Lộ trình học C++

 **C++ cơ bản**:

* Cú pháp cơ bản: biến, kiểu dữ liệu, toán tử.
* Cấu trúc điều khiển: if, else, switch, for, while.
* Hàm: định nghĩa, tham số, đệ quy.

tham số: có 3 cách truyền pass by value, pass by pointer và pass by references

1, pass by value (truyền tham số bằng giá trị): void ham(int x) không thay đổi giá trị biến x

//trong hàm: ham(x) truyền giá trị của biến x vào hàm

2, pass by pointer(truyền tham số bằng con trỏ): void ham(int \*x) truyền con trỏ trỏ đến x vào hàm, muốn thay đổi giá trị của x trong hàm cần dùng toán tử giải tham chiếu vd: \*x += 5;

/ void ham(int \*x){

\*x += 5;

}

//Trong hàm main: ham(&x) : truyền con trỏ của x vào hàm

3, pass by references (truyền tham số bằng tham chiếu) Khi một tham số được truyền bằng tham chiếu, hàm sẽ nhận một tham chiếu trực tiếp đến biến gốc, và mọi thay đổi trong hàm sẽ ảnh hưởng đến biến đó.VD: void ham(int &x) truyền trực tiếp biến x vào hàm, trong hàm ta biến đổi như trong main vd: x += 5;

// void ham(int &x){

x++;

}

//Trong hàm main: ham(x) : truyền trực tiếp biến x vào hàm

 **Cấu trúc dữ liệu và thuật toán**:

* Mảng, chuỗi, danh sách, ngăn xếp, hàng đợi.
* Thuật toán sắp xếp, tìm kiếm.
* Struct

 **C++ nâng cao**:

* Con trỏ, dynamic memory.
* Lập trình hướng đối tượng (OOP): class, object, kế thừa, đa hình.

 **Thư viện chuẩn C++ (STL)**:

* Vector, map, set, pair.

 **C++ chuyên sâu**:

* Quản lý bộ nhớ, smart pointers.
* Xử lý lỗi, multithreading.

 **Dự án thực tế**:

* Xây dựng ứng dụng thực tiễn với C++.

cout << fixed << setprecision(2) << y <<endl;

dùng thư viện #include <iomanip> để in mấy chữ số sau dấu phẩy

abs(x) tính giá trị tuyệt đối

pow(a, b) tính a mũ b, trả về double

round(x) làm tròn x (lên hoặc xuống)

sqrt(x) căn x, trả về double

khai báo nguyên mẫu hàm (để dùng hàm tùy vị trí)

tên hàm + (kiểu dữ liệu không ghi biến);

vd:

bool nt(int);

int sum(int);

int min(int, int); //nhiều biến in

//ở dưới:

bool nt(int n) {

...

}

int sum(int n) {

...

}

int min(int a, int b) {

...

}

- Cách dùng define

#define pi 3.14

#define s(x) ((x)\*(x))

#define songuyen long long

#define for(i, a, b) for(int i = a; i < b; i++)

int main(){

songuyen x;

cin >> x;

cout <<x << endl;

cout << pi <<"\n";

cout << s(5);

for(i, 0, 100){

cout << i << " ";

}

}

- cách tắt để nhập và in cả mảng ra

int n; cin >> n;

int a[n];

for(int &x : a) cin >> x;

for(int x : a){

cout << x << " ";

}

- Tạo mảng string dùng như mảng int

string mang[6];

string tmp = “blabla”

mang[i] = tmp; //i có thể là 1 số int từ 0 đến 5

- Cách dùng vector: push được thêm bao nhiêu phần tử vào vector thì push, vector không tự sắp xếp thứ tự phần tử như set. Thằng nào vào sau thì nằm sau, vào trước thì nằm trước

A screenshot of a computer

Description automatically generated

vector<kiểu\_dữ\_liệu> tên\_vector;

v.push\_back(phần tử muốn nhập)

v.size() : kích thước vector

v.cleare() : dọn sạch các phần tử trong vector

v.back(): trả về giá trị phần tử cuối cùng trong vector(not null)

v.begin() và v.rbegin() : là 2 con trỏ trỏ đến phần tử đầu và cuối của vector. CÓ thể giải tham chiếu 2 con trỏ này để lấy giá trị

cout << \*v.begin();

có thể truy cập vào các phần tử trong vector bằng v[i]

vector<data\_type> vector\_name(n, x0) : tạo vector gồm n phần tử toàn x0

vector<int> ten[n] : ma trận có n hàng, mỗi hàng là 1 vector

- Nhập phần tử vào vector mảng đến khi gặp dấu # thì dừng

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

while(true){

string input;

cin >> input;

if(input == "#"){

break;

