

# FPS-2

# VECTORS

---

Introduction, Common functions,  
Searching and Sorting

# VECTORS

- A vector is very similar to an Array, but the only difference is that vector's size is dynamic.
- Vector has the ability to resize itself automatically when an element is inserted or deleted, with its storage being handled automatically by the container.
- Vectors are used to store elements of similar data types.

# *Vectors vs Arrays*

- In the vector representation, there are additional empty slots available for future elements. This demonstrates the ability of vectors to dynamically adjust their size to accommodate new elements without manual resizing, which is not possible in case of arrays .
- Vectors can be used for any data type unlike arrays.

# VECTORS VS ARRAYS

Arrays:

```
+---+---+---+---+---+
| 1 | 2 | 3 | 4 | 5 |
+---+---+---+---+---+
```

Vectors:

```
+---+---+---+---+---+---+---+---+---+
| 1 | 2 | 3 | 4 | 5 |   |   |   |
+---+---+---+---+---+---+---+---+---+
```

Vectors offer flexibility in terms of adding or removing elements. The empty slots in the vector representation signify that elements can be inserted or deleted at any position without disrupting the existing elements, whereas arrays have a fixed structure, and modifying their size requires shifting or rearranging elements manually.

# *DECLARATION*



```
#include <bits/stdc++.h>
using namespace std;
```

```
int main(){
    vector<int> vec;
}
```



```
vector<int> vec;
vector<char> vec_char;
vector<float> vec_float;
```

```
//we can make a vector of
//vectors as well!
vector<vector<int>> vec_of_vec;
```

## *COMMON FUNCTIONS:* push\_back

Inserts an element  
x at the end of the  
vector



```
vec.push_back(x);
```

## pop\_back

Removes the last  
element of the  
vector



```
vec.pop_back();
```

## EXAMPLE

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

```
vec.push_back(9);
```

```
vec.push_back(2);
```

```
vec.push_back(7);
```

```
vec.pop_back();
```

```
vec.push_back(8);
```

```
vec.push_back(2);
```

```
vec.pop_back();
```

