Date:
Sat Sun Mon Tue wed Thu Fri

Theme: Standard Template Library

* 1 Library for all → <bits/stdc++.h>
* To shorten :: (std::cin >>a); we use 'using namespace std;'

1. Functions

ⓐ Void
→ it will not return us anything.

ⓑ Return Type function
→ int sum(int a, int b){ return a+b;}
→ int main(){ int a,b;
or just, int s = sum(1,5);
cout << s;
}

can be any datatype

STL is divided into 4 parts.

→ Algorithms, containers, functions, iterators
Learning this

Pairs

→ Part of utility library
→ store couple of "Datatypes"
→ How? → pair <int, int> P = {1,3};
Declaration → cout << p.first << " " << p.second;
for 3 variables; pair <int, pair<int,int>> P = {1,{3,4}}
cout << P.first << P.second.first << P.second.second;
1 3 4

Theme: Date: / /
Sat Sun Mon Tue wed Thu Fri

→ what if we can declare pair array

pair <int, int> arr [] = { {1,2}, {2,5}, {5,2} };
index: 0 1 2
cout << arr[1].second;

4) Vectors
→ we cant modify size of a previously declared array.
→ vector is a dynamic array (container)

⇒ Declaration : vector <int> v; → name
↳ Dtype → { }

⇒ add elements: v.push_back(1); → {1, }
v.emplace_back(2); → {1, 2}

dynamically increases size and adds elem

5) Pair : vector <pair <int, int>> vec; → { }
vec.push_back({1, 2}); → {{1,2}, }
vec.emplace_back(1,2);

Automatically gets that its a pair. so take {1,2}

6) Predefined Vectors
5 instances of 100
vector <int> vac(5, 100) → {100, 100, 100, 100, 100}
or, vector <int> vect(5) → {-, -, -, -, -}

BD SCHOOL SHOP

Theme:

Date: / /

Sat Sun Mon Tue wed Thu Fri

⑤ Copying vectors

```
vector <int> v1 (3, 20);
vector <int> v2 (v1)
```

Accessing elements on a vector:

① → Array like : v[0], v[1]

* ② Iterator :

→ v { 1, 2, 3, 4 }

Iterator declaration:

```
vector <int>:: iterator name = v.begin();
```

where 1 is staying

→ Points to the memory address

→ it ++ ; goes to '2' → shifted memory

→ cout << *(it) << " ";

↳ accessed '2'

same way, it = it + 2;

cout << *(it) << " ";

↳ prints 4

* vector <int>:: iterator it = v.end();

points to some-thing just after the last element;

Theme:

Date: / /

Sat Sun Mon Tue wed Thu Fri

* v.rend();
→ {1, 2, 3, 4, 5}

Not used

* v.rbegin();
→ {1, 2, 3} → points to 3

but, it++ means it will move to 2;

Print

* v[0] → prints 1 , v.at(0) → prints 1
* v.back() → prints 3.

Printing vector

a {10, 20, 30}

```
for(vector<int>::iterator it = v.begin(); it != v.end(); it++)
{ cout << *(it) << " "; }
```

short cut

```
for (auto it = v.begin(); it != v.end(); it++)
{ cout << *(it) << " "; }
```

auto defines datatype

For each loop

```
for (auto it : v) { cout << it << " "; }
```

int

Theme:

Date: / /

Sat Sun Mon Tue wed Thu Fri

Delete in vector

v.erase(iterator)

{10, 20, 21, 22}

single: v.erase(v.begin()+1); → ~~20~~ & reshuffle

multiple: v.erase(v.begin()+2, v.begin()+1)

[start, end) → after what I want to delete

Insert function

vector<int> v(2,100) → {100,100}

v.insert(v.begin(), 300) → {300,100,100}

v.insert(v.begin()+1, 2, 10) {300,10,10,100,100}

How many numbers ↙ ↘ what I include

vector<int> copy(2,50); // {50,50}

v.insert(v.begin(); copy.begin(); copy.end());

---

v.size() → // How many elements

v.pop_back → // deletes last ele~

v1.swap(v2) // v1 → {30,40}, v2 → {10,20}

v.clear() → Is your vector empty?

