LECTURE 01

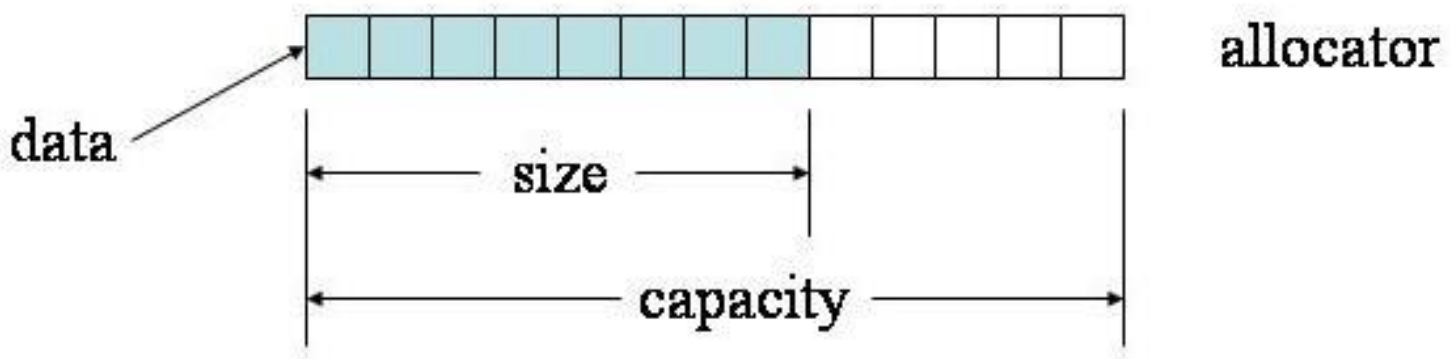
DYNAMIC ARRAY

Phạm Nguyễn Sơn Tùng

Email: sontungtn@gmail.com

Dynamic array là gì?

Dynamic array là một cấu trúc dữ liệu mảng động, người dùng không cần khai báo trước số lượng phần tử là bao nhiêu.



C++: `vector`

Java: `Arraylist`

Python: `list`

Cách khai báo sử dụng Dynamic array

C++

Thư viện:

```
#include <vector>
using namespace std
```

Khai báo:

```
vector<data_type> name;
```

Ví dụ:

```
vector<int> v;
```

python

Thư viện:

```
NULL
```

Khai báo:

```
<variable> = [value1, value2, ..]
```

Ví dụ:

```
list1 = []
list2 = [5, 7, 8, 3, 6]
```

Thêm phần tử vào Dynamic array

push_back(int)

```
vector<int> v;
v.push_back(5);
```

append(obj)

```
list = []
list.append(5)
```

Lấy phần tử đầu tiên Dynamic array

C++

python

front()

```
int result = v.front();
cout<<result;
```

Dùng vị trí để trả về giá trị đầu tiên.

```
result = list[0]
print(result)
```

Kết quả

5

Lấy phần tử cuối cùng Dynamic array

C++

python

back()

```
int result = v.back();
cout<<result;
```

Dùng vị trí để trả về giá trị cuối cùng.

```
result = list[-1]
print(result)
```

Kết quả

6

Chèn một giá trị vào Dynamic array

C++

python

insert(iterator, val): Chèn **một** giá trị.

```
vector<int>::iterator it;
it = v.begin() + 2;
v.insert(it, 9);
```

insert(pos, val): chèn giá trị val vào vị trí pos.

```
list = [5, 7, 8, 3, 6]
list.insert(2, 9)
```

Kết quả

Chèn hàng loạt giá trị và Dynamic array

C++

python™

insert(iterator, size_type, value)

```
vector<int>::iterator it;
it = v.begin() + 2;
v.insert(it, 3, 9);
```

<variable>[pos:pos] = n*[<value>]

```
list = [5, 7, 8, 3, 6]
list[2:2] = 3*[9]
```

Kết quả

Xóa phần tử cuối khỏi Dynamic array

C++

python

pop_back()

```
v.pop_back();
```

pop()

```
list.pop()
```

Kết quả

Xóa phần tử ở vị trí bất kỳ trong DA

C++

python

erase(iterator): Xóa **một** giá trị.

```
vector<int>::iterator it;
it = v.begin() + 2;
v.erase(it);
```

pop(pos): Xóa **giá trị** trong list ở vị trí bất kì và trả về giá trị bị xóa.

```
list.pop(2)
```

