LAB REPORT # 03

Data Structure and Object Oriented Programming

Name: M. Faisal Anwar Khan

Reg. Number: 325252

Degree: MTS-41

Syn: A

Date: 17/11/2020

Instructor: Ma'am Arshia Arif

LAB REPORT # 03

Differences between Vectors and Arrays:

1 Initialization:

Arrays are initialized as

```
int array1[10];
```

Vectors are initialized as

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

2 Indexing:

- Vector is a sequential container to store elements and not index based.

In vectors we can enter value from start one by one till end or we can enter values from end till start one by one. We cannot enter value randomly in between the values.

```
18      // inserts 24 to the last position
19      vec.push_back(24);
20      int s = vec.size();
21      cout << "The last element is: " << vec[s - 1]<<endl;
22      cout<<vec[s]<<endl;
23
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL                    1: powershell
The last element is: 24
0
The vector elements after push back are:
12
12
12
12
12
12
12
24
```

Like in above code , we can see that by using push_back command we enter values from end one by one until we reach start . if we want to enter a single value in any special single declared places then we can't enter directly.

```
    // 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;
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL                    1: powershell
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
```

Similarly using vec.emplace function we can enter values both from front and back but we should mention that. In spite of that we cannot enter value in between anywhere.

In contrast Array,

Array has capability that we can enter elements in array based on index number,

We can change a particular index element or we can enter a specific number on specific index number.

```
#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;
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE                    1: powershell
11
Enter value
5
8
0
4199705
0
8
0
22
0
6952656
0
PS D:\C++\LAB#03>
```

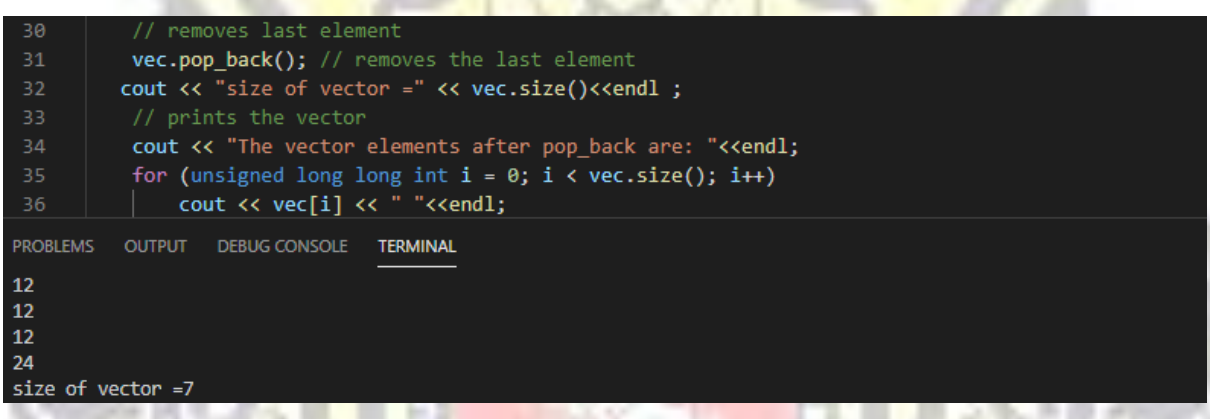
3 Length:

In vector we can change size of array at any time .if initially our size is 7 and we want to enter number on place that requires 8 place so vector increase size and allow us to enter value there. It has dynamic memory .

```
    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
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL                    1: powershell
PS D:\projects\vectors> .\vector.exe
The vector elements are:
12
12
12
12
12
12
12
size of vector =8
```



```
    // 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;
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
12
12
12
24
size of vector =7
```

While in Array,

Array has fixed size once initialized in beginning of program, it can't be changed anywhere else in program called as static memory allocation.

4 Time

In Vectors Accessing elements is time consuming.

```cpp

int main()
{
    vector<int> vec1;
   srand(time(NULL));
    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;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL                    1: powershell

PS D:\projects\vectors> g++ -o vector vectors.cpp
PS D:\projects\vectors> .\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 5
6 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
48 milsec
PS D:\projects\vectors>
```


While Arrays are not time consuming .They take less time to access any element.

```cpp
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++)
    {
        cout<<arr1[b]<<" ";
    }
    end = clock();
    milsec = end - strt;
    cout<<"The time will be="<<milsec<<endl;
    return 0;
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL                    1: powershell

PS D:\projects\vectors> g++ -o array Array.cpp
PS D:\projects\vectors> .\array.exe
1875946976 0 1875893184 0 1875947560 0 6421704 0 1875947560 0 1875855620 0 1875947768 0 1875865606 0 1875947552 0 1875746568 0 3 0 1875937088 0 14811136 0 18
75947560 0 1875942272 0 1875865824 0 30 0 1875208332 0 14811976 0 268501009 0 14813024 0 -2010994042 32767 4206152 0 0 0 4200027 0 16 0 0 0 0 0 8 0 4199705 0
 4206152 0 4206144 0 14816992 0 16 0 0 0 4200107 0 4199744 0 8 0 0 0 268501009 0 14813032 0 -2010994042 32767 8 0 0 0 4200203 0 24 0 0 0 0 0 8 0 4199705 0 Th
e time will be=25
PS D:\projects\vectors>
```