(Ghi chú)

Ngôn ngữ lập trình cpp

**Thực hiện: Nguyễn Văn Hào**

**Email:** [hao2205tb@gmail.com](mailto:hao2205tb@gmail.com)

(Tài liệu được biên xoạn lại từ nhiều nguồn và không dùng trong mục đích kinh doanh kiếm tiền)

# Mục lục

[Mục lục 1](#_Toc126778272)

[Chuỗi 10](#_Toc126778273)

[Khái niêm 10](#_Toc126778274)

[Khai báo và Khởi tạo chuỗi 10](#_Toc126778275)

[in chuỗi 10](#_Toc126778276)

[Strlen(): tính độ dài chuỗi 10](#_Toc126778277)

[Con trỏ chuỗi 11](#_Toc126778278)

[Khai báo 11](#_Toc126778279)

[Truy suất các ký tự bằng con trỏ 12](#_Toc126778280)

[Nhập chuỗi bằng con trỏ 13](#_Toc126778281)

[Xuất chuỗi bằng con trỏ 13](#_Toc126778282)

[Chèn chuỗi 13](#_Toc126778283)

[Cách 1: dịch chuyển phần sau vị trí chèn ra sau và chép chuỗi vào vị trí chèn 13](#_Toc126778284)

[Cách 2: cấp phát bộ nhớ động và chép từng phần của chuỗi đích vào bộ nhớ. 14](#_Toc126778285)

[Copy chuỗi 16](#_Toc126778286)

[Strcpy(): copy toàn bộ chuỗi 16](#_Toc126778287)

[Strcpy\_c(): copy toàn bộ chuỗi thay thế cho strcpy() ở một số trình biên dịch. 17](#_Toc126778288)

[Strncpy(): copy n ký tự 18](#_Toc126778289)

[Nối chuỗi 19](#_Toc126778290)

[Strcat() nối 2 chuỗi 19](#_Toc126778291)

[Strncat(): nỗi 2 chuỗi với n ký tự 19](#_Toc126778292)

[Sprintf() nối nhiều chuỗi 20](#_Toc126778293)

[Tách chuỗi 21](#_Toc126778294)

[Strtok() 21](#_Toc126778295)

[Tiếp tục sử dụng strtok() để tách toàn bộ chuỗi 22](#_Toc126778296)

[Tách số trong chuỗi 23](#_Toc126778297)

[Cắt chuỗi 24](#_Toc126778298)

[Xoá ký tự trong chuỗi 25](#_Toc126778299)

[Xoá chuỗi con trong chuỗi 26](#_Toc126778300)

[Xắp xếp chuỗi 27](#_Toc126778301)

[Xắp xếp các ký tự của chuỗi theo bảng chữ cái 27](#_Toc126778302)

[Sắp xếp các chuỗi theo thứ tự chữ cái 28](#_Toc126778303)

[Đảo ngược chuỗi 29](#_Toc126778304)

[Strstr(): tìm kiếm chuỗi con 30](#_Toc126778305)

[Strchr(): Tìm kiếm ký tự trong chuỗi 31](#_Toc126778306)

[Strcmp(): so sánh chuỗi 31](#_Toc126778307)

[Strncmp(): so sánh 2 chuỗi với n ký tự 32](#_Toc126778308)

[Hàm isspace(): 34](#_Toc126778309)

[Xoá khoảng trắng 35](#_Toc126778310)

[Xoá toàn bộ kí tự khoảng trắng 35](#_Toc126778311)

[Xoá khoảng trắng đầu và khoảng trắng cuỗi 35](#_Toc126778312)

[Chuyển số thành chuỗi 37](#_Toc126778313)

[Sprintf() 37](#_Toc126778314)

[Itoa() 38](#_Toc126778315)

[Chuyển chuỗi thành số 38](#_Toc126778316)

[Atoi(): chuyển chuỗi sang int 38](#_Toc126778317)

[Atol(): chuyển chuỗi sang long 40](#_Toc126778318)

[Atof(): chuyển chuỗi sang float 40](#_Toc126778319)

[Tolower(): chuyển chữ hoa sang chữ thường 41](#_Toc126778320)

[Tạo hàm chuyển đổi chữ hoa chữ thường 42](#_Toc126778321)

[Nhập chuỗi 43](#_Toc126778322)

[Nhập 1 ký tự 43](#_Toc126778323)

[Nhập 1 chuỗi 43](#_Toc126778324)

[Nhập nhiều chuỗi 43](#_Toc126778325)

[Nhập chuỗi có khoảng trắng 44](#_Toc126778326)

[Tạo chuỗi ngẫu nhiên 45](#_Toc126778327)

[String 46](#_Toc126778328)

[Tổng quát 46](#_Toc126778329)

[Khởi tạo 46](#_Toc126778330)

[Cách 1: sử dụng dấu ngoặc {} 46](#_Toc126778331)

[Cách 2: sử dụng dấu = 47](#_Toc126778332)

[Cách 3: Khởi tạo string với n ký tự lặp lại 47](#_Toc126778333)

[Truy xuất ký tự trong string 47](#_Toc126778334)

[Lấy kích thước string 48](#_Toc126778335)

[Length() 48](#_Toc126778336)

[Size() 49](#_Toc126778337)

[Nhập xuất string 49](#_Toc126778338)

[Nhập string 49](#_Toc126778339)

[Nhập nhiều string 49](#_Toc126778340)

[Nhập string có khoảng trắng 50](#_Toc126778341)

[Xuất string 51](#_Toc126778342)

[Thêm ký tự 51](#_Toc126778343)

[Nối string bằng toán tử + 51](#_Toc126778344)

[Hàm push\_back(): Thêm ký tự vào cuối tring C++ bằng 51](#_Toc126778345)

[Thêm chuỗi ký tự vào cuối tring C++ bằng toán tử gán 52](#_Toc126778346)

[Chèn 1 ký tự vào vị trí chỉ định trong string C++ bằng hàm insert 53](#_Toc126778347)

[Chèn 1 chuỗi ký tự vào vị trí chỉ định trong string C++ bằng hàm insert 54](#_Toc126778348)

[Xoá ký tự 54](#_Toc126778349)

[pop\_back() 54](#_Toc126778350)

[Xóa 1 ký tự trong string bằng hàm erase 55](#_Toc126778351)

[Xóa các ký tự trong một phạm vi chỉ định trong string C++ 56](#_Toc126778352)

[Kiểm tra chuỗi 57](#_Toc126778353)

[Kiểm tra chuỗi rỗng bằng hàm empty 57](#_Toc126778354)

[Kiểm tra chuỗi rỗng trong C++ bằng độ dài của nó 57](#_Toc126778355)

[Hàm substr(): copy chuỗi con từ chuỗi string ban đầu. 58](#_Toc126778356)

[Hàm copy(): copy chuỗi con từ chuỗi string ban đầu. 59](#_Toc126778357)

[Copy n ký tự từ đầu chuỗi string 60](#_Toc126778358)

[Copy n ký tự từ vị trí m trong string 61](#_Toc126778359)

[Copy toàn bộ chuỗi string trong C++ 61](#_Toc126778360)

[Tách string 62](#_Toc126778361)

[Tách string bằng một ký tự phân cách 62](#_Toc126778362)

[Tách string bằng một chuỗi ký tự phân cách 63](#_Toc126778363)

[Cắt chuỗi 64](#_Toc126778364)

[Cắt ký tự đầu tiên từ string trong C++ bằng hàm front 64](#_Toc126778365)

[Cắt ký tự cuối cùng từ string trong C++ bằng hàm back 65](#_Toc126778366)

[Chuyển đổi kiểu dữ liệu 66](#_Toc126778367)

[Chuyển số sang string trong C++ bằng hàm template 66](#_Toc126778368)

[Chuyển số sang string trong C++ bằng hàm to\_string 67](#_Toc126778369)

[Tìm ký tự hoặc chuỗi trong chuỗi 68](#_Toc126778370)

[Tìm ký tự hoặc chuỗi ký tự từ đầu string trong C++ bằng hàm find. 68](#_Toc126778371)

[Tìm ký tự hoặc chuỗi ký tự từ cuối string trong C++ bằng hàm rfind. 69](#_Toc126778372)

[Hàm replace trong C++ 71](#_Toc126778373)

[Thay thế một chuỗi ký tự tại phạm vi chỉ định trong string C++ bằng hàm replace 71](#_Toc126778374)

[Thay thế ký tự tại vị trí chỉ định trong string C++ bằng hàm replace 71](#_Toc126778375)

[Thay thế toàn bộ ký tự trong string trong C++ bằng hàm replace 72](#_Toc126778376)

[Con trỏ 73](#_Toc126778377)

[Khai báo 73](#_Toc126778378)

[Khởi tạo 73](#_Toc126778379)

[Sử dụng 73](#_Toc126778380)

[Thao tác đến giá trị của biến mà con trỏ trỏ tới 73](#_Toc126778381)

[Thao tác với địa chỉ lưu trong con trỏ 74](#_Toc126778382)

[Con trỏ của con trỏ 75](#_Toc126778383)

[Khai báo 75](#_Toc126778384)

[Khởi tạo 75](#_Toc126778385)

[Sử dụng 76](#_Toc126778386)

[Mảng 1 chiều 77](#_Toc126778387)

[Khởi tạo 77](#_Toc126778388)

[Truy cập phần tử mảng 77](#_Toc126778389)

[Lấy kích thước mảng 78](#_Toc126778390)

[Độ dài mảng 79](#_Toc126778391)

[Nhập xuất mảng 79](#_Toc126778392)

[Nhập mảng 79](#_Toc126778393)

[Xuất mảng 80](#_Toc126778394)

[Con trỏ mảng 80](#_Toc126778395)

[Khai báo 80](#_Toc126778396)

[Khởi tạo 80](#_Toc126778397)

[Địa chỉ 80](#_Toc126778398)

[Địa chỉ con trỏ mảng trỏ tới 80](#_Toc126778399)

[Chỉ định địa chỉ trong mảng thông qua dịch chuyển con trỏ 80](#_Toc126778400)

[Truy suất phần tử trong mảng bằng con trỏ 81](#_Toc126778401)

[Nhập xuất mảng bằng con trỏ 82](#_Toc126778402)

[Nhập mảng bằng con trỏ 82](#_Toc126778403)

[Xuất mảng bằng con trỏ 82](#_Toc126778404)

[Sao chép mảng 82](#_Toc126778405)

[Cách 1: Sao chép từng phần tử của mảng nguồn tới mảng đích 82](#_Toc126778406)

[Cách 2: sử dụng hàm memcpy() 83](#_Toc126778407)

[Thêm phần tử vào mảng 84](#_Toc126778408)

[Thêm phần tử vào cuối mảng 84](#_Toc126778409)

[Chèn phần tử vào vị trí bất kỳ 84](#_Toc126778410)

[Ghép 2 mảng 86](#_Toc126778411)

[Xoá phần tử 87](#_Toc126778412)

[Xoá một phần tử 87](#_Toc126778413)

[Xoá nhiều phần tử 88](#_Toc126778414)

[Tách mảng chẵn lẻ 90](#_Toc126778415)

[Đảo ngược mảng 92](#_Toc126778416)

[Xắp xếp mảng 93](#_Toc126778417)

[Hàm qsort() 93](#_Toc126778418)

[Xắp xếp tăng dần 94](#_Toc126778419)

[Xắp xếp giảm dần 95](#_Toc126778420)

[Tìm min max 96](#_Toc126778421)

[Tìm min 96](#_Toc126778422)

[Tìm max 96](#_Toc126778423)

[Tìm min max bằng cách xếp mảng 97](#_Toc126778424)

[So sánh mảng 98](#_Toc126778425)

[Cách 1: so sánh 2 mảng bằng cách đối chiếu từng cặp phần tử 98](#_Toc126778426)

[So sánh 2 mảng bằng hàm memcmp() 99](#_Toc126778427)

[Kiểm tra phần tử trùng 100](#_Toc126778428)

[Kiểm tra phần tử trùng trong mảng C++ bằng cách so sánh từng phần tử 100](#_Toc126778429)

[Kiểm tra phần tử trùng trong mảng C++ bằng cách sắp xếp mảng 102](#_Toc126778430)

[Lấy phần tử trùng 103](#_Toc126778431)

[Xoá phần tử trùng trong mảng đã được xắp xếp 104](#_Toc126778432)

[Xoá phần tử trùng trong mảng chưa được xắp xếp 106](#_Toc126778433)

[Đếm số phần tử trùng nhau 107](#_Toc126778434)

[Đếm số phần tử khác nhau 108](#_Toc126778435)

[Liệt kê phần tử 110](#_Toc126778436)

[Liệt kê các phần tử duy nhất trong mảng 110](#_Toc126778437)

[Liệt kê các phần tử xuất hiện một lần trong mảng 111](#_Toc126778438)

[Liệt kê các phần tử xuất hiện nhiều hơn 1 lần trong mảng 112](#_Toc126778439)

[Tìm kiếm phần tử 113](#_Toc126778440)

[Tìm kiếm một phần tử trong mảng 113](#_Toc126778441)

[Tìm phần tử xuất hiện nhiều nhất trong mảng 114](#_Toc126778442)

[Mảng 2 chiều 116](#_Toc126778443)

[Khai báo 116](#_Toc126778444)

[Khởi tạo 116](#_Toc126778445)

[Truy cập phần tử 117](#_Toc126778446)

[Nhập xuất 117](#_Toc126778447)

[Nhập 117](#_Toc126778448)

[Xuất 118](#_Toc126778449)

[In toàn bộ phần tử trong mảng 2 chiều 119](#_Toc126778450)

[Mảng chuỗi 2 chiều 120](#_Toc126778451)

[Khởi tạo 120](#_Toc126778452)

[Truy cập phần tử 120](#_Toc126778453)

[Lấy kích thước 122](#_Toc126778454)

[Lấy độ dài 122](#_Toc126778455)

[Lấy số phần tử mảng 2 chiều 123](#_Toc126778456)

[Con trỏ mảng 2 chiều 123](#_Toc126778457)

[Cấp phát bộ nhớ động cho mảng hai chiều 123](#_Toc126778458)

[Khai báo mảng 2 chiều trong C++ dưới dạng con trỏ trong mảng 2 chiều 124](#_Toc126778459)

[Khai báo mảng 2 chiều trong C++ dưới dạng con trỏ trong mảng 1 chiều 126](#_Toc126778460)

[Sử dụng con trỏ mảng 2 chiều trong hàm 127](#_Toc126778461)

[Sử dụng con trỏ mảng 2 chiều trong struct 128](#_Toc126778462)

[Kiểu cấu trúc 129](#_Toc126778463)

[Tổng quan 129](#_Toc126778464)

[Khai báo 130](#_Toc126778465)

[Khởi tạo 131](#_Toc126778466)

[Định nghĩa kiểu dữ liệu mới với typedef 132](#_Toc126778467)

[Khai báo 132](#_Toc126778468)

[Khởi tạo 132](#_Toc126778469)

[Truy cập vào thành viên 133](#_Toc126778470)

[Lấy kích thước 133](#_Toc126778471)

[Con trỏ cấu trúc 134](#_Toc126778472)

[Truy cập tới thành viên 134](#_Toc126778473)

[Sử dụng con trỏ cấu trúc trong hàm 135](#_Toc126778474)

[Mảng cấu trúc 136](#_Toc126778475)

[Khai báo 136](#_Toc126778476)

[Khởi tạo 137](#_Toc126778477)

[Truy cập vào phần tử và thành viên 137](#_Toc126778478)

[Cấu trúc và hàm 139](#_Toc126778479)

[Truyền cấu trúc cho hàm 139](#_Toc126778480)

[Trả về cấu trúc từ hàm 139](#_Toc126778481)

[Sao chép cấu trúc 140](#_Toc126778482)

[Cách 1: sử dụng toán tử = 140](#_Toc126778483)

[Cách 2: Sử dụng con trỏ 141](#_Toc126778484)

[So sánh 2 struct 142](#_Toc126778485)

[Cách 1: So sánh các thành viên với nhau 142](#_Toc126778486)

[Cách 2: Sử dụng hàm memcmp() 143](#_Toc126778487)

[Xắp xếp theo thành viên 145](#_Toc126778488)

[Tham khảo 149](#_Toc126778489)

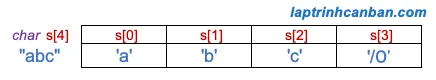
# Chuỗi

## Khái niêm

Chuỗi là tập hợp của nhiều kí tự và được viết trong dấu ngặc kép “”.

