Conversions

① hex to decimal

```
int hex_int = 0x70;
int dec_int;
string stream stream;
stream << std::dec << hex_int << '\n';
stream >> dec_int;
```

② String to int.

```
string text = "123" ; int d;
string stream stream(text)
stream >> d;
```

/ stoi

③ int to string

```
int i = 123
string s = to_string(d)
```

④ int to binary (string)

```
cout << bitset<32>(12345).to_string();
```

↳ no to represent.

↳ length of representation

⑤ random int

```
rand()%100+123
```

⑥ random float

↳ static_cast<float>(rand())

/ static float (RAND_MAX)

Char Array to String

```
char a[5] = {'a', 'b', 'c', 'd', 'e'};
string s(a);
```

or

```
Char a[] = "abcde";
string s = a
```

String to C-string / ~~char array~~

~~s.c_str();~~ s.c_str()

String to char array

```
string s = "abcde"
char a [s.length()];
strcpy(c, s.c_str())
```

initializing a string with part of other str

```
String s (str2, 6, 6)
```

↑ first index; ↑ length of part from 1st index

String operations

1. ~~erase()~~ erase. (index no, length)
2. substr (start index, length)
3. replace (start index, string to replace)
4. append (String to append, start index, length)
5. length().
6. ~~find (string (s.find(ch, index to start from);~~
7. compare (ch)
8. find
9. clear (empties the string)
10. ~~getline (cin, str).~~
11. swap (s=swap(ch));
12. insert (s=insert(insert-loc, "string to insert");

~~i/p or o/p~~

getline (cin, str)

~~A1 S = {s1, ... sn~~

DATE
PAGE

Exception handling.

```
try {

}

catch(...) {
}.
```

(std::vector::iterator it)

Vectors

1. vector (int x, int y) → ~~copy~~ a ~~vector of x "y"~~ s
   ↳ {100, 100, 100, 100}
2. ~~vector third(second.begin(), second.end());~~
3. vector fourth(third);
4. ~~vector fifth (arr array_name, array_length)~~ (arry_name + array_length)
   ↳ eg:- int src[] = {1,2,3,4,5}
   int length = sizeof(src)/sizeof(int).
   std::vector<int> dest (src, src+length)
5. ~~size()~~
6. push_back()
7. pop_back().
8. ~~insert() →~~ it = ~~myvector.insert(it, 2, 300)~~
   ~~→ insert(iterator, array, array+length))~~

→ PTO

8) insert

→ myvector.insert(iterator where you want to insert)

→ it = iterator pointing pos where to cpy (fr. begin)

myvector.insert(it, iterator1, iterator2)

- it → position
- iterator1 → specifies the starting position where to cpy
- iterator2 → end point from where to copy.

9) erase

myvector.erase(begin it, end it)

→ erase b/w them.

10)

DATE
PAGE

List (Doubly Linked List)

1. front() → returns value of first elem in list.
2. back() → " " " last " " "
3. push_front → adds at beginning of list
4. push_back " " " back " " "
5. pop_front, pop_back
6. begin(), end()
7. insert()
   - insert(pos, element)
   - (3, 2) → insert 2 at 5th index
8. erase()
   - list.erase(begin(), end()) → multiple elements or (begin()) → single element
9. reverse()
10. size()
11. sort()
12. unique() → removes all duplicate elements.

Queue

→ push()
→ pop()
→ size()
→ front() → reference to first element
→ back() → " " last "

Stack

push()
pop()
size()
top() → reference to topmost element in stack

Priority Queue

size()
top() → reference to top element of queue
push()
pop()

~~An~~ heap of user defined class

class Point {
int x;
int y;

```
Point (int x1, int y1) {
public:
    x = x1,
    y = y1
};
comparator {
public:
    operator() (Point a, Point b) {
        return a.x > b.x
    }
};
main() { priority_queue < Point, vector<Point>, comparator > pq; }
```

~~Set()~~ Unordered ~~Set~~ Map

→ (unordered_map<T1, T2> umap)

T1 → string, T2 → int.

umap["Hello"] = 2.

umap["Man"] = 3

umap.insert(make_pair("Me", 3))

string key = "Meow"

umap.find(key) != umap.end() (→ key not found)

→ insert()

→ find() → returns iterator to element

→ erase() → iterator / key.

