VECTOR

#include <vector>

NỘI DUNG

1. Khai báo vector
2. Vector Iterator
3. Truy cập phần tử trong vector
4. Duyệt vector
5. Các hàm phổ biến trong vector

KHÁI QUÁT VỀ VECTOR

Vector được coi như mảng động có thể tự động thay đổi kích thước khi thêm hay xóa phần tử khỏi vector.

Lưu trữ được đa dạng các kiểu dữ liệu: int, string, pair, class, struct, vector, set,...

Các phần tử trong vector được lưu trữ trong các ô nhớ liên tiếp và truy cập được thông qua chỉ số.

KHÁI QUÁT VỀ VECTOR

VECTOR

Phù hợp để thay thế mảng, dễ dàng thêm phần tử vào vector.

Không phù hợp khi bài toán yêu cầu nhiều thác tác tìm kiếm hay xóa phần tử được thực hiện nhiều lần.

1. KHAI BÁO VECTOR

Syntax: `vector <data_type> vector_name;`

Khai báo vector rỗng: `vector <int> v;`

Khai báo vector với các phần tử cho trước:

```
vector <int> v = { 1, 2, 3, 4, 5 };
```

Khai báo vector với n phần tử: `vector <int> v (n);`

Khai báo vector với n phần tử có cùng giá trị val: `vector <int> v (n, val);`

Khai báo vector từ mảng a[ ] có n phần tử, hoặc từ một vector a khác:

```
vector <int> v (a, a + n);
vector <int> v (a.begin(), a.end());
```

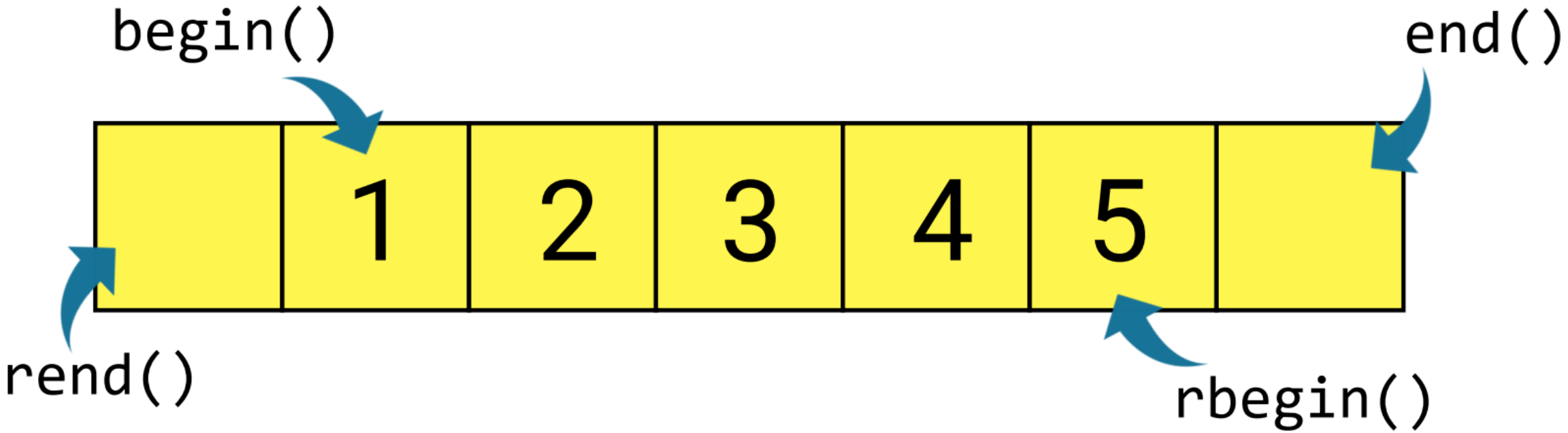
2. VECTOR ITERATOR

Iterator được coi như là một con trỏ thông minh, các bạn có thể hiểu đơn giản iterator là một con trỏ trỏ tới phần tử trong vector, tức là thông qua iterator bạn có thể truy cập, thay đổi phần tử mà nó đang quản lý.

`begin()`	Iterator đến phần tử đầu tiên
`end()`	Iterator đến phần tử sau phần tử cuối
`rbegin()`	Iterator đến phần tử cuối trong vector
`rend()`	Iterator đến phần tử trước phần tử đầu tiên

2. VECTOR ITERATOR

```
vector <int> v = { 1, 2, 3, 4, 5 };
```



2. VECTOR ITERATOR

- Để truy cập vào phần tử mà iterator đang quản lý các bạn phải dùng toán tử giải tham chiếu : *
- Các bạn có thể cộng trừ iterator, hoặc sử dụng toán tử ++ và -- với iterator, đối với vector thì việc khai báo iterator tương đối dài nên các bạn có thể dùng auto cho tiện.
- Nếu muốn truy cập đến iterator trỏ vào phần tử có chỉ số x trong mảng ta có thể sử dụng iterator : begin() + x

VÍ DỤ:

```cpp
#include <bits/stdc++.h>
using namespace std;
using ll = long long;

int main(){
    vector<int> a = {1, 2, 3, 4, 5, 6, 7, 8};
    vector<int>::iterator it1 = a.begin();
    cout << *it1 << endl; // 1

    vector<int>::iterator it2 = a.end();
    cout << *it2 << endl; // ?????

    vector<int>::iterator it3 = a.end();
    --it3; // dịch iterator này sang trái 1 phần tử
    cout << *it3 << endl; // 8

    cout << *(it1 + 4) << endl; // 5
}
```