VD biểu diễn của chuỗi “abc”

"abc" = 'a' + 'b' + 'c' + '\0'



## Khai báo và Khởi tạo chuỗi

VD

char s[4];

char s[5] = "love";

char s[] = "love";

## in chuỗi

VD

#include <iostream>

using namespace std;

int main(){

  char str1[] = "C";

  char str2[] = "C#";

  char str3[] = "C++";

  cout << str1 <<' ' << str2 <<' ' << str3;

  return 0;

}

// C C# C++

## Strlen(): tính độ dài chuỗi

Hàm strlen() sẽ trả về độ dài chuỗi ký tự được gán trong chuỗi. Lưu ý rằng đây là độ dài (số ký tự) của chuỗi giá trị gán trong chuỗi, do đó nó sẽ không bao gồm cả ký tự kết thúc chuỗi \0 khi tính độ dài.

Lưu ý là chúng ta cần phải viết thêm dòng #include <cstring> để load header file cstring để có thể sử dụng được hàm strlen() trong chuơng trình.

VD:

#include <iostream>

#include <cstring>

using namespace std;

int main()

{

    char str[] = "I Love C/C++";

    int len;

    len = strlen(str);

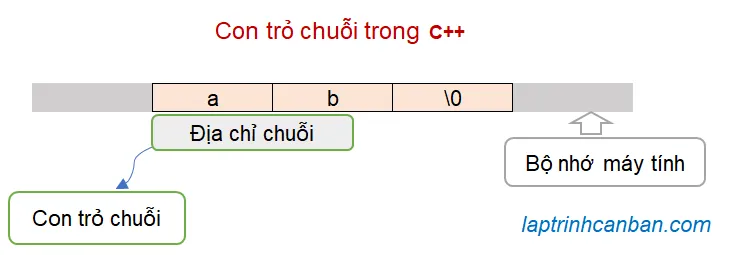
    cout << "Length of "<< str << " is " << len;

    return 0;

}

//Length of "I Love C/C++" is 12.

## Con trỏ chuỗi



### Khai báo

char a[] = "Hello", \*p1;

p1 = a;

char b[100], \*p2;

p2 = b;

### Truy suất các ký tự bằng con trỏ

Cú pháp:

\*(p + index);

| **index** | **Truy xuất bằng chuỗi** | **Truy xuất bằng con trỏ** |
| --- | --- | --- |
| 0 | a[0] | \*p |
| 1 | a[1] | \*(p + 1) |
| 2 | a[2] | \*(p + 2) |
| 3 | a[3] | \*(p + 3) |
| … | … | … |
| n | a[n] | \*(p + n) |

VD

#include <iostream>

using namespace std;

int main(){

    char a[] = "Hello", \*p;

    p = a;

    cout <<  a[2] <<endl;

    cout <<  \*(p +2)<<endl ;

    cout << a[4]<<endl;

    cout <<  \*(p +4)<< endl;

    return 0;

}

l

l

o

o

### Nhập chuỗi bằng con trỏ

char str[100], \*p;

p = str;

cin >> p;

### Xuất chuỗi bằng con trỏ

char str[100], \*p;

p = str;

while(\*p != '\0'){

    cout << \*p;

    ++p;

}

## Chèn chuỗi

Ví dụ như chèn "xyz" vào giữa chuỗi "abcdef" ở vị trí thứ 2

### Cách 1: dịch chuyển phần sau vị trí chèn ra sau và chép chuỗi vào vị trí chèn

* Bắt đầu từ vị trí cần chèn trong chuỗi ban đầu, chúng ta dịch chuyển dần về phía sau lượng ký tự bằng đúng với số ký tự của chuỗi cần chèn và tạo ra các ô trống.
* Sao chép chuỗi ký tự cần chèn để chèn vào các ô trống vừa tạo.

VD

#include <iostream>

#include <cassert>

#include <cstring>

using namespace std;

#define SIZE\_OF\_ARRAY(array)    (sizeof(array)/sizeof(array[0]))

/\*

    Chèn chuỗi trong C++

    Tham số

        str1:      Chuỗi ban đầu (không chấp nhận Null)

        str1\_size: Độ dài chuỗi str1

        pos:       Vị trí chèn (tính từ đầu chuỗi str1)

        str2:      Chuỗi chèn vào (không chấp nhận Null pointer)

    Giá trị trả về

        str1

\*/

char\* str\_insert(char\* str1, size\_t str1\_size, size\_t pos, const char\* str2)

{

    assert(str1 != NULL);

    assert(str2 != NULL);

    size\_t len1 = strlen(str1);

    size\_t len2 = strlen(str2);

    // Kiểm tra str1 có đủ độ dài để chèn str2 không

    assert(len1 + len2 < str1\_size);

    // Kiểm tra vị trí chèn có nằm trong chuỗi str1 không

    assert(len1 >= pos);

    // Làm trống một phạm vi dài với len2 ký tự, từ  str1[pos])

    // Dịch chuyển chuỗi ban đầu về sau, bao gồm cả ký tự `\0'

    memmove(&str1[pos + len2], &str1[pos], len1 - pos + 1);

    // Copy str2 và dán vào khoảng trống mới tạo

    memcpy(&str1[pos], str2, len2);

    return str1;

}

int main()

{

    char str[10] = "abcde";

    cout <<"before: "<< str <<endl;

    cout<< "after: "<<str\_insert(str, SIZE\_OF\_ARRAY(str), 2, "xyz");

    return 0;

}

### Cách 2: cấp phát bộ nhớ động và chép từng phần của chuỗi đích vào bộ nhớ.

#include <iostream>

#include <cassert>

#include <cstdlib>

#include <cstring>

using namespace std;

#define SIZE\_OF\_ARRAY(array)    (sizeof(array)/sizeof(array[0]))

/\*

    Chèn chuỗi trong C++

    Tham số

        str1:      Chuỗi ban đầu (không chấp nhận Null)

        pos:       Vị trí chèn (tính từ đầu chuỗi str1)

        str2:      Chuỗi chèn vào (không chấp nhận Null pointer)

    Giá trị trả về

        Con trỏ mảng chứa kết quả. Nếu thực thi thất bại thì trả về Null

\*/

char\* str\_alloc\_and\_insert(const char\* str1, size\_t pos, const char\* str2)

{

    assert(str1 != NULL);

    assert(str2 != NULL);

    size\_t len1 = strlen(str1);

    size\_t len2 = strlen(str2);

    // Kiểm tra vị trí chèn có nằm trong chuỗi str1 không

    assert(len1 >= pos);

    size\_t result\_size = len1 + len2 + 1;

    char\* result = (char\*) malloc(sizeof(char) \*result\_size);

    if (result == NULL) {

        return NULL;

    }

    // Copy phạm vi từ đầu chuỗi ban đầu tới trước vị trí cần chèn vào chuỗi kết quả

    memcpy(result, str1, pos);

    // Copy chuỗi cần chèn vào cuối kết quả

    memcpy(&result[pos], str2, len2);

    // Copy phạm vi từ vị trí cần chèn tới cuối chuỗi ban đầu vào chuỗi kết quả

    // Thêm ký tự kết thúc chuỗi `\0` vào kết quả

    strcpy(&result[pos + len2], &str1[pos]);

    return result;

}

int main()

{

    char str[] = "abcde";

    cout <<"before: "<< str <<endl;

    char\* result = str\_alloc\_and\_insert(str, 2, "xyz");

    if (result != NULL) {

        cout <<"after: "<< result;

        free(result);

    }

    return 0;

}

## Copy chuỗi

### Strcpy(): copy toàn bộ chuỗi

Cú pháp:

strcpy(target, source);

Trong đó:

* targe là chuỗi đích
* source là chuỗi nguồn.

Hàm strcpy() sẽ copy toàn bộ nội dung của chuỗi source và phục chế nó vào chuỗi target. Lưu ý là hai chuỗi này cần phải tồn tại trước đó, thông qua việc khởi tạo hay khai báo chuỗi trong C++.

VD

#include <iostream>

#include <cstring>

using namespace std;

int main(){

    char str1[30];

    char str2[30] = "I love C/C++";

    cout << "str2: " << str2 <<endl;

    //Copy toàn bộ chuỗi str2 vào chuỗi str1

    strcpy(str1, str2);

    cout << "str1: " << str1;

}

### Strcpy\_c(): copy toàn bộ chuỗi thay thế cho strcpy() ở một số trình biên dịch.

Khi tiến hành compile chương trình chứa hàm strcpy() thì lỗi compile sẽ xảy ra, và có thêm màn hình hướng dẫn chúng ta hãy sử dụng tới hàm strcpy\_c() để thay thế.

Lý do cũng dễ hiểu, bởi việc copy chuỗi bằng hàm strcpy() có một nhược điểm là nếu kích thước chuỗi cần copy lớn hơn kích thước của chuỗi đích chứa nó, thì hiện tượng tràn bộ nhớ sẽ bị xảy ra.

Và để khắc phục điểm yếu này của hàm strcpy() mà hàm strcpy\_c() đã được ra đời. Trong hàm strcpy\_c() có thêm chỉ định kích thước chuỗi đích, giúp chúng ta quản lý và không để xảy ra hiện tượng tràn bộ nhớ trong quá trình copy chuỗi trong C++.

Cú pháp

strcpy\_c(target, rsize\_t, source);

Trong đó:

* target là chuỗi đích
* rsize\_t là kích thước chuỗi đích
* source là chuỗi nguồn.

Lưu ý là chúng ta cũng cần phải viết thêm dòng #include <cstring> để load header file cstring để có thể sử dụng được hàm strcpy\_c() trong chuơng trình.

VD

#include <iostream>

#include <cstring>

using namespace std;

int main() {

    char str1[16] = "Hello World!";

    char str2[16];

    cout << "str2: " << str2 <<endl;

    //Copy toàn bộ chuỗi str2 vào chuỗi str1

    strcpy\_s(str2, 16, str1);

    cout << "str1: " << str1;

    return 0;

}

//Hello World!

### Strncpy(): copy n ký tự

Cú pháp

strncpy(target, source, n);

Trong đó:

* target là chuỗi đích
* source là chuỗi nguồn.
* n là số ký tự lớn nhất có thể copy

Hàm strncpy() sẽ copy từ chuỗi source một số lượng n ký tự, và dán nó vào chuỗi target. Hai chuỗi này cần phải tồn tại trước đó, thông qua việc khởi tạo hay khai báo chuỗi trong C++.

Cần chú ý là nếu chỉ định giá trị của n nhỏ hơn độ dài của chuỗi nguồn khi sử dụng hàm strncpy() thì ký tự kết thúc chuỗi \0 sẽ không được copy vào chuỗi đích, do đó chúng ta cần phải thêm xử lý nối ký tự này vào chuỗi đích target.

Và chúng ta cần phải viết thêm dòng #include <cstring> để load header file cstring để có thể sử dụng được hàm strlen() trong chuơng trình.

VD

#include <iostream>

#include <cstring>

using namespace std;

int main(){

    char str1[30];

    char str2[30] = "I love C++/C++";

    cout << "str2: " << str2 <<endl;

    //Copy nhiều nhất 6 ký tự từ chuỗi str2 vào chuỗi str1

    strncpy(str1, str2,6);

    str1[6] = '\0'; //Gán ký tự kết thúc chuỗi \0

    cout << "str1: " << str1;

}

## Nối chuỗi

## Strcat() nối 2 chuỗi

Cú pháp

strcat(target, source);

Trong đó:

* target là chuỗi đích
* source là chuỗi dùng để nối vào chuỗi target

Hàm strcat() sẽ copy toàn bộ nội dung của chuỗi source, sau đó gán vào vị trí cuối cùng của chuỗi target và trả về kết quả là con trỏ của chuỗi kết quả.

Và chúng ta cần phải viết thêm dòng #include <cstring> để load header file cstring để có thể sử dụng được hàm strcat() trong chuơng trình.

VD

#include <iostream>

#include <cstring>

using namespace std;

int main(){

    char str1[20] = "I Lov";

    char str2[10] = "e You";

    strcat(str1, str2);

    cout <<  str1;

}

//I Love You

## Strncat(): nỗi 2 chuỗi với n ký tự

Cú pháp:

strncat(target, source, n);

Trong đó:

* target là chuỗi đích
* source là chuỗi dùng để nối
* n là số ký tự tối đa được lấy ra từ đầu chuỗi source để nối vào chuỗi target

Hàm strncat() sẽ copy n ký tự tính từ đầu chuỗi source, sau đó gán chúng vào vị trí cuối cùng của chuỗi target và trả về kết quả là con trỏ của chuỗi kết quả.

Và chúng ta cần phải viết thêm dòng #include <cstring> để load header file cstring để có thể sử dụng được hàm strncat() trong chuơng trình.

VD:

#include <iostream>

#include <cstring>

using namespace std;

int main(){

    char str1[20] = "I Lov";

    char str2[10] = "e You";

    strncat(str1, str2,3);

    cout <<  str1;

}

// I Love Y

### Sprintf() nối nhiều chuỗi

Cú pháp:

sprintf (target, \*format, srt1 , srt2 , ...);

Trong đó

* target là chuỗi đích
* srt là các chuỗi dùng để nối vào chuỗi target
* \*format là chuỗi định dạng, là tập hợp các định dạng %s của các chuỗi str. Có bao nhiêu chuỗi str được chỉ định thì có bấy nhiêu định dạng tương ứng của chúng được viết bên trong chuỗi định dạng. Xem thêm: Định dạng nhập xuất trong C++.

VD:

#include <iostream>

#include <cstring>

using namespace std;

int main() {

    char str1[10] = "Hello";

    char str2[10] = "C++";

    char str3[10] = "World!";

    char target[100];

    sprintf(target, "%s %s %s\n", str1, str2,str3);

    cout << target;

    return 0;

}

//Hello C++ World!

## Tách chuỗi

### Strtok()

Hàm strtok() trong C++ là một hàm có sẵn trong header file cstring, giúp chúng ta tách chuỗi trong chuỗi C++ bằng ký tự chỉ định.

