Array

Array in C is the collection of similar data type, all the elements are stored at the contiguous memory location. In c or c++ language we can create a static or dynamic array, the static array has the fixed size but the size of the dynamic array can be changed at runtime. The dynamic array is created using the pointer and library functions (malloc, calloc) or new operator which are used to allocate the memory from the heap or free store.

Use the memset to clear all block of the character array, it provides the surety that every block of the array contains zero.

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

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

(a + i) = &a[i];

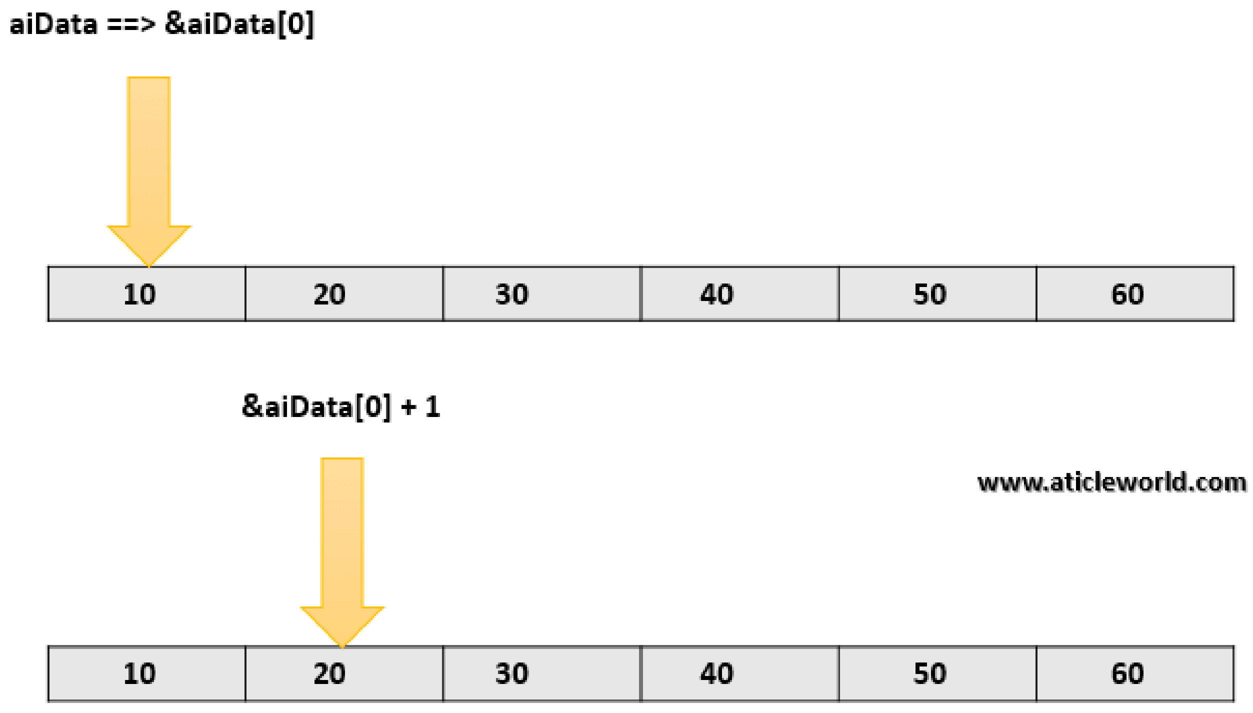
If the value of i is 0.

(a + 0) = &a[0];

a = &a[0];

means that array name is the address of its first element.

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



Note: But when you put the ampersand(&) before the array name then its type change.It becomes a pointer to the array.

In the short, we can say.

a = Pointer to the first element of the array .

&a = Pointer to an array of 6 elements.

&a + 1 = Address of next memory block (Address ahead of 6 integers)

*(&a + 1) = Dereferencing to *(&a + 1) gives the address of first element of second memory block.

*(&a + 1) – a = Since *(&a + 1) points to the address ahead of 6 integers , the difference between two is 6.