→ 1 → false

2 → True

Theme:

Date: / /

Sat Sun Mon Tue wed Thu Fri

List

→ same as vectors but slightly cheaper

```
list<int> ls;
ls.push_back(2);        {2}
ls.emplace_back(4);     {2,4}
ls.push_front(1);       {1,2,4}   ⟶ cheaper than v.insert()
ls.push_front(1);       {1,2,4}
ls.emplace_front();     {2,4};
```

all other functions are similar to vector

Deque

→ same vec, list

→ deque<int> dq;

```
dq.push_back(1);
dq.emplace_back(2);
dq.push_front(3);       {3,1,2}
dq.emplace_front(3);    {3,4,1,2}

dq.pop_back();          {3,4,1}
dq.pop_front();         {4,1}

dq.back();
dq.front();
```

Theme:

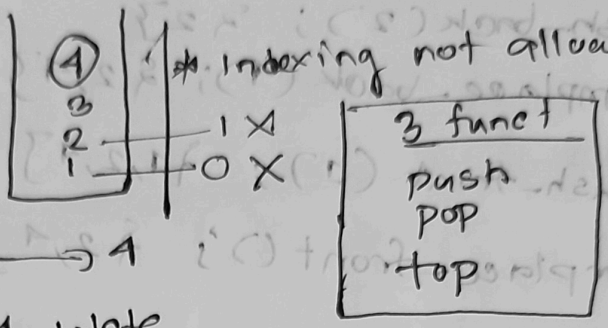
Date: / /

Sat Sun Mon Tue wed Thu Fri

Stack

LIFO → Last in first out

```
stack <int> st;
st.push(1)
  .push(2)
  .push(3)
  .emplace(4)
```




```
cout << st.top(); ⟶ 4
st.pop(); ⟶ 4 delete
st.size(); // 3
st.empty(); ⟶ false
st.swap();
```

Queue ⟶ FIFO → first in, first out

```
queue <int> q;
q.push(1);
q.push(2);
q.emplace(3);
q.back()+=5; ⟶ (1, 2, 8)
cout < q.back() → 8
cout < q.front() ⟶ 1
q.pop() → (2, 8)
cout < q.frontA() ⟶ 2
```

Theme: Date: / /

Sat Sun Mon Tue wed Thu Fri

4. Priority Queue

→ max elem stays at top

```
priority_queue <int> pq;
pq.push(5);
pq.push(2);
pq.emplace(8);
pq.top() ⟶ 8
```

8
5
2

Minimum order / heap

```
priority_queue <int, vector<int>, greater<int>> pq;
pq.push(5);
pq.push(2);
push(1);
```

1
2
5

(*) Push → log N
top → O(1)
pop → log N

5. Set

→ stores unique, everything in sorted order.

```
set <int> st;
st.insert(1)
  .insert(2)
  .emplace(2)
  .insert(4)
```

4
2
1

```
auto it = st.find(2);
   ↳ iterator points at 2
auto it = st.find(45);
   ↳ points at afterward
```

```
st.erase(4);
```

```
int cnt = st.count(1);
```

if have → 1
if dont have → 0

st.lower_bound(2)
st.upper_bound(3)

compl: log( )

{1, 2, 3, 4, 5}

```
auto it1 = st.find(2)
    it2 = st.find(4);
st.erase(it1, it2) ⇒ {1, 4, 5}
```

range [st, end)

BD SCHOOL SHOP
ONE STOP SCHOOL SHOPPING

Theme:

