

# TIN HỌC ĐẠI CƯƠNG

## BÀI 9: VÉC-TƠ VÀ CẤU TRÚC

---

Phạm Xuân Cường  
Khoa Công nghệ thông tin  
[cuongpx@tlu.edu.vn](mailto:cuongpx@tlu.edu.vn)

1. Lớp vector
2. Kiểu dữ liệu cấu trúc (struct)

# Lớp vector

---

- Dãy phần tử có kích thước thay đổi được
- Có sẵn các phương thức để thao tác với dãy:
  - Thêm phần tử
  - Xóa phần tử
  - Xác định kích thước của dãy
  - v.v. . .
- Cần dòng định hướng bộ tiền xử lý sau:

```
#include <vector>
```

- Cách 1: Khai báo một vector rỗng  
`vector<kiểu-phần-tử> tên-vector;`
- Ví dụ:  
`vector<float> day_so;`  
`vector<string> day_xau;`

## Khai báo vector

- Cách 2: Khai báo vector có n phần tử  
`vector<kiểu-pt> tên-vector(n);`
- Cách 3: Khai báo vector có n phần tử với giá trị ban đầu gt

`vector<kiểu-pt> tên-vector(n, gt);`

- Ví dụ:

`vector<int> day_nguyen(10);`

`vector<double> day_thuc(20, 1.2);`

## Truy nhập phần tử của vector

- Dùng phương thức at:  
tên-vector.at(chỉ-số)

- Dùng chỉ số:  
tên-vector[chỉ-số]

- Ví dụ:

```
vector<int> vec(2);  
vec[0] = 10; // chỉ số bắt đầu từ 0  
vec[1] = 22;  
cout << vec[0] + vec[1]; // in ra 32
```

## Nhập và hiển thị vector

```
#include <iostream>
#include <vector>
using namespace std;
int main()
{
    vector<int> vec(10);
    for (int i = 0; i < 10; i++)
    {
        cout << "vec[" << i << "] = ";
        cin >> vec[i];
    }
    cout << "Day so nguyen vua nhap:" << endl;
    for (int i = 0; i < 10; i++)
        cout << vec[i] << " ";
    return 0;
}
```



## Vector với kích thước nhập từ bàn phím

```
#include <iostream>
#include <vector>
using namespace std;
int main()
{
    vector<int> vec;
    int n;
    cout << "Nhap so phan tu: "; cin >> n;
    vec.resize(n); // kích thước vector bằng n phần tử
    for (int i = 0; i < 10; i++) {
        cout << "vec[" << i << "] = ";
        cin >> vec[i];
    }
    cout << "Day so nguyen vua nhap:" << endl;
    for (int i = 0; i < 10; i++)
        cout << vec[i] << " ";
    return 0;
}
```

## Một số phương thức hữu ích trong lớp vector

| Phương thức                    | Mô tả  |
|--------------------------------|--|
| <code>vec.front()</code>       | Trả về phần tử đầu tiên  |
| <code>vec.back()</code>        | Trả về phần tử cuối cùng   |
| <code>vec.clear()</code>       | Xóa tất cả các phần tử   |
| <code>vec.push_back(pt)</code> | Thêm phần tử pt vào cuối vector  |
| <code>vec.pop_back()</code>    | Xóa phần tử cuối cùng  |
| <code>vec.empty()</code>       | Trả về <code>true</code> nếu vector rỗng, <code>false</code> ngược lại |
| <code>vec.size()</code>        | Trả về số phần tử hiện tại trong vector                                |

## Ví dụ về các phương thức vector

```
#include <iostream>
#include <vector>
using namespace std;
int main(){
    vector<int> vec(2);
    vec[0] = 12;
    vec[1] = 35;
    cout << vec.size() << endl; // in ra 2
    vec.push_back(100);
    vec.push_back(200);
    cout << vec.size() << endl; // in ra 4
    // In ra 12 35 100 200
    for (int i = 0; i < vec.size(); i++)
        cout << vec[i] << " ";
    return 0;
}
```

- Nhập một dãy số thực rồi tính trung bình cộng của các số dương trong dãy:
  - Nhập số phần tử
  - Nhập dãy
  - Tính trung bình cộng
  - Hiển thị dãy
  - Hiển thị trung bình cộng

- Vị trí chèn/xóa dưới dạng một giá trị kiểu **iterator** nằm bên trong kiểu vector (không dùng chỉ số ở đây)
- Ví dụ:

```
vector<int>::iterator itr;
```

itr là một biến có kiểu iterator, được gọi là **biến lặp**, nó có thể trỏ đến một phần tử cụ thể của vector

## Chèn/xóa trên vector

- Khởi tạo biến lặp:

