Vector

Nguyễn Mạnh Hiển
hiennm@tlu.edu.vn

Nội dung

1. Cấu trúc dữ liệu là gì?
2. Vector
3. Chèn phần tử
4. Xóa phần tử
5. Thời gian chạy

1. Cấu trúc dữ liệu là gì?

Cấu trúc dữ liệu

- Là cách tổ chức dữ liệu trong máy tính sao cho các thao tác xử lý dữ liệu (như tìm, chèn, xóa) trở nên hiệu quả hơn
- Ví dụ cấu trúc dữ liệu:
  - Vector
  - Danh sách liên kết
  - Ngăn xếp/Hàng đợi
  - Cây
  - Bảng băm

Cài đặt cấu trúc dữ liệu

Mỗi cấu trúc dữ liệu được cài đặt bằng một lớp C++:

```
template <typename T>
class Tên-Cấu-Trúc-Dữ-Liệu {
    public:
        hàm tạo (constructor)
        hàm hủy (destructor)
        các thao tác xử lý
    private:
        các trường dữ liệu
        các thao tác trợ giúp
};
```

(T là kiểu dữ liệu của các phần tử trong cấu trúc dữ liệu)

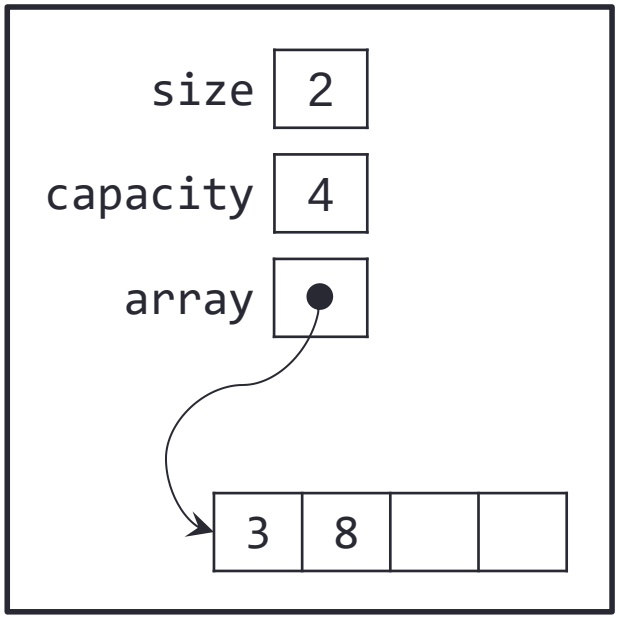
2. Vector

Vector

- Lưu trữ một dãy phần tử có kích thước thay đổi được (trong khi kích thước của mảng cố định sau khi khai báo)
- Các thao tác chính:
  - Chèn và xóa phần tử ở cuối vector
  - Chèn và xóa phần tử ở giữa vector
  - Lấy kích thước vector
  - Truy nhập phần tử dùng chỉ số

Cài đặt vector

```
template <typename T>
class Vector {
    public:
        hàm tạo, hàm hủy, toán tử gán
        lấy kích thước vector
        truy nhập phần tử dùng chỉ số
        các thao tác chèn và xóa
        các thao tác khác
    private:
        int size;     // kích thước vector (số phần tử)
        int capacity; // dung lượng vector (sức chứa)
        T * array;    // con trỏ tới mảng chứa các phần tử
        các thao tác trợ giúp
};
```

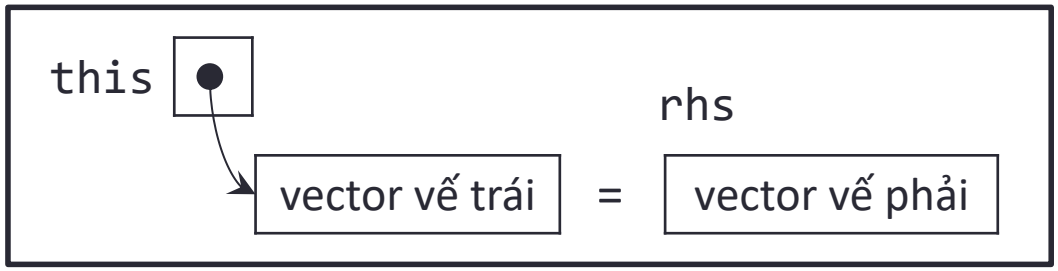


Hàm tạo và hàm hủy

```
// initCapacity là dung lượng ban đầu của
// vector, có giá trị ngầm định bằng 16
Vector(int initCapacity = 16) {
    size = 0;
    capacity = initCapacity;
    array = new T[capacity];
}

~Vector() {
    delete[] array;
}
```

Toán tử gán

```
// rhs (right-hand side) là vector ở vế phải của phép gán.
// this là con trỏ tới vector hiện hành, tức là vế trái.
Vector & operator=(Vector & rhs) {
    if (this != &rhs) {           // ngăn cản tự sao chép
        delete[] array;           // xóa mảng hiện tại
        size = rhs.size;          // đặt kích thước mới
        capacity = rhs.capacity; // đặt dung lượng mới
        array = new T[capacity]; // tạo mảng mới

        // Sao chép các phần tử từ vế phải sang vế trái
        for (int i = 0; i < size; i++)
            array[i] = rhs.array[i];
    }

    return *this;
}
```



Kích thước vector và truy nhập phần tử

```
// Trả về kích thước vector
int getSize() {
    return size;
}

// Nếu vector rỗng, trả về true;
// ngược lại trả về false
bool isEmpty() {
    return (size == 0);
}

// index là chỉ số của phần tử cần truy nhập
T & operator[](int index) {
    return array[index];
}
```

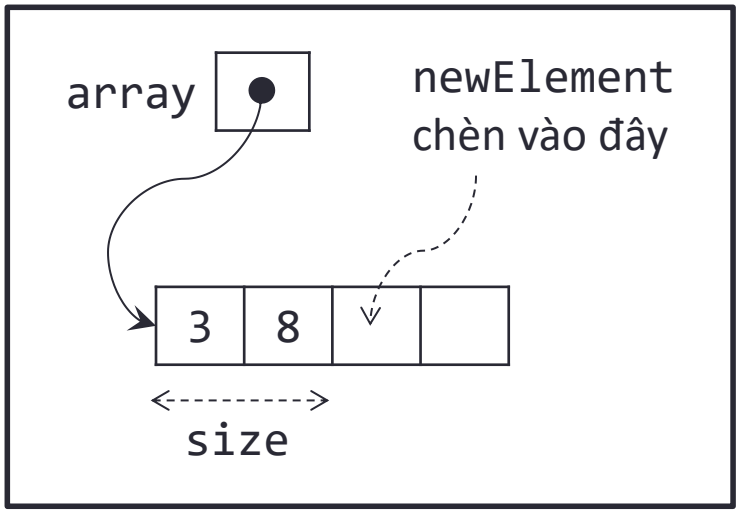
3. Chèn phần tử

Tăng dung lượng vector

```
// Đây là thao tác trợ giúp cho các thao tác chèn.
// newCapacity là dung lượng mới (phải lớn hơn kích thước).
void expand(int newCapacity) {
    if (newCapacity <= size)
        return;

    T * old = array;              // old trỏ tới mảng cũ
    array = new T[newCapacity]; // array trỏ tới mảng mới

    for (int i = 0; i < size; i++)
        array[i] = old[i];        // sao chép cũ sang mới

    delete[] old;                 // xóa mảng cũ
    capacity = newCapacity;       // đặt dung lượng mới
}
```

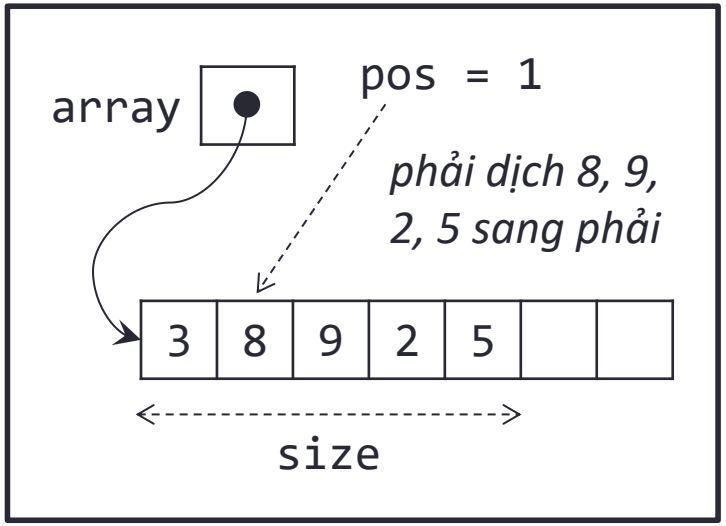
Chèn phần tử vào cuối vector

```
// newElement là phần tử mới cần chèn vào cuối vector
void pushBack(T newElement) {
    // Gấp đôi dung lượng nếu vector đã đầy
    if (size == capacity)
        expand(2 * size);

    // Chèn phần tử mới vào ngay sau phần tử cuối cùng
    array[size] = newElement;

    // Tăng kích thước 1 đơn vị
    size++;
}
```



Chèn phần tử vào giữa vector

```
// pos (position) là vị trí chèn.
// newElement là phần tử mới cần chèn.
void insert(int pos, T newElement) {
    // Gấp đôi dung lượng nếu vector đã đầy
    if (size == capacity)
        expand(2 * size);

    // Dịch các phần tử ở pos và sau pos sang phải 1 vị trí
    for (int i = size; i > pos; i--)
        array[i] = array[i – 1];

    // Đặt phần tử mới vào vị trí pos
    array[pos] = newElement;

    // Tăng kích thước 1 đơn vị
    size++;
}
```



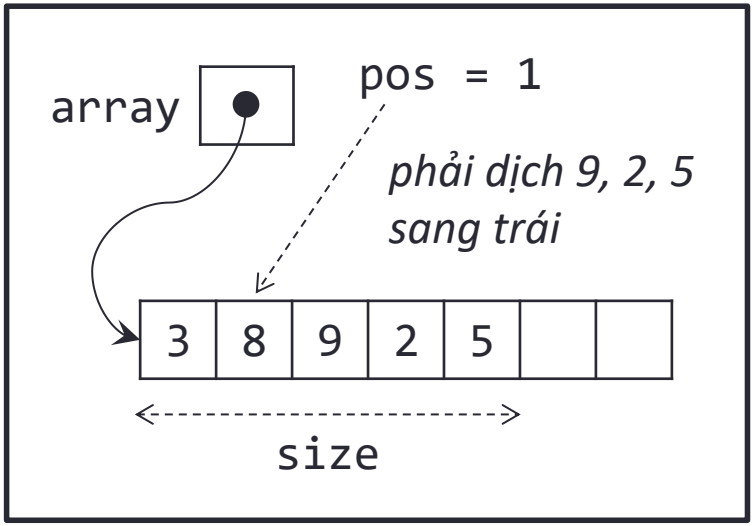
4. Xóa phần tử

Xóa phần tử

```
// Xóa phần tử ở cuối vector
void popBack() {
    size--;
}

// Xóa tất cả các phần tử
void clear() {
    size = 0;
}
```

Xóa phần tử ở giữa vector

```
// pos (position) là vị trí của phần tử cần xóa
void erase(int pos) {
    // Dịch các phần tử sau pos sang trái 1 vị trí
    for (int i = pos; i < size - 1; i++)
        array[i] = array[i + 1];

    // Giảm kích thước 1 đơn vị
    size--;
}
```



5. Thời gian chạy

Phân tích thời gian chạy

- Hàm tạo, hàm hủy: O(1)
- Toán tử gán: O(n) – vì phải sao chép các phần tử
- getSize, isEmpty, operator[]: O(1)
- expand: O(n) – vì phải sao chép các phần tử
- pushBack: O(1)
- insert: O(n) – vì phải dịch các phần tử sang phải
- popBack: O(1)
- clear: O(1)
- erase: O(n) – vì phải dịch các phần tử sang trái