Object Oriented Programming and Data Structures

Lab Report

VECTORS AND ARRAYS

PC Muhammad Awais Fazal e Qadeer

41 - MTS A

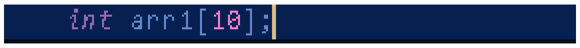
Regn No 10791

LE Arshia

Differences between Vectors and Arrays

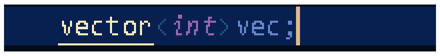
1. Initialization

Arrays are initialized by syntax *arrayname[]*



```
int arr1[10];
```

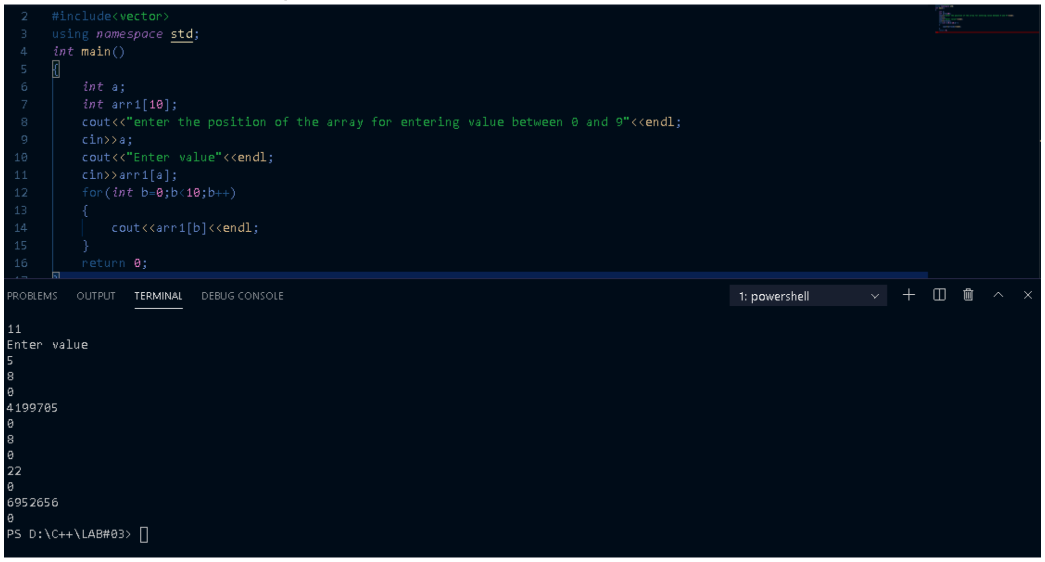
Whereas, Vectors can be initialized by *Vector<datatype>obj1,ob2;*



```
vector<int> vec;
```

2. Indexing

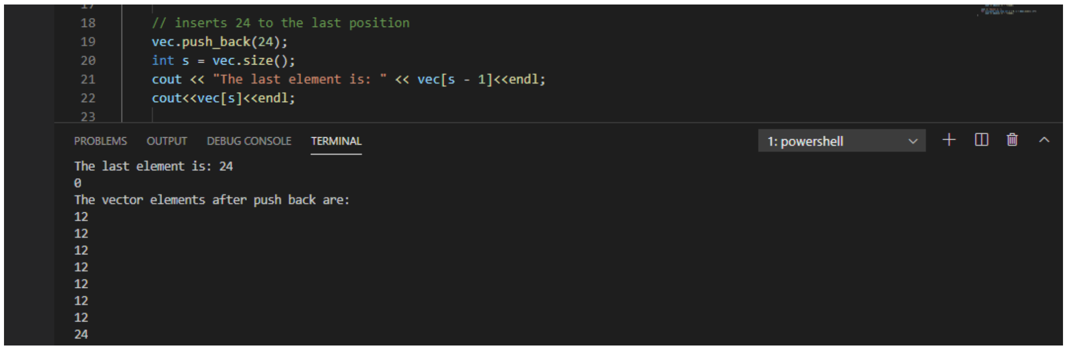
Arrays have the option to enter values based upon index, therefore values maybe issued randomly.



```
#include<vector>
using namespace std;
int main()
{
    int a;
    int arr1[10];
    cout<<"enter the position of the array for entering value between 0 and 9"<<endl;
    cin>>a;
    cout<<"Enter value"<<endl;
    cin>>arr1[a];
    for(int b=0;b<10;b++)
    {
        cout<<arr1[b]<<endl;
    }
    return 0;
```

