/ /

#include <algorithm>

* Functions :-
→ sort
→ search

عشان أقارن algorithm لازم أقارن وقت، صح قارن وقت بدي
يكون عندي نفس ال machine ، اشيلها هاي الكوة بي
total number of operation) بدل ما أشوف الوقت بشوف كم
operations عندي بكل كود

عائلات
:- ~~[illegible]~~ أقسم لذلك رح بفوقة عندي ال (exact number) *

2^n — n qubic — n^2 — linear

* n ⇐ هو عدد العناصر إلي بتعامل معها

* عدد العمليات ثابت (shifting فيها) ~~[illegible]~~

O(1)

* عدد العمليات مش ثابت ويعتمد على n زي أدو لكونة عندي shifting

O(n)

Scanned by CamScanner

* pointer special variable //
بخزن values بخزن addresses

& x ⇒ جيبلي ال address تبع المتغير x
* y ⇒ pointer

* & ⇒ جيبلي محتويات ال memory location

int x=5 // متغير عادي

int *ptr = &x // pointer
بتخزن فيه (address)

cout << x; 5
cout << *&x; 5

* بدي أعدل قيمة x ⇐ (بطريقتين)

1. x = 3;
2. *y = 3;

* ال datatype تبعت ال pointer هنا ال
datatype إلي بدو يأشر عليه ال pointer

* لما بعمل عمليات على ال pointer أنا بعملها على addresses

Scanned by CamScanner

pair

* priority - queue ← بتضيف على الأخر لكن internally
يتم الترتيب.

* pair :-

pair < DT1 , DT > pair_name ;

first second

↳ (structure DT)

int	string
ID	Name

comment :-

① DT1 ≠ DT2 ⇒ مش شرط زي بعض
② DT1 , DT2 ⇒ simple ⇒ & not simple ممكن يكونوا
⇏

* Exp :- ① pair < int , string > P1 ; ⇒ مبدأياً بحتاج تحجز rubbish

P1 . first = 134
P2 . second = "Ali"

int	string
134	Ali

② pair < int , pair < string , double > > P2 ;
↳ إني بكون جوا ما إلو اسم

P2 . first = 134 ;
P2 . second . first = "Ali"
P2 . second . second = 31.5

First	second first	second second
134	Ali	31.5

ALADIB

Scanned by CamScanner

int	string	double	char
134	Ali	31.5	P

* ممكن أعملها بالشكل دي بطريقة :-

① pair<pair<int,string>, pair<double,char>> P3
② pair<int, pair<string, pair<double,char>>> P4

* ممكن أعملها كـ شكل (array of pairs) :-

* pair<int, pair<string, pair<double, char>>> P4[3]

first, second, first, second

~~P4~~ [i].second.first = "Ali" ;

pair<int,string> P3 (P1) ؛
لازم يكونوا نفس النوع

* بدل ما أكتب (P3) من الأول بشدعي (P1) وبستغل ال constructor .

* swap ⇐ بيدل الجهتين لازم يكونوا من نفس النوع

First ↔ first , second ↔ second

* أنا بعبي ال pair بـ 3 طرق :

① cin ② Assignment ③ make pair

Scanned by CamScanner

vector

* عا بدي اصرح size ← ال array

① * include <vectors>

* vector :- 1 Dim array with dynamic size

② vector <DT> vector_name;
↳ vector, simple, pair -----

vector <int> v; // size = 0
vector <int> v(10); initial size = 10

vector <int> v(10, -1)
initial size initial value

* عا بعمل (resize) ← ما بضيف فوق ال element القديم بضيف بعد الزيادة.

v1 = v2 ⇒ حتى لو مش نفس ال size بتعمل ال (right hand size)