}

a[i].push\_back(stoi(input));

}

}

VD:

input:

3 //số vector

1 2 3 #

3 4 5 6 #

1 2 3 #

output:

1 2 3

3 4 5 6

1 2 3

- Xóa phần tử khỏi vector

v.erase(v.begin() + i); trong đó v.begin() trỏ vào phần tử đầu tiên của vector, i là thứ tự phần tử muốn xóa trong vector(chạy từ 0 đến n-1, vector có n phần tử)

 v.erase(v.begin() + 1, v.begin() + 4); // Xóa từ vị trí thứ 2 đến thứ 4 (giá trị 2, 3, 4) hay xóa từ v[1] đến v[3]

1. v.erase(remove(v.begin(), v.end(), 2), v.end()); // Xóa tất cả phần tử có giá trị là 2
2. Các phần tử còn lại trong vector sẽ được tự động dịch lên sau khi xóa.

- Nhập 1 xâu string từ bàn phím

th1: xâu k có dấu cách cin >> s; vd: nguyen

th2: xâu có dấu cách: getline(cin, s); vd:nguyen tuan

- Sau khi cin bộ đệm sẽ lưu phím enter làm lệnh getline phía sau không hoạt động nên cần cin.ignore(); để ăn phím enter đó

string s0;

cin >> s0;

cout << s0 << endl;

cin.ignore();

string s;

getline(cin, s);

cout << s;

return 0;

\* nếu trước getline là một getline khác thì không cần cin.ignore vì lệnh getline sẽ ăn luôn phím enter ngay sau nó. Nếu giữa 2 lệnh getline có ignore thì lệnh getline sau sẽ thiếu mất 1 kí tự đầu

string s0;

getline(cin, s0);

cout << s0 << endl;

string s;

getline(cin, s);

cout << s;

return 0;

- số phần tử của string là s.size() hoặc s.length()

- Cộng 2 xâu

string a;

cin >> a;

string b;

cin >> b;

string c = a + " " + b;

cout << c;

có thể dùng a += b; // thay đổi xâu a

không có a -= b;

- Cách so sánh 2 xâu: dùng > hoặc == hoặc < (so sánh phần tử đầu tiên của 2 xâu trước, nên sẽ có chuyện 1910 < 230)

**không dùng strcmp(s1, s2)**

- Cắt xâu dùng substr

string a;

cin >> a;

string b = a.substr(2, 4);

cout << b;

return 0;

Kết quả: abcdef bị cắt 2 phần tử đầu ab, và lấy 4 phần tử sau đó là cdef

- Chuyển 1 xâu string thành int (stoll để chuyển thành long long nếu xâu dài quá)

string a = "123435";

int x = stoi(a);

cout << x;

- Chuyển 1 số int hoặc float thành xâu string

int n = 12345; //thay int bằng float n = 123.345 cx được

string s = to\_string(n);

cout << s;

- Dùng stringstream để cắt nhỏ string ra bằng những khoảng space

int main(){

string s = "java c++ php python lap trinh";

stringstream ss(s);

string tmp;

vector<string> mang; //tạo vector lưu các phần tử được cắt

while(ss >> tmp){ //lưu phần tử được cắt vào tmp

mang.push\_back(tmp); //lưu vào vector

cout << tmp << endl;

}

cout << endl;

//in các phần tử trong vector ra

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

cout << mang[i] << endl;

}

}

- Dùng stringstream để cắt nhỏ string ra bằng những ký hiệu tùy ý (ở trong ví dụ này là dấu chấm)

string s = "java.php.python.laptrinh.hello.world";

vector<string> v;

stringstream ss(s);

string tmp;

while(getline(ss, tmp, '.')){ //để cắt bằng dấu chấm, dùng ‘.’

v.push\_back(tmp);

}

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

cout << v[i] << endl;

}

- Cách dùng pair trong cpp: thay int bằng các kiểu dữ liệu khác được

pair<int, int> p;