Cú pháp:

strtok(str, sep);

Trong đó:

* str là chuỗi cần tách
* sep là ký tự phân tách. (sep viết tắt separator)

Hàm strtok() sẽ trả về vị trí đầu chuỗi của chuỗi con đầu tiên được phân tách bằng dấu phân cách từ chuỗi ban đầu dưới dạng con trỏ chuỗi trong C++. Trong trường hợp không tìm thấy ký tự phân tách , giá trị NULL sẽ được trả về.

Cơ chế của hàm strtok() là điền ký tự kết thúc chuỗi \0 vào chuỗi ban đầu mỗi khi tìm thấy ký tự phân tách, do đó cần lưu ý là hàm strtok() sẽ làm biến đổi chuỗi ban đầu.

VD:

#include <iostream>

#include <cstring>

using namespace std;

int main(){

    char str[50] = "Ha Noi,Ninh Binh,Nam Dinh,Thanh Hoa";

    //Khai báo con trỏ chuỗi để chứa kết quả

    char \* p;

    p = strtok(str, ",");

    cout <<  p;

}

//Ha Noi

Ở ví dụ này, hàm strtok() sẽ hoạt động với các bước như sau:

* Hàm strtok() bắt đầu tìm vị trí ký tự phân tách là dấu , ở bên trong chuỗi str.
* Sau khi tìm thấy vị trí dấu phẩy đầu tiên, hàm strtok() sẽ điền ký tự kết thúc chuỗi \0 vào vị trí tìm thấy. Khi đó, chuỗi ban đầu sẽ chuyển thành dạng Ha Noi\0Ninh Binh,Nam Dinh,Thanh Hoa.
* Hàm strtok() quay trở về đầu chuỗi ban đầu và trả về vị trí đầu chuỗi này dưới dạng con trỏ chuỗi trong C++. Cuối cùng khi in con trỏ chuỗi này, chuỗi sẽ được in ra từ vị trí đầu chuỗi là ký tự H cho đến ký tự thúc chuỗi \0 (Ha Noi\0), do đó chuỗi kết quả Ha Noi sẽ được in ra màn hình.

### Tiếp tục sử dụng strtok() để tách toàn bộ chuỗi

Ở phần trên chúng ta đã biết hàm strtok() trong C++ chỉ có thể giúp chúng ta tách một chuỗi con bằng dấu phân tách từ chuỗi ban đầu trong mỗi lần sử dụng mà thôi.

Do đó, để có thể tách toàn bộ chuỗi con từ chuỗi ban đầu trong C++ thì sau lần tách đầu tiên, chúng ta cần phải tạo ra một vòng lặp để lần lượt tách các chuỗi con con lại từ chuỗi ban đầu bằng hàm strtok().

Tham số NULL được truyền vào để báo hàm tiếp tục tách từ vị trí hoàn thành trước.

VD

#include <iostream>

#include <cstring>

using namespace std;

int main(){

    char str[50] = "Ha Noi,Ninh Binh,Nam Dinh,Thanh Hoa";

    char \* p;

    //Tách chuỗi con lần đầu tiên

    p = strtok(str, ",");

    cout << p <<endl;

    //Tách chuỗi con từ lần thứ 2 trở đi

    //Bằng cách sử dụng hàm strok cho tới khi kết quả NULL được trả về.

    while(p != NULL) {

        //Chỉ dịnh đối số NULL trong hàm strtok để tiếp tục tách chuỗi ban đầu

        p = strtok(NULL, ",");

        if(p != NULL) {

            cout<< p<<endl;

        }

    }

    return 0;

}

Ha Noi

Ninh Binh

Nam Dinh

Thanh Hoa

## Tách số trong chuỗi

Sử dụng isdigit()

VD

#include <iostream>

#include <cctype>

#include <cstring>

using namespace std;

int main(){

    char str[] = "A100B20C3000D40";

    char result[100];

    int i=0;

    int n=0;

    for(i=0; i<=strlen(str); ++i){

        /\* Kiểm tra ký tự thứ i trong chuỗi đã cho có phải là số hay không\*/

        /\* Nếu là số thì lưu vào chuỗi result\*/

        if(isdigit(str[i])){

            result[n] = str[i];

            n++;

            result[n] = '\0';

        }else if(n){ /\* Nếu chuỗi result chứa phần tử thì in chuỗi result\*/

            cout << result<<endl;

            n=0; /\* Reset và làm rỗng chuỗi result\*/

        }

    }

    return 0;

}

100

20

3000

40

## Cắt chuỗi

Phương pháp sử dụng strncpy()

strncpy(target, source + begin, n);

Trong đó:

* target là chuỗi đích để chứa kết quả cắt
* source là chuỗi nguồn.
* begin là vị trí bắt đầu cắt trong chuỗi nguồn
* n là số ký tự sẽ cắt từ vị trí begin.

Hiểu đơn giản thì cú pháp trên sẽ cắt n ký tự từ vị trí begin trong chuỗi source và trả về kết quả là chuỗi target.

Cần chú ý là nếu chỉ định giá trị của n nhỏ hơn độ dài của chuỗi nguồn khi sử dụng hàm strncpy() thì ký tự kết thúc chuỗi \0 sẽ không được copy vào chuỗi đích, do đó chúng ta cần phải thêm xử lý nối ký tự này vào chuỗi đích target.

VD

#include <iostream>

#include <cstring>

using namespace std;

int main() {

    char str[] = "01234567890";

    char t[64];

    strncpy( t, str+3, 5 ); /\*Cắt 5 ký tự từ vị trí thứ 3 \*/

    t[5] = '\0';            /\*Thêm ký tự kết thúc chuỗi vào kết quả\*/

    cout << t ;

}

//34567

## Xoá ký tự trong chuỗi

Phương pháp: lấy từng kí tự trong chuỗi ban đầu ra. Nếu không phải kí tự cần xoá thì lưu và chuỗi đích.

VD

#include<iostream>

using namespace std;

char\* pop\_str(char\* a, char chr, char\* b){

  int a\_cnt;

  int b\_cnt = 0;

  /\* So sánh từng ký tự trong chuỗi a với ký tự C++\*/

  for ( a\_cnt = 0; a[a\_cnt] != '\0'; ++a\_cnt ) {

    /\* Nếu ký tự khác C++ thì lấy ra và thêm vào kết quả \*/

    if ( a[a\_cnt] != chr ) {

      b[b\_cnt] = a[a\_cnt];

      ++b\_cnt;

    }

  }

  b[b\_cnt] = '\0'; /\* Gán ký tự kết thúc chuỗi '\0' \*/

  return b;

}

int main(){

  char a[50] = "abcdcde"; /\* Chuỗi ban đầu\*/

  cout<< "before: "<<a<<endl;

  char b[50]; /\* Chuỗi kết quả\*/

  char chr = 'a'; /\* Ký tự cần xóa\*/

  cout<< "after : "<<pop\_str(a,chr,b)<<endl;

  return 0;

}

## Xoá chuỗi con trong chuỗi

Phương pháp: Tìm kiếm vị trí của chuỗi con trong chuỗi mẹ bằng hàm strstr(), sau đó tách chuỗi mẹ ra từ vị trí tìm thấy, và nối các phần này lại với nhau bằng hàm strcat().

VD

#include <iostream>

#include <cstring>

using namespace std;

void pop\_child\_str(char\* str1,char\* str2){

  const int len = strlen(str2);

  char str3[128] = "";

  char \*p1 = str1, \*p2;

    /\* Tạo vòng lặp để xóa hết chuỗi con \*/

  while((p2 = strstr(p1,str2)) != NULL) { /\*Tìm vị trí chuỗi con bằng hàm strstr\*/

    strncat(str3,p1,p2 - p1);   /\* Nối các phần không chứa chuỗi con \*/

    p1 = p2 + len;      /\* Dịch chuyển con trỏ sang vị trí tìm kiếm tiếp theo \*/

  }

  strcat(str3,p1);

  cout << str3;

}

int main()

{

  char str1[128], str2[128];

  cout << "Nhap chuoi me: ";

  cin >>  str1;

  cout << "Nhap chuoi con can xoa: ";

  cin >>  str2;

  pop\_child\_str(str1,str2);

  return(0);

}

Nhap chuoi me: abcde

Nhap chuoi con can xoa: bc

ade

## Xắp xếp chuỗi

### Xắp xếp các ký tự của chuỗi theo bảng chữ cái

VD

#include <iostream>

#include <cstring>

using namespace std;

int main (void) {

  char str[256] ;

  /\* Nhập các chuỗi vào chương trình \*/

  cout << "Nhap chuoi: ";

  cin >>  str;

  /\* Tìm độ dài chuỗi\*/

  int length = strlen(str);

  /\* Tạo vòng lặp để tiến hành sắp xếp ký tự trong chuỗi\*/

  int i, j;

  char temp;

  for (i = 0; i < length-1; i++) {

      for (j = i+1; j < length; j++) {

         /\* So sánh lần lượt các ký tự và hoán đổi giá trị nhờ biến temp\*/

         if (str[i] > str[j]) {

            temp = str[i];

            str[i] = str[j];

            str[j] = temp;

         }

      }

  }

  cout << "Ket qua: "<< str;

  return 0;

}

Nhap chuoi: akjhu

Ket qua: ahjku

### Sắp xếp các chuỗi theo thứ tự chữ cái

#include<iostream>

#include<cstring>

using namespace std;

int main()

{

  int i, j, num;

  /\* Lệnh nhập số chuỗi cần nhập \*/

  cout << "So chuoi can nhap: ";

  cin >> num;

  /\* Khai báo chuỗi sử dụng để sắp xếp \*/

  char str[num][20], tmp[20];

  /\* Nhập các chuỗi vào chương trình \*/

  cout << "\nHay nhap "<<num<<" chuoi vao chuong trinh\n";

  for (i=0;i<num;i++) {

    cout << "Chuoi thu "<<i+1<<"= ";

    cin >>  str[i];

  }

  /\* Sắp xếp \*/

  for(i=1;i<num;i++){

    for(j=1;j<num;j++){

      if(strcmp(str[j-1], str[j])>0){

        strcpy(tmp, str[j-1]);

        strcpy(str[j-1], str[j]);

        strcpy(str[j], tmp);

      }

    }

  }

  /\* In kết quả sắp xếp\*/

  cout << endl;

  for (i=0;i<num;i++)

    cout <<str[i]<<endl;

  return 0;

}

So chuoi can nhap: 3

Hay nhap 3 chuoi vao chuong trinh

Chuoi thu 1= xyz

Chuoi thu 2= abcd

Chuoi thu 3= mnk

abcd

mnk

xyz

## Đảo ngược chuỗi

#include<iostream>

using namespace std;

/\* Tạo hàm đảo ngược chuỗi trong C++\*/

void str\_reverse(char str[]) {

  int lo = 0;

  int length;

  for (length = 0; str[length] != '\0' ; length++) {}

  while ( lo < length ) {

    --length;

    char tmp = str[lo];

    str[lo] = str[length];

    str[length] = tmp;

    ++lo;

  }

}

int main() {

    char str[100];

    /\* Nhập chuỗi cần đảo ngược từ bàn phím\*/

    cout << "Nhap chuoi: ";

    cin >>  str;

    /\* Gọi hàm đảo ngược chuỗi\*/

    str\_reverse(str);

    /\* In kết quả\*/

    cout << "Chuoi viet nguoc: "<< str;

    return 0;

}

Nhap chuoi: abc123xyz

Chuoi viet nguoc: zyx321cba

## Strstr(): tìm kiếm chuỗi con

Cú pháp:

strstr(str, keyword);

Trong đó:

* str là chuỗi ban đầu
* keyword là chuỗi ký tự cần tìm trong chuỗi str.

Hàm strstr() sẽ trả về địa chỉ của vị trí đầu tiên tìm thấy của chuỗi ký tự keyword trong chuỗi str tính từ đầu chuỗi, và trả về vị trí này dưới dạng con trỏ chuỗi trong C++.

Và chúng ta cần phải viết thêm dòng #include <cstring> để load header file cstring để có thể sử dụng được hàm strstr() trong chuơng trình.

VD

#include <iostream>

#include <cstring>

using namespace std;

int main(){

    char str[30] = "Good School";

    char \* p;

    p = strstr(str, "oo");

    cout << p;

}

//ood School

## Strchr(): Tìm kiếm ký tự trong chuỗi

Cú pháp:

strchr(str, chr);

Trong đó:

* str là chuỗi ban đầu
* chr là ký tự cần tìm trong chuỗi str.

Hàm strchr() sẽ trả về vị trí đầu tiên tìm thấy của ký tự chr trong chuỗi str tính từ đầu chuỗi, và trả về vị trí này dưới dạng con trỏ chuỗi trong C++.

Và chúng ta cần phải viết thêm dòng #include <cstring> để load header file cstring để có thể sử dụng được hàm strchr() trong chuơng trình.

VD:

#include <iostream>

#include <cstring>

using namespace std;

int main(){

    char str[30] = "Good School";

    //Khai báo con trỏ chuỗi để chứa kết quả

    char \* p;

    p = strchr(str, 'o');

    cout <<  p;

}

//ood School

## Strcmp(): so sánh chuỗi

Cú pháp:

strcmp(str1, str2);

Trong đó str1 và str2 là 2 chuỗi cần được so sánh.

Hàm strcmp() sẽ tiến hành so sánh bắt đầu từ ký tự đầu tiên trong hai chuỗi, và nếu chúng giống nhau, các ký tự tiếp theo sẽ lần lượt được so sánh cho tới khi xuất hiện cặp ký tự khác nhau đầu tiên trong hai chuỗi. Khi đó, mã ký tự ASSCII của ký tự này sẽ đại diện cho cả chuỗi và được dùng để so sánh lớn nhỏ.

Sau khi đã xác định được cặp ký tự đầu tiên khác nhau giữa 2 chuỗi, hàm strcmp() sẽ trả về hiệu mã ký tự ASSCII giữa chúng.

#include<iostream>

#include<cstdlib>

#include<cstring>

using namespace std;

int main()

{

  char s1[] = "ab";

  char s2[] = "aAb";

  cout << "Ma ASCII cua a : " << (int)('a')<<endl;

  cout << "Ma ASCII cua A : " << (int)('A')<<endl;

  cout << "Ma ASCII cua b : " << (int)('b')<<endl;

  cout << "Hieu ma ascii: "<<strcmp(s1,s2)<<endl;

  if(strcmp(s1,s2)==0){

       cout <<s1<<" bang "<<s2;

  } else if(strcmp(s1,s2)>0){

       cout <<s1<<" lon hon "<<s2;

  } else

       cout <<s1<<" nho hon "<<s2;

    return 0;

}

Ma ASCII cua a : 97

Ma ASCII cua A : 65

Ma ASCII cua b : 98

Hieu ma ascii: 33

ab lon hon aAb

## Strncmp(): so sánh 2 chuỗi với n ký tự

Cú pháp:

strncmp(str1, str2, n);

Trong đó str1 và str2 là 2 chuỗi cần được so sánh, và n là phạm vi số ký tự tối đa tính từ đầu hai chuỗi được dùng để so sánh.