Date: / /

Sat Sun Mon Tue wed Thu Fri

Multiset

→ obeys sorted & duplicate elements also

```
myMiset<int> ms;
ms.push(1);        Not push → insert
  .push(1);
  .push(1);   {1,1,1}
ms.erase(1); // all 1's are erased

ms.erase(ms.find(1));
        → erases only that portion

ms.erase(ms.find(1), ms.find(1)+2);
                      ↑
                  Go till 2
```

→ rest are same as set

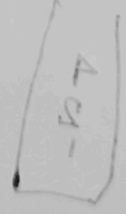
unordered set

→ doesn't store in a sorted order

→ Unique elements

→ O(1) worst, else O(N) But lower / upper bound doesn't work

```
unordered.set<int> st;
```



Theme:

Date: / /

Sat Sun Mon Tue wed Thu Fri

Map

→ same but different [roll numbers] (key → unique) and [Name] (value → can be similar)

→ stores everything in values and keys

→ Log(n)

→ map <key, value> name;

map → unique keys in sorted order

```
map <int, int> mpp;
map <int, pair<int, int>> mpp;
map < pair<int, int>, int> mpp;
```

mpp[1] = 2; // 1 is the key & 2 is the value

mpp.emplace({3,1});

mpp.insert({2,4});

{1,2}
{3,1}
{2,4}
→
{1,2} — begin
{2,4}
{3,1} — end

mpp[{2,3}] = 10;
key <> ↳ val

{2,3}, 10

Printing:

```
for(auto it : mpp) { cout << it.first << " " << it.second;
```

1	2
2	4
3	1

cout << mpp[1] → 2

cout << ~~mpp~~[5] → 0 → Null

auto it = mpp.find(3);

cout << *(it).second; (3), 1

auto it = mpp.find(5) → end()

Multimap

→ can store duplicate keys

unordered Map

→ Not store in sorted, unique keys

BD SCHOOL SHOP
ONE-STOP SCHOOL SHOPPING

Theme: Basic Algorithm

Date: / /

Sat Sun Mon Tue wed Thu Fri

Sort numbers (vectors & arrays)

A[4] = { 3, 2, 1, 4 }

sort (a, a+4)

first iterator → a

a+4 → last iterator

(start, end)

* if using vector,

sort (v.begin(), v.end());

* { 3, 1, 2, 4 } want to sort this part

sort (a+2, a+4)

* what if we want to sort in descending order?

sort (a, a+n, greater<int>())

→ Inbuilt comperator

* My Way

Let's say we have a pair,

→ pair<int, int> a[] = { {1,2}, {2,1}, {4,1} }

what if I want to sort it by second element, in order of increasing second e, if the s.ele is same the do it in first element but in descending order.

Theme:

Date: / /

Sat Sun Mon Tue wed Thu Fri

① → {2,1} {4,1} {1,2}

same → so first element ভিত্তে sort করব in desc. order

{4,1} {2,1} {1,2}

{ , } { , }
P_1 ≤ P_2

My correct order

My comp (How to make ?) :)

```
bool comp (P1 copy past data type, P2 have couple of that) {
    if (p1.second < p2.second) return true;
    if (p1.second > p2.second) return false;
    if (p1.first > p2.first) return true;
    return false;
}
```

My incorrect order

---

* built in popcount

→ How many set bits or 1's are there.

int n = 7; ⟶ Binary : 111 ⇒ 3

int cnt = __builtin_popcount( );

long long myl = 1 2 3 9 5 .... ;

= __builtin_popcountll ( );

BD SCHOOL SHOP
ONE-STOP SCHOOL SHOPPING

Theme:

Date: / /

Sat Sun Mon Tue wed Thu Fri

* Next Permutation (combinations):

→ start in sorted fashion

```
string s= "123";
sort(s.begin(), s.end());
```

→ do { cout << s << endl;

```
} while(next_permutation(s.begin(), s.end()))
```

* Maximum & Minimum element

```
int maxi = *max_element(a, a+n)
int mini = *min_element(....)
```

End of STL

Base