Notes

C++ STL for C.P

Vector of Pairs

⇒ Simple decleration of ~~pairs~~ ~~in a~~ vector of pairs — (without taking input.

```
void printvec(vector<pair<int, int>> &v){
    cout << "size" << v.size() << endl;
    for(int i=0; i< v.size(); ++i){
        cout << v[i].first << " " << v[i].secound << endl;
    }
    cout << endl;
}

int main(){
    vector<pair<int, int>> v = {{1, 2}, {2, 3}
                                {4, 5}};

    printVec(v);
}
```

⇒ Decleration of vector of pair by taking input from user —

```
void printVec(vector<pair<int, int>> &v){
    cout << "size:" << v.size() << endl;
    for(int i=0; i< v.size(); ++i){
        cout << v[i].first << " " << v[i].secound;
    }
    cout << endl;
}

int main(){
```

make_pair() → always work for map, set, vector but { } work for vector only

```
print Vec(V);
int n;
cin >> n;
for (int i=0; i<n; ++i)
    int x, y;
    cin >> x >> y;
    v.push_back ({x, y});
        or
    v.push_back (make_pair(x, y));

print Vec(v);
```

~~Vector(q/ Array etc~~

Input —	Output
3	size : 0
1 2	size : 3
2 3	1 2
4 5	2 3
	4 5

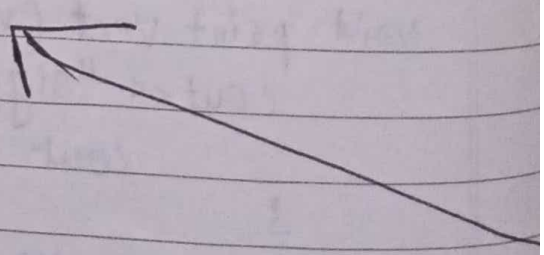
Vector & Array (or Array & Vector)

```
void print Vec (vector <int> &v){
    cout << "size :" << v.size() << endl;
        cout << v[i] << " ";
}
cout << endl;
```

⇒ By using vector & Array we have fixed rows but dynamic columns.

```
int main(){
    int N;
    cin>>N;
    vector<int> v[N];
    for(int i=0; i<N; ++i){
        int n;
        cin>>n;
        for(int j=0; j<n; ++j){
            int x;
            cin>>x;
            v[i].push_back(x);
        }
    }
    for(int i=0; i<N; ++i){
        print vec(v[i])
```

Input	Output
N → 3	size : 3
N=1 → 3	1 2 3
1 2 3	size : 3
N=2 → 3	3 4 5
3 4 5	size : 2
N=3 → 2	1 2
1 2	



vector of Vector ——

⇒ By using vector of vector we have dynamic row ~~and columns~~ as well as column.

```
void printVec ( vector<int> &v ){
    cout << "size:" << v.size() << endl;
        cout << v[i] << " ";
    }
    cout << endl;
}.
int main() {
    int N;
    cin >> N;
    vector < vector < int > > v;
    for (int i=0; i<N; ++i){
        int n;
        cin >> n;
        vector<int> temp;
        for( int j = 0; j < n; ++j){
            int x;
            cin>>x;
            temp.push_back(x);
        }
        v.push_back(temp);
    }
    for( int i=0; i<N ; ++i){   // or v.size();
        printVec( v[i] );
    }.
```

→ Input output same ().

To access the value of iterator directly without alloting any iterator

Note

⇒ (cout << *s1.begin())

OR

s1 is a set

(s1.begin())

↓ iterator

```
int main(){
    int N;
    cin>>N;
    vector<vector<int>> v;
    for(int i=0; i<N; ++i){
        int n;
        cin>>n;
        v.push_back(vector<int>());
        for(int j=0; j<n; ++j){
            int x;
            cin>>x;
            v[i].push_back(x);
        }
    }
    for(int i=0; i<v.size(); ++i){
        printVec(v[i]);
    }
}
```

⇒ Iterators

syntax of decleration of iterators.

container<data typ>:: iterator it;

eg: vector<int>:: iterator it;

vector<pair<int,int>>:: iterator it;

↓ int

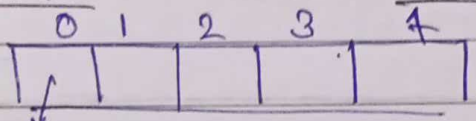
how to

⇒ Using Iterator to iterate over vector —

Sol^n
```
vector<int> :: iterator it = v.begin();

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

Difference bet (it ++) AND (it + 1)



(it ++) ⟶ next iterator को point करता है

(it + 1) ⟶ next location को point करता है.

Note

(it + 1) only work in (vector) not in set and other container because in other contain the next iterator ≠ next location like vector.

⇒ Iterate over pairs —

```
vector< pair<int, int>> v_p = {{1,2}, {2,3}, {3,4}};
vector < pair<int, int>> :: iterator it;

for( it = v_p.begin(); it != v_p.end(); ++it)
    { cout << (*it).first << " " << (*it).second << endl;
```

This code also written as.

Iterate over pairs

```
for (it = v.p.begin(); it != v.p.end(); ++it)
  {   cout << (it -> first) << " " << (it -> second)
                                   << endl;
}
```

Important trick

Important trick.

output

```
1  2
2  3
3  4.
```

Note

$(*it).first \iff (it \rightarrow first)$

Range Based Loop

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

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

To replace this

```
for( int value : v) {
    value ++;
}
for( int value : v) {
    cout << value << " ";
}
cout << endl;
```

In range based loop copy of values are taken so use & to access actual value

Output —

2 3 5 6 7
3 4 6 7 8

For a vector of pairs — (Ranged based loop)

```
vector<pair<int, int>> v_p = {{1,2}, {2,3}};
for (pair<int, int> value : v_p) {
        cout << value.first << " " << value.second
             << endl;
}
cout << endl;
```

Output

1 2
2 3

Auto Key word (auto)

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

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

or

for (auto value : v)
        cout << value << " ";
```