Giống như Kiyoshi đã hướng dẫn về hàm strcmp() ở trên thì hàm strncmp() cũng sử dụng hiệu mã ký tự ASSCII giữa hai chuỗi để so sánh chúng. Điểm khác biệt duy nhất đó là thay vì so sánh trên toàn bộ 2 chuỗi ban đầu, thì chúng ta chỉ tiến hành so sánh trên một phạm vi được quyết định bởi đối số n khi dùng hàm strcmp() mà thôi.

#include<iostream>

#include<cstdlib>

#include<cstring>

using namespace std;

int main()

{

  char s1[10] = "abA";

  char s2[10] = "aba";

  int n = 2;

  cout << "Ma ASCII cua a : " << (int)('a')<<endl;

  cout << "Ma ASCII cua A : " << (int)('A')<<endl;

  cout << "Ma ASCII cua b : " << (int)('b')<<endl;

  cout << "Hieu ma ascii: "<<strncmp(s1,s2,2)<<endl;

  if(strncmp(s1,s2,2)==0){

    cout << n<<" ky tu dau cua 2 chuoi bang nhau";

  } else if(strncmp(s1,s2,2)>0){

    cout <<n<<" ky tu dau cua "<<s1<<" lon hon "<<s2;

  } else

    cout <<n<<" ky tu dau cua "<<s1<<" nho hon "<<s2;

    return 0;

}

Ma ASCII cua a : 97

Ma ASCII cua A : 65

Ma ASCII cua b : 98

Hieu ma ascii: 0

2 ky tu dau cua 2 chuoi bang nhau

## Hàm isspace():

isspace(char);

Trong đó char là ký tự cần được kiểm tra.

Hàm isspace() sẽ trả về 1 nếu như ký tự được chỉ định là khoảng trắng, và trả về 0 trong các trường hợp còn lại.

Các giá trị được coi là khoảng trắng trong C++ như bảng sau, và nếu ký tự được chỉ định thuộc một trong các giá trị này, thì hàm isspace() sẽ trả về 1.

| **Ký hiệu** | **Code** | **Ý nghĩa** |
| --- | --- | --- |
| \t | 0x09 | tab ngang (TAB) |
| \n | 0x0A | dòng mới(LF) |
| \v | 0x0B | tab dọc (VT) |
| \f | 0x0C | ngắt trang(FF) |
| \r | 0x0D | hồi quy(CR) |
| ‘ ‘ | (0x20) | khoảng trống (SPACE) |

VD

#include<iostream>

#include<cctype>

using namespace std;

int main(){

    char chr[5] = {'\n', '2', 1, ' ', ','};

    for(int i = 0; i < 5; i++){

        if(isspace(chr[i])) {

                cout <<"Vi tri "<< i <<" la khoang trang" << endl;

        }

    }

}

Vi tri 0 la khoang trang

Vi tri 3 la khoang trang

## Xoá khoảng trắng

### Xoá toàn bộ kí tự khoảng trắng

VD

#include <iostream>

#include <cstring>

using namespace std;

char\* delete\_space(char \*str,char \*result) {

  int pos = 0;

  const int len = strlen(str); /\*Tìm độ dài chuỗi ban đầu\*/

  for(int i = 0; i < len; i++) {

      char chr = str[i];

      /\*Nếu ký tự lấy ra là ký tự trắng thì bỏ qua\*/

      if (chr == '\r' || chr == '\n' || chr == ' ') {

        continue;

      }

      /\*Thêm ký tự lấy ra vào kết quả\*/

      result[pos++] = chr;

  }

  return result;

}

int main() {

    char str[] = "   Hello World     ";

    cout << "before: ["<< str<<"]\n";

    char result[100];

    cout << "after : ["<< delete\_space(str,result)<<"]\n";

}

before: [ Hello World ]

after : [HelloWorld]

### Xoá khoảng trắng đầu và khoảng trắng cuỗi

#include <iostream>

#include <cstdlib>

#include <cctype>

#include <cstring>

using namespace std;

char \*trimString(char \*str){

    char \*end; /\*Khai con trỏ của ký tự cuối cùng\*/

    /\*Vòng lặp đầu tiên để xóa khoảng trắng từ đầu chuỗi\*/

    while(isspace((unsigned char)\*str)) str++;

    /\*Nếu vòng lặp thực hiện hết cả chuỗi thì trả về chuỗi str\*/

    if(\*str == 0) return str;

    /\*end là vị trí cuối chuỗi, cũng là vị trí bắt đầu vòng lặp thứ 2\*/

    end = str + strlen(str) - 1;

    /\*Vòng lặp thứ hai để xóa khoảng trắng từ cuối chuỗi\*/

    while(end > str && isspace((unsigned char)\*end)) end--;

    end[1] = '\0';

    return str;

}

int main() {

    const char \*str1 = "   Hello World     ";

    cout <<"Before:["<< str1 <<"]\n";

    /\*Dùng hàm strdup để copy chuỗi ban đầu và trả về con trỏ của chuỗi đó\*/

    /\*Gán con trỏ ở tạo bởi strdup vào con trỏ tmp\*/

    char \*tmp = strdup(str1);

    cout <<"After :[" <<trimString(tmp) <<"]";

    /\*Trách nhiệm giải phóng bộ nhớ thuộc về bên gọi\*/

    free(tmp);

    exit(EXIT\_SUCCESS);

}

Before:[ Hello World ]

After :[Hello World]

## Chuyển số thành chuỗi

### Sprintf()

Cú pháp

sprintf (target, format, num);

Trong đó

* target là chuỗi đích
* num là số cần chuyển thành chuỗi
* format là chuỗi định dạng của số đó. Tuỳ thuộc số này là số nguyên hay số thực mà sẽ có định dạng tương ứng. Xem thêm: Định dạng nhập xuất trong C++.

Bằng cách sử dụng hàm sprintf(), chúng ta có thể thực hiện các phép ép kiểu số trong C++ như chuyển int sang string trong C++, chuyển float sang string trong C++ tuỳ theo định dạng mà chúng ta chỉ định trong hàm.

VD

#include <iostream>

using namespace std;

int main() {

    int num1 = 88;

    float num2 = 10.8;

    //chuyển int sang string trong C++

    char target1[100];

    sprintf(target1, "%d\n", num1);

    cout << target1;

    //chuyển float sang string trong C++

    char target2[100];

    sprintf(target2, "%f\n", num2);

    cout <<target2;

    return 0;

}

### Itoa()

Chuyển số nguyên (kiểu int) sang chuỗi.

Cú pháp:

itoa (num, target, base);

Trong đó

* num là số nguyên cần chuyển thành chuỗi
* target là chuỗi đích để chứa num sau khi chuyển thành chuỗi
* base là cơ số chuyển đổi, nhằm xác định một giá trị số nguyên, chuyển đổi nó thành một giá trị cơ bản và lưu trữ nó trong một bộ đệm. Nếu cơ số là 10 và giá trị là âm, thì chuỗi kết quả được đặt trước dấu trừ ( -).

VD

#include <iostream>

#include <cstdlib>

#include <cstring>

using namespace std;

int main()

{

    int number,l;

    char string[20];

    cout << ("Enter a number: ");

    cin >>  number;

    itoa(number,string,10);

    cout << "String value = "<< string;

    return 0;

}

## Chuyển chuỗi thành số

### Atoi(): chuyển chuỗi sang int

VD

#include <iostream>

#include <cstdlib>

using namespace std;

int main()

{

  /\* Chuỗi chứa số nguyên int\*/

    char str0[] = "123";

  /\* Chuỗi chứa số thực có một phần số nguyên\*/

    char str1[] = "123";

    /\* Chuỗi chứa cả phần có thể và không thể chuyển sang int\*/

    char str2[] = "-123ab";

    /\* Chuỗi chứa ký tự trắng đầu chuỗi\*/

    char str3[] = "    \n\t123ab";

    /\* Chuỗi không thể chuyển sang số nguyên\*/

    char str4[] = "a123b";

    int num0 = atoi(str0);

    int num1 = atoi(str1);

    int num2 = atoi(str2);

    int num3 = atoi(str3);

    int num4 = atoi(str4);

    cout << str0<< " => " << num0 <<endl;

    cout << str1<< " => " << num1 <<endl;

    cout << str2<< " => " << num2 <<endl;

    cout << str3<< " => " << num3 <<endl;

    cout << str4<< " => " << num4 <<endl;

    return 0;

}

123 => 123

123 => 123

-123ab => -123

123ab => 123

a123b => 0

### Atol(): chuyển chuỗi sang long

VD

#include <iostream>

#include <cstdlib>

#include <cstring>

using namespace std;

int main () {

   long val;

   char str[20];

   strcpy(str, "11111.22abac");

   val = atof(str);

   cout <<  str <<" => "<< val;

   return(0);

}

//1111.22abac => 11111

### Atof(): chuyển chuỗi sang float

#include <iostream>

#include <cstdlib>

#include <cstring>

using namespace std;

int main () {

   float val;

   char str[20];

   strcpy(str, "11.22abac");

   val = atof(str);

   cout <<  str <<" => "<< val;

   return(0);

}

// 11.22abac => 11.22

## Tolower(): chuyển chữ hoa sang chữ thường

Cú pháp:

tolower(chr);

Trong đó chr là ký tự cần chuyển về chữ thường trong C++.

Hàm tolower() sẽ trả về chữ thường sau khi chuyển đổi từ ký tự chỉ định nếu có thể, và trả về chính ký tự chỉ định, nếu ký tự đó không thể chuyển về chữ thường.

VD

#include <iostream>

#include  <cctype>

using namespace std;

int main(){

    char chr;

    /\*Trường hợp có thể chuyển về chữ thường\*/

    chr = tolower('H');

    cout <<chr; //h

    chr = tolower('A');

    cout << chr; //a

    /\*Trường hợp không thể chuyển về chữ thường\*/

    chr = tolower('a');

    cout <<  chr; //a

    chr = tolower('8');

    cout <<  chr; //8

    return 0;

}

Toupper(): chuyển chữ thường sang chữ hoa

Cú pháp:

toupper(chr);

Trong đó chr là ký tự cần chuyển về chữ hoa trong C++.

Hàm toupper() sẽ trả về chữ hoa sau khi chuyển đổi từ ký tự chỉ định nếu có thể, và trả về chính ký tự chỉ định, nếu ký tự đó không thể chuyển về chữ hoa.

VD

#include <iostream>

#include  <cctype>

using namespace std;

int main(){

    char chr;

    /\*Trường hợp có thể chuyển về chữ hoa\*/

    chr = toupper('h');

    cout << chr <<endl; //H

    chr = toupper('a');

    cout << chr <<endl;; //A

    /\*Trường hợp không thể chuyển về chữ hoa\*/

    chr = toupper('H');

    cout << chr <<endl; //H

    chr = toupper('!');

    cout << chr <<endl; //!

    return 0;

}

## Tạo hàm chuyển đổi chữ hoa chữ thường

VD

#include <iostream>

using namespace std;

/\*Hàm chuyển chữ thường thành chữ hoa trong C++\*/

char upper(char chr){

    if('a' <= chr && chr <= 'z'){

        chr = chr - ((int)('a') - (int)('A'));

    }

    return chr;

}

/\*Hàm chuyển chữ hoa thành chữ thường trong chr\*/

char lower(char chr){

    if('A' <= chr && chr <= 'Z'){

        chr = chr + ((int)('a') - (int)('A'));

    }

    return chr;

}

int main(){

    char chr;

    chr = lower('L');

    cout <<  chr <<endl;

    chr = upper('l');

    cout <<  chr;

    return 0;

}

l

L

## Nhập chuỗi

### Nhập 1 ký tự

char a;

cin >> a;

### Nhập 1 chuỗi

char str[n];

cin >> str

### Nhập nhiều chuỗi

char str1[n1], str1[n2], str3[n3];

cin >> str1 >> str2 >>str3 ;

VD

#include <iostream>

using namespace std;

int main(){

    cout << "Nhap cac chuoi: ";

    char str1[16], str2[16], str3[16];

    cin >> str1 >> str2 >>str3 ;

    cout <<"Cac chuoi vua nhap: "<<str1<<' '<<str2<<' ' <<str3;

}

Nhap cac chuoi: a bc defg

Cac chuoi vua nhap: a bc defg

### Nhập chuỗi có khoảng trắng

Cú pháp

fgets(str, sizeof(str)+1, stdin);

Trong đó:

* str là biến để chứa chuỗi cần nhập từ bàn phím
* sizeof(str)+1 để chỉ định số ký tự lớn nhất có thể nhập bởi hàm fgets(). Ở đây hàm sizeof(str) sẽ trả về kích cỡ của chuỗi str, bao gồm cả ký tự kết thúc chuỗi \0, nên chúng ta cần cộng thêm 1 đơn vị vào.
* stdin được sử dụng như là con trỏ file trong hàm fgets(). stdin là viết tắt của file header “iostream”, giúp chúng ta có thể nhận đầu vào tiêu chuẩn từ bàn phím.

VD

#include <iostream>

using namespace std;

int main(){

    cout << "Nhap chuoi co ky tu trang: ";

    char str[10];

    fgets(str, sizeof(str)+1, stdin);

    cout << "Chuoi vua nhap: ";

    cout << str;

    return 0;

}

Nhap chuoi co ky tu trang: Viet Nam

Chuoi vua nhap: Viet Nam

## Tạo chuỗi ngẫu nhiên

VD

#include <iostream>

#include <cstdlib>

#include <time.h>

#include <cstring>

using namespace std;

void rand\_text(int length, char \*result) {

    int i, rand\_int;

    char char\_set[] = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz&quot";

    for (i = 0; i <length; i++) {

        result[i] = char\_set[rand() % sizeof(char\_set)];

    }

    result[length] = 0;

}

int main(){

    cout << "Nhap đo dai chuoi: ";

    int length;

    cin >> length;

    char result[length + 1];

    srand(time(NULL));

    rand\_text(length, result);

    cout << result;

}

// Kết quả lần 1:

Nhap đo dai chuoi: 5

XtoI

// kết quả lần 2

Nhap đo dai chuoi: 5

v9DBu

# String

## Tổng quát

std::string là một class với các chức năng linh hoạt được sử dụng làm tiêu chuẩn để xử lý chuỗi trong C++.

Ưu điểm lớn nhất của std::string đó chính là khả năng tự thay đổi kích thước.

Mặc dù chúng ta có thể sử dụng mảng ký tự để xử lý chuỗi string trong C++, tuy nhiên khác với mảng ký tự có kích thước cố định và chúng ta rất khó có thể thao tác và thay đổi kích thước của chuỗi string lưu trong nó, thì kích thước của std::string là kích thước động, do vậy chúng ta có thể linh hoạt hơn khi xử lý chuỗi ký tự được lưu trong nó.

std::string được cài sẵn trong header file string và để sử dụng được chức năng này, chúng ta cần thêm dòng 「#include <string>」 vào đầu chương trình.

VD

#include <string>

using namespace std;

int main()

{

    string str1;

    string str2;

}

## Khởi tạo

### Cách 1: sử dụng dấu ngoặc {}

VD:

std::string name{"Kiyoshi"};

### Cách 2: sử dụng dấu =

VD

std::string job = "BrSE";

### Cách 3: Khởi tạo string với n ký tự lặp lại

Cú pháp:

std::string str(n, chr);

Trong đó str là tên biến string, chr là ký tự cần lặp và n là số ký tự lặp trong string cần khởi tạo.

