Difference between pointer and array in C?

Pointers are used for storing address of dynamically allocated arrays and for arrays which are passed as arguments to functions

- Behavior of sizeof operator
- sizeof(array) returns the amount of memory used by all elements in array
- sizeof(pointer) only returns the amount of memory used by the pointer variable itself

  int arr[] = {10, 20, 30, 40, 50, 60};
  int *ptr = arr;

  Size of arr[] 24
  Size of ptr 8

- Assigning any address to an array variable is not allowed.
- the & operator
  &array is an alias for &array[0] and returns the address of the first element in array
  &pointer returns the address of pointer
- a string literal initialization of a character array
  char array[] = “abc” sets the first four elements in array to ‘a’, ‘b’, ‘c’, and ‘\0’
  char *pointer = “abc” sets pointer to the address of the “abc” string (which may be stored in read-only memory and thus unchangeable)

Similarities

- Pointer name gives address of first element of array.
- Array members are accessed using pointer arithmetic.
- Array parameters are always passed as pointers, even when we use square brackets.

Dangling pointer

A pointer pointing to a memory location that has been deleted (or freed) is called dangling pointer. There are **three** different ways where Pointer acts as dangling pointer

- De-allocation of memory
- Function Call
- Variable goes out of scope
- Never return the address of the local variable from the function. It will be the causes of the dangling pointer.

How to avoid the dangling pointer errors

The behavior of the dangling pointer is undefined, so it is very important to avoid the born of dangling pointers. The common mistake that is done by many programmers is that not assigning the NULL explicitly after freeing the dynamic memory. So It is very good habits to assign the NULL after deallocation of the dynamically allocated memory.

Apart from that, another mistake is to return the address of the local variable (stack variable) from the function, it is also a cause to create a dangling pointer. Using the static variable we can resolve the problem because the lifetime of the static variable is the entire run of the program.

Void pointer

A void pointer in c is called a generic pointer, it has no associated data type. It can store the address of any type of object and it can be type-casted to any types. According to C standard, the pointer to void shall have the same representation and alignment requirements as a pointer to a character type. A void pointer declaration is similar to the normal pointer, but the difference is that instead of data types we use the void keyword.

Using the indirection operator (*) we can get back the value which is pointed by the pointer, but in case of void pointer we cannot use the indirection operator directly. This is because a void pointer has no data type that creates a problem for the compiler to predict the size of the pointed object. So before dereferencing the void * we have to type cast it, it enables the compiler to predict the data types.

Why is void pointers use?

A very important feature of the void pointer is reusability. Using the void pointer we can store the address of any object and whenever required we can get back the object through the indirection operator with proper casting.

Application of void pointer in C

Application of void pointers are very wide, we can not cover all the application in one article. Here I am taking one of the most popular applications of the void pointer in qsort function.

A qsort is a C standard library function that use to sort arrays. Using the qsort function, we can sort the array of integer, double, long etc.

Following is the declaration for qsort() function,

void qsort(void *arr, size_t elements, size_t size, int (*comp)(const void *, const void*));

Parameters of qsort:

arr − pointer to the first element of the array.

elements− number of elements in the array.

size − size(in bytes) of the element in the array.

comp − compare function that is used to compares two elements.
int comp(const void* a, const void* b);

Advantages of the void pointer in c

- Using the void pointer we can create a generic function that can take arguments of any data type. The memcpy and memmove library function are the best examples of the generic function, using these function we can copy the data from the source to destination.

Important Points

1. void pointers cannot be dereferenced. It can however be done using typecasting the void pointer
2. Pointer arithmetic is not possible on pointers of void due to lack of concrete value and thus size.

Size of Void is 1 byte and Size of void * is 8 bytes

The reason malloc, calloc or realloc library function return void *. Due to the void * these functions are used to allocate memory to any data type.

- Using the void * we can create a generic linked list.For more information see this link: How to create generic Link List.

NULL Pointer

NULL Pointer is a pointer which is pointing to nothing. In case, if we don't have address to be assigned to a pointer, then we can simply use NULL.

Important Points

1. NULL vs Uninitialized pointer – An uninitialized pointer stores an undefined value. A null pointer stores a defined value, but one that is defined by the environment to not be a valid address for any member or object.
2. NULL vs Void Pointer – Null pointer is a value, while void pointer is a type

To check for null pointer before accessing any pointer variable. By doing so, we can perform error handling in pointer related code e.g. dereference pointer variable only if it’s not NULL.

Size of Null is 8 bytes on 64 bit compiler

Dereference is not possible

Use of null pointer in C

A pointer that is not pointing the address of a valid object or valid memory should be initialized to NULL. It prevents the pointer to become a dangling pointer and ensure the programmer that pointer is not pointing anywhere.

Wild pointer

A pointer which has not been initialized to anything (not even NULL) is known as wild pointer. The pointer may be initialized to a non-NULL garbage value that may not be a valid address.

Why C treats array parameters as pointers?

If the pointer coming in really is the base address of a whole array, then we should use [ ].

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.

Complex pointer :

1. Pointer to function
2. Pointer to array
    - Pointer to array of integer
    - Pointer to array of function
    - Pointer to array of character
    - Pointer to array of structure
    - Pointer to array of union

    - Pointer to array of array
    - Pointer to two dimensional array
    - Pointer to three dimensional array
    - Pointer to array of string
    - Pointer to array of pointer to string
- Pointer to structure
- Pointer to union
- Multilevel pointers

What are near, far and huge pointers?

- Near pointer is used to store 16 bit addresses means within current segment on a 16 bit machine. The limitation is that we can only access 64kb of data at a time.
- A far pointer is typically 32 bit that can access memory outside current segment. To use this, compiler allocates a segment register to store segment address, then another register to store offset within current segment.
- Like far pointer, huge pointer is also typically 32 bit and can access outside segment. In case of far pointers, a segment is fixed.In far pointer, the segment part cannot be modified, but in Huge it can be

restrict keyword in C

- restrict keyword is mainly used in pointer declarations as a type qualifier for pointers.
- It doesn’t add any new functionality. It is only a way for programmer to inform about an optimizations that compiler can make.

- When we use restrict with a pointer ptr, it tells the compiler that ptr is the only way to access the object pointed by it and compiler doesn’t need to add any additional checks.
- If a programmer uses restrict keyword and violate the above condition, result is undefined behavior.
- restrict is not supported by C++. It is a C only keyword.

Pointer to an Array | Array Pointer

int (*ptr)[10];

On dereferencing a pointer expression we get a value pointed to by that pointer expression. Pointer to an array points to an array, so on dereferencing it, we should get the array, and the name of array denotes the base address. So whenever a pointer to an array is dereferenced, we get the base address of the array to which it points.

Difference between ++*p, *p++ and *++p