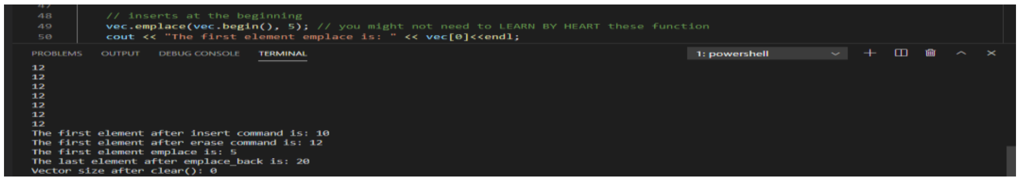
```
11
Enter value
5
8
0
4199705
0
8
0
22
0
6952656
0
PS D:\C++\LAB#03>
```

Whereas, vectors are dynamic in nature therefore size increases with increase in declaration of values.



```
    // inserts 24 to the last position
    vec.push_back(24);
    int s = vec.size();
    cout << "The last element is: " << vec[s - 1]<<endl;
    cout<<vec[s]<<endl;
```

```
The last element is: 24
0
The vector elements after push back are:
12
12
12
12
12
12
12
24
```

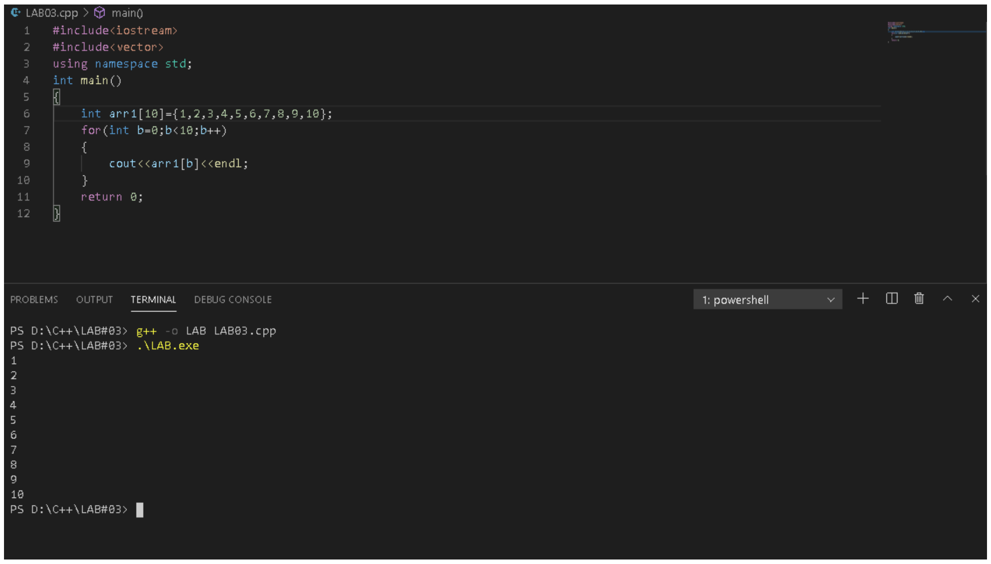


```cpp
    // inserts at the beginning
    vec.emplace(vec.begin(), 5); // you might not need to LEARN BY HEART these function
    cout << "The first element emplace is: " << vec[0]<<endl;
```

```
12
12
12
12
12
12
12
The first element after insert command is: 10
The first element after erase command is: 12
The first element emplace is: 5
The last element after emplace_back is: 20
Vector size after clear(): 0
```