VD

std::string str(10, 'a'); // aaaaaaaaaa

## Truy xuất ký tự trong string

Cú pháp:

str[index];

Trong đó str là tên biến string và index là vị trí của ký tự cần truy xuất.

VD

#include <iostream>

#include <string>

using namespace std;

int main()

{

    string str{"Love"};

    cout << str[0] <<endl;

    cout << str[1] <<endl;

    cout << str[2] <<endl;

    cout << str[3] <<endl;

    cout << str[4] <<endl;

    return 0;

}

L

o

v

e

## Lấy kích thước string

### Length()

Cú pháp

str.length();

Trong đó str là chuỗi cần kiểm tra.

Hàm length sẽ trả về tổng số byte được tạo thành từ các ký tự tạo nên chuỗi. Lưu ý đây không phải là kích thước thực của string trong bộ nhớ, mà là kích thước được tính từ tổng các thành phần tạo nên string.

VD

#include <iostream>

#include <string>

using namespace std;

int main ()

{

    string str1("Hello world");

    cout<<str1.length()<<endl;

    string str2= "Tokyo";

    cout<<str2.length()<<endl;

    return 0;

}

11

5

### Size()

str.size();

Trong đó str là chuỗi cần kiểm tra.

Hàm size sẽ trả về tổng số byte được tạo thành từ các ký tự tạo nên chuỗi. Lưu ý đây không phải là kích thước thực của string trong bộ nhớ, mà là kích thước được tính từ tổng các thành phần tạo nên string.

VD

#include <iostream>

#include <string>

using namespace std;

int main ()

{

    string str1("Hello world");

    cout<<str1.size()<<endl;

    string str2= "Tokyo";

    cout<<str2.size()<<endl;

    return 0;

}

11

5

Hàm size và hàm length là các hàm đồng nghĩa, và giá trị trả về của chúng đều giống nhau. Bởi vậy hai hàm này trong C++ thực ra là giống nhau, và chúng ta có thể lựa chọn bất kỳ một trong số chúng để lấy kích thước string trong C++.

## Nhập xuất string

### Nhập string

string str;

cin >> str;

### Nhập nhiều string

cin >> str1 >> str2 >>str3 ;

VD

#include <iostream>

#include <cstring>

using namespace std;

int main(){

    cout << "Nhap cac chuoi: ";

    string str1, str2, str3;

    cin >> str1 >> str2 >>str3 ;

    cout <<"Cac chuoi vua nhap: "<<str1<<' '<<str2<<' ' <<str3;

}

Nhap cac chuoi: ab cd123 ef8

Cac chuoi vua nhap: ab cd123 ef8

### Nhập string có khoảng trắng

std::getline(std::cin, str, delimiter );

Trong đó:

* std::cin để chỉ định stream nhận dữ liệu, giúp chúng ta có thể nhận đầu vào tiêu chuẩn từ bàn phím.
* str là tên biến string sẽ lưu string nhập vào.
* delimiter là ký tự phân tách sử dụng để tách string nhập vào. Nếu delimiter được lược bỏ thì sử dụng giá trị mặc định là ký tự xuống dòng.

VD

#include <iostream>

#include <cstring>

using namespace std;

int main(){

    cout << "Nhap chuoi co ky tu trang: ";

    string str;

    getline(cin,str);

    cout << "Chuoi vua nhap: ";

    cout << str;

    return 0;

}

Nhap chuoi co ky tu trang: I love you

Chuoi vua nhap: I love you

### Xuất string

cout >> str;

## Thêm ký tự

### Nối string bằng toán tử +

Để nối string trong C++ chúng ta sử dụng toán tử + với cú pháp sau đây:

str = str1 + str2 + str3 + ...;

VD3

#include <iostream>

#include <string>

using namespace std;

int main ()

{

  string str;

  string str1= "Hello";

  string str2= "\t";

  string str3= "Vietnam";

  str = str1 + str2 + str3;

  cout << str;

  return 0;

}

Hello Vietnam

### Hàm push\_back(): Thêm ký tự vào cuối tring C++ bằng

Cú pháp:

str.push\_back(chr);

Trong đó str là tên string cần thêm ký tự chr vào vị trí cuối cùng

Hàm push\_back thuộc kiểu void, do đó nó sẽ không trả giá trị. Bởi vậy nó sẽ thêm ký tự vào vị trí cuối cùng trong string chứ không trả về chuỗi kết quả.

#include <iostream>

#include <string>

using namespace std;

int main ()

{

  string str ("Hello world");

  str.push\_back('a');

  cout << str;

  return 0;

}

//> Hello worlda

### Thêm chuỗi ký tự vào cuối tring C++ bằng toán tử gán

Cú pháp

str += str1;

Trong đó str1 là một hoặc một chuỗi ký tự cần thêm vào cuối string str.

VD

#include <iostream>

#include <string>

using namespace std;

int main ()

{

  string str ("abcd");

  str += "e"; //Thêm ký tự a vào cuối string

  cout << str <<endl;

  str += "gkh";//Thêm chuỗi gkh vào cuối string

  cout << str;

  return 0;

}

abcde

abcdegkh

### Chèn 1 ký tự vào vị trí chỉ định trong string C++ bằng hàm insert

Cú pháp:

str.insert(p, chr);

Trong đó str là string ban đầu, chr là ký tự cần chèn, và p là trình lặp trỏ đến vị trí cần chèn trong string.

Ở đây trình lặp (iterator) là một vòng lặp có tác dụng giống như con trỏ, giúp truy cập đến các ký tự ở vị trí cụ thể trong string.

Trong trường hợp cần chỉ đến vị trí index thứ n trong string, chúng ta sẽ viết trình lặp p như sau:

str.insert(str.begin() + i, chr);

Trong đó str.begin() chỉ đến vị trí đầu tiên trong string, và i là index của vị trí ký tự cần chỉ đến.

#include <iostream>

#include <string>

using namespace std;

int main ()

{

  string str ("abcd");

  str.insert(str.begin()+2, 'g'); //Thêm ký tự 'g' vào vị trí thứ 2

  cout << str <<endl;

 str.insert(str.begin()+4, 'H'); //Thêm ký tự 'H' vào vị trí thứ 4

  cout << str;

  return 0;

}

abgcd

abgcHd

### Chèn 1 chuỗi ký tự vào vị trí chỉ định trong string C++ bằng hàm insert

str.insert(position, str1);

Trong đó str là string ban đầu, str1 là string cần chèn vào str, và position là vị trí cần chèn vào string str.

#include <iostream>

#include <string>

using namespace std;

int main ()

{

  string str("abcd");

  string str2("123");

  string str3("FG");

  str.insert(2, str2); //Thêm str2 vào vị trí thứ 2

  cout << str <<endl;

  str.insert(4, str3); //Thêm str3 vào vị trí thứ 4

  cout << str;

  return 0;

}

ab123cd

ab12FG3cd

## Xoá ký tự

### pop\_back()

Hàm pop\_back là một hàm thành viên trong class std:string, có tác dụng xóa ký tự cuối cùng.

Cú pháp sử dụng hàm pop\_back để xóa ký tự cuối cùng trong string như sau:

str.pop\_back();

Trong đó str là tên string cần xóa ký tự cuối cùng. Lưu ý là ký tự cuối cùng ở đây không bao gồm ký tự kết thúc chuỗi \0.

Hàm pop\_back thuộc kiểu void, do đó nó sẽ không trả giá trị. Bởi vậy nó sẽ xóa ký tự cuối cùng trong string chứ không trả về ký tự đó.

#include <iostream>

#include <string>

int main ()

{

  std::string str ("Hello world");

  str.pop\_back();

  std::cout << str;

  return 0;

}

//> Hello worl

### Xóa 1 ký tự trong string bằng hàm erase

Hàm erase là một hàm thành viên trong class std:string, có tác dụng xóa một hoặc nhiều ký tự.

Cú pháp:

str.erase(p);

Trong đó str là string ban đầu, và p là trình lặp trỏ đến vị trí cần xóa trong string.

Trong trường hợp cần chỉ đến vị trí index thứ n trong string, chúng ta sẽ viết trình lặp p như sau:

str.erase(str.begin() + i);

Trong đó str.begin() chỉ đến vị trí đầu tiên trong string, và i là index của vị trí ký tự cần chỉ đến.

VD

#include <iostream>

#include <string>

using namespace std;

int main ()

{

    string str("Hello World");

    str.erase(str.begin() + 2);//Xóa ký tự ở vị trí thứ 2

    cout << str<<endl;

    str.erase(str.begin() + 6);//Xóa ký tự ở vị trí thứ 6

    cout << str;

    return 0;

}

Helo World

Helo Wrld

### Xóa các ký tự trong một phạm vi chỉ định trong string C++

Cú pháp:

str.erase( iterator\_first, iterator\_last);

Trong đó str là chuỗi ban đầu, iterator\_first và iterator\_last là phạm vi cần xóa được chỉ định trong trình lặp trỏ đến vị trí cần xóa trong string.

Tương tự như khi xóa 1 ký tự thì cú pháp sử dụng thực tế để xóa các ký tự trong một phạm vi chỉ định trong string C++ bằng hàm erase sẽ là:

str.erase(str.begin() + start, str.begin() + end);

Trong đó start và end là vị trí index của phạm vi xóa trong string.

VD

#include <iostream>

#include <string>

using namespace std;

int main ()

{

    string str("Hello World");

    //Xóa ký tự trong phạm vi index từ 1 đến 3

    str.erase(str.begin() + 1, str.begin() + 3);

    cout << str<<endl;

    //Xóa ký tự trong phạm vi index từ 2 đến 5

    str.erase(str.begin() + 2, str.begin() +5);

    cout << str;

    return 0;

}

## Kiểm tra chuỗi

### Kiểm tra chuỗi rỗng bằng hàm empty

Hàm empty là một hàm thành viên trong class std:string, có tác dụng kiểm tra chuỗi rỗng trong C++.

Cú pháp:

str.empty();

Trong đó str là chuỗi string cần kiểm tra.

Hàm empty sẽ trả về 1 (true) nếu string rỗng, và 0 (false) trong trường hơp còn lại.

#include <iostream>

#include <string>

using namespace std;

int main ()

{

    string str1 ("Hello world");

    cout<<str1.empty()<<endl;

    string str2;

    cout<<str2.empty()<<endl;

    return 0;

}

0

1

### Kiểm tra chuỗi rỗng trong C++ bằng độ dài của nó

Ý tưởng ở đây là nếu một string là chuỗi rỗng, thì độ dài của nó bằng 0, nên chỉ cần tìm được độ dài của nó là có thể xác định được nó rỗng hay không.

VD

#include <iostream>

#include <string>

using namespace std;

void check\_empty\_string(string str){

    if (str.length() == 0) cout << "Chuoi rong"<<endl;

    else cout <<"Khong phai chuoi rong"<<endl;

}

int main ()

{

    string str ("Hello world");

    check\_empty\_string(str);

    string str2;

    check\_empty\_string(str2);

    return 0;

}

Khong phai chuoi rong

Chuoi rong

## Hàm substr(): copy chuỗi con từ chuỗi string ban đầu.

Hàm substr là một hàm thành viên trong class std:string

Cú pháp:

str.substr(start, length);

Trong đó:

* str là chuỗi ban đầu
* start là vị trí bắt đầu copy
* length là độ dài (số ký tự) cần copy. Nếu không chỉ định length thì mặc định sẽ copy từ vị trí chỉ định cho tới cuối chuỗi ban đầu.

Substr() dùng copy chuỗi con từ chuỗi ban đầu và không làm thay đổi chuỗi ban đầu.

VD1: Copy string từ vị trí chỉ định tới cuối

#include <iostream>

#include <string>

using namespace std;

int main() {

    string str = "Hello World!";

    string substr = str.substr(6);

    cout << substr << endl;

    return 0;

}

World!

VD2 Copy một phạm vi trong string

#include <iostream>

#include <string>

using namespace std;

int main() {

    string str = "Hello World!";

    string substr = str.substr(1, 3);

    cout << substr << endl;

    string substr2 = str.substr(1,6);

    cout << substr2 << endl;

    return 0;

}

ell

ello W

## Hàm copy(): copy chuỗi con từ chuỗi string ban đầu.

Hàm copy là một hàm thành viên trong class std:string, có tác dụng sao chép một phần hoặc toàn bộ chuỗi string ban đầu.

Cú pháp:

src.copy(des, length, pos);

Trong đó:

* src là chuỗi string ban đầu
* des là mảng ký tự chứa kết quả.
* pos là vị trí bắt đầu copy. Nếu lược bỏ pos thì sẽ bắt đầu copy từ đầu chuỗi string
* length là độ dài (số ký tự) cần copy.

Hàm copy sẽ trả về số ký tự đã copy từ chuỗi src vào mảng ký tự des.

Lưu ý, trong trường hợp chúng ta copy một phần của chuỗi ban đầu thì ký tự kết thúc chuỗi \0 sẽ không bao gồm trong kết quả, vì thế đừng quên thêm ký tự \0 sau kết quả sao chép chuỗi nhé.

### Copy n ký tự từ đầu chuỗi string

Để copy n ký tự từ đầu chuỗi string, chúng ta lược bỏ đi đối số pos khi sử dụng hàm copy trong C++. Ví dụ cụ thể:

#include <iostream>

#include <string>

using namespace std;

int main ()

{

    //String nguồn

    string str("Hello world");

    //Khai báo mảng ký tự chứa kết quả

    char des[100];

    size\_t length;

    length= str.copy(des,6); //Copy 6 ký tự từ đầu chuỗi string vào des

    des[length]='\0'; //Thêm ký tự kết thúc chuỗi

    cout<<des<<endl;

    length= str.copy(des,3); //Copy 3 ký tự từ đầu chuỗi string vào des

    des[length]='\0'; //Thêm ký tự kết thúc chuỗi

    cout<<des<<endl;

    return 0;

}

Hello

Hel

### Copy n ký tự từ vị trí m trong string

Khi chỉ định đầy đủ các đối số pos và length trong hàm copy, chúng ta có thể tiến hành copy n ký tự từ vị trí m trong string. Ví dụ cụ thể:

#include <iostream>

#include <string>

using namespace std;

int main ()

{

    //String nguồn

    string str("Hello world");

    //Khai báo mảng ký tự chứa kết quả

    char des[100];

    size\_t length;

    //Copy 3 ký tự từ vị trí thứ 2 vào des

    length= str.copy(des,3, 2);

    des[length]='\0';

    cout<<des<<endl;

    //Copy 6 ký tự từ vị trí thứ 2 vào des

    length= str.copy(des,6, 2);

    des[length]='\0';

    cout<<des<<endl;

    return 0;

}

llo

llo wo

### Copy toàn bộ chuỗi string trong C++

Để copy toàn bộ chuỗi string trong C++, chúng ta cần phải chỉ định số lượng ký tự cần copy chính bằng độ dài chuỗi string ban đầu là xong. Ví dụ cụ thể:

#include <iostream>

#include <string>

using namespace std;

int main ()

{

    //String nguồn

    string str("Hello world");

    //Khai báo mảng ký tự chứa kết quả

    char des[100];

    size\_t length;

    //Copy toàn bộ chuỗi string vào des

    length= str.copy(des,str.size());

    des[length]='\0';

    cout<<des<<endl;

    return 0;

}

Hello world

## Tách string

### Tách string bằng một ký tự phân cách

Ý tưởng: Sử dụng vòng lặp, tìm các vị trí của ký tự phân tách trong chuỗi bằng hàm find() và tách chuỗi ra tại các vị trí tìm thấy.

