

AUGUST 2008						
Mon	04	11	18	25		
Tue	05	12	19	26		
Wed	06	13	20	27		
Thu	07	14	21	28		
Fri	08	15	22	29		
Sat	09	16	23	30		
Sun	10	17	24	31		

July  
Tuesday



JUL

~~C++~~

## Vector class of STL

⇒ including & initialising vectors -

#include <vector>

    ↑ type → name

vector<int> arr(m, 0);

arr.resize(g); → resizing

    ↑ initialising value  
for all elements

arr.resize(g, 3);

    → size of vector

⇒ getting spaced inputs in C++.

int l, r.

cin >> l >> r;

⇒ size of vector → arr.size()

⇒ arr.push\_back(i) → append i at the last of vector

⇒ arr.pop\_back() → to remove an element  
from last

⇒ Pbds → policy based data structures

find-by-order(k)

    ↑ return the pointer  
to k<sup>th</sup> largest  
element  
 $k \in [0, \text{arr}]$

order-of-Key()

Appointment

To avoid any error "comparison b/w signed & unsigned  
int": use "size\_t" data type for storing indices

AUG

SEP

OCT

NOV

DEC

JULY			
Mon	07	14	21 28
Tue	08	15	22 29
Wed	09	16	23 30
Thu	10	17	24 31
Fri	11	18	25
Sat	12	19	26
Sun	13	20	27

02

July

Wednesday

→ 1<sup>st</sup> largest

1, 2, 3, 4

0<sup>th</sup> largest element → min<sup>imo</sup>

⇒ vector<vector<int>> vvi;

⇒ if (vvv)

{

continue;

}

[This statement won't run]

⇒ #include <bits/stdc++.h>

→ min(a, b)

→ max(a, b)

→ swap(a, b)

→ double pow(double base, double exponent)  
 ↳ all values are double type

→ cout << fixed << setprecision(10) << num << "m";

⇒ Global variables - defined on top

- initialised with zero (By default)

- visible everywhere

Local variables - defined in functions  
 - initialised randomly (garbage value)  
 - visible just in the function

Notes

Appointment

⇒ vector<vector<int>> vvi(5, vector<int>(10, 23));

• vector<int>(10, 23);

	AUGUST 2008				
	04	11	18	25	
Mon	05	12	19	26	
Tue	06	13	20	27	
Wed	07	14	21	28	
Thu	08	15	22	29	
Fri	09	16	23	30	
Sat	10	17	24	31	
Sun	11				

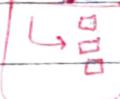
July

Thursday

03

→ we can rewrite global variables by declaring local variable in a function with same name as the local variable.

⇒ block of code - code between braces

⇒ codeblocks debugger -  go to the line then  
S press next line button.

watcher ≡ workspace in MATLAB

Q) Appearance array -

mode of - 1, 7, 7, 7, 3, 2, 5, 10

make an array of size 11 & increment each index when you read it.

⇒ stack is a list with a restriction that insertion and deletion can be performed only from one end, called top.

→ also called last in, first out Data Structure

→ push() - pushes a value on top   
of the stack

→ pop() - removes value from top   
of the stack

→ isEmpty() - returns true if stack is empty

→ top() → returns top value of a stack

04

July  
Friday

Mon	07	14
Tue	01	08
Wed	02	09
Thu	03	10
Fri	04	11
Sat	05	12
Sun	06	13
	20	27

→ stack - CPP

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

```
int stack[100];
int ind;
void push(int x) {
    ++ind;
    stack[ind] = x;
}
```

```
bool isEmpty() {
    if (ind >= 1) return false;
    else return true;
}
```

```
void pop() {
    stack[ind] = 0;
    ind--;
}
```

```
int top() {
    return stack[ind];
}
```

```
int main()
```

```
{
```

Notes      ind = 0

```
push(1);
push(2);
pop();
```

```
cout << top(); }
```

Appointment

04	11	18	25
05	12	19	26
06	13	20	27
07	14	21	28
01	08	15	22
02	09	16	23
03	10	17	24
Sun			31

July  
Saturday

05

AUG

SEP

C T

J N

C E

→ To find a given string of brackets is valid or not, use stack open & close brackets do push & pop. if stack is empty in end, strings are valid.

⇒ Queue - A queue is a list with the restriction that insertion can be performed only from one end (called back) & deletion can be performed only from the other end (called front)

→ operations - push() - inserts at the back of queue  
 - pop() - deletes element from back  
 - isEmpty() - Returns true if queue is empty  
 - front() - returns the value at front end

→ It is called first in, first out o/s.