3. Size

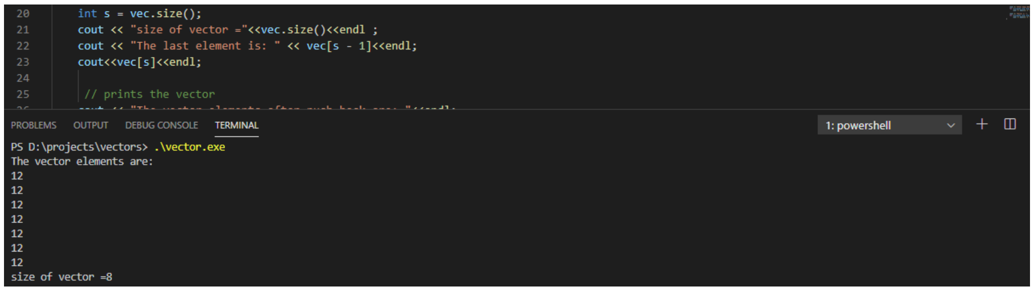
Arrays have fixed size therefore, once it is initialized it cannot be changed.



```cpp
#include<iostream>
#include<vector>
using namespace std;
int main()
{
    int arr1[10]={1,2,3,4,5,6,7,8,9,10};
    for(int b=0;b<10;b++)
    {
        cout<<arr1[b]<<endl;
    }
    return 0;
}
```

```
PS D:\C++\LAB#03> g++ -o LAB LAB03.cpp
PS D:\C++\LAB#03> .\LAB.exe
1
2
3
4
5
6
7
8
9
10
PS D:\C++\LAB#03>
```

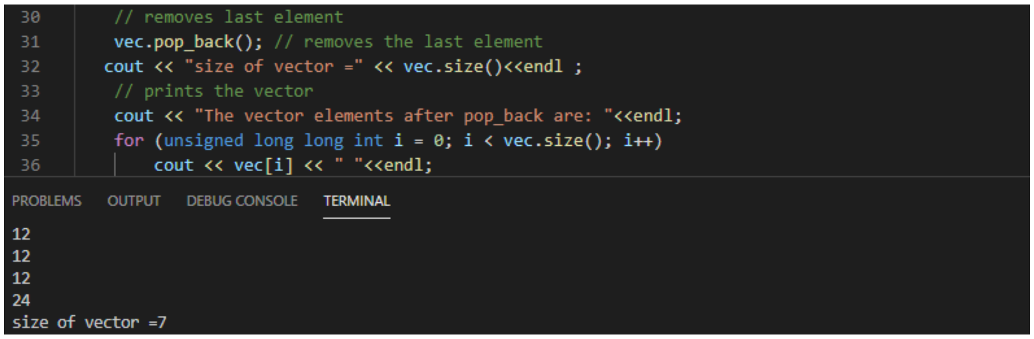
Vector has a dynamic memory and is increased with increase in insertion of newer values.



```cpp
    int s = vec.size();
    cout << "size of vector ="<<vec.size()<<endl ;
    cout << "The last element is: " << vec[s - 1]<<endl;
    cout<<vec[s]<<endl;

     // prints the vector
```

```
PS D:\projects\vectors> .\vector.exe
The vector elements are:
12
12
12
12
12
12
12
size of vector =8
```

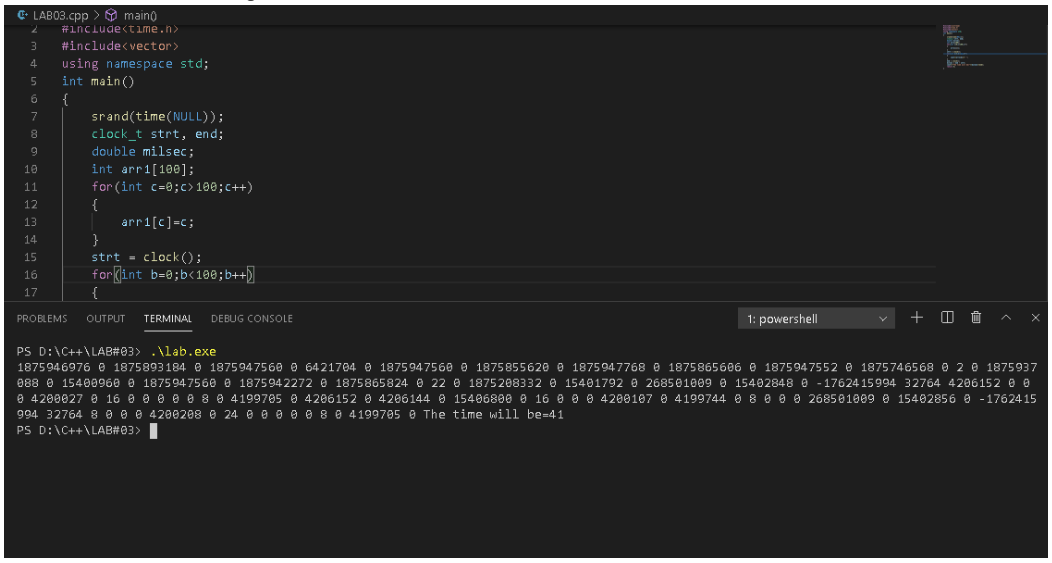


```cpp
     // removes last element
     vec.pop_back(); // removes the last element
    cout << "size of vector =" << vec.size()<<endl ;
     // prints the vector
     cout << "The vector elements after pop_back are: "<<endl;
     for (unsigned long long int i = 0; i < vec.size(); i++)
         cout << vec[i] << " "<<endl;
```

```
12
12
12
24
size of vector =7
```