VD

#include <iostream>

using namespace std;

string split(string str, string delimiter){

    size\_t pos = 0;

    string token;

    while ((pos = str.find(delimiter)) != string::npos) {

        token = str.substr(0, pos);

        cout << token << endl;

        str.erase(0, pos + delimiter.length());

    }

    return str;

}

int main() {

    //Tách string bằng ký tự trống

    cout << split("Hello new world!", " ") << endl;

    //Tách string bằng dấu tab

    cout << split("Tom\tand\tJerry", "\t") << endl;

    return 0;

}

Hello

new

world!

Tom

and

Jerry

### Tách string bằng một chuỗi ký tự phân cách

Ý tưởng: Sử dụng vòng lặp, tìm các vị trí của ký tự phân tách trong chuỗi bằng hàm find() và tách chuỗi ra tại các vị trí tìm thấy.

#include <iostream>

using namespace std;

string split(string str, string delimiter){

    size\_t pos = 0;

    string token;

    while ((pos = str.find(delimiter)) != string::npos) {

        token = str.substr(0, pos);

        cout << token << endl;

        str.erase(0, pos + delimiter.length());

    }

    return str;

}

int main() {

    //Tách string bằng chuỗi ký tự " and "

    cout << split("Tom and you and me", " and ") << endl;

    //Tách string bằng chuỗi ký tự "\*\_\*"

    cout << split("Good\*\_\*Morning\*\_\*MyLove", "\*\_\*") << endl;

    return 0;

}

Tom

you

me

Good

Morning

MyLove

## Cắt chuỗi

### Cắt ký tự đầu tiên từ string trong C++ bằng hàm front

Hàm front là một hàm thành viên trong class std:string, có tác dụng tham chiếu đến ký tự đầu tiên trong chuỗi string.

Chúng ta sử dụng hàm front trong C++ với cú pháp sau đây:

str.front();

Trong đó str là chuỗi string cần cắt ra ký tự ban đầu.

Hàm front() sẽ trả về tham chiếu tới ký tự đầu tiên của chuỗi string, qua đó chúng ta có thể cắt lấy giá trị của ký tự đầu tiên này, hoặc là gán một ký tự khác để thay đổi nó.

#include <iostream>

#include <string>

using namespace std;

int main() {

    string str = "Hello World!";

    //Tạo tham chiếu tới vị trí ký tự đầu tiên của chuỗi string

    char frontstr = str.front();

    cout << frontstr << endl;

    //Thay đổi ký tự đầu tiên

    str.front() = 'K';

    cout << str << endl;

    return 0;

}

H

Kello World!

### Cắt ký tự cuối cùng từ string trong C++ bằng hàm back

Ngược lại với hàm front() chính là hàm back().

Hàm back là một hàm thành viên trong class std:string, có tác dụng tham chiếu đến ký tự cuối cùng trong chuỗi string.

Chúng ta sử dụng hàm back trong C++ với cú pháp sau đây:

str.back();

Trong đó str là chuỗi string cần cắt ra ký tự cuối cùng.

Hàm back() sẽ trả về tham chiếu tới ký tự cuối cùng của chuỗi string, qua đó chúng ta có thể cắt lấy giá trị của ký tự cuối cùng này, hoặc là gán một ký tự khác để thay đổi nó.

Lưu ý chúng ta không dùng hàm back() cho chuỗi string trống nhé.

VD

#include <iostream>

#include <string>

using namespace std;

int main() {

    string str = "Hello World!";

    //Tạo tham chiếu tới vị trí ký tự cuối cùng của chuỗi string

    char backstr = str.back();

    cout << backstr << endl;

    //Thay đổi ký tự cuối cùng.

    str.back() = '?';

    cout << str << endl;

    return 0;

}

!

Hello World?

## Chuyển đổi kiểu dữ liệu

### Chuyển số sang string trong C++ bằng hàm template

#include <sstream>

template <typename T> std::string tostr(const T& t)

{

    std::ostringstream os; os<<t; return os.str();

}

Hàm template trên có tác dụng chuyển đổi kiểu số bao gồm cả int lẫn float sang kiểu string. Sau khi đã khai báo template này, chúng ta có thể sử dụng nó nhiều lần trong chương trình.

Ví dụ cụ thể, chúng ta có thể chuyển float và int sang string như chương trình sau. Lưu ý chúng ta sử dụng tới hàm typeid.name() để kiểm tra kiểu của dữ liệu trước và sau khi thay đổi bằng hàm template.

#include <iostream>

#include <sstream>

#include <typeinfo>

using namespace std;

template <typename T> string tostr(const T& t)

{

    ostringstream os; os<<t; return os.str();

}

int main() {

    int a = 123;

    float b = 3.45;

    cout << "Kieu cua a: "<< typeid(a).name() <<endl;

    cout << "Kieu cua b: "<<typeid(b).name() <<endl;

    //Kiểu sau khi thay đổi

    cout << "Kieu thay doi cua a: "<< typeid(tostr(123)).name() <<endl;

    cout << "Kieu thay doi cua b: "<<typeid(tostr(3.1415)).name() <<endl;

    return 0;

}

Kieu cua a: i

Kieu cua b: f

Kieu thay doi cua a: NSt7\_\_cxx1112basic\_stringIcSt11char\_traitsIcESaIcEEE

Kieu thay doi cua b: NSt7\_\_cxx1112basic\_stringIcSt11char\_traitsIcESaIcEEE

### Chuyển số sang string trong C++ bằng hàm to\_string

Để sử dụng hàm template ở trên, chúng ta đều phải khai báo template ở đầu mỗi chương trình, và chuyện này thật là rườm rà phải không nào? Do đó, từ phiên bản C++11 trở đi, các nhà phát triển C++ đã thêm vào một hàm mới có chức năng tương tự nhưng lại có thể sử dụng dễ dàng hơn, đó chính là hàm to\_string.

Hàm to\_string có thể chuyển tất cả các định dạng kiểu số như int, float, double sang string trong C++ với cú pháp đơn giản và ngắn gọn.

Chúng ta sử dụng hàm to\_string() trong C++ với cú pháp sau đây:

to\_string(num);

Trong đó num là số cần chuyển sang kiểu chuỗi string.

Ví dụ:

#include <iostream>

#include <typeinfo>

using namespace std;

int main() {

    int a = 123;

    float b = 3.45;

    cout << "Kieu cua a: "<< typeid(a).name() <<endl;

    cout << "Kieu cua b: "<<typeid(b).name() <<endl;

    //Kiểu sau khi thay đổi

    cout << "Kieu thay doi cua a: "<< typeid(to\_string(123)).name() <<endl;

    cout << "Kieu thay doi cua b: "<<typeid(to\_string(3.1415)).name() <<endl;

    return 0;

}

Kieu cua a: i

Kieu cua b: f

Kieu thay doi cua a: NSt7\_\_cxx1112basic\_stringIcSt11char\_traitsIcESaIcEEE

Kieu thay doi cua b: NSt7\_\_cxx1112basic\_stringIcSt11char\_traitsIcESaIcEEE

## Tìm ký tự hoặc chuỗi trong chuỗi

### Tìm ký tự hoặc chuỗi ký tự từ đầu string trong C++ bằng hàm find.

Hàm find là một hàm thành viên trong class std:string, có tác dụng vị trí xuất hiện đầu tiên của một hoặc một chuỗi ký tự chỉ định từ đầu tới cuối string trong C++.

Cú pháp của hàm find trong C++ như sau:

base.find(str, pos);

Trong đó:

* base là chuỗi ban đầu
* str là ký tự hoặc chuỗi string cần tìm từ trong base
* pos là vị trí bắt đầu tìm kiếm. Nếu lược bỏ pos thì sẽ tiến hành tìm từ đầu string.

Hàm find sẽ trả về vị trí index tìm thấy đầu tiên của đối tượng cần tìm kiếm trong string ban đầu. Và nếu đối tượng đó không tồn tại trong string, giá trị lớn nhất của kiểu dữ liệu size\_t (lấy giá trị bằng: string::npos ) sẽ được trả về.

Ví dụ:

#include <iostream>

#include <cstring>

using namespace std;

int main(){

    string base = "Good School";

    //Tìm vị trí xuất hiện đầu tiên từ đầu string của ký tự

    cout << base.find('o') << endl;

    //Tìm vị trí xuất hiện đầu tiên từ đầu string của chuỗi ký tự

    cout << base.find("oo") << endl;

    //Tìm vị trí xuất hiện đầu tiên từ vị trí thứ 3 đến cuối string

    cout << base.find("oo", 3) << endl;

    //Kết quả khi không tìm thấy ký tự chỉ định trong string

    cout << base.find("m") << endl;

    return 0;

}

Vi tri tim thay ky tu o:1

Vi tri tim thay chuoi oo:1

Vi tri tim thay chuoi oo tinh tu ky tu thu 3:8

Vi tri tim thay ky tu m:18446744073709551615

VD: so sánh giá trị trả về với string::npos trong tạo hàm split (tách chuỗi) để biết find() có tìm được vị trí của ký tự hay chuỗi cần tìm hay không.

string split(string str, string delimiter){

    size\_t pos = 0;

    string token;

    while ((pos = str.find(delimiter)) != string::npos) {

        token = str.substr(0, pos);

        cout << token << endl;

        str.erase(0, pos + delimiter.length());

    }

    return str;

}

### Tìm ký tự hoặc chuỗi ký tự từ cuối string trong C++ bằng hàm rfind.

Ngược lại với hàm find ở trên thì chúng ta có hàm rfind() để giúp tìm kiếm từ cuối string trở về đầu string.

Cú pháp của hàm rfind trong C++ như sau:

base.rfind(str, pos);

Trong đó:

* base là chuỗi ban đầu
* str là ký tự hoặc chuỗi string cần tìm từ trong base
* pos là vị trí bắt đầu tìm kiếm. Nếu lược bỏ pos thì sẽ tiến hành tìm từ cuối string trở về đầu string.

Hàm rfind sẽ trả về vị trí index tìm thấy đầu tiên của đối tượng cần tìm kiếm trong string ban đầu. Và nếu đối tượng đó không tồn tại trong string, giá trị lớn nhất của kiểu dữ liệu size\_t sẽ được trả về.

Ví dụ cụ thể, chúng ta sử dụng hàm rfind để tìm ký tự hoặc chuỗi ký tự trong nhiều trường hợp khác nhau như sau:

#include <iostream>

#include <cstring>

using namespace std;

int main(){

    string base = "Good School";

    //Tìm vị trí xuất hiện đầu tiên từ cuối string của ký tự

    cout << base.rfind('o') << endl;

    //Tìm vị trí xuất hiện đầu tiên từ đầu string của chuỗi ký tự

    cout << base.rfind("oo") << endl;

    //Tìm vị trí xuất hiện đầu tiên từ vị trí thứ 3 lên đầu string

    cout << base.rfind("oo", 3) << endl;

    //Kết quả khi không tìm thấy ký tự chỉ định trong string

    cout << base.rfind("m") << endl;

    return 0;

}

9

8

1

18446744073709551615

## Hàm replace trong C++

Hàm replace là một hàm thành viên trong class std:string, có tác dụng thay thế một phạm vi trong string ban đầu bằng một nội dung mới. Phép thay thế chuỗi string bằng hàm replace sẽ làm thay đổi string ban đầu.

Cú pháp của hàm replace trong C++ như sau:

base.replace(pos,len,str);

Trong đó:

* base là chuỗi ban đầu
* pos là vị trí bắt đầu thay thế Nếu lược bỏ pos thì sẽ tiến hành thay thế từ đầu string.
* len là số ký tự của phạm vi cần thay thế tính từ vị trí pos.
* str là chuỗi ký tự dùng để thay thế phạm vi ở trên

Hàm replace sẽ thay thế một phạm vi bắt đầu từ vị trí pos và có độ dài len ký tự, bằng nội dung mới là chuỗi str, và trả về chuỗi ban đầu sau khi đã được thay thế.

### Thay thế một chuỗi ký tự tại phạm vi chỉ định trong string C++ bằng hàm replace

#include <iostream>

#include <cstring>

using namespace std;

int main(){

    string base = "Good School";

    base.replace(5, 6, "Day");

    cout <<  base << endl;

    return 0;

}

Good Day

### Thay thế ký tự tại vị trí chỉ định trong string C++ bằng hàm replace

Để thay thế ký tự tại vị trí chỉ định trong string C++, chúng ta chỉ định đối số độ dài len bằng 1.

#include <iostream>

#include <cstring>

using namespace std;

int main(){

    string base = "Good School";

    base.replace(2, 1, "AAA");

    cout <<  base << endl;

    return 0;

}

GoAAAd School

### Thay thế toàn bộ ký tự trong string trong C++ bằng hàm replace

Để thay thế toàn bộ ký tự trong string trong C++, chúng ta chỉ định phạm vi thay thế là toàn bộ chuỗi ban đầu, với vị trí bắt đầu pos bằng 0, và độ dài len bằng chính độ dài chuỗi ban đầu.

#include <iostream>

#include <cstring>

using namespace std;

int main(){

    string base = "Good School";

    base.replace(0, base.length(), "Bad boy");

    cout <<  base << endl;

    return 0;

}

Bad boy

# Con trỏ

## Khai báo

type \*p;

Trong đó type là kiểu dữ liệu của con trỏ, và p là tên con trỏ. Lưu ý là kiểu dữ liệu của con trỏ phải giống với kiểu dữ liệu của dữ liệu cần lưu địa chỉ trong con trỏ.

VD

int \*p1, \*p2, \*p3;

char \*p4, \*p5;

## Khởi tạo

type \*p = &va ;

VD

#include <iostream>

using namespace std;

int main() {

    int num = 10;

    int \*p = &num;

    cout << "pointer: "<< p <<endl;

    cout << "num    :"<< num;

    return 0;

}

//pointer: 0xbd621ff7a4

//num    :10

## Sử dụng

### Thao tác đến giá trị của biến mà con trỏ trỏ tới

\*p;

VD

#include <iostream>

using namespace std;

int main() {

    int x = 10;

    int \*p = &x;

    /\* Lấy giá trị tại địa chỉ lưu trong con trỏ C++\*/

    cout << "Gia tri tai đia chi luu trong con tro= "<< \*p;

    return 0;

}

Gia tri tai đia chi luu trong con tro= 10

### Thao tác với địa chỉ lưu trong con trỏ

Giá trị của con trỏ chính là địa chỉ của biến mà con trỏ trỏ đến. Do đó, thao tác với giá trị của biến con trỏ chính là thao tác với địa chỉ mà nó lưu.

p

VD

#include <iostream>

using namespace std;

int main() {

  char chr = 'a';

  int num = 1;

  char \*p1 = &chr;

  int \*p2 = &num;

  cout << "Gia tri ban dau cua con tro kieu char: "<<p1<< endl;

  p1 += 1;

  cout << "Gia tri sau do cua con tro kieu char : "<< p1<< endl ;

  cout << "\nGia tri ban dau cua con tro kieu int : "<<p2 << endl;

  p2 += 1;

  cout << "Gia tri sau do cua con tro kieu int  : "<<p2 << endl;

  return 0;

}

Gia tri ban dau cua con tro kieu char: 1290950691

Gia tri sau do cua con tro kieu char : 1290950692

Gia tri ban dau cua con tro kieu int : 1290950692

Gia tri sau do cua con tro kieu int : 1290950696

## Con trỏ của con trỏ

Chính vì con trỏ cũng là một biến, nên bản thân biến con trỏ cũng được gán một địa chỉ ở đâu đó trong bộ nhớ. Và bạn cũng có thể tạo một con trỏ khác để lưu trữ địa chỉ của biến con trỏ này. Đây được gọi là con trỏ của con trỏ trong C++.