//khởi tạo các giá trị trong pair

//cach 1:

p.first = 1;

p.second = 2;

cout << p.first << " " << p.second;

//cách 2:

pair<int, int> v = make\_pair(100, 200);

hoặc pair<int, int> v = {12, 24};

hoặc pair<int, int> v {12, 24};

- tuple: giống pair nhưng được nhiều phần tử hơn

tuple<int, int, int, int> t {1, 2, 3, 4};

cout << get<3>(t) << " "; //get được từ 0 đến 3, không dùng biến i được mà phải là hằng số

- set: giống vector nhưng khi push các phần tử vào sẽ check xem trong set đã có phần tử đó hay chưa

Dùng s.insert(giá trị) thay vì s.push\_back(gt)

Các phần tử trong set được sắp xếp từ bé đến lớn

s.begin() : con trỏ trỏ vào phần tử đầu tiên

s.rbegin() : con trỏ trỏ vào phần tử cuối cùng

s.end(): con trỏ trỏ vào phần kết thúc của set (sau đít phần tử cuối cùng)

s.size() :kích thước set

s.count(x): trả về 1 nếu có x trong set, 0 nếu ko có

s.erase(x) : xóa phần tử có giá trị x trong set

s.upper\_bound(k) : trả về iterator trỏ đến phần tử liền sau k (có thể trỏ đến s.end() nếu k là pt cuối của set

s.find(x): trả về con trỏ trỏ đến phần tử x trong set. Nếu x ko nằm trong set, nó sẽ trả về con trỏ kết thúc của set là s.end(). s.find(x) là iterator

set<kieu\_du\_lieu>::iterator it;

//thì it là con trỏ trỏ đến các phần tử trong set có kieu\_du\_lieu

cách dùng:

it++

it--

advance(it, 3)

it = s.begin();

lấy giá trị mà con trỏ it trỏ đến(không sửa đổi) dùng \*it

s.find(x) ở trên chính là 1 kiểu iterator

set<int>::iterator it = s.find(x);

không thể it = it + 1;

- in ra phần tử s[3] trong set (không cout << s[3] như vector được)

set<int> s;

auto x = s.begin();

advance(x, 3);

cout << \*x;

\*auto x = s.begin() tương đương với

set<int>::iterator p = s.begin();

-Muốn xóa phần tử có thứ tự nhất định thì khai báo x như trên rồi s.erase(x) hoặc s.erase(\*x) đều được

auto p = s.begin();

advance(p, 2);

s.erase(p); //xóa pt có tt là 2 trong set

- không thể p = p + 1; mà chỉ có thể dùng p++, p--, advance(p, 3)

- cách in cả set ra dùng iterator

vd:

set<int>::iterator it;

for(it = s.begin(); it != s.end(); it++){

cout << \*it << " ";

}

- multiset: giống set về các hàm insert, size, erase, count, begin, end, rbegin, find ... nhưng cho phép các phần tử trùng nhau. Vẫn sắp xếp thứ tự phần tử từ bé đến lớn như set. Thích hợp để xử lí bài toán sắp xếp chuỗi số như cửa sổ trượt

multiset<int> ms;

- Kiểu dữ liệu map: lưu dữ liệu theo các cặp ({key, value}). Mỗi key chỉ có 1 value

Key value có thể âm

Khai báo:

map<kiểu dữ liệu của key, kiểu dữ liệu của value> tên\_map;

vd: map<int, int> mp;

\* mp.insert({key, value}) : nhập 1 cặp key value vào map

vd: mp.insert({36, 69});

\* mp[key] : trả về value mà key đang quản lí

vd: cout << mp[36]; sẽ ra 69

vd: nếu key là string và value là struct

A screen shot of a computer code

Description automatically generated

\* mp.count(100); nếu mp[100] == 0, hàm trả về 0

nếu mp[100] != 0, hàm trả về 1

\* có thể sửa value của 1 key đã được insert value

vd: mp[36] = 53;

\* các key chưa được insert value sẽ có value = 0

vd: cout << mp[-1]; kết quả là 0

\* số lượng phần tử trong map bằng số key đã được insert value

vd: cout << mp.size();

\* có thể dùng value để đếm số lần key xuất hiện bằng mp[key]++;

