C_47 → Arrays in C - Part 2

Initialization of arrays

* Arrays can be initialized in 2 ways
  - → Compile time
  - → Run time

① Compile time :-

* when we declare array; at that time we initialize array.

✓ → int a[5] = { 0, -1, 11, 10, 2 }; ✓ correct

✗ → int a[ ]; ✗ wrong

✓ → int a[ ] = { 0, 1, 2, 0, -1, 6, 7 }; ✓ correct

✓ → int a[5] = { 0, 1, -1 }; ✓ correct

* Size is 5 but only 3 values; So remaining 2 values are 0, 0.

0	1	-1	0	0

✓ → int a[5]; ✓ correct * But garbage value.

any garbage values are assigned.

→ Designated initialization

→ a[5] = {[0] = 1, [1] = 2, [2] = 3, [3] = 4, [4] = 5}

→ int a[5] = {1, 2, 3, 4, 5, 6, 7} wrong

* because here size is 5; but we give 7 values so wrong.

→ int a[5]; Correct

a[0] = 1 a[1] = 2 a[2] = 3 a[3] = 4
a[4] = 5

* Index of array; generally starts with zero.

→ int a[5] = {0}; Correct

→ int a[5] = { }; wrong error.

→ char b[] = {'j', 'e', 'n', 'n', 'y'};

→ char b[10] = {'j', 'e', 'n', 'n', 'y'};

* Remaining 5 values are null.

* when we fix the size of array at run time we cannot change the array size and their values.

* But when we want the array size at the run time; we use run time initialization of array.

② Run time :

At run time we use standard functions to initialize array.

getting input at run time

```
int i;
int a[5];
printf(" Enter the elements of array: ");
for (i=0; i<5 ; i++)
{
    scanf("%d", &a[i]);
}
```

	0	1	2	3	4
a	0	1	-1	0	2
	a[0]	a[1]	a[2]	a[3]	a[4]

i: ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~

array size − 1

$\boxed{5-1 \Rightarrow 4}$

printing values of array index

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

When to initialize array :-

* When initialization have small value (ie) 1-5; then we can initialize at compile time.

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

* But if size of array is large (ie) 1-100; we can initialize at runtime.

```
int a[100], i;
for(i=0; i<100; i++)
{if(i<30)
    a[i]=1;
else
    a[i]=0;
}
```

First 30 nos in array is 1 and remaining 0

CODE 1:

```c
#include <stdio.h>
#include <stdlib.h>
/* 1 - 1D ARRAY INITIALIZATION */
//COMPILE TIME INITIALIZATION
int main()
{
int i,a[5];
a[0]=1;
a[1]=2;
a[2]=3;
a[3]=4;
a[4]=5;
for(i=0;i<5;i++)
printf("%d\n",a[i]);
}
```

"D:\1. C NOTEBOOK\C LANGUAGE\C PROGRAMS\PART 5_Jennys Lectures\PART 4_JENNYS LECTURE_ARRAYS\1_ONE DIMENSIONAL A...

```
1
2
3
4
5
```

CODE 2:

```c
#include <stdio.h>
#include <stdlib.h>
/* 2 - 1D ARRAY INITIALIZATION */
//COMPILE TIME INITIALIZATION
int main()
{
    int i,a[5]={1,2,3,4,5};  //Preferred Method to Initialize
    //a[]; wrong
    //a[-5]; wrong
    //a[2]={1,2,3}; wrong -> array size is 2 but elements are 3
    //a[5]={}; wrong
    //array elements should be of same types either integer or float...
    //a[]={1,2,3,4,5}; correct
    //a[5]={0}; correct
    //a[5]={1,2,3}; correct
    char c[10]={'j','a','y','a','n','t','h','i'};
    for(i=0;i<5;i++)
    printf("%d ",a[i]);
    printf("\n");
    for(i=0;i<8;i++)
     printf("%c",c[i]);
     getch();
```

"D:\1. C NOTEBOOK\C LANGUAGE\C PROGRAMS\PART 5_Jenn

```
1 2 3 4 5
jayanthi
```

CODE 3:

```c
#include <stdio.h>
#include <stdlib.h>
/* 3 - 1D ARRAY INITIALIZATION */
//RUN TIME INITIALIZATION
int main()
{
 int i,a[5];
 printf("Enter the elements:\n");
 for(i=0;i<5;i++)
  scanf("%d",&a[i]);
 printf("Array Elements are:\n");
 for(i=0;i<5;i++)
  printf("%d\n",a[i]);
  getch();
}
/* We use for loop to get the array elements since we no need not write
   the scanf() function for 5 times to get the element values */
```

"D:\1. C NOTEBOOK\C LANGUAGE\C PROGRAMS\PART 5_Jennys Lectures\PART 4_JENNYS LECTURE_ARRAYS\1_ONE DIMENSIONAL A...

```
Enter the elements:
3
4
-1
5
7
Array Elements are:
3
4
-1
5
7
```

CODE 4:

```c
#include <stdio.h>
#include <stdlib.h>
/* 4 - 1D ARRAY INITIALIZATION */
//RUN TIME INITIALIZATION
int main()
{
  int i,n,a[100];
  printf("Enter the limit of array size:\n");
  scanf("%d",&n);
  printf("Enter the array elements:\n");
  for(i=0;i<n;i++)
   scanf("%d",&a[i]);
  printf("The array element values are:\n");
  for(i=0;i<n;i++)
   printf("%d\n",a[i]);
}
/* Run time initialization of array will be used when array size is larger..
   Here we assign array size as 100 but we set limit till 50 only..*/
```

"D:\1. C NOTEBOOK\C LANGUAGE\C PROGRAMS\PART 5_Jennys Lectures\PART 4_JENNYS LECTURE_ARRAYS\1_ONE DIM

```
Enter the limit of array size:
5
Enter the array elements:
1
2
3
4
5
The array element values are:
1
2
3
4
5

Process returned 0 (0x0)   execution time : 7.033 s
Press any key to continue.
```

CODE 5:

```c
#include <stdio.h>
#include <stdlib.h>
/* 5 - 1D ARRAY INITIALIZATION */
int main()
{
    int i,a[10];

    for(i=0;i<10;i++)
    {
     if(i<5)
     printf("%d\n",a[i]=1);
    else
     printf("%d\n",a[i]=2);
    }
    getch();
}
```

"D:\1. C NOTEBOOK\C LANGUAGE\C PROGRAMS\PART 5_

```
1
1
1
1
1
2
2
2
2
2
```

CODE 6:

```c
#include <stdio.h>
#include <stdlib.h>
#define A 3  // if 5 means size of array will be changed to 5
/* 6 - 1D ARRAY INITIALIZATION */
//ARRAY INITIALIZATION USING MACRO NAME
int main()
{
  int i,a[A];
  printf("Enter the array elements:\n");
  for(i=0;i<A;i++)
  scanf("%d",&a[i]);
  printf("The array elements are:\n");
  for(i=0;i<A;i++)
  printf("%d\n",a[i]);
}
/* MACRO names are easy when we have so many lines of array size and need not
 to change each and every line instead we can change the macro value in header
 file.... */
```

```
"D:\1. C NOTEBOOK\C LANGUAGE\C PROGRAMS\PART 5_Jennys Lectures\PART 4_JENNYS LECTURE
Enter the array elements:
3
2
1
The array elements are:
3
2
1

Process returned 0 (0x0)   execution time : 5.344 s
Press any key to continue.
```