Bằng cách sử dụng con trỏ của con trỏ, chúng ta có thể tiến hành các tham chiếu gián tiếp kép tới giá trị của một biến thông qua địa chỉ của nó trong chương trình.

### Khai báo

type \*\*p;

### Khởi tạo

VD

#include <iostream>

using namespace std;

int main() {

    int x = 10;

    int \*p1 = &x;

    cout << "x = "<< x<<endl;

    cout << "p1= "<< p1 <<endl;

    int \*\*p2 = &p1;

    cout << "p2= "<< p2;

    return 0;

}

### Sử dụng

VD1: lấy giá trị tại địa chỉ gán trong con trỏ của con trỏ

#include <iostream>

using namespace std;

int main() {

    int x = 10;

    int \*p1 = &x;

    int \*\*p2 = &p1;

    cout << "p1             =  " <<  p1 << endl;

    cout << "địa chỉ của p1 =  " <<  &p1 << endl;

    cout << "p2             =  " <<  p2 << endl;

    return 0;

}

p1 = 0x7fff257d6624

địa chỉ của p1 = 0x7fff257d6628

p2 = 0x7fff257d6628

VD2: dịch chuyển địa chỉ được gán trên bộ nhớ

#include <iostream>

using namespace std;

int main() {

    int x = 10;

    int \*p1 = &x;

    int \*\*p2 = &p1;

    cout << "p2 before=  " <<  p2 << endl;

    p2 += 1;

    cout << "p2 after =  " <<  p2 << endl;

    return 0;

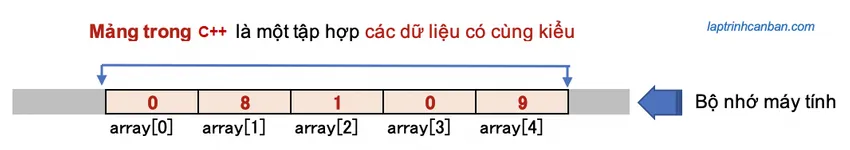
}

p2 before= 0x7ffeea36cb20

p2 after = 0x7ffeea36cb28

# Mảng 1 chiều

Mảng trong C++, hay còn gọi là kiểu mảng trong C++ là tập hợp các dữ liệu có cùng kiểu, và các dữ liệu chứa trong mảng được gọi là phần tử của mảng đó. Chúng ta không thể kết hợp các kiểu dữ liệu khác nhau trong cùng một mảng.



## Khởi tạo

type name[length] = {value1, value2, value3,...};

type name[] = {value1, value2, value3,...};

## Truy cập phần tử mảng

name[index]

name[index] = value;

Trong đó name là tên mảng, index là vị trí phần tử cần gán giá trị, và value là giá trị cần gán.

VD

#include <iostream>

using namespace std;

int main(){

    /\*Khai báo mảng số\*/

    int int\_arr[4];

    /\*Truy cập và gán giá trị vào phần tử trong mảng\*/

    int\_arr[0]=2;

    int\_arr[1]=0;

    int\_arr[2]=2;

    int\_arr[3]=1;

    /\*In từng phần tử và kiểm tra kết quả \*/

    for(int i = 0; i < 4; i++) {

      cout << int\_arr[i] <<' ';

    }

}

//> 2 0 2 1

## Lấy kích thước mảng

sizeof a;

VD1 lấy kích thước mảng chuỗi

#include <iostream>

using namespace std;

int main(){

    char a[] = "abcdef";

    int size = sizeof a;

    cout << size;

}

//7

VD lấy kích thước mảng số

#include <iostream>

using namespace std;

int main(){

    int a[] ={1,2,3,4,5};

    int size = sizeof a;

    cout << size;

}

//20

## Độ dài mảng

Độ dài mảng (số phần tử) = [Kích thước mảng] / [Kích thước kiểu mảng]

int length = sizeof a / sizeof(int);

VD2: Tính độ dài mảng chuỗi

#include <iostream>

using namespace std;

int main(){

    char a[] = "abcdef";

    int length = sizeof a / sizeof(char);

    cout <<length;

}

//7

Độ dài mảng chuỗi tính cả ký tự \0.

VD3: Tính độ dài mảng số

#include <iostream>

using namespace std;

int main(){

    int a[] ={1,2,3,4,5};

    int length = sizeof a / sizeof(int);

    cout <<length;

}

//5

## Nhập xuất mảng

### Nhập mảng

/\*Tạo hàm nhập mảng 1 chiều trong C++\*/

void input\_array(int array[], int length){

    //array: tên mảng

    //length: độ dài mảng

    for (short i = 0; i < length; i++) cin >> array[i];

}

### Xuất mảng

/\*Tạo hàm xuất mảng 1 chiều trong C++\*/

void show\_array(int array[], int length){

    //array: tên mảng

    //length: độ dài mảng

    for(short i = 0; i < length; i++)  cout << array[i] <<' ';

    cout << endl;

}

## Con trỏ mảng

### Khai báo

type \*p;

### Khởi tạo

Gán con trỏ mảng không cần ký tự &

int nums[] = {1,2,3,4}, \*p;

p = nums;

## Địa chỉ

### Địa chỉ con trỏ mảng trỏ tới

Con trỏ mảng biểu thị địa chỉ của điểm bắt đầu mảng trong bộ nhớ cũng là địa chỉ của phần tử đầu tiên.

### Chỉ định địa chỉ trong mảng thông qua dịch chuyển con trỏ

VD

#include <iostream>

using namespace std;

int main(){

    int nums[] = {10,20,30,44,55}, \*p;

    p = nums;

    cout << "array[0] address: "<< p << endl; //Địa chỉ phần tử đầu tiên

    cout << "array[1] address: "<< p + 1 << endl; //Địa chỉ phần tử thứ hai

    cout << "array[3] address: "<< p + 3 ; //Địa chỉ phần tử thứ tư

    return 0;

}

array[0] address: 2039942224

array[1] address: 2039942228

array[3] address: 2039942236

## Truy suất phần tử trong mảng bằng con trỏ

Cú pháp:

\*(p + index);

| **index** | **Truy xuất bằng mảng** | **Truy xuất bằng con trỏ** |
| --- | --- | --- |
| 0 | a[0] | \*p |
| 1 | a[1] | \*(p + 1) |
| 2 | a[2] | \*(p + 2) |
| 3 | a[3] | \*(p + 3) |
| … | … | … |
| n | a[n] | \*(p + n) |

VD

#include <iostream>

using namespace std;

int main(){

    int nums[] = {10,20,30,44,55}, \*p;

    p = nums;

    cout <<  nums[2] <<endl;

    cout <<  \*(p +2) <<endl;

    cout <<  nums[4] <<endl;

    cout <<  \*(p +4) <<endl;

    return 0;

}

30

30

55

55

## Nhập xuất mảng bằng con trỏ

### Nhập mảng bằng con trỏ

VD

/\*Tạo hàm nhập mảng bằng con trỏ trong C++\*/

void input\_array(int array[], int length){

    //array: tên mảng

    //length: độ dài mảng

    for (short i = 0; i < length; i++) cin >> \*(array + i);

}

### Xuất mảng bằng con trỏ

VD

/\*Tạo hàm xuất mảng 1 chiều trong C++\*/

void show\_array(int array[], int length){

    //array: tên mảng

    //length: độ dài mảng

    for(short i = 0; i < length; i++)  cout << array[i] <<' ';

    cout << endl;

}

## Sao chép mảng

### Cách 1: Sao chép từng phần tử của mảng nguồn tới mảng đích

VD

#include <iostream>

using namespace std;

int main(){

    /\*Khởi tạo chuỗi ban đầu\*/

    short data1[10] = {5,7,2,8,1,0,3,9,6,4};

    /\*Khai báo chuỗi kết quả có cùng độ dài\*/

    short data2[10];

    /\*Tạo vòng lặp, lấy và copy từng phần tử từ mảng cũ và gán vào mảng mới \*/

    for (short i = 0; i < 10; i++) {

        data2[i] = data1[i];

    }

    /\*Kiểm tra mảng mới\*/

    for(short i = 0; i < 10; i++) {

      cout << data2[i] <<' ';

    }

}

//> 5 7 2 8 1 0 3 9 6 4

### Cách 2: sử dụng hàm memcpy()

Cú pháp memcpy()

memcpy (dest\_address, src\_address, size);

Trong đó dest\_address là con trỏ chỉ vị trí đầu vùng bộ nhớ đích, src\_address là con trỏ chỉ vị trí đầu vùng bộ nhớ nguồn, và size là kích thước cần copy. Ở đây, dest là viết tắt của destination có nghĩa là điểm đến, còn str là viết tắt của source có nghĩa là nguồn.

VD

#include <iostream>

#include <string.h>

using namespace std;

int main()

{

    short src[10] = {5,7,2,8,1,0,3,9,6,4};

    short dest[10];

    /\*Sao chép mảng trong C++ bằng hàm memcpy\*/

    memcpy((void \*)dest, (void \*)src, sizeof(src));

    /\*Kiểm tra mảng mới\*/

    for(short i = 0; i < 10; i++) {

      cout <<  dest[i] << ' ';

    }

}

//> 5 7 2 8 1 0 3 9 6 4

Tham khảo: <http://laptrinhcanban.com/cpp/lap-trinh-cpp-co-ban/mang-trong-cpp/sao-chep-mang-trong-cpp/>

## Thêm phần tử vào mảng

### Thêm phần tử vào cuối mảng

VD

#include <iostream>

using namespace std;

int main(){

    /\*Mảng ban đầu mới có 4 giá trị cụ thể được khởi tạo\*/

    /\*Các vị trí khác chưa có giá trị cụ thể và chúng ta có thể thêm phần tử vào\*/

    int arr[6] = {1, 3, 4, 5};

    cout << ("Mang ban dau:\n");

    for(short i = 0; i < 6; i++) cout << arr[i] << ' ';

    /\*Thêm phần tử vào các vị trí cuối mảng chưa có giá trị cụ thể\*/

    arr[4] = 4;

    arr[5] = 8;

    cout << ("\nMang sau khi them phan tu:\n");

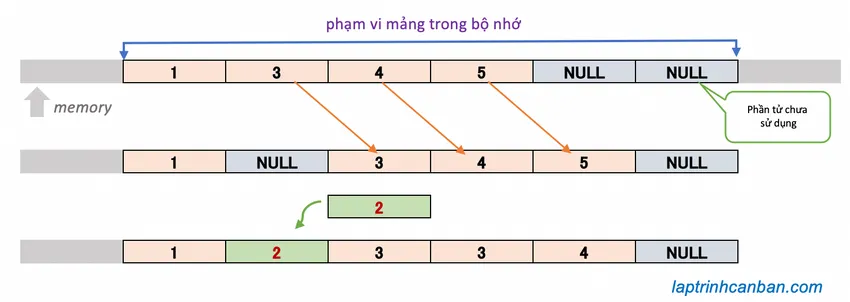
    for(short i = 0; i < 6; i++) cout << arr[i] << ' ';

}

### Chèn phần tử vào vị trí bất kỳ

Các bước như sau:

* Dịch chuyển các phần tử từ vị trí cần chèn về phía cuối mảng 1 đơn vị, và làm trống vị trí cần chèn
* Chèn phần tử cần thêm vào vị trí trống mới tạo ở trên



VD

#include <iostream>

using namespace std;

/\*Tạo hàm nhập mảng 1 chiều trong C++\*/

void input\_array(int array[], int length){

    //array: tên mảng

    //length: độ dài mảng

    for (short i = 0; i < length; i++) cin >> array[i];

}

/\*Tạo hàm in phần tử trong mảng\*/

void show\_array(int array[], int length){

    for(short i = 0; i < length; i++)  cout << array[i] <<' ';

    cout << endl;

}

int main()

{

   int array[100], position, i, n, value;

   cout << "Nhap so phan tu: ";

   cin >>  n;

   cout << "Nhap phan tu:\n";

   input\_array(array, n);

   cout << "Mang da nhap:";

   show\_array(array, n);

   cout << "\nNhập vi tri chen phan tu: \n";

   cin >>  position;

   cout << "Nhập phan tu muon chen: \n";

   cin >>  value;

   // Dịch chuyển các phần tử từ vị trí cần chèn (position) về phía sau 1 đơn vị

   // Và làm trống vị trí position

   for (i = n - 1; i >= position - 1; i--)

      array[i+1] = array[i];

   /\*Chèn phần tử vào chỗ trống vừa tạo\*/

   array[position-1] = value;

   cout << ("Mang sau khi chen:\n");

   show\_array(array, n);

   return 0;

}

Nhap so phan tu: 4

Nhap phan tu:

1 2 3 4

Mang da nhap:1 2 3 4

Nhập vi tri chen phan tu:

3

Nhập phan tu muon chen:

88

Mang sau khi chen:

1 2 88 3

### Ghép 2 mảng

Sử dụng memcpy() sao chép lần lượt 2 mảng vào mảng đích.

VD

#include <iostream>

#include <cstring>

using namespace std;

int main(){

    int array1[] = {1,2,3};

    int array2[] = {4,5};

    /\*Tìm kích thước và chiều dài của các mảng cần ghép\*/

     int size1 = sizeof array1,  length1 = sizeof array1/ sizeof(int);

     int size2 = sizeof array2,  length2 = sizeof array2/ sizeof(int);

    /\* Khai báo mảng kết quả với độ dài đủ để chứa 2 mảng trên\*/

    int result[length1 + length2];

     /\* Copy mảng đầu tiên vào mảng kết quả\*/

    memcpy(result,array1,size1);

    /\* Copy mảng thứ 2 vào vị trí cuối mảng 1 trong mảng kết quả\*/

    memcpy(&result[length1],array2,size2);

    /\*Kiểm tra kết quả ghép nối mảng\*/

    for(short i = 0; i < 5; i++) {

      cout << result[i]<<' ';

    }

}

//> 1 2 3 4 5

## Xoá phần tử

Xoá phần tử tương đương với thay thế phần tử bằng NULL. NULL trong C++ nếu ở dạng char thì là ký tự \0, và nếu ở dạng int thì là số 0. Tuy nhiên về bản chất thì NULL có nghĩa là ký tự rỗng có nghĩa là không tồn tại giá trị.

### Xoá một phần tử

Ý tưởng: Dịch chuyển các phần tử sau phần tử xoá lên một đơn vị.

Các bước:

* Thay đổi giá trị của tất cả các phần tử từ vị trí cần xóa trong mảng, bằng giá trị của phần tử đứng đằng sau nó.
* Thay đổi giá trị của phần tử cuối cùng thành NULL.

