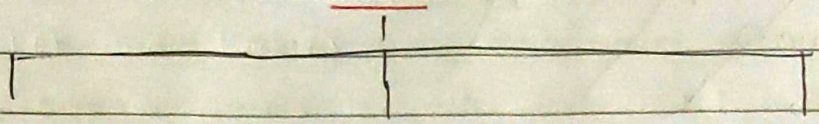
DATE

STL



1) Containers.

- Collection of classes.
- implemented as generic class templates etc
- helps us to implement and replicate simple and complex data structures very easily.
- Can be used to hold different kind of objects.

Common containers -

① vectors
② queue
③ stack
④ priority_queue
⑤ list
⑥ set
⑦ map.

DATE

★ Array class

- Array is a linear collection of similar elements.
- Array container in STL provides us the implemetation of static array
- Use header array
  - # include <array>

- Creating array object :

```
array < obj_type , array_size > array_name ;
```

It creates an empty array of obj-type with max size of array-size.

```
array<int, 4> EvenNumber = {2,4,6,8};
```
— Example

- Member functions

1. **at**
   - The method returns value in the array at given rage.
   - If the given range is greater than the size of array, exception is thrown.

   Ex:

```
cout << EvenNumber.at(2);      output : 6
cout << EvenNumber.at(5);      ERROR
```

2. **[]**
   - Use is same as normal array.

   Ex:

```
cout << EvenNumber[2]          output : 6
```

DATE

3. Front() and back()

- Front() returns ther first element.
- Back() returns the last element.

Ex:

```
cout << EvenNumber.front() << EvenNumber.back()
```

2, 8.

4. Fill()

- This method assigns the given value to every element of array.

Ex:

```
EvenNumber.fill(1);
Cout << EvenNumber;
```

1 1 1 1

5. Size()

- This returns the size of array.

Ex:

```
cout << EvenNumber.size() ;
```

4

6. begin() and end()

- begin() returns the iterator pointing to first element
- end() -"- — "- — next to last elem

DATE

★ Pair class

- Pair is a template class in STL.
- Pair is not a part of container.
- Syntax for using pair object

```
pair < T1 , T2 > pair1 ;
```

Ex:

```
pair < int , string > p2 ;
```

- Inserting value

```
p2 = make_pair ( 16 , "Rahul" ) ;
```

- Accessing pair members

```
cout << p2.first          // 16
cout << p2.second         // Rahul
```

- We can compare two pairs

```
==
!=
<
>
<=
>=
```

} All these are applicable.

PAGE

classmate

DATE

★ Tuple Class

- Just like in pair, we can pair two heterogeneous object, in tuple we can pair multiple objects.

Example :

```
tuple <string, int, int> t1 ;
t1 = make_tuple ("India", 16, 10) ;
cout << get <0> (t1)
cout << get <1> (t1)
cout << get <2> (t1)
```

- Making
- Inserting values

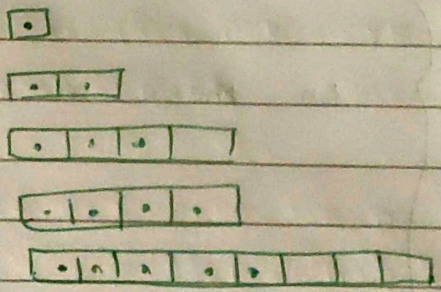
} Accesing values

We can compare two tuples

★ Vector Class

- The most general container
- Supports a dynamic array

capacity : 1 → 2 → 4 → 8. Size doubles



(by default when obj is made cap is 1)

DATE

- `vector <int> V1` — blank vector.
- `vector <int> V1 {1,2,3}` — Declaration of vector. In this case, capacity will go on like 3 → 6 → 12 → 24.
- `vector < int> V2 (4);` — vector of size 4.
- `vector < string> V3 ( 3, "hello");` — vector of size 3 with hello in each block.
- Subscript operator '[]' is also defined for vector.

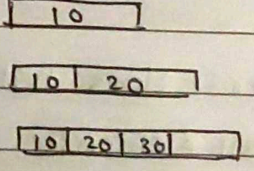
```
Ex:  cout << V3 [0]   // hello
     cout << V3 [1]   // hello
     cout << V3 [2]   // hello
```
} output

1. push_back ()

It is a member function to add value to the vector at end.

Ex:

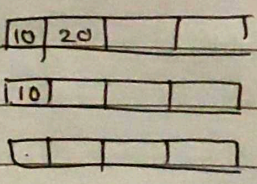
```
vector <int> V1;
V1. pushback (10);
V1.pushback (20);
V1. pushback (30);
```



2. pop_back ()

It removes the last element
capacity doesn't decrease

```
V1. pop-back ();
V1. ppp-back ();
V1. ppp-back ();
```



DATE

3. Capacity() and size()

capacity returns no. of element vector can hold
size returns no of elements present.

4. Clear()

Removes all elements

5. Front() and back()

returns first and last value. resp.

* List class

- supports bidirectional linear list
- vector supports random access but list can only be accessed sequentially.
- Creating list object.

```
list <int> L1 {11, 12, 13}
list <string> L2 { "India", "Pakistan"}
```

- [] operator does not work here.
- To print all elements. (use iterator)

```
list < int > :: iterator p= L1.begin() ;
while (p != L1.end())
{
cout << *p ;
p++ ;
}
```

DATE

member functions.

Ex: 5 3 2 1 4

1. push-back ();
2. push-front ();
3. pop-back ();
4. pop-front ();
5. sort ();
6. reverse ();
7. remove (4);
8. clear ();

added at the end

added at the front.

5 3 2 1

3 2 1 4

1 2 3 4 5

4 1 2 3 5

5 3 2 1

List is cleared.

* ~~String~~ Map Class

- Used to replicate associative arrays.
- Maps contain sorted Key-Value pair, in which each key is unique and it can be inserted or deleted but can not be altered.
- Value associated with key can be altered.

Customer No.	Customer Name
100	Gajendra
123	Dilip
145	Aditya.

```
map < int, string > customer;
customer [100] = "Gajendra";
Customer [123] = "Dilip";
Customer [145] = "Aditya";
```

DATE

★ String Class.

- String is another container class.
- need to include string header.

```
# include <string>
```

- Constructors

  string class supports many constructors

  eg :
  - string()
  - string (const char *str)
  - string (const string &str).

- Operators

=	!=	>=	} all these can be used.
+	<	[]	
+=	<=		
==	>		

- Insertion and extraction operators can be used

```
string s1;
cin >> s1;
cout << s1;
```

- You can mix string objects with another ~~e~~ of string objects or character array.

0 1 2 3 4
H e l l o

DATE

1) assign () S1 = "Hello" OR S1.assign("Hello")

2) append () S2 = S1 + "1" OR S2 = S1.append("1")
↳ output Hello1

3) insert () S1.insert(2, "123") He123llo
index no, inserting value.

4) replace () S1.replace(2, 2, "A") HeAo
index no, length

5) S1.replace(1, 3, "A") HAo

5) erase () S1.erase() No output

6) find () int i = S1.find("llo") i = 2 (Hello)

7) compare ()

```
string S1 = "Amar";
string S2 = "Amit";
int i = S1.compare(S2);
cout << i
```

output -1
not same.

There are three results of compare

	0	-1	1
	same	different (in dictionary order)	Not in dict order
S1	amit	amar	amit
S2	amit	amit	amar

classmate

PAGE