## EXAMPLE

9
---

9	2
---	---

9	2	7
---	---	---

9	2
---	---

9	2	8
---	---	---

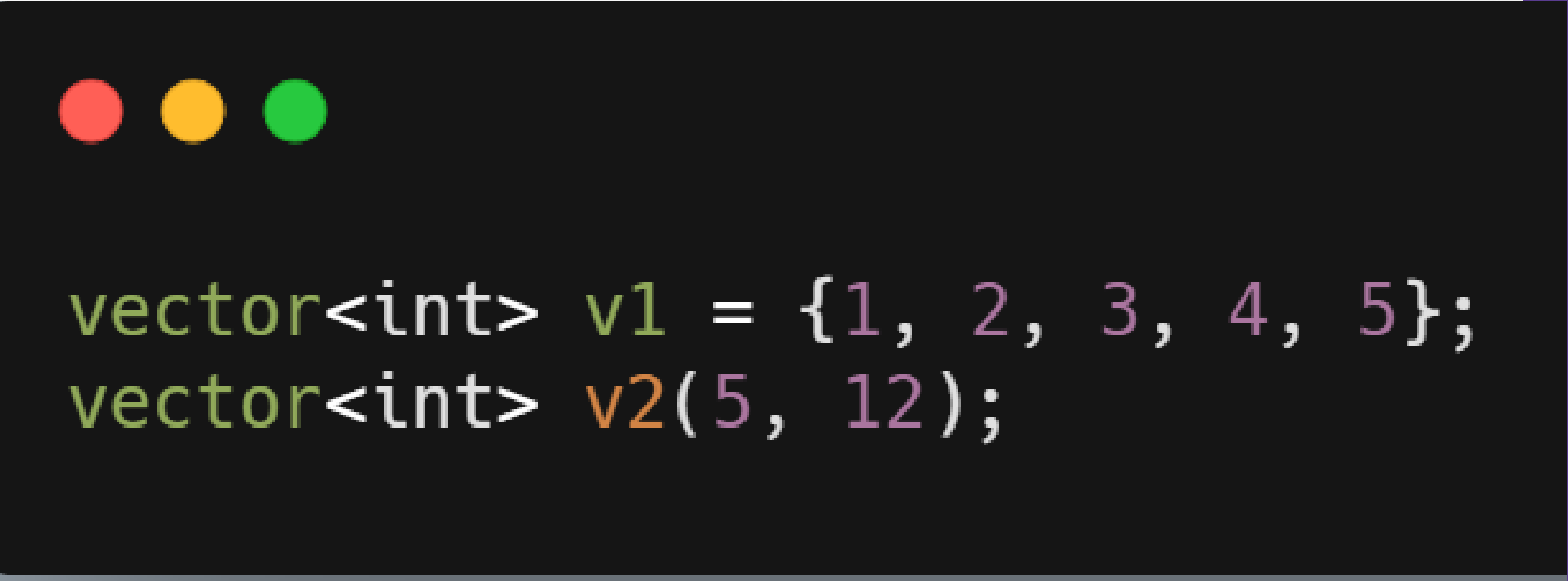
9	2	8	2
---	---	---	---

9	2	8
---	---	---

```
vector<int> vec;  
vec.push_back(9);  
vec.push_back(2);  
vec.push_back(7);  
vec.pop_back();  
vec.push_back(8);  
vec.push_back(2);  
vec.pop_back();
```



# INITIALIZATION



```
vector<int> v1 = {1, 2, 3, 4, 5};  
vector<int> v2(5, 12);
```

- Vector v1 is initialized by providing values directly to the vector. Now, contents of vector v1 are 1,2,3,4,5.
- v2: 5 is the size of the vector and 12 is the value. A vector of size 5 is created and initialized with the value of 12. So, the vector v2's contents are 12,12,12,12,12.

