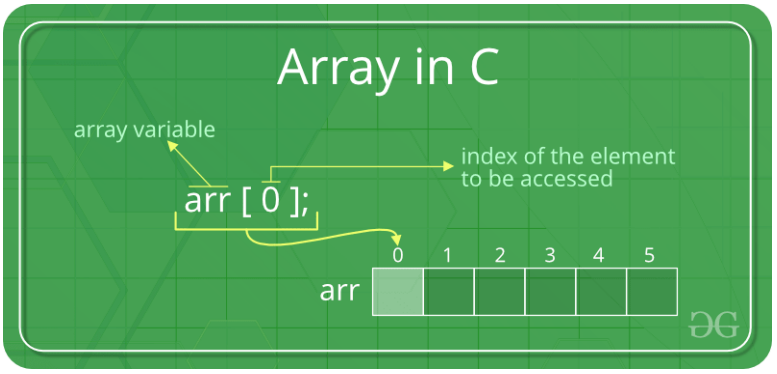
Accessing Array Elements in C++

Accessing Array Elements:

- Array elements are accessed by using an integer index. Array index starts with 0 and goes till the size of the array minus 1.Its same as C language.
- The name of the array is also a pointer to the first element of the array.




```
#include <iostream>
using namespace std;

int main()
{
    int arr[5];
    arr[0] = 5;
    arr[2] = -10;

    // this is same as arr[1] = 2
    arr[3 / 2] = 2;
    arr[3] = arr[0];

    cout << arr[0] << " " << arr[1] << " " << arr[2] << " "
        << arr[3];

    return 0;
}
```

Output

```
5 2 -10 5
```

No Index Out of bound Checking:

There is no index out of bounds checking in C++, for example, the following program compiles fine but may produce unexpected output when run.

```
// This C++ program compiles fine
// as index out of bound
// is not checked in C.

#include <iostream>
using namespace std;

int main()
{
    int arr[2];

    cout << arr[3] << " ";
    cout << arr[-2] << " ";

    return 0;
}
```

Output

```
0 0
```

The elements are stored at contiguous memory locations

Example:

```
// C++ program to demonstrate that array elements
// are stored contiguous locations

#include <iostream>
using namespace std;
```

```
int main()
{
    // an array of 10 integers.
    // If arr[0] is stored at
    // address x, then arr[1] is
    // stored at x + sizeof(int)
    // arr[2] is stored at x +
    // sizeof(int) + sizeof(int)
    // and so on.
    int arr[5], i;

    cout << "Size of integer in this compiler is "
        << sizeof(int) << "\n";

    for (i = 0; i < 5; i++)
        // The use of '&' before a variable name, yields
        // address of variable.
        cout << "Address arr[" << i << "] is " << &arr[i]
            << "\n";

    return 0;
}
```

Output

```
Size of integer in this compiler is 4
Address arr[0] is 0x7ffeb5b3c850
Address arr[1] is 0x7ffeb5b3c854
Address arr[2] is 0x7ffeb5b3c858
Address arr[3] is 0x7ffeb5b3c85c
Address arr[4] is 0x7ffeb5b3c860
```