



# COMP 345 Week 6

[h\\_lai@encs.concordia.ca](mailto:h_lai@encs.concordia.ca)

# Array / Dynamic Array



# Array

1. When declare an array, you allocate a sequential memory cells;
2. The name of the array is the pointer to the first memory cell of the array;

access array through index

1 2 3 4 5

access array through pointer

6 7 8 9 10

```
array.cpp
1 #include <iostream>
2
3 int main(int argc, char const *argv[])
4 {
5     const int SIZE = 5;
6
7     int arr1[SIZE], arr2[SIZE];
8
9     for (int i = 0; i < SIZE; ++i)
10    {
11        arr1[i] = i + 1;
12        arr2[i] = SIZE + i + 1;
13    }
14
15    std::cout << "access array through index" << std::endl;
16    for (int i = 0; i < SIZE; ++i)
17    {
18        std::cout << arr1[i];
19        if (i == SIZE - 1)
20        {
21            std::cout << std::endl;
22        }
23        else {
24            std::cout << " ";
25        }
26    }
27
28    std::cout << "access array through pointer" << std::endl;
29    for (int i = 0; i < SIZE; ++i)
30    {
31        std::cout << *(arr2 + i);
32        if (i == SIZE - 1)
33        {
34            std::cout << std::endl;
35        }
36        else {
37            std::cout << " ";
38        }
39    }
40
41
42    return 0;
43 }
```


Line 40, Column 1

# Attention

```
1  #include <iostream>
2
3  int main(int argc, char const *argv[])
4  {
5      const int SIZE = 5;
6
7      int arr[SIZE];
8
9      for (int i = 0; i < SIZE; ++i)
10     {
11         arr[i] = i + 1;
12     }
13
14     std::cout << "access array through index" << std::endl;
15     for (int i = 0; i < 10; ++i)
16     {
17         std::cout << arr[i] << " ";
18     }
19
20     return 0;
21 }
```

access array through index

1 2 3 4 5 32767 -869728034 1503577280 1338197440 32767



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

#define HEIGHT 3
#define WIDTH 4

int main()
{
    int table[HEIGHT][WIDTH] = {2,4,6,8,10,12,14,16,18,20,22,24};

    // use index notation
    for (int i = 0; i < HEIGHT; i++)
        for (int j = 0; j < WIDTH; j++)
            cout << table[i][j] << ' ';
    cout << endl;

    // use pointer arithmetic on each row sequentially
    for (int i = 0; i < HEIGHT; i++)
        for (int j = 0; j < WIDTH; j++)
            cout << *((table + i) + j) << ' ';
    cout << endl;

    // use pointer arithmetic to access the entire array sequentially as stored
    for (int i = 0; i < HEIGHT*WIDTH; i++)
        cout << *((table + 0) + i) << ' ';
    cout << endl;

    cout << *((table + 1)) << endl;    //10
    cout << *((table + 1) + 7) << endl; //24
    cout << *((table + 2) - 8) << endl; //2
    int n; cin >> n;
    return 0;
}
```



# Dynamic array

1. When you don't know your array's length, you cannot use array
2. You can use dynamic array with keyword "new"
3. You have to free the memory manually using keyword "delete []"

# Observer Pattern





The cases when certain objects need to be informed about the changes occurring in other objects are frequent. To have a good design means to decouple as much as possible and to reduce the dependencies. The Observer Design Pattern can be used whenever a subject has to be observed by one or more observers.