Ways to passing a 2D array as parameter to the function

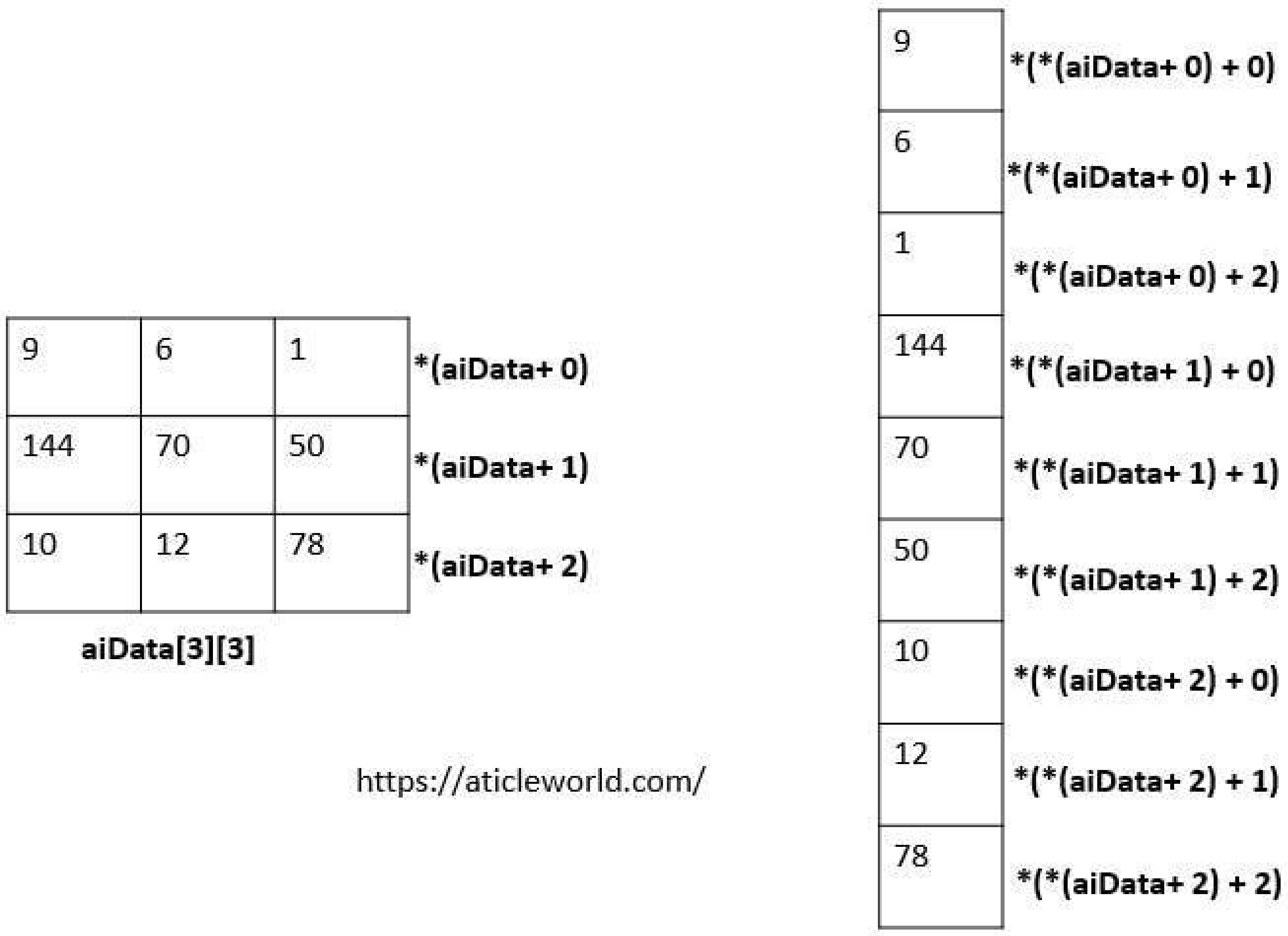
Similar to the 1D array we can pass the 2D array as a parameter to the function. It is important to remember that when we pass a 2D array as a parameter decays into a pointer to an array, not a pointer to a pointer.