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

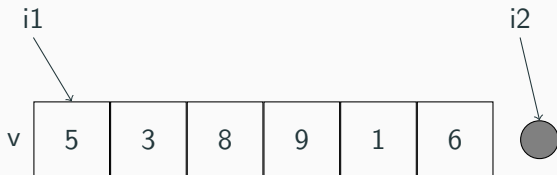
```
...
```

```
// i1 trỏ đến phần tử đầu
```

```
vector<int>::iterator i1 = v.begin();
```

```
// i2 trỏ đến vị trí ngay sau phần tử cuối
```

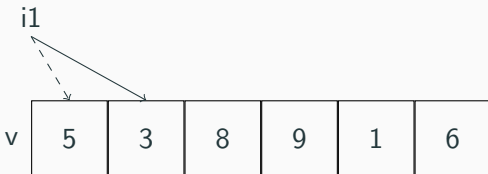
```
vector<int>::iterator i2 = v.end();
```



## Chèn/xóa trên vector

- Cho biến lặp trở tới phần tử kế tiếp bằng phép ++
- Lấy phần tử được trở bằng phép \*

```
// Cho i1 trở tới phần tử đầu (5)
vector<int>::iterator i1 = v.begin();
// Cho i1 trở tới phần tử kế tiếp (3)
i1++;
cout << *i1 << endl; // in ra 3
```



## Chèn phần tử vào vector

Chèn x vào trước vị trí pos: **v.insert(pos, x)**

```
vector<int> v;  
v.push_back(6); // v = {6}  
v.push_back(2); // v = {6, 2}  
v.push_back(7); // v = {6, 2, 7}  
vector<int>::iterator i1;  
i1 = v.begin(); // i1 trỏ tới 6  
i1++; // i1 trỏ tới 2  
v.insert(i1, 9); // Chèn 9 vào trước i1  
// Do đó, v = {6, 9, 2, 7}
```



## Xóa một phần tử khỏi vector

Xóa phần tử ở vị trí pos: **v.erase(pos)**

```
vector<int> v;  
v.push_back(6); // v = {6}  
v.push_back(2); // v = {6, 2}  
v.push_back(7); // v = {6, 2, 7}  
v.push_back(4); // v = {6, 2, 7, 4}  
vector<int>::iterator i1;  
i1 = v.begin(); // i1 trỏ tới 6  
i1++; // i1 trỏ tới 2  
v.erase(i1); // Xóa 2 ==> v = {6, 7, 4}
```

## Xóa một dải phần tử khỏi vector

Xóa các phần tử từ vị trí `pos1` đến vị trí ngay trước `pos2` (không xóa phần tử ở vị trí `pos2`): **`v.erase(pos1, pos2)`**

```
vector<int> v;  
v.push_back(6); // v = {6}  
v.push_back(2); // v = {6, 2}  
v.push_back(7); // v = {6, 2, 7}  
v.push_back(4); // v = {6, 2, 7, 4}  
vector<int>::iterator i1 = v.begin(); // i1 trỏ 6  
i1++; // i1 trỏ 2  
v.erase(i1); // Xóa 2 ==> v = {6, 7, 4}
```

## Kiểu dữ liệu cấu trúc (struct)

---

## Kiểu dữ liệu cấu trúc (struct)

- Khai báo biến cấu trúc: tên-cấu-trúc tên-biến-cấu-trúc;
- Truy nhập các trường trong cấu trúc:  
tên-biến-cấu-trúc.tên-trường
- Có thể kết hợp định nghĩa kiểu cấu trúc và khai báo biến cấu trúc:

```
struct tên-cấu-trúc
{
    kiểu-1 tên-trường-1;
    kiểu-2 tên-trường-2;
    :
    kiểu-n tên-trường-n;
} tên-biến-cấu-trúc;
```

## Ví dụ về kiểu cấu trúc

```
#include <iostream>
#include <cmath>
using namespace std;
struct diem // Cau truc diem voi hai truong x va y
{
    double x;
    double y;
};
int main()
{
    diem diemA, diemB;
    diemA.x = 1; diemA.y = 3;
    diemB.x = 4; diemB.y = 3;
    double d = sqrt(pow(diemB.x - diemA.x, 2) +
                    pow(diemB.y - diemA.y, 2));
    cout << "Khoang cach giua A va B la " << d << endl;
    return 0;
}
```

Thực hiện các công việc sau:

- Định nghĩa kiểu cấu trúc `hoc_sinh` có hai trường dữ liệu sau:
  - `ho_ten` kiểu `string` (họ tên)
  - `diem_tk` kiểu `double` (điểm tổng kết)
- Khai báo một vector để chứa danh sách học sinh
- Nhập số lượng học sinh
- Nhập danh sách học sinh
- Tìm và hiển thị học sinh giỏi nhất đầu tiên

**Questions?**