4. Time Difference

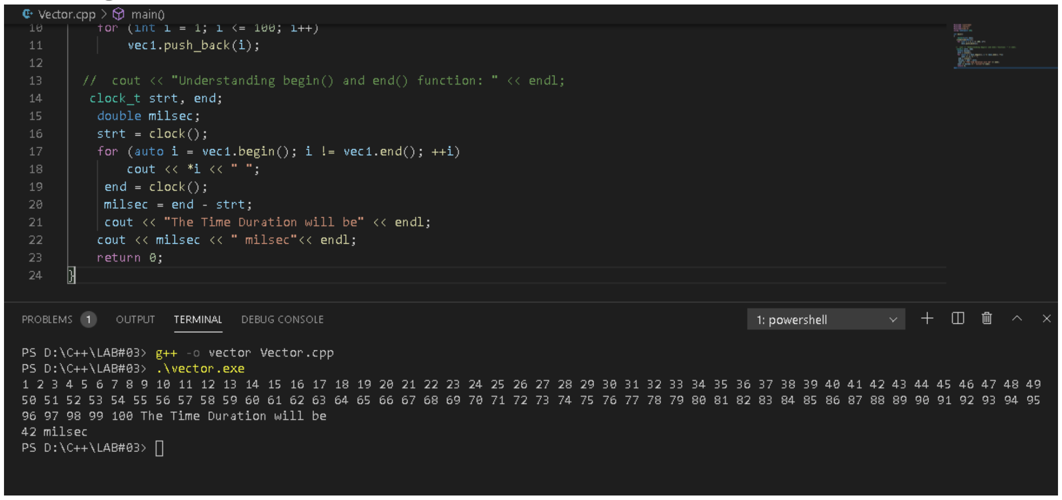
Arrays take less time accessing values in memory. It is faster and more efficient while doing this.



```cpp
#include<time.h>
#include<vector>
using namespace std;
int main()
{
    srand(time(NULL));
    clock_t strt, end;
    double milsec;
    int arr1[100];
    for(int c=0;c>100;c++)
    {
        arr1[c]=c;
    }
    strt = clock();
    for(int b=0;b<100;b++)
    {
```

```
PS D:\C++\LAB#03> .\lab.exe
1875946976 0 1875893184 0 1875947560 0 6421704 0 1875947560 0 1875855620 0 1875947768 0 1875865606 0 1875947552 0 1875746568 0 2 0 1875937
088 0 15400960 0 1875947560 0 1875942272 0 1875865824 0 22 0 1875208332 0 15401792 0 268501009 0 15402848 0 -1762415994 32764 4206152 0 0
0 4200027 0 16 0 0 0 0 0 8 0 4199705 0 4206152 0 4206144 0 15406800 0 16 0 0 0 4200107 0 4199744 0 8 0 0 0 268501009 0 15402856 0 -1762415
994 32764 8 0 0 0 4200208 0 24 0 0 0 0 0 8 0 4199705 0 The time will be=41
PS D:\C++\LAB#03>
```

Vectors don’t have an index therefore they are less efficient and more time consuming.



```cpp
    for (int i = 1; i <= 100; i++)
        vec1.push_back(i);

  //  cout << "Understanding begin() and end() function: " << endl;
   clock_t strt, end;
    double milsec;
    strt = clock();
    for (auto i = vec1.begin(); i != vec1.end(); ++i)
        cout << *i << " ";
     end = clock();
     milsec = end - strt;
     cout << "The Time Duration will be" << endl;
    cout << milsec << " milsec"<< endl;
    return 0;
}
```

```
PS D:\C++\LAB#03> g++ -o vector Vector.cpp
PS D:\C++\LAB#03> .\vector.exe
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95
96 97 98 99 100 The Time Duration will be
42 milsec
PS D:\C++\LAB#03>
```