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. In other contexts, arrays and pointer are two different things, see the following programs to justify this statement.

Behavior of sizeof operator

```c
// 1st program to show that array and pointers are different
#include <stdio.h>

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

   // sizof(int) * (number of element in arr[]) is printed
   printf("Size of arr[] %d\n", sizeof(arr));

   // sizeof a pointer is printed which is same for all type
   // of pointers (char *, void *, etc)
   printf("Size of ptr %d", sizeof(ptr));
   return 0;
}
```

Output:

```
Size of arr[] 24
Size of ptr 8
```

Assigning any address to an array variable is not allowed.

```c
// IInd program to show that array and pointers are different
#include <stdio.h>

int main()
{
   int arr[] = {10, 20}, x = 10;
   int *ptr = &x; // This is fine
   arr = &x;  // Compiler Error
   return 0;
}
```

Output:

```
Compiler Error: incompatible types when assigning to
            type 'int[2]' from type 'int *'
```

Although array and pointer are different things, following properties of array make them look similar.

1) *Array name gives address of first element of array.*

Consider the following program for example.

```c
#include <stdio.h>

int main()
{
   int arr[] = {10, 20, 30, 40, 50, 60};
   int *ptr = arr;  // Assigns address of array to ptr
   printf("Value of first element is %d", *ptr);
   return 0;
}
```

Output:

```
Value of first element is 10
```

2) *Array members are accessed using pointer arithmetic.*

Compiler uses pointer arithmetic to access array element. For example, an expression like "arr[i]" is treated as *(arr + i) by the compiler. That is why the expressions like *(arr + i) work for array arr, and expressions like ptr[i] also work for pointer ptr.

```c
#include <stdio.h>

int main()
{
   int arr[] = {10, 20, 30, 40, 50, 60};
   int *ptr = arr;
   printf("arr[2] = %d\n", arr[2]);
   printf("*(arr + 2) = %d\n", *(arr + 2));
   printf("ptr[2] = %d\n", ptr[2]);
   printf("*(ptr + 2) = %d\n", *(ptr + 2));
   return 0;
}
```

Output:

```
arr[2] = 30
*(arr + 2) = 30
ptr[2] = 30
*(ptr + 2) = 30
```

3) *Array parameters are always passed as pointers, even when we use square brackets.*

```c
#include <stdio.h>

int fun(int ptr[])
{
    int x = 10;

    // size of a pointer is printed
    printf("sizeof(ptr) = %d\n", sizeof(ptr));

    // This allowed because ptr is a pointer, not array
    ptr = &x;

    printf("*ptr = %d ", *ptr);

    return 0;
}
// Driver code
int main()
{
    int arr[] = {10, 20, 30, 40, 50, 60};
    fun(arr);
    return 0;
}
```

Output:

```
sizeof(ptr) = 8
*ptr = 10
```