Passing 2d array to function in c using pointers

The first element of the multi-dimensional array is another array, so here, when we will pass a 2D array then it would be split into a pointer to the array.

For example,

If int aiData[3][3], is a 2D array of integers, it would be split into a pointer to the array of 3 integers (int (*)[3]).



Why “&” is not used for strings in scanf() function?

In case of a string (character array), the variable itself points to the first element of the array in question. Thus, there is no need to use the ‘&’ operator to pass the address.

Important Points

1. ‘&’ is used to get the address of the variable. C does not have a string type, String is just an array of characters and an array variable stores the address of the first index location.
2. By default the variable itself points to the base address and therefore to access base address of string, there is no need of adding an extra ‘&’

Difference between Structure and Array in C

Array in C

An array is collection of items stored at continuous memory locations.

Array uses subscripts or “[ ]” (square bracket) for element access

Array is pointer as it points to the first element of the the collection.

Bit filed is not possible in an Array.

Array size is fixed and is basically the number of elements multiplied by the size of an element.

Array declaration is done simply using [] and not any keyword.

Arrays is a primitive datatype

Structure in C

A structure is a user defined data type in C/C++. A structure creates a data type that can be used to group items of possibly different types into a single type.

Structure uses “.” (Dot operator) for element access

Structure is not a pointer

Structure size is not fixed as each element of Structure can be of different type and size.

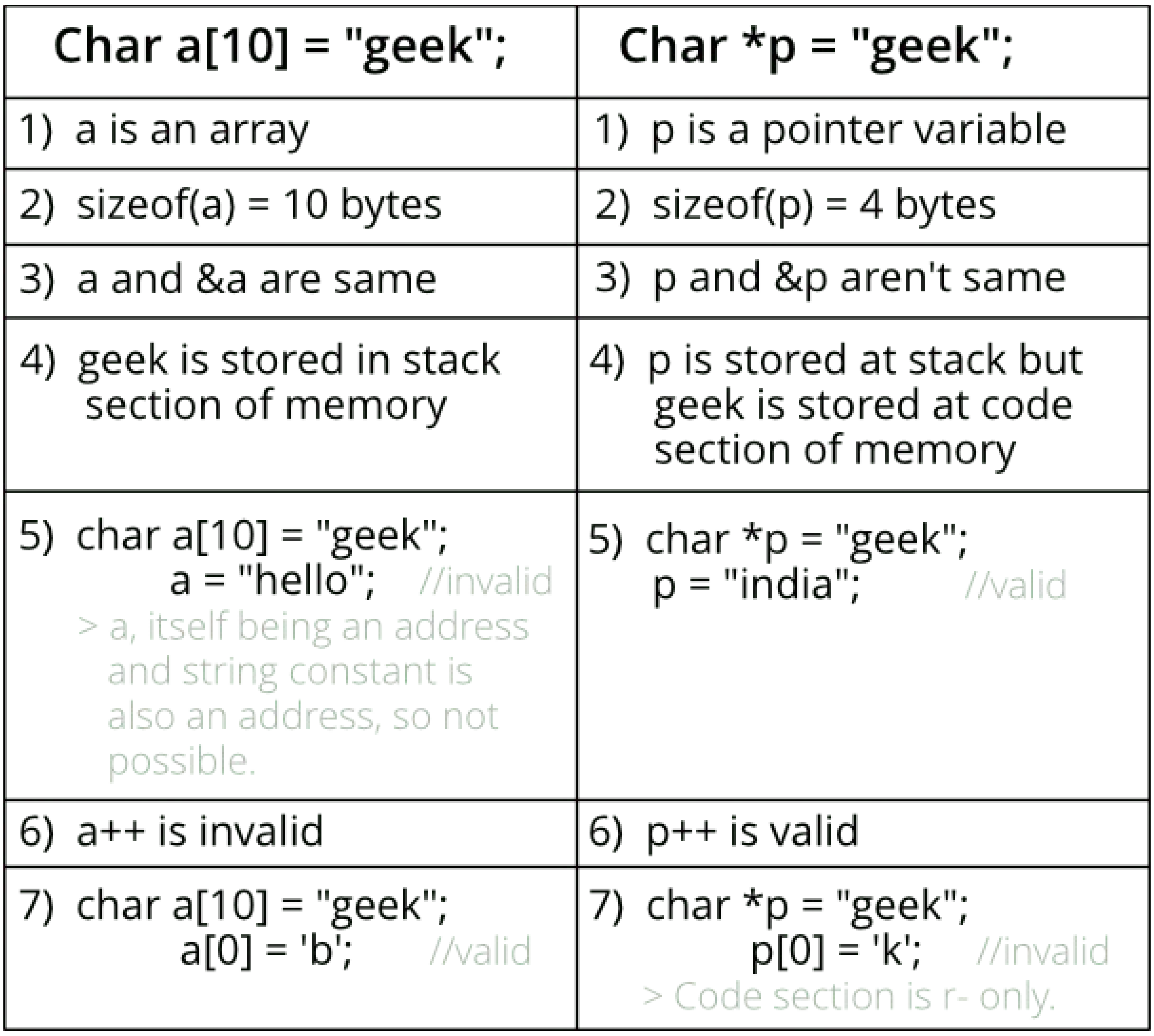
Bit filed is possible in an Structure.

