها
بالذاكرة

1D - Array

copy arrays

ولكن ماذا لو قمنا بتعريف array اخرى ك pointer وجعلها تساوي ال array الأولى ؟

```
int array1[] = { 4 , 2 , 11 , 2 , 0 }  
;
```

```
Int *array2 = array1 ;
```

or

```
int array1[] = { 4 , 2 , 11 , 2 , 0 }  
;
```

```
Int *array2 = &array1 ;
```



array2[]

1D - Array

copy arrays

```
int array1[] = { 4 , 2 , 11 , 2 , 0 } ;
```

```
int *array2 = array1 ;
```

في حالة مساواتهما معا وكانا يتكونا من نفس النوع
من ال **dataType** فإنه يتم تخزينهما معا في نفس
المكان بالذاكرة , وأي تغيير على أحدهما يعني التغيير
على الآخر

array1:int[]
array2:int[]

1D - Array

coupy arrays

```
int array1[] = {4, 2, 11, 2, 0};  
int *array2 = array1;  
  
array2[0] = 1111;  
array1[1] = 710;  
  
for (int i = 0; i < 5; i++) {  
    printf("%d\n", array1[i]);  
}
```

output

| |
|--|
| |
| |
| |
| |
| |

1D - Array

copy arrays

```
int array1[] = {4, 2, 11, 2, 0};  
int *array2 = array1;  
  
array2[0] = 1111;  
array1[1] = 710;  
  
for (int i = 0; i < 5; i++) {  
    printf("%d\n", array1[i]);  
}
```

output

1111

710

11

2

0

1D - Array

copy arrays

```
double array1[ ]= {4.1 , 2.9 , 11 , 2.5 , 0.2}  
;  
int array2[5] ;  
  
for (int i = 0; i < 5; i++) {  
    array2[i] = array1[i] ;  
}  
  
for (int i = 0; i < 5; i++) {  
    printf("%d\n", array2[i]);  
}  
}
```

output

| |
|--|
| |
| |
| |
| |
| |

1D - Array

copy arrays

```
double array1[] = {4.1, 2.9, 11, 2.5, 0.2};  
int array2[5]; // Assuming both arrays have 5 elements  
  
// Copying elements from array1 to array2  
for (int i = 0; i < 5; i++) {  
    array2[i] = array1[i];  
}  
  
// Printing elements of array2  
for (int i = 0; i < 5; i++) {  
    printf("%d\n", array2[i]);  
}
```

في هذه الحالة نحن نخزن
array من نوع
double الى اخرى من
نوع int

هنا سيقوم باهمال القيم بعد
الفاصله وعمل
casting

output

| |
|--|
| |
| |
| |
| |
| |

1D - Array

copy arrays

```
double array1[] = {4.1, 2.9, 11, 2.5, 0.2};  
int array2[5]; // Assuming both arrays have  
5 elements  
  
// Copying elements from array1 to array2  
for (int i = 0; i < 5; i++) {  
    array2[i] = array1[i];  
}  
  
// Printing elements of array2  
for (int i = 0; i < 5; i++) {  
    printf("%d\n", array2[i]);  
}
```

output

4

2

11

2

0

1D - Array

Bubble Sort array

```
int size=5;
int array[5]={9,5,3,7,1};
for (int i = 0; i < size - 1; i++) {
    for (int j = 0; j < size -i-1; j++) {
        if (array[j] > array[j + 1]) {
            int temp = array[j];
            array[j] = array[j + 1];
            array[j + 1] = temp;
        }
    }
    for (int k = 0; k < size; k++) {
        printf("%d\n",array[k]);
    }
}
```

قام بترتيب العناصر تصاعديا

output

1

3

5

7

9

1D - Array

```
int size = 5;
int array[] = {9, 1, 3, 7, 5};

for (int i = 0; i < size - 1; i++) {
    for (int j = i + 1; j < size; j++) {
        if (array[j] < array[i]) {
            int temp = array[i];
            array[i] = array[j];
            array[j] = temp;
        }
    }
}

for (int k = 0; k < size; k++) {
    printf("%d\n", array[k]);
}
```

Selection Sort array

قام بترتيب العناصر تصاعديا

| output |
|--------|
| 1 |
| 3 |
| 5 |
| 7 |
| 9 |

1D - Array

Bubble Sort array

```
int size=5;
int array[5]={9,1,3,7,5};
for (int i = 0; i < size - 1; i++) {
    for (int j = 0; j < size - 1; j++) {
        if (array[j] < array[j + 1]) {
```

```
            int temp = array[j];
            array[j] = array[j + 1];
            array[j + 1] = temp;
```

```
        }
    }
}
for (int k = 0; k < size; k++) {
    printf("%d\n", array[k]);
}
```

قام بترتيب العناصر تنازلي

| output |
|--------|
| 9 |
| 7 |
| 5 |
| 3 |
| 1 |

1D - Array

نريد ان نبحث عنه داخل المصفوفه \\key=3 int

```
int size = 5;
```

```
int array[] = {9, 1, 3, 7, 5};
```

```
for (int i = 0; i < size; i++) {  
    if(array[i]==key) {  
        printf("index of key is:%d",i);  
        break;  
    }  
}
```