3. TRUY CẬP PHẦN TỬ

Truy cập thông qua chỉ số tương tự như mảng 1 chiều

```
vector <int> a =  {1, 2, 3, 4};
cout << a[2]; // 3
```

front ( ) Trả về phần tử đầu tiên trong vector

```
vector <int> a =  {1, 2, 3, 4};
cout << a.front(); // 1
```

back ( ) Trả về phần tử cuối cùng trong vector

```
vector <int> a =  {1, 2, 3, 4};
cout << a.back(); // 4
```

4. DUYỆT VECTOR

```
vector <int> v = { 1, 2, 3, 4, 5 };
```

Duyệt thông qua chỉ số:

```
for(int i = 0; i < a.size(); ++i){
    cout << a[i] << ' ';
}
```

Duyệt bằng for each:

```
for(int x : a){
    cout << x << ' ';
}
```

Duyệt bằng Iterator:

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

Hoặc thay `vector<int>::iterator` bằng `auto`

5. CÁC PHƯƠNG THỨC

1. Hàm push_back

- **Chức năng:** Thêm 1 phần tử vào cuối vector, vector sẽ tự động mở rộng kích thước.
- **Hàm emplace_back** cũng có chức năng tương tự như hàm push_back.

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

int main(){
  vector <int> v;
  v.push_back(1); // 1
  v.push_back(5); // 1 5
  v.push_back(3); // 1 5 3
  v.push_back(2); // 1 5 3 2
  v.push_back(4); // 1 5 3 2 4
  for(int i = 0; i < v.size(); i++){
     cout << v[i] << ' ';
  }
}
```

OUTPUT: 1 5 3 2 4

5. CÁC PHƯƠNG THỨC

2. Hàm size, clear, empty

- **Hàm size:** Trả về số lượng phần tử trong vector
- **Hàm clear:** Xóa toàn bộ các phần tử trong vector
- **Hàm empty:** Trả về true nếu vector rỗng, ngược lại trả về false

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

int main(){
    vector<int> v;
    for(int i = 1; i <= 5; i++){
        v.push_back(i);
    }
    cout << v.size() << endl;
    v.clear(); // xoa toan bo
    if(v.empty()) cout << "EMPTY\n";
    else cout << "NOT EMPTY\n";
}
```

OUTPUT: 5
NOT EMPTY

5. CÁC PHƯƠNG THỨC

3. Hàm insert

- **Chức năng:** Thêm 1 phần tử vào vị trí bất kì trong vector thông qua iterator.
- **Cú pháp:** insert(Vị trí iterator, val)

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

int main(){
    vector<int> v;
    for(int i = 1; i <= 5; i++){
        v.push_back(i);
    }
    for(int i = 0; i < v.size(); i++){
        cout << v[i] << ' ';
    }
    v.insert(v.begin() + 2, 100); // thêm
// 100 vào chỉ số 2
    for(int x : v){
        cout << x << ' ';
    }
}
```

OUTPUT:

```
1 2 3 4 5
1 2 100 3 4 5
```

5. CÁC PHƯƠNG THỨC

4. Hàm erase

- **Chức năng:** Xóa 1 phần tử trong vector thông qua iterator.
- **Cú pháp:** erase (Vị trí iterator)

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

int main(){
    vector<int> v;
    for(int i = 1; i <= 5; i++){
        v.push_back(i);
    }
    for(int i = 0; i < v.size(); i++){
        cout << v[i] << ' ';
    }
    v.erase(v.begin() + 2); // xóa phần
// tử ở chỉ số 2
    for(int x : v){
        cout << x << ' ';
    }
}
```

OUTPUT:

```
1 2 3 4 5
1 2 4 5
```

5. CÁC PHƯƠNG THỨC

5. Hàm pop_back

- **Chức năng:** Xóa phần tử cuối cùng trong vector

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

int main(){
    vector<int> v;
    for(int i = 1; i <= 5; i++){
        v.push_back(i);
    }
    for(int i = 0; i < v.size(); i++){
        cout << v[i] << ' ';
    }
    v.pop_back();
    for(int x : v){
        cout << x << ' ';
    }
}
```

OUTPUT:

```
1 2 3 4 5
1 2 3 4
```

5. CÁC PHƯƠNG THỨC

6. Hàm assign

- **Chức năng:** Gán các phần tử trong vector với cùng một giá trị
- **Cú pháp:** assign (số phần tử, val)

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

int main(){
    vector<int> v;
    v.assign(5, 100);
    for(int x : v) cout << x << ' ';
}
```

OUTPUT:

100 100 100 100 100

5. CÁC PHƯƠNG THỨC

7. Hàm resize

- **Chức năng:** Thay đổi kích thước của vector
- **Cú pháp:** resize(n)

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

int main(){
    vector<int> v(5, 28);
    for(int x : v) cout << x << ' ';
    cout << endl;
    v.resize(3);
    for(int x : v) cout << x << ' ';
}
```

OUTPUT:

```
28 28 28 28 28
28 28 28
```