Kết quả

Xóa hàng loạt giá trị trong DA

C++

erase(iterator1, iterator2)

```
vector<int>::iterator it1;
vector<int>::iterator it2;
it1 = v.begin() + 1;
it2 = v.begin() + 3;
v.erase(it1, it2);
```

python

del <variable> [first: last]

Sẽ xóa các phần tử từ first đến trước last.

```
del list[1: 3]
```

Kết quả

Xóa toàn bộ các phần tử trong DA

C++

python

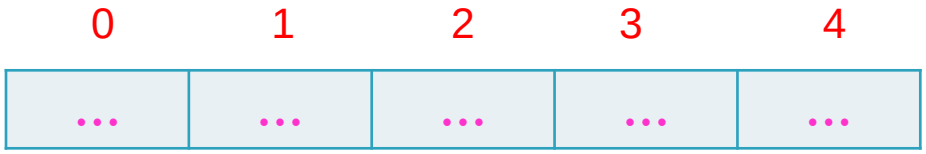
clear()

```
v.clear();
```

clear()

```
list.clear()
```

Kết quả



Lấy kích thước của mảng

size()

```
int n = v.size();
cout<<n;
```

len(obj)

```
n = len(list)
print(n)
```

5

Thay đổi kích thước lớn lên

C++

python

resize(size_type): Thay đổi và gán giá trị = 0 **nếu có thể**.

```
v.resize(7);
```

extend(list): nối 2 list lại với nhau.

```
list = [5, 7, 8, 3, 6]
list.extend(2*[0])
```

Kết quả

Thay đổi kích thước nhỏ lại

C++

```
v.resize(2);
```

python

```
list = [5, 7, 8, 3, 6]
list = list[0:2]
```

Kết quả

Kiểm tra xem DA có rỗng hay không

empty()

```
vector<int> v;
if(v.empty() == true)
    cout<<“DA is empty!";
else
    cout<<"DA is not empty!";
```

Sử dụng lại hàm len

```
list = []
if len(list) == 0:
    print(“DA is empty")
else:
    print(“DA is not empty")
```

DA is empty!

CÁC LƯU Ý KHI SỬ DỤNG DYNAMIC ARRAY

Truy cập ngẫu nhiên vào DA

Có thể truy cập vào thành phần vector để lấy giá trị hoặc thay đổi giá trị nếu chỉ số hợp lệ.

C++

```
v[2] = 9;
int a = v[2];
cout<<a;
```

python

```
list[2] = 9
a = list[2]
print(a)
```

Kết quả

9

Duyệt xuôi trong Dynamic Array

C++

```
for(int i=0; i<v.size(); i++)
{
    cout<<v[i]<<", ";
}
```

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

python

```
for i in range(0, len(list)):
    print(list[i],end=', ')
```

Kết quả

5, 7, 8, 10, 6

Duyệt ngược trong Dynamic Array

C++

```
for(int i=v.size(); i>=0; i--)
{
    cout<<v[i]<<", ";
}
```

```
ector<int>::reverse_iterator it;
for(it=v.rbegin(); it!=v.rend(); it++)
{
    cout<<*it<<", ";
}
```

python

```
for i in range(len(list)-1, -1, -1):
    print(list[i],end=', ')
```

6, 10, 8, 7, 5

Sử dụng vector như mảng 2 chiều

Khai báo:

```
vector<vector<data_type> > variable_name;
```

Ví dụ:

```
vector<vector<int> > v;
```

```
void Input(vector<vector<int> > &a, int &m, int &n)
{
    cin>>m>>n;
    a.resize(m);
    for(int i=0; i<m; i++)
    {
        a[i].resize(n);
        for(int j=0; j<n; j++)
        {
            cout<<"a["<<i<<"]"<<"["<<j<<"]: ";
            cin>>a[i][j];
        }
    }
}
```

Sử dụng list như mảng 2 chiều

List là kiểu dữ liệu động, mỗi phần tử có thể là 1 list khác, 1 giá trị,... Vì vậy có thể sử dụng như mảng 2 chiều. Khai báo vẫn như bình thường, tuy nhiên ở mỗi phần tử thì sẽ khai báo nó thành 1 list (list lồng list).

```
def inputArray2D(arr2d):
    n, m = [int (x) for x in input().split()]

    for i in range(n):
        arr2d.append([])
        arr2d[i] = map(int, input().split())
    return n, m
```

Hỏi đáp