07

July

Monday

JULY 2008						
Mon	07	14	21	28		
Tue	08	15	22	29		
Wed	09	16	23	30		
Thu	10	17	24	31		
Fri	11	18	25			
Sat	12	19	26			
Sun	13	20	27			

=> Queue - CPP

```
#include <bits/stdc++.h>
```

```
using namespace std;
```

```
int Queue[1000], backInd, frontInd;
```

```
void push(int x) {
```

```
    backInd++;

```

```
    Queue[backInd] = x;
}
```

}

```
void pop() {
```

```
    Queue[frontInd] = 0

```

```
    frontInd++;
}
```

}

```
bool isEmpty() {
```

```
    if (backInd < frontInd) return true;

```

```
    else return false;
}
```

}

```
int front() {
```

```
    return Queue[frontInd];
}
```

}

```
int main()
```

{

Notes      backInd = -1 ;

frontInd = 0 ;

push(5);

push(7);

}

Appointment

AUGUST		2008			
Mon		04	11	18	25
Tue		05	12	19	26
Wed		06	13	20	27
Thu		07	14	21	28
Fri	01	08	15	22	29
Sat	02	09	16	23	30
Sun	03	10	17	24	31

July  
Tuesday

08

AUG

SEP

C  
T

INDU

E  
C

=> Binarysearch .cpp

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

```
int A[1000], number of elements;
```

```
int binarySearch(int x) {
```

```
    int left = 0, right = number of Elements, mid;
```

```
    while (left <= right) {
```

```
        mid = (left + Right) / 2;
```

```
        if (x == A[mid]) return mid; // found
```

```
        else if (x < A[mid]) right = mid - 1;
```

```
        else left = mid + 1;
```

```
}
```

```
return -1;
```

```
}
```

```
int main()
```

```
{
```

```
    f >> numberElements;
```

```
    for (int i = 1; i <= no. of elements; i++)
```

```
        f >> A[i]
```

```
    cout << binarySearch(2);
```

```
    return 0;
```

```
}
```

09

July

Wednesday

A[7] = [1, 2, 5, 3]

JULY		2008				
		07	14	21	28	
Mon		01	08	15	22	29
Tue		02	09	16	23	30
Wed		03	10	17	24	31
Thu		04	11	18	25	
Fri		05	12	19	26	
Sat		06	13	20	27	

=> #include <bits/stdc++.h>

sort(A+0, A+4)

↓      ↓ , last index + 1  
1st index

=> to sort in descending order

bool cmp(int x, int y)

{

return x > y;

}

main ↓

sort(A+0, A+4, cmp);

=> #include <string>

int sum = 5;

string sum = to\_string(sum);

#include <array>

array<int, 5> a = {0, 1, 2, 3, 4}

for (int i = a)

{     a[i] = a[i] + 1; i = i + 1

Notes

for {  
int i = a;  
} cout << a << endl;

Appointment

AUGUST 2008						
Mon	04	11	18	25		
Tue	05	12	19	26		
Wed	06	13	20	27		
Thu	07	14	21	28		
Fri	01	08	15	22	29	
Sat	02	09	16	23	30	
Sun	03	10	17	24	31	

July  
Thursday

10

## NPTEL

### String class of STL

=> # include < string >

string v = "abcdab";

string w(v) // equal to w = v

string x = v + w

v[2] = v[3]; // v = "abddab" now

cout << v.substr(2); // substring "ddab"  
v.substr(1, 3)

↳ starts at 1 of length 3

int i = v.find("ab"); // position of 1<sup>st</sup> occurrence  
· find("ab", 1);

↳ starts for searching  
from given index

=> if string find to "doesn't find  
anything, it returns a string  
"string::npos"

=> for reading a sentence

# include < bits/stdc++ >

char c[100];

cin.getline(c, 100);

Notes => vector<int> v(10)

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

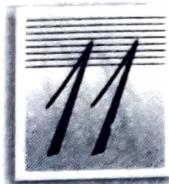
AUG

SEP

OCT

NOV

DEC



July  
Friday

JULY			
	2008		
Mon	07	14	21
Tue	01	08	15
Wed	02	09	16
Thu	03	10	17
Fri	04	11	18
Sat	05	12	19
Sun	06	13	20

=> STL - maps -

- general form:

map <index type, value type> map name;

- Example:

map <string, double> population;

=> population["India"] = 1.37;

population.count(country);

cout << population["India"];

# include <<sup>utility</sup> pair>

pair <int, char> P;

P = make\_pair(65, 'A');

cout << P.first;

cout << P.second;

=> Hybrid containers -

array < pair<string, int>, 3> stu;

AUGUST		2008			
Mon		04	11	18	25
Tue		05	12	19	26
Wed		06	13	20	27
Thu		07	14	21	28
Fri		01	08	15	22
Sat		02	09	16	23
Sun		03	10	17	24
					31

Harshit  
withub  
stl



July  
Saturday



AUG

SEP

OCT

## # Pointers - datatypes

=> &a → address of variable a

=> \*p → value stored at address p.

=> int \*p;

\*p = a;

If p = 1245

then, p + 1 = 1249

{ Difference of 4  
because integer  
is of 4 bytes