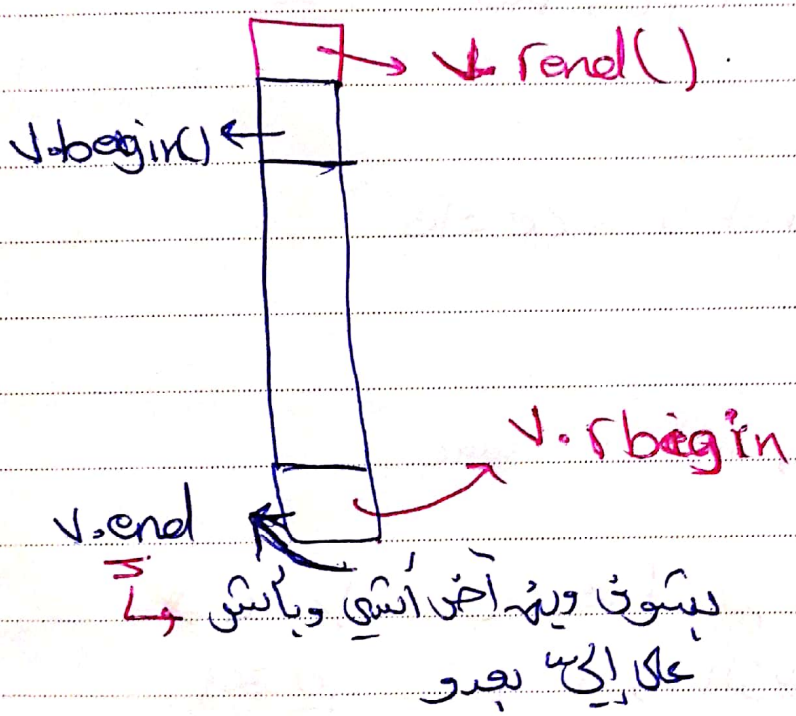
* functions :- as vector ال

① stack :- v.push_back(30);
v.pop_back();
② list :- v.insert(address, ▭);
v.erase(address);

كلهم عبارة عن member function

* vector <int> v(5); أول أشي بضاف ال size
v.push_back(30); بعدين بنقل ال element القديم
على الجديد.

* بدي ال address تبع أول index ← نفس اسم ال array
* بدي ال address تبع آخر index ← اسم ال array + size



begin()
end()
rend()
rebegin()

Iterators

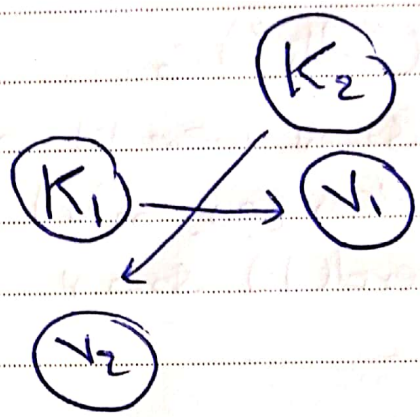
```
vector<int> v(5);
v.size();// 5
v.push_back(30); // O(n)
v.size(); // 6
v.push_back(100); // O(1)
v.size(); // 7
v.pop_back(); // O(1)
v.size(); // 6
```

ALADIB
Scanned by CamScanner

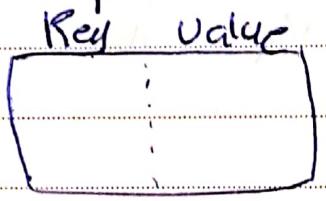
clear(); ⟹ بتمسح كل إلي فيها افضل مكان استوعبها فيه هو ال destructor

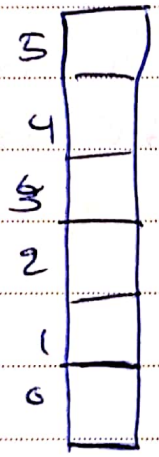
vector<vector<int>> vecofvec;

بشبه ال (2Dim array) ⟸



* map :-





* بتلف عن طريقه ال (iterator) فا بقدر ألف عن طريقه ال (for loop) لأنهم حش مجموعة continuously.

* include <map>

map <DT1, DT2> map name;

Key ⟸ value

map<string, int> mp;

ال Key فا بتكرر

mp["Key"] = value;

mp["Ahmad"] = 10;

Scanned by CamScanner

mp.insert(make_pair("Ali", 99);

mp.erase("ahmad");

بياخد Key معينة وبتمسح ال Key وال value إلي معها.

mp.find("Ali");

بتمسح Key معينة إذا موجود برجعلي iterator واقف عنده (كأنه برجع address).
أما إذا مش موجود برجعلي (mp.end())

* cout << (*it).first;

عشان الأقواس ال (.) أقوى
فبنحط الأقواس حتى تصير الأولوية لل (*)

ara $\xrightarrow{*+.}$

cout << it → first;

يعني . + * صدق
العلاقة مع بعض

* set :-

1. sorted
2. No repetition

ALADIB
Scanned by CamScanner