Structure declaration is done with the help of “struct” keyword.

Structure is a user-defined datatype.

What’s difference between char s[] and char *s in C?

```
char s[] = "geeksquiz";
char *s  = "geeksquiz";
```



Char a[10] = "geek";	Char *p = "geek";
1) a is an array	1) p is a pointer variable
2) sizeof(a) = 10 bytes	2) sizeof(p) = 4 bytes
3) a and &a are same	3) p and &p aren't same
4) geek is stored in stack section of memory	4) p is stored at stack but geek is stored at code section of memory
5) char a[10] = "geek"; a = "hello"; //invalid > a, itself being an address and string constant is also an address, so not possible.	5) char *p = "geek"; p = "india"; //valid
6) a++ is invalid	6) p++ is valid
7) char a[10] = "geek"; a[0] = 'b'; //valid	7) char *p = "geek"; p[0] = 'k'; //invalid > Code section is r- only.

The statements ‘**char s[] = “geeksquiz”**‘ creates a character array which is like any other array and we can do all array operations. The only special thing about this array is, although we have initialized it with 9 elements, its size is 10 (Compiler automatically adds ‘\0’)

The statement ‘**char *s = “geeksquiz”**‘ creates a string literal. The string literal is stored in read only part of memory by most of the compilers. The C and C++ standards say that string literals have static storage duration, any attempt at modifying them gives undefined behavior.
s is just a pointer and like any other pointer stores address of string literal.

Why C treats array parameters as pointers?

Array parameters treated as pointers because of efficiency. It is inefficient to copy the array data in terms of both memory and time; and most of the times, when we pass an array our intention is to just tell the array we interested in, not to create a copy of the array.

Difference between strlen() and sizeof() for string in C

sizeof()

Sizeof operator is a compile time unary operator which can be used to compute the size of its operand.

- The result of sizeof is of unsigned integral type which is usually denoted by size_t.
- sizeof can be applied to any data-type, including primitive types such as integer and floating-point types, pointer types, or compound datatypes such as Structure, union etc.

strlen()

strlen() is a predefined function in C whose definition is contained in the header file “string.h”.

strlen() accepts a pointer to an array as argument and walks through memory at run time from the address we give it looking for a **NULL** character and counts up how many memory locations it passed before it finds one.

