**Assignment No. 6**

**AIM:- Create a class template to represent a generic vector ,including**

**1. To create the vector**

**2. To modify the value of a given element**

**3. To multiply by a scalar value**

**4. To display the vector in the form (10,20,30…)**

**OBJECTIVE**:-

Vectors are sequence containers that have dynamic size. In other words, vectors are dynamic arrays. Just like arrays, vector elements are placed in contiguous storage location so they can be accessed and traversed using iterators. To traverse the vector, we need the position of the first and last element in the vector which we can get through **begin()** and **end()** or we can use indexing from 0 to **size()**. Let us see how to construct a vector.

**Theory** :-

### A class template by itself is not a type, or an object, or any other entity. No code is generated from a source file that contains only template definitions. For any code to appear, a template must be instantiated: the template arguments must be provided so that the compiler can generate an actual class (or function, from a function template).

### Vectors are same as dynamic arrays with the ability to resize itself automatically when an element is inserted or deleted, with their storage being handled automatically by the container. Vector elements are placed in contiguous storage so that they can be accessed and traversed using iterators. In vectors, data is inserted at the end. Inserting at the end takes differential time, as sometimes there may be a need of extending the array. Removing the last element takes only constant time because no resizing happens. Inserting and erasing at the beginning or in the middle is linear in time.

**Source code :-**

/\*

Create a class template to represent a generic vector. Include following member functions:

1]To create the vector.

2]To modify the value of a given element.

3]To multiply by a scalar value.

4]To display the vector in the form (10,20,30.....)

\*/

#include<iostream>

using namespace std;

template<class T>

class vector

{

T \*a;

int size;

public:

vector(int m) // parameterized constructor to create NULL vector

{

a=new T[size=m];

for(int i=0;i<size;i++)

a[i]=0;

}

void create()

{

for(int i=0;i<size;i++)

{

cout<<"a["<<i<<"] = ";

cin>>a[i];

}

}

void modify()

{

int pos;

cout<<"enter position to make changes :";

cin>>pos;

cout<<"Enter new Value :";

cin>>a[pos];

}

void multiply()

{

T sc;

cout<<"Enter scaler Number to multiply with vector : ";

cin>>sc;

for(int i=0;i<size;i++)

a[i]=a[i]\*sc;

}

void display()

{

int i;

cout<<"(";

for(i=0;i<size-1;i++)

{

cout<<a[i]<<",";

}

cout<<a[i]<<")\n";

}

};

int main()

{

int size;

cout<<"enter size of vector: ";

cin>>size;

vector<int> vi(size); //creates int vector

vi.create();

vi.display();

p:int choice;

cout<<"\nEnter your choice\n~press 1 to MODIFY\n~press 2 to MULTIPLY\n";

cin>>choice;

switch(choice)

{

case 1: vi.modify();

vi.display();

break;

case 2: vi.multiply();

vi.display();

break;

default: cout<<"ERROR! Enter correct choice\n";

break;

}

char ch1;

cout<<"\nDo you want to continue? (~press Y/N) : ";

cin>>ch1;

cout<<"\n";

if(ch1=='y'||ch1=='Y')

goto p;

return 0;

}

**Output:-**

/\*output:

enter size of vector: 4

a[0] = 2

a[1] = 6

a[2] = 12

a[3] = 9

(2,6,12,9)

Enter your choice

~press 1 to MODIFY

~press 2 to MULTIPLY

1

enter position to make changes :2

Enter new Value :42

(2,6,42,9)

Do you want to continue? (~press Y/N) : y

Enter your choice

~press 1 to MODIFY

~press 2 to MULTIPLY

2

Enter scaler Number to multiply with vector : 2

(4,12,84,18)

Do you want to continue? (~press Y/N) : n

\*/

**Conclusion:-**

Vectors allows our code to be more reusable, solid, and robust. If we take advantage of STL, you can make your life efficient through simplicity. Vector is also extensible, letting you add your own containers and algorithms.  
  
Learning to use vector is no small task. The C++ standard has a wide range of concepts and methodologies that can be derived from the library.