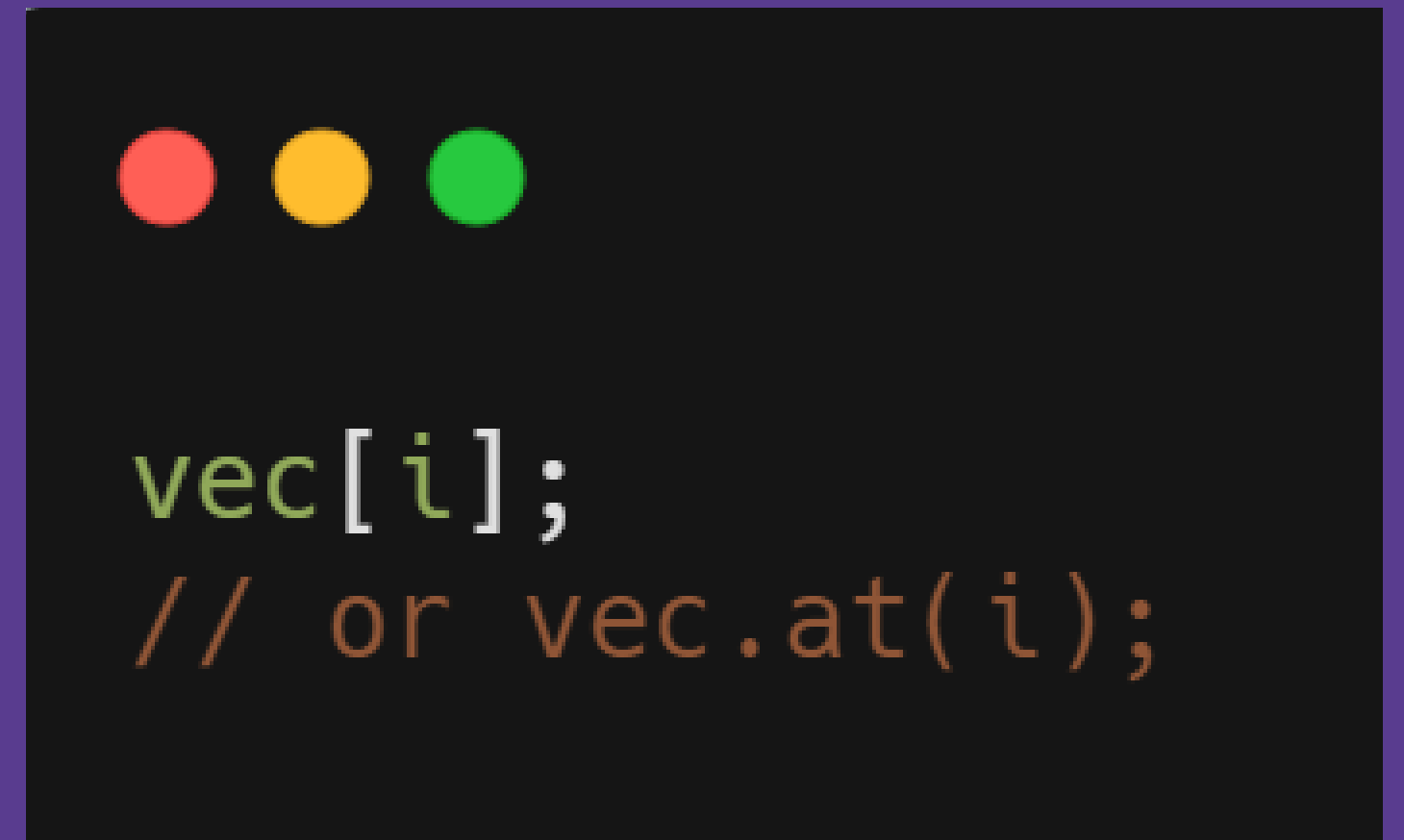
# MORE FUNCTIONS



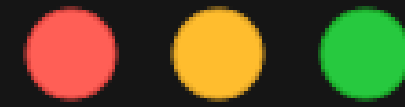
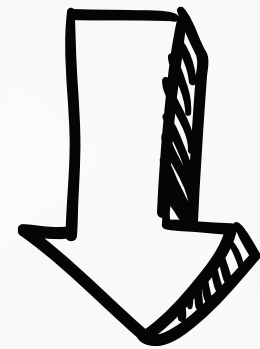
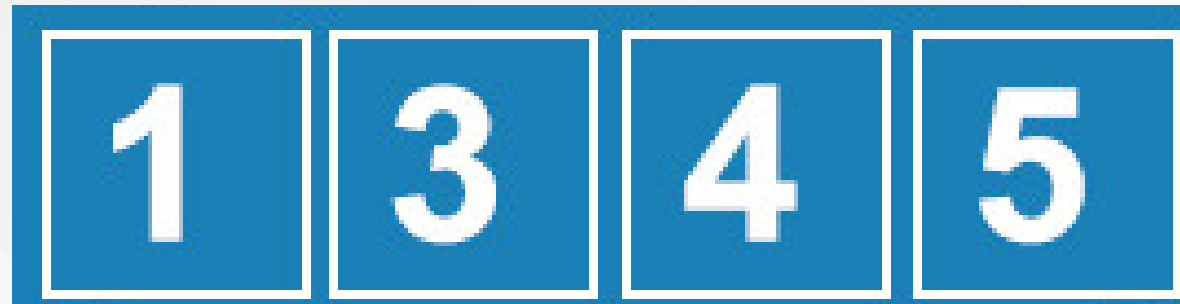
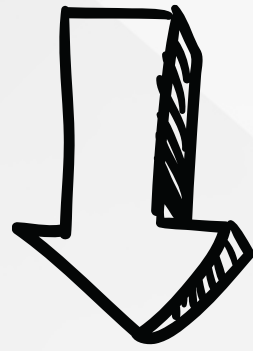
Returns the number of elements  
in the vector

Returns the element at (i+1)th  
position in the vector

	9	2	7	8	2
Index :	0	1	2	3	4
Position :	1	2	3	4	5



- **vec.clear()** : Erases the vector vec
- **vec.erase(vec.begin()+k)** : Removes the element at k+1th position, shifting the remaining elements.
- **vec.insert(vec.begin()+k,x)** : inserts an element x at the k+1th position of vector vec, shifting all the elements with index  $\geq k$  by 1 position to the right.
- **vec1.swap(vec2)** : swaps the contents of vectors vec1 and vec2.
- **vec.front()** : Returns the first element of vector
- **vec.back()** : Returns the last element of vector



```
vector<int> v{1,3,4,5};
```

```
//inserts 7 at 3rd position
```

```
v.insert(v.begin()+2,7);
```

```
//erases the 4th element
```

```
v.erase(v.begin()+3);
```

# SEARCHING

Suppose you have a number  $x$ . You want to find out whether  $x$  is present in a given vector  $v$  or not (if yes, at which position?). How do you implement this?

