

# C++ Programming

## Pointers and Arrays

**Mostafa S. Ibrahim**

*Teaching, Training and Coaching since more than a decade!*

*Artificial Intelligence & Computer Vision Researcher*

*PhD from Simon Fraser University - Canada*

*Bachelor / Msc from Cairo University - Egypt*

*Ex-(Software Engineer / ICPC World Finalist)*



# 1D Array in memory

- `Int arr[4] = {5, 6, 7, 8};`
  - `cout<<arr[2] ⇒ 7`
  - 4 integers defined **consecutively** in memory
    - Addresses: 1008, 1010, 1012, 1014

|      |      |      |      |
|------|------|------|------|
| 5    | 6    | 7    | 8    |
| 1008 | 1010 | 1012 | 1014 |

# 2D Array in memory

- Recall flattening array 3x4 to 12 elements
- The memory is just 12 numbers consecutive
- In row-major order style (typical)
  - First row, then 2nd row and so on
- In col-major order style
  - Column 1, then 2nd column and so on
- Same logic for 3D and beyond

```
int num[3][4] = {  
    {1, 2, 3, 4},  
    {5, 6, 7, 8},  
    {9, 10, 11, 12}  
};
```

# 2D Array in memory

|         |             |      |      |      |             |      |      |      |             |      |      |      |
|---------|-------------|------|------|------|-------------|------|------|------|-------------|------|------|------|
|         | <— row 0 —> |      |      |      | <— row 1 —> |      |      |      | <— row 2 —> |      |      |      |
| value   | 1           | 2    | 3    | 4    | 5           | 6    | 7    | 8    | 9           | 10   | 11   | 12   |
| address | 1000        | 1002 | 1004 | 1006 | 1008        | 1010 | 1012 | 1014 | 1016        | 1018 | 1020 | 1022 |

↑  
first element of the array num

- Same logic for 3D and beyond: `arr[2][3][4]`
- Same order as accessing: loop `i` (0-1), loop `j` (0-2), loop `k` (0-3): `arr[i][j][k]`

# Pointer to array

```
4 int main() {  
5     int arr[] { 3, 5, 7, 9 };  
6  
7     // value of array is address: to first element  
8     cout << arr << "\n";    // 0x7f  
9  
10    int &val = arr[0];  
11  
12    cout << val << " " << &val << "\n"; // 3 0x7f  
13    cout << &arr[0] << "\n";    // 0x7f  
14  
15    int *ptr = arr;  
16    // also value of ptr is address  
17    // 3 0x7f 0x2aa  
18    cout << *ptr << " " << ptr << " " << &ptr << "\n";  
19  
20    // point to array: you can use it as same as array  
21    ptr[0] = 10, ptr[1] = 20;  
22  
23    for (auto &val : arr)  
24        cout << val << " ";  
25    cout << "\n";    // 10 20 7 9  
26}
```

# Offset Notation

```
5  int arr[] { 3, 5, 7, 9 };
6
7  int *ptr = arr;
8
9  // 0x7ffc528b6470 0x7ffc528b6470
10 cout << ptr + 0 << " " << &arr[0] << "\n";
11 // 0x7ffc528b6474 0x7ffc528b6474
12 cout << ptr + 1 << " " << &arr[1] << "\n";
13 // 0x7ffc528b6478 0x7ffc528b6478
14 cout << ptr + 2 << " " << &arr[2] << "\n";
15
16 // 3 3
17 cout << *(ptr + 0) << " " << arr[0] << "\n";
18 // 5 5
19 cout << *(ptr + 1) << " " << arr[1] << "\n";
20 // 7 7
21 cout << *(ptr + 2) << " " << arr[2] << "\n";
22 // 7 7
23 cout << *(ptr + 2) << " " << *(arr + 2) << "\n";
24
25 // arr[index] = subscript notation
26 // *(ptr+2) = offset notation
27
28 // 9 6 BE CAREFUL
29 // *ptr + 3 = 3 + 3 = 6 = get value of current cell and increment 3
30 // *(ptr + 3) = move to extra 3 cells then get value
31 cout << *(ptr + 3) << " " << *ptr + 3 << "\n";
32
```

# Pointers Arithmetic

```
4 int main() {
5     int arr[] { 3, 5, 7, 9 };
6
7     int *ptr = arr;
8     cout << *ptr << "\n";    // 3
9
10    ++ptr;                    // move to next memory cell
11    cout << *ptr << "\n";    // 5
12
13    cout << *ptr++ << "\n"; // 5
14    cout << *ptr << "\n";    // 7
15
16    cout << *++ptr << "\n"; // 9
17    cout << *ptr << "\n";    // 9
18
19    cout << *--ptr << "\n"; // 7
20    cout << *ptr << "\n";    // 7
21
22    ptr -= 2;    // go back 2 positions
23    cout << *ptr << "\n";    // 3
24
25    --ptr; // now this is 1 step before array. BE CAREFUL
26    //++arr;    CE
27
28    ptr = arr + 3;
29    cout << ptr - arr << "\n"; // 3 cells
30
31    ptr = arr;
32    cout << ++*ptr << "\n";    // get value then increment 4
33    cout << ptr - arr << "\n"; // 0 = still same location
34
```

# Iterating over array

- We can compare addresses, so let's use it to iterate also!
  - Be careful from going beyond limits

```
4 int main() {  
5     int arr[] { 3, 5, 7, 9 };  
6  
7     int *ptr = arr;  
8  
9     while (ptr != arr + 4)  
10         cout << *ptr++ << " ";  
11     cout<<"\n";        // 3 5 7 9  
12  
13     ptr = arr;  
14     while (ptr != arr + 5)  
15         cout << *ptr++ << " ";  
16     cout<<"\n";        // 3 5 7 9 -520862016  
17  
18     return 0;  
19 }
```



# Iterating over char array

```
4 int main() {  
5     char arr[] = "hello";  
6     char * str = arr;  
7  
8     while(*str != '\0')  
9         cout<<*str++;  
10  
11     return 0;  
12 }  
13
```

# Comparing Pointers

```
4 int main() {  
5     string str1 = "mostafa";  
6     string str2 = "mostafa";  
7  
8     cout << (str1 == str2) << "\n"; // true  
9  
10    string *p1 = &str1;  
11    string *p2 { &str2 }; // c++ style  
12  
13    // BE CAREFUL: do u wanna compare addresses or values?  
14    cout << (p1 == p2) << "\n"; // false  
15    cout << (*p1 == *p2) << "\n"; // true  
16  
17    return 0;  
18 }  
19
```

*“Acquire knowledge and impart it to the people.”*

*“Seek knowledge from the Cradle to the Grave.”*