→ size()

unordered_map:: iterator it.

for(it = umap.begin(); it != umap.end(); it++)

cout << it→first << " " << it→second

(key) (value)

Unordered_set

unordered_set<string> set.

set.insert("hello")

set.insert("how")

set.insert("delete")

set.erase("delete me")

set.erase(set.begin())

set.find("Japan") → returns iterator to element

DATE
PAGE

Math

abs()
ceil()
cos()
cosh()
exp(x) → e^x
floor(x)
fmax()
fmin()
log()
log10()
nearbyint()
pow(base, exponent)
sqrt()

~~HashMap~~ Map

map<int, string> mp
mp.insert(make_pair(3, "Manik"))
. .find() → returns iterator
key
~~Update~~ .lower_bound()
.upper_bound()
.erase()

DATE
PAGE

Complex

```
complex<int> c(1,3);
c.real()   c.imag()
abs(c)
norm(c)
arg(c)
```

Numeric.

```
struct accumulator {
public:
  int operator()(int a, int b){
      return a-b;
  }
};

struct multiplier {
public:
  int operator()(int a, int b){
      return a*b;
  }
};

int main(){
  int a[] = {1,3,4,12,41,4,3};
```

DATE
PAGE

cout << accumulate(a, a+sizeof(a)/sizeof(a[0]), 0, custom)

int result[7];

adjacent_difference(a, a+7, result, accumulator);

int b[] = {2, 4, 14, 1, 4, 2, 4};

cout << inner_product(a, a+7, b, 0, accumulate, multiplies)

return 0

}

// limit

int main(){

cout << numeric_limits<int>::min() << endl;

cout << numeric_limits<float>::max() << endl;

}

// bitset

{

bitset<10> b(24);

cout << b << endl

cout << b[3] << endl;

cout << b.to_ulong() << endl;

return 0;

}

// algorithm

sort(a, a+5) → compare / stable_sort / is_sorted

reverse(a, a+2)

nth_element(a, a+2, a+5)

binary_search(a, a+5, 3-0);

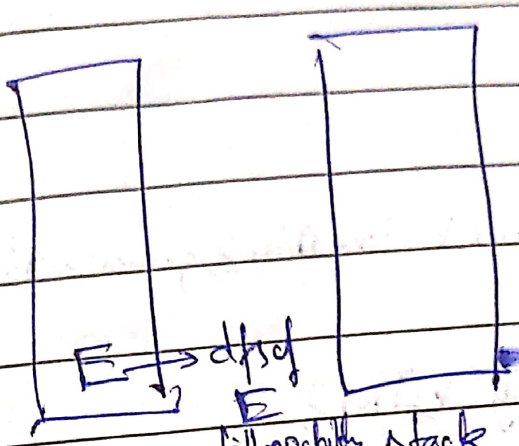
lower_bound, upper_bound

DATE
PAGE

min_element (a, a+5);
next_permutation (a, a+5);
merge (first, first+5, arr, arr+5, v.begin())
inplace_merge (first, first+5, second);
set_union (first, first.End, arr, arr+5, third);
find (InputIterator first, last, val)
count (" " ")
equal (" " , InputIterator second)
is_permutation (first1, last1, first2)
search (first1, last1, first2, last2)
replace (first, last, old_value, new_value)
remove (first, last, value)
unique (first, last)
rotate (first, middle, last)
random_shuffle (first, last)
stable_sort (first, last)
is_sorted (fir, last)

Graphs

Topological Sort



Set (visited vertices) (vertices in order)

priority-queue<iPair, vector<iPair>, greater<iPair>> pq;

vector<int> dist(V, ∞);

Dijkstra (src)

dist [∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞ ∞]

pq → empty (every item is pair<int, int>)

weight, vertex

insert source into pq

while(pq != empty){

u = pq.top(); pq.pop();

for dist(pos(u)) loop all vertices of u i.e v

if (dist[v] > dist[u] + weight[u,v])

dist[v] = dist[u] + weight[u,v]

}

Implementation of Weighted Graph (STL)

addEdge (int u, int v, int w)

{ adj[u].push_back(make_pair(v,w))

adj[v].push_back(u,w);

~~print~~

~~int v, w~~ x.

Prims

```
priority_queue <iPair, vector<iPair>, greater<iPair>> pq;
int src = 0
vector <int> key(V, INF);
vector<int> parent (V, -1);
vector <bool> inMST (V, false);
pq.push (make_pair (0, src));
  key[src] = 0
while (!pq.empty()) {
  int u = pq.top().second();
  pq.pop();
inMST [u] = true;
for (i = adj[u].begin(); i != end) {
    int v = (*i).first
      weight = (*i)->second;
if (inMST [v] == false && key[v] > weight)
{
  key[v] = weight.
  pq.push (make_pair (key[v], weight))
  parent [v] = u
}
}
}
```