Linear Search in
array

output

index of
key is:3

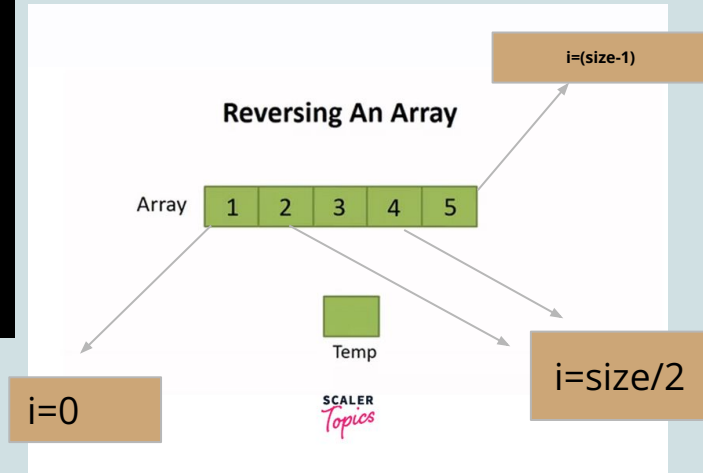
1D - Array

```
int size = 5;
int array[] = {1, 2, 3, 4, 5};

for (int i = 0; i < size/2; i++) {
    int temp=array[i];
    array[i]=array[size-i-1];
    array[size-i-1]=temp;
}

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

Reverse array



output

5

4

3

2

1

Array in function

1D - Array

طرق لجعل ال function يستقبل Array

```
void fun(int *arr)
{
    int i;
    int n = 8;
    for (i = 0; i < n; i++)
        printf("%d ", arr[i]);
}

int main()
{
    int arr[] = { 1, 2, 3, 4, 5, 6, 7, 8 };
    fun(&arr);
    return 0;
}
```

```
void fun(int arr[ ])
{
    int i;
    int n = 8;
    for (i = 0; i < n; i++)
        printf("%d ", arr[i]);
}

int main()
{
    int arr[] = { 1, 2, 3, 4, 5, 6, 7, 8 };
    fun(arr);
    return 0;
}
```

Array in function

1D - Array

طرق ارسال Array الى function

```
void fun(int *arr)
{
    int i;
    int n = 8;
    for (i = 0; i < n; i++)
        printf("%d ", arr[i]);
}

int main()
{
    int arr[] = { 1, 2, 3, 4, 5, 6, 7, 8 };
    fun(&arr);
    return 0;
}
```

```
void fun(int arr[ ])
{
    int i;
    int n = 8;
    for (i = 0; i < n; i++)
        printf("%d ", arr[i]);
}

int main()
{
    int arr[] = { 1, 2, 3, 4, 5, 6, 7, 8 };
    fun(arr);
    return 0;
}
```

Array in function

1D - Array

```
void fun(int arr[ ])
{
    int i;
    int n = 8;
    for (i = 0; i < n; i++)
        scanf("%d", arr[i]);
}

int main()
{
    int arr[] = { 1, 2, 3, 4, 5, 6, 7, 8 };
    fun(arr);

    return 0;
}
```

عندما نرسل array الى function فاننا
نرسل ال addresses الخاص بها

هذا يعني انه اي تعديل يحدث على array
داخل ال function سوف يحدث ايضا على
قيمتها داخل ال main

Array in function

1D - Array

output

9

6

4

7

9

3

5

```
int main() {
    int n=7;
    int arr[] = { 1, 2, 3, 4, 5, 6, 7 };
    fun(arr,n);

    for (int i = 0; i <7; i++) {

        printf("%d ", arr[i]); }
    return 0;
}

void fun(int arr1[ ],int n){
    int arr2[] = {9,6, 4, 7, 9, 3, 5};

    for (int i = 0; i < n; i++) {
        arr1[i] = arr2[i];
    }
}
```

What the output of following code

- A)61
- B)39
- c)none
- d)78

```
int x[5] = {15, 22, 39, 50, 21};
```

```
int s = x[5 % 2] + x[2];
```

```
printf("The value of s is: %d\n", s);
```

- x[0] = 15
- x[1] = 22
- x[2] = 39
- x[3] = 50
- x[4] = 21

$X[5\%2] = X[1] = 22$

$S = x[1] + x[2]$

$S = 22 + 39 = 61$

```

int main() {
    int b[7] = {7, 8, 4, 2, 13, 87, 31}, i, m =
0;
    char c[7];
    m = F1(b, c, 7);
    for (i = 0; i < 7; i++) {
        printf("\n%c", c[i]);
    }
    printf("\n# of passes: %d\n", m);
    return 0;
}

int F1(int a[], char c[], int n) {
    int i, pass = 0;
    float x = 0;
    for (i = 0; i < n; i++) {
        x = x + a[i];
    }
    x = x / n;
    for (i = 0; i < n; i++) {
        if (a[i] > x) {
            c[i] = 'P';
            pass++;
        } else {
            c[i] = 'F';
        }
    }
    return pass;
}

```

Output:

```

{
F
F
F
F
F
P
P
# of passes: 2
}

```

```
int k[6] = {0, 0, 0, 0, 0, 0};  
int i, n;  
for (i = 3; i < 6; ++i) {  
    scanf("%d", &n);  
    k[n] = i;  
}
```

What will be the values of k[1] and k[3] after execution of the code segment below using the given data: 2 0 1?

k[1]=5
k[3]=0