- The main task of strlen() is to count the length of an array or string.

sizeof vs strlen()

1. **Type:** Sizeof operator is a unary operator whereas strlen() is a predefined function in C
2. **Data types supported:** Sizeof gives actual size of any type of data (allocated) in bytes (including the null values) whereas get the length of an array of chars/string.
3. **Evaluation size:** sizeof() is a compile-time expression giving you the size of a type or a variable's type. It doesn't care about the value of the variable.
   Strlen on the other hand, gives you the length of a C-style NULL-terminated string.
4. **Summary:** The two are almost different concepts and used for different purposes.

```
// C program to demonstrate difference
// between strlen() and sizeof()

#include<stdio.h>
#include<string.h>

int main()
{

char str[] = "November";
printf("Length of String is %d\n", strlen(str));
printf("Size of String is %d\n", sizeof(str));

}
```

```
Length of String is 8
Size of String is 9
```

Since size of char in C is 1 byte but then also we find that strlen() gave one less value than sizeof().

Explanation : We know, that every string terminates with a NULL character (“\0”).
strlen() searches for that NULL character and counts the number of memory address passed, So it actually counts the number of elements present in the string before the NULL character, here which is 8.
sizeof() operator returns actual amount of memory allocated for the operand passed to it. Here the operand is an array of characters which contains 9 characters including Null character and size of 1 character is 1 byte. So, here the total size is 9 bytes.

Anything written in sizeof() is never executed in C

In C/C++ sizeof() operator is used to find size of a date type or variable. Expressions written in sizeof() are never executed.

Is sizeof for a struct equal to the sum of sizeof of each member?

The sizeof for a struct is not always equal to the sum of sizeof of each individual member. This is because of the padding added by the compiler to avoid alignment issues. Padding is only added when a structure member is followed by a member with a larger size or at the end of the structure.

Different compilers might have different alignment constraints as C standards state that alignment of structure totally depends on the implementation.

What is the difference between single quoted and double quoted declaration of char array?

In C/C++, when a character array is initialized with a double quoted string and array size is not specified, compiler automatically allocates one extra space for string terminator ‘\0’.

When character array is initialized with comma separated list of characters and array size is not specified, compiler doesn’t create extra space for string terminator ‘\0’

Storage for Strings in C

In C, a string can be referred to either using a character pointer or as a character array.

Strings as character arrays

When strings are declared as character arrays, they are stored like other types of arrays in C. For example, if str[] is an auto variable then string is stored in stack segment, if it’s a global or static variable then stored in data segment, etc.

Strings using character pointers
Using character pointer strings can be stored in two ways:

1) Read only string in a shared segment.
When a string value is directly assigned to a pointer, in most of the compilers, it’s stored in a read-only block (generally in data segment) that is shared among functions.

```
char *str = "GfG";
```

In the above line “GfG” is stored in a shared read-only location, but pointer str is stored in a read-write memory. You can change str to point something else but cannot change value at present str. So this kind of string should only be used when we don’t want to modify string at a later stage in the program.

2) Dynamically allocated in heap segment.
Strings are stored like other dynamically allocated things in C and can be shared among functions.

```
int main()
{
  int size = 4;
  /* Stored in heap segment like other dynamically allocated things */
  char *str = (char *)malloc(sizeof(char)*size);
  *(str+0) = 'G';
  *(str+1) = 'f';
  *(str+2) = 'G';
  *(str+3) = '\0';
  *(str+1) = 'n';  /* No problem: String is now GnG */
   getchar();
   return 0;
}
```

Do not use sizeof for array parameters

The function fun() receives an array parameter arr[] and tries to find out number of elements in arr[] using sizeof operator.
In C, array parameters are treated as pointers . So the expression sizeof(arr)/sizeof(arr[0]) becomes sizeof(int *)/sizeof(int) which results in 1 for IA 32 bit machine (size of int and int * is 4) and the for loop inside fun() is executed only once irrespective of the size of the array.

Therefore, sizeof should not be used to get number of elements in such cases. A separate parameter for array size (or length) should be passed to fun()