# SEARCHING

Suppose you have a number  $x$ . You want to find out whether  $x$  is present in a given vector  $v$  or not (if yes, at which position?). How do you implement this?

Let  $x$  be called 'key'. Traverse the vector till you find the key.

- When you are at the  $i$ th element of the vector, compare that element with key.
- If it does not match, go to the next element. If it matches, you have found the key.



```
int key= 25;
vector<int> v{10,34,54,78,0,25,67};

//linear search
for(int i=0;i<v.size();i++){
    if(v[i]==key){
        cout<<"key has been found at position "<<i+1<<endl;
        break;}
}
```

As we traverse the elements of the vector in a linear fashion, this is called '**linear search**'.

There is a function 'find' which helps us achieve the same.

Other types of searching algorithms such as binary search also exist.



# An alternate way to write the 'for' loop

- Range-based for loop executes a for loop over a range.
- The auto keyword specifies that the type of the variable that is being declared will be automatically deducted from its initializer
- Used as a more readable equivalent to the traditional for loop operating over a range of values, such as all elements in a container.



```
int key = 25;
vector<int> v{10, 34, 54, 78, 0, 25, 67};

int index = 0;
for (auto element : v) {
    if (element == key) {
        cout << "Key has been found at position " << index +
1 << endl;
        break;
    }
    index++;
}
```



# SORTING

- Sorting refers to arranging data in a logical fashion.

# SORTING

- Sorting refers to arranging data in a logical fashion.
- A vector is said to be sorted when its elements are in non-increasing (or non-decreasing) order. To sort a vector, we use various techniques or sorting algorithms (like bubble sort, insertion sort, quick sort, radix sort, etc.)
- Applications: to calculate the  $n$ th largest or smallest value in a vector, etc.

# SORT FUNCTION



```
// sorting elements of vector vec in increassing order  
sort(vec.begin(),vec.end());
```

```
//sorting elements of vector vec in decreasing order  
sort(vec.begin(),vec.end(),greater<int>())
```

```
vector<int> v={1,5,2,4,3};

for(int i=0;i<5;i++){
    v.push_back(i+6);
}

cout<<"size of v: "<<v.size()<<endl;

for(int i=0;i<3;i++){
    v.pop_back();
}

cout<<"size of v: "<<v.size()<<endl;
//printing vector
for(int i=0;i<v.size();i++){
    cout<<v[i]<<" ";
}

cout<<endl;
sort(v.begin(),v.end());
//printing vector
cout<<"after sort: ";
for(int i=0;i<v.size();i++){
    cout<<v[i]<<" ";
}
cout<<endl;
```



# EXAMPLE

```
vector<int> v={1,5,2,4,3};

for(int i=0;i<5;i++){
    v.push_back(i+6);
}

cout<<"size of v: "<<v.size()<<endl;

for(int i=0;i<3;i++){
    v.pop_back();
}

cout<<"size of v: "<<v.size()<<endl;
//printing vector
for(int i=0;i<v.size();i++){
    cout<<v[i]<<" ";
}

cout<<endl;
sort(v.begin(),v.end());
//printing vector
cout<<"after sort: ";
for(int i=0;i<v.size();i++){
    cout<<v[i]<<" ";
}
cout<<endl;
```

# OUTPUT

size of v: 10

size of v: 7

1 5 2 4 3 6 7

after sort: 1 2 3 4 5 6 7

# PRACTICE PROBLEMS

- Vanya and Fence

**Problem - 677A**

Codeforces. Programming competitions and contests, programming community

 Codeforces

- Medium Number

**Problem - 1760A**

Codeforces. Programming competitions and contests, programming community

 Codeforces

# PRACTICE PROBLEMS

- Array Rearrangement

**Problem - 1445A**

Codeforces. Programming competitions and contests, programming community

 Codeforces

***THANK YOU***