\* các key là duy nhất, tăng dần, mỗi một key chỉ có 1 value

- In các cặp ({key, value}) trong map dùng pair

//auto có công dụng tự xác định datatype phù hợp cho x

vd:

for(pair<int, int> x : mp){

cout << x.first << " " << x.second << endl;

}

- có thể thay pair<int, int> bằng auto

vd:

for(auto x : mp){

cout << x.first << " " << x.second << endl;

}

mp.begin() : trỏ đến key đầu

mp.rbegin() :trỏ đến key cuối

mp.end() trỏ đến key kết thúc (sát sau key cuối)

mp.erase(100) :xóa cặp key value có key == 100

-multimap: giống map nhưng có thể lưu được nhiều key giống nhau nhưng phải dùng insert chứ k đc mp[100] = 200;

multimap<int, int> mp;

mp.insert({1, 100});

mp.insert({1, 200});

mp.insert({2, 100});

for(auto x : mp){

cout << x.first << " " << x.second << endl;

}

- unordered\_map: giống map nhưng không sắp xếp key theo thứ tự bé đến lớn mà sx theo trước sau. Các key không được trùng nhau

\* Cách dùng struct

#include <bits/stdc++.h>

using namespace std;

struct sinhvien{

string msv;

string name;

int diem;

//constructor

sinhvien(){

msv = "20225772";

}

void in(){

cout << msv << " " << name << " " << diem ;

}

void nhap(){

cin >> name;

cin >> diem;

}

};

int main(){

sinhvien sv1;

sv1.nhap();

sv1.in();

}

\*nếu sinhvien không có constructor có thể khai báo kiểu này:

sinhvien sv2{"2022", "Tuna", 69};

- Các hàm thuật toán thông dụng trong c++

Với mảng: int a[n];

Tham số: (a + x, a + y + 1) duyệt từ a[x] đến a[y]

vd: (a, a+n) duyệt cả mảng

(a, b): input là a và b

({a, b, c, d,...}): input là nhiều phần tử

Hàm: min(tham số), max(tham số): trả về giá trị lớn nhất/nhỏ nhất trong đống input

vd: min(a, a+n) : trả về thằng nhỏ nhất từ a0 đến a[n-1]

max(a, a+n)

min\_element(), max\_element(): trả về con trỏ trỏ đến phần tử lớn nhất/nhỏ nhất trong đống input

vd: min\_element(a, a+n) : trả về con trỏ trỏ đến thằng nhỏ nhất

swap(a[0], a[1]); đổi chỗ x và y hoặc 2 pt trong mảng

sort(a+x, a+y+1): sắp xếp từ a[x] đến a[y] theo thứ tự tăng dần

sort(a+x, a+y+1, greater<int>()); như trên nhưng giảm dần

sort(a.begin(), a.end(), greater<int>()); như trên nhưng dành cho vector

- cách dùng find tìm xem trong mảng a có giá trị nào = x không

Nếu tìm được, hàm find trả về con trỏ trỏ đến phần tử x trong mảng

Nếu k tìm được, hàm find trả về a+n là phần tử kết thúc của mảng hoặc a.end đối với vector or set

vd:

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

auto it = find(a, a+5, x); //trong đó x là một giá trị cụ thể nào đó của phần tử trong mảng

if(it == a+5){

cout << "find k tim duoc, tro den a.end";

}

else cout << "find tim duoc phan tu, it tro den con tro cua x";

vd2: đối với vector

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

auto it = find(a.begin(), a.end(), 6);

if(it == a.end()){

cout << "find k tim duoc, tro den a.end";

}

else cout << "find tim duoc phan tu, it tro den con tro cua x";

- Cách dùng binary\_search

binary\_search(a, a+n, x) đối với mảng a có n phần tử hoặc binary\_search(a.begin(), a.end(), x) đối với vector, set,...

Hàm trả về true nếu tìm thấy, false nếu k tìm thấy

\*lưu ý: khi dùng binary\_search, a phải được sắp xếp theo thứ tự tăng dần

set<int> a;

int x;

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

cin >> x;

a.insert(x);

}

cout << binary\_search(a.begin(), a.end(), 5);

- Con trỏ ptr trong c++

Khai báo: int \*ptr; mặc định ptr == nullptr

Toán tử giải tham chiếu \*: \*ptr truy cập vào giá trị của a luôn

int a = 69;

int \*ptr = &a;

\*ptr = 100;

cout << a;