VD

/\*Tạo hàm xóa 1 phần tử trong mảng\*/

int delete\_one\_element\_from\_array(int array[], int length, int position){

   int count = 0;

    /\* Dịch phần tử về đầu mảng từ vị trí xóa \*/

    for(int i = position; i < length-1; i++) {

        array[i] = array[i+1];

    }

    /\*Thay đổi giá trị phần tử cuối cùng thành NULL\*/

    array[length-1] = 0;

    count++;

    return count;

}

### Xoá nhiều phần tử

Ý tưởng: sử dụng hàm xoá một phần tử ở trên để xoá từng phần tử cần xoá.

VD

#include <iostream>

using namespace std;

/\*Tạo hàm nhập mảng 1 chiều trong C++\*/

void input\_array(int array[], int length){

    //array: tên mảng

    //length: độ dài mảng

    for (short i = 0; i < length; i++) cin >> array[i];

}

/\*Tạo hàm in phần tử trong mảng\*/

void show\_array(int array[], int length){

    for(short i = 0; i < length; i++)  cout << array[i] <<' ';

    cout << endl;

}

/\*Tạo hàm xóa 1 phần tử trong mảng\*/

int delete\_one\_element\_from\_array(int array[], int length, int position){

   int count = 0;

    /\* Dịch phần tử về đầu mảng từ vị trí xóa \*/

    for(int i = position; i < length-1; i++) {

        array[i] = array[i+1];

    }

    /\*Thay đổi giá trị phần tử cuối cùng thành NULL\*/

    array[length-1] = 0;

    count++;

    return count;

}

/\*Tạo hàm xóa nhiều phần tử trong mảng\*/

int delete\_elements\_from\_array(int array[], int length, int positions[], int delete\_num){

    int n = 0;

    int count=0;

    for (int i=0; i< delete\_num; i++){

        int position= positions[i] - count;

        n += delete\_one\_element\_from\_array(array,length, position);

        count++;

    }

    return n;

}

int main(){

    int array[100], positions[100], length, delete\_num;

    cout << "Nhap so phan tu: ";

    cin >>  length;

    cout << "Nhap phan tu:\n";

    input\_array(array,length);

    cout << "Nhap so phan tu can xoa: ";

    cin >>  delete\_num;

    cout << "Nhap vi tri xoa:\n";

    input\_array(positions,delete\_num);

    cout << "Mang truoc khi xoa:\n";

    show\_array(array,length);

    int count= delete\_elements\_from\_array(array,length, positions,delete\_num);

    cout << "Mang sau khi xoa:\n";

    show\_array(array,length);

    cout << "So phan tu bi xoa: "<<count;

    /\*Tạo một mảng mới để lưu kết quả xóa phần tử từ mảng ban đầu\*/

    int n = length -count;

    int result[n];

    for(short i = 0; i < n; i++)  result[i] = array[i];

    cout << "\nMang ket qua:\n";

    show\_array(result,n);

    return 0;

}

Nhap so phan tu: 6

Nhap phan tu:

1 2 3 4 5 6

Nhap so phan tu can xoa: 2

Nhap vi tri xoa:

3 5

Mang truoc khi xoa:

1 2 3 4 5 6

Mang sau khi xoa:

1 2 3 5 0 0

So phan tu bi xoa: 2

Mang ket qua:

1 2 3 5

## Tách mảng chẵn lẻ

VD

#include <iostream>

using namespace std;

/\*Tạo macro SIZE\_OF\_ARRAY để lấy độ dài (số phần tử) của mảng chỉ định\*/

#define SIZE\_OF\_ARRAY(array)  (sizeof(array)/sizeof(array[0]))

/\*Tạo hàm in phần tử trong mảng\*/

void show\_array(int array[], int length){

    for(short i = 0; i < length; i++)  cout << array[i] <<' ';

    cout << endl;

}

/\*Tạo hàm kiểm tra số chẵn lẻ trong C++\*/

int check\_odd\_even(int n){

    //flag = 1 => số lẻ

    //flag = 0 => số chẵn

    int flag = 1;

    if( n % 2 == 0 ) flag= 0;

    return flag;

}

int main(){

    //Khai báo 2 mảng chứa kết quả

    int odd\_array[100], even\_array[100];

    //Khởi tạo mảng cần tách ra các số chẵn lẻ

    int array[] = {-2,-1,0,1,2,3,4,5,6,7,8};

    //Sử dụng macro ở trên để lấy độ dài mảng

    int length = SIZE\_OF\_ARRAY(array);

    int odd\_count = 0, even\_count = 0;

    for (int i = 0; i < length; i++){

        //Kiểm tra phần tử là số chẵn hay lẻ

        //Và gán số tìm thấy vào các mảng chẵn lẻ tương ứng

        //Cũng như tăng dần số phần tử trong mảng kết quả

        if (check\_odd\_even(array[i]) == 1) {

          odd\_array[odd\_count]=array[i];

          ++ odd\_count;

        } else {

          even\_array[even\_count]=array[i];

          ++ even\_count;

        }

    }

    //In ra mảng chẵn lẻ đã tách ra

    show\_array(odd\_array, odd\_count);

    show\_array(even\_array, even\_count);

    return 0;

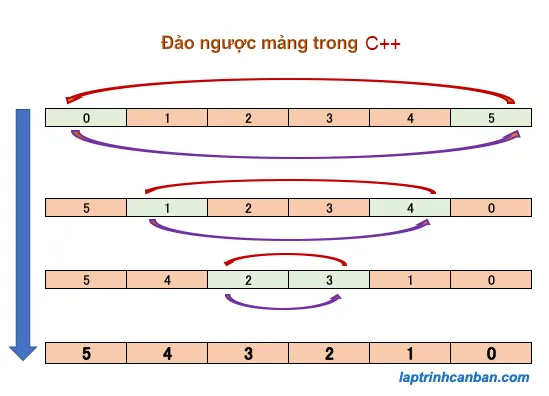
}

-1 1 3 5 7

-2 0 2 4 6 8

## Đảo ngược mảng

Ý tưởng: hoán đổi vị trí của các cặp phần tử từ ngoài vào trong



VD

#include <iostream>

using namespace std;

//Định nghĩa maco có tác dụng hoán đổi 2 phần tử

#define SWAP(type,x,y) do{type tmp = x; x = y; y = tmp;}while(0)

//Tạo hàm đảo ngược mảng

void reverse(int\* array, int size)

{

    for (int i = 0; i < size / 2; ++i) {

        SWAP(int, array[i], array[size - i - 1]);

    }

}

/\*Tạo hàm in phần tử trong mảng\*/

void show\_array(int array[], int length){

    for(short i = 0; i < length; i++)  cout << array[i] <<' ';

    cout << endl;

}

int main()

{

    int array1[] = {11, 22, 33, 44,};

    int array2[] = {0, 1, 2, 3, 4, 5};

    //Lấy độ dài mảng ban đầu

    int size1 = sizeof array1 / sizeof(int);

    int size2 = sizeof array2 / sizeof(int);

    reverse(array1, size1);

    reverse(array2, size2);

    //in mảng kết quả

    show\_array(array1,size1);

    show\_array(array2,size2);

    return 0;

}

44 33 22 11

5 4 3 2 1 0

## Xắp xếp mảng

### Hàm qsort()

Hàm qsort trong C++ là một hàm có sẵn trong header file cstdlib, giúp chúng ta sắp xếp mảng trong C++. Cú pháp sử dụng hàm qsort() trong C++ như sau:

qsort(array, length, size, compare\_fuct);

Trong đó:

* array là mảng cần so sánh
* length là độ dài (số phần tử) của mảng đó. Xem thêm: Độ dài mảng trong C++.
* size là kích thước một phần tử trong mảng. Ví dụ như kiểu int là 4 byte, char là 1 byte chẳng hạn. Xem thêm: Kiểu dữ liệu trong C++.
* compare\_fuct là hàm so sánh để quyết định thứ tự sắp xếp

### Xắp xếp tăng dần

Ý tưởng: sử dụng qsort()

VD

#include <iostream>

using namespace std;

#include <cstdlib>

/\*Định nghĩa macro SIZE\_OF\_ARRAY để lấy độ dài (số phần tử) trong mảng chỉ định\*/

#define SIZE\_OF\_ARRAY(array) (sizeof(array)/sizeof(array[0]))

/\*Tạo hàm in phần tử trong mảng\*/

void show\_array(int array[], int length){

    for(short i = 0; i < length; i++)  cout << array[i] <<' ';

    cout << endl;

}

/\*Tạo hàm so sánh tăng dần sử dụng trong hàm qsort\*/

int compareIntAsc(const void\* a, const void\* b){

    int aNum = \*(int\*)a;

    int bNum = \*(int\*)b;

    return aNum - bNum;

}

int main(){

    int array1[] = {5, 4, 7, 2, 8, 7, 3};

    int array2[] = {99, 4, 5, 2, 80, 7, 3};

    /\*Sử dụng hàm qsort để sắp xếp mảng tăng dần\*/

    qsort(array1, SIZE\_OF\_ARRAY(array1), sizeof(int), compareIntAsc);

    qsort(array2, SIZE\_OF\_ARRAY(array2), sizeof(int), compareIntAsc);

    /\*Xem kết quả sắp xếp mảng\*/

    show\_array(array1, SIZE\_OF\_ARRAY(array1));

    show\_array(array2, SIZE\_OF\_ARRAY(array2));

    return 0;

}

2 3 4 5 7 7 8

2 3 4 5 7 80 99

### Xắp xếp giảm dần

VD

#include <iostream>

#include <cstdlib>

using namespace std;

/\*Định nghĩa macro SIZE\_OF\_ARRAY để lấy độ dài (số phần tử) trong mảng chỉ định\*/

#define SIZE\_OF\_ARRAY(array) (sizeof(array)/sizeof(array[0]))

/\*Tạo hàm in phần tử trong mảng\*/

void show\_array(int array[], int length){

    for(short i = 0; i < length; i++)  cout << array[i] <<' ';

    cout << endl;

}

/\*Tạo hàm so sánh giảm dần sử dụng trong hàm qsort\*/

int compareIntDesc(const void\* a, const void\* b){

    int aNum = \*(int\*)a;

    int bNum = \*(int\*)b;

    return bNum - aNum;

}

int main(){

    int array1[] = {5, 4, 7, 2, 8, 7, 3};

    int array2[] = {99, 4, 5, 2, 80, 7, 3};

    /\*Sử dụng hàm qsort để sắp xếp mảng giảm dần\*/

    qsort(array1, SIZE\_OF\_ARRAY(array1), sizeof(int), compareIntDesc);

    qsort(array2, SIZE\_OF\_ARRAY(array2), sizeof(int), compareIntDesc);

    /\*Xem kết quả sắp xếp mảng\*/

    show\_array(array1, SIZE\_OF\_ARRAY(array1));

    show\_array(array2, SIZE\_OF\_ARRAY(array2));

    return 0;

}

8 7 7 5 4 3 2

99 80 7 5 4 3 2

## Tìm min max

### Tìm min

#include <cassert>

/\*Hàm tìm min trong mảng C++\*/

int minElement(const int\* array, size\_t size){

    /\*Kiểm tra và báo lỗi nếu đối số truyền vào không thoả mãn điều kiện chạy hàm\*/

    assert(array != NULL);

    assert(size >= 1);

    //Giả định giá trị nhỏ nhất là giá trị đầu tiên của mảng.

    int min = array[0];

    /\*So sánh từng phần tử trong mảng với giá trị đầu tiên để tìm ra giá trị nhỏ nhất\*/

    for (size\_t i = 1; i < size; ++i) {

        if (min > array[i]) { //Thay đổi giá trị min nếu tìm ra số nhỏ hơn

            min = array[i];

        }

    }

    return min;

}

### Tìm max

#include <cassert>

/\*Hàm tìm max trong mảng C++\*/

int maxElement(const int\* array, size\_t size){

    /\*Kiểm tra và báo lỗi nếu đối số truyền vào không thoả mãn điều kiện chạy hàm\*/

    assert(array != NULL);

    assert(size >= 1);

    //Giả định giá trị lớn nhất là giá trị đầu tiên của mảng.

    int max = array[0];

    /\*So sánh từng phần tử trong mảng với giá trị đầu tiên để tìm ra giá trị lớn nhất\*/

    for (size\_t i = 1; i < size; ++i) {

        if (max < array[i]) { //Thay đổi giá trị max nếu tìm ra số lớn hơn

            max = array[i];

        }

    }

    return max;

}

### Tìm min max bằng cách xếp mảng

#include <iostream>

#include <cstdlib>

using namespace std;

/\*Định nghĩa macro SIZE\_OF\_ARRAY để lấy độ dài (số phần tử) trong mảng chỉ định\*/

#define SIZE\_OF\_ARRAY(array) (sizeof(array)/sizeof(array[0]))

/\*Tạo hàm so sánh sử dụng trong hàm qsort\*/

int compareIntDesc(const void\* a, const void\* b){

    int aNum = \*(int\*)a;

    int bNum = \*(int\*)b;

    return bNum - aNum;

}

int main(){

    int array[] = {5, 4, 7, 2, 8, 7, 3};

    int length= SIZE\_OF\_ARRAY(array);

    /\*Sử dụng hàm qsort để sắp xếp mảng giảm dần\*/

    qsort(array, length, sizeof(int), compareIntDesc);

    /\*max là phần tử đầu tiên, min là phần tử cuối cùng\*/

    cout << "max= "<< array[0] <<endl;

    cout << "min= "<< array[length-1];

    return 0;

}

max= 8

min= 2

## So sánh mảng

### Cách 1: so sánh 2 mảng bằng cách đối chiếu từng cặp phần tử

#include <cassert>

#include <cstdbool>

#include <iostream>

using namespace std;

/\*Tạo macro SIZE\_OF\_ARRAY để lấy độ dài (số phần tử) của mảng chỉ định\*/

#define SIZE\_OF\_ARRAY(array)  (sizeof(array)/sizeof(array[0]))

/\*Hàm so sánh 2 mảng trong C++\*/

int array\_equal(const int\* array1, size\_t size1, const int\* array2, size\_t size2){

    //flag = 1 => 2 mảng đã cho giống nhau

    //flag = 0 => 2 mảng đã cho khác nhau

    int flag = 1;

    /\*Kiểm tra đối số truyền vào mảng có thỏa mãn điều kiện chạy hàm không\*/

    assert(array1 != NULL);

    assert(array2 != NULL);

    assert(size1 != 0);

    assert(size2 != 0);

    /\*Nếu số phần tử của 2 mảng khác nhau, thì chúng sẽ khác nhau\*/

    if (size1 != size2)  return flag = 0;

    /\*Kiểm tra từng cặp phần tử tương ứng xem có cặp nào khác nhau không\*/

    for (size\_t i = 0; i < size1; ++i) {

        if (array1[i] != array2[i])  return flag = 0;

    }

    return flag;

}

int main()

{

    int array1[] = {0, 1, 2, 3, 4};

    int array2[] = {0, 1, 4};

    int array3[] = {0, 1, 2, 3, 4};

    int array4[] = {0, 1, 2, 3, 4, 5};

    cout <<  array\_equal(array1, SIZE\_OF\_ARRAY(array1), array1, