void v1() {

vector<int> V(3);

V[0] = 5;

V[1] = 6;

V[2] = 4;

for (int i = 0; i < V.size(); i++) {

cout << V[i] << endl;

}

}

void v2() {

int iWordNum;

vector<string> arrWords;

cout << "Enter number of words = ";

cin >> iWordNum;

arrWords.resize(iWordNum);

for (int i = 0; i < arrWords.size(); i++)

{

cout << "Enter word " << i << " = ";

cin >> arrWords[i];

}

cout << "After entering data..." << endl;

for (int i = 0; i < arrWords.size(); i++)

cout << arrWords[i] << endl;

}

//Các phương thức của stack: push\_back() và pop\_back()

void v3() {

int i;

vector<int> V;

for (i = 0; i < 5; i++)

V.push\_back(i\*10);

cout << endl << "Mang ban dau:" << endl;

for (i = 0; i < V.size(); i++)

cout << V[i] << endl;

V.pop\_back();

cout << endl << "Xoa phan tu cuoi:" << endl;

for (i = 0; i < V.size(); i++)

cout << V[i] << endl;

}

/\*Xóa tại vị trí bất kỳ, xóa trắng\*/

/\*template <class T>

void print(const vector<T>&v)

{

for (int i = 0; i < v.size(); i++)

cout << v[i] << endl;

}

void v4() {

const char \*chao[] = { "Xin", "chao", "tat", "ca", "cac", "ban" };

int n = sizeof(chao) / sizeof(\*chao);

vector<char\*> v(chao, chao + n);//đây là 1 cách khởi tạo vector

cout << "vector truoc khi xoa" << endl;

print(v);

v.erase(v.begin() + 2, v.begin() + 5);//xóa từ phần tử thứ 2 đến phần tử thứ 5

v.erase(v.begin() + 1);//xóa phần tử thứ 1

cout << "vector sau khi xoa" << endl;

print(v);

v.clear();//Xóa toàn bộ các phần tử

cout << "Vector sau khi clear co "<< v.size() << " phan tu" << endl;

}\*/

/\*Tham chiếu back(), front()\*/

void v5(){

int a[] = { 3,2,3,1,2,3,5,7 };

int n = sizeof(a) / sizeof(\*a);

vector<int> v(a, a + n);

cout << "phan tu dau la " << v.front() << endl;

cout << "phan tu cuoi la " << v.back() << endl;

cout << "gan phan tu cuoi la 9 ..." << endl;

v.back() = 9;

cout << "gan phan tu dau la 100 ..." << endl;

v.front() = 100;

cout << "kiem tra lai vector: ";

for (int i = 0; i < v.size(); i++)

cout << v[i] << " ";

cout << endl;

}

/\*Sự khác biệt giữa 2 hàm size() và capacity()\*/

void v6() {

vector<int >so1,so2[10];

so1.reserve(10);

cout << "Kich thuoc toi da:" << so1.capacity();

cout << "\n Kich thuoc hien tai cua mang 2 " << so2->size() << endl;

}

/\*Ví dụ nếu ko có reserve() thì capacity sẽ là 4 :\*/

void v7() {

vector< int > my\_vect;

my\_vect.reserve(8);

my\_vect.push\_back(1);

my\_vect.push\_back(2);

my\_vect.push\_back(3);

cout << my\_vect.capacity() << "\n";

}

/\*Kiểm tra tràn chỉ số mảng\*/

void v8() {

try {

vector<long> V(3, 10); // Khởi tạo vector gồm 3 thành phần

cout << "V[0]=" << V[0] << endl; // Đưa thành phần 0 ra màn hình

cout << "V[1]=" << V[1] << endl; // Đưa thành phần 1 ra màn hình

cout << "V[2]=" << V[2] << endl; // Đưa thành phần 2 ra màn hình

cout << "V[3]=" << V[3] << endl; // Thành phần 3 (lệnh này hoạt động không

cout << "V[4]=" << V[4] << endl; // Thành phần 4 (càng không đúng)

cout << "V[0]=" << V.at(0) << endl; // Không sử dụng [], dùng phương thức at

cout << "V[1]=" << V.at(1) << endl; // Thành phần 1, OK

cout << "V[2]=" << V.at(2) << endl; // Thành phần 2, OK

cout << "V[3]=" << V.at(3) << endl; // Thành phần 3: Lỗi, chương trình dừng

cout << "V[4]=" << V.at(4) << endl;

getchar();

}

catch (exception &e) {

cout << "Tran chi so ! " << endl;

}

}

/\*Ví dụ làm việc với thư viện STL với lớp vector và con trỏ kiểu iterator\*/

void v9() {

vector<int> v;

for (int i = 10; i < 15; i++) v.push\_back(i);

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

cout << "Xuat:";

while (it != v.end())

{

cout << \*it << " ";

it++;

}

}

/\*Dùng 1 số hàm cơ bản trong thư viện algorithm\*/

void printint(const int &i)

{

cout << i << " ";

}

void v10() {

vector<int> v(8); // v: 0 0 0 0 0 0 0 0

fill(v.begin(), v.begin() + 4, 5); // v: 5 5 5 5 0 0 0 0

fill(v.begin() + 3, v.end() - 2, 8); // v: 5 5 5 8 8 8 0 0

cout << "\nVector Contains:";

for\_each(v.begin(), v.end(), printint);

}

/\*Copy iterator ( tương tự memcpy() đối với pointer\*/

void v11() {

int a[] = { 1, 2, 3, 4, 5, 6 };

int n = sizeof(a) / sizeof(\*a);

vector<int> v1(a, a + n);

vector<int> v2(n);

copy(v1.begin(), v1.end(), v2.begin());

//copy\_n(v1.begin(), v1.size(), v2.begin());

for (int i = 0; i < n; i++)

cout <<"\n"<< v2[i] << " ";

}

/\*Đảo ngược (reverse)\*/

void v12() {

vector<int> a;

// set some values:

for (int i = 1; i < 10; ++i) a.push\_back(i); // 1 2 3 4 5 6 7 8 9

reverse(a.begin(), a.end()); // 9 8 7 6 5 4 3 2 1

// print out content:

cout << "\nVec contains:";

int i, n = a.size();

for (i = 0; i < n; i++)

cout << a[i] << " ";

cout << endl;

}

/\*Thay thế các giá trị (replace)\*/

void v13() {

int myints[] = { 10, 20, 30, 30, 20, 10, 10, 20 };

vector<int> a(myints, myints + 8); // 10 20 30 30 20 10 10 20

replace(a.begin(), a.end(), 20, 99); // 10 99 30 30 99 10 10 99

cout << "\nVec contains: ";

int i, n;

for (i = 0, n = a.size(); i < n; i++)

cout << a[i] << " ";

}

/\*Thay thế các giá trị theo điều kiện (replace\_if)\*/

inline bool SoLe(int i) { return ((i % 2) == 1); }

void v14() {

vector<int> a;

for (int i = 1; i < 10; i++) a.push\_back(i); // 1 2 3 4 5 6 7 8 9

replace\_if(a.begin(), a.end(), SoLe, 0); // 0 2 0 4 0 6 0 8 0

cout << "\nVec contains: ";

int i, n;

for (i = 0, n = a.size(); i < n; i++)

cout << a[i] << " ";

}

/\*Xóa với remove và remove\_if\*/

bool IsOdd(int x)

{

return x % 2;

}

void v15() {

//remove

int A[] = { 3,1,4,1,5,9 };

int N = sizeof(A) / sizeof(\*A);

vector<int> V(A, A + N);

copy(V.begin(), V.end(), ostream\_iterator<int>(cout, " "));

//The output is "3 1 4 1 5 9".

cout << endl;

vector<int>::iterator new\_end =

remove(V.begin(), V.end(), 1);

V.resize(new\_end - V.begin());

copy(V.begin(), V.end(), ostream\_iterator<int>(cout, " "));

// The output is "3 4 5 9

}

void v16() {

int a[] = { 3,1,4, 8, 5, 2, 9 };

int n = sizeof(a) / sizeof(\*a);

vector<int> vec(a, a + n);

vector<int>::iterator new\_end =

remove\_if(vec.begin(), vec.end(), IsOdd);

vec.erase(new\_end, vec.end());

cout << "\n";

copy(vec.begin(), vec.end(), ostream\_iterator<int>(cout, " "));

// The output is "4 8 2".

}

/\*Sắp xếp container (ở đây là vector)\*/

void v17() {

int A[] = { 3,2,3,1,2,3,5,3 };

int n = sizeof(A) / sizeof(\*A);

vector<int> V(A, A + n);

cout << endl << "Danh sach ban dau" << endl;

for\_each(V.begin(), V.end(), printint);

sort(V.begin(), V.end());

vector<int>::iterator vi;

cout << endl << "Danh sach sau khi sap xep" << endl;

for\_each(V.begin(), V.end(), printint);

}

/\*Tìm kiếm tuyến tính\*/

void v18() {

int a[] = { 2,4,2,6,9,1,2,3,2,3,4,5,6,4,3,2,1 };

int n = sizeof(a) / sizeof(\*a);

vector<int> v(a, a + n);

int first = find(a, a + n, 1) - a;//các hàm thuật toán hoàn toàn có thể thao tác trên mảng & con trỏ thường

cout << "\n[Array] So thu tu cua phan tu dau tien = 1: " << first << endl;

first = find(v.begin(), v.end(), 1) - v.begin();

cout << "[Vector] So thu tu cua phan tu dau tien = 1: " << first << endl;//find\_if

vector<int>::iterator it;

it = find\_if(v.begin(), v.end(), IsOdd);

first = it - v.begin();

cout << "Phan tu le dau tien la " << \*it << " O vi tri thu " << first << endl;

}

/\*Đếm & tìm min max\*/

void v19() {

int a[] = { 3,2,3,1,2,4,5,3 };

int n = sizeof(a) / sizeof(\*a);

vector<int> v(a, a + n);

cout << "\nSo luong so 3 trong mang: " << count(v.begin(), v.end(), 3) << endl;

cout << "So luong so le trong mang: " << count\_if(v.begin(), v.end(), IsOdd) << endl;

cout << "So nho nhat trong mang: " << \*min\_element(v.begin(), v.end()) << endl;

cout << "So lon nhat trong mang: " << \*max\_element(v.begin(), v.end()) << endl;;

}

/\*Hàm chuyển đổi hàng loạt transform()\*/

template <class T>

T inc(T x)

//hàm dùng để chuyển đổi

{

return x + 1;

}

void v20() {

int a[] = { 3,2,3,1,2,3,5,3 };

int n = sizeof(a) / sizeof(\*a);

vector<int> v(a, a + n);

transform(v.begin(), v.end(), v.begin(), inc<int>);

copy(v.begin(), v.end(), ostream\_iterator<int>(cout, " "));

// The output is "4 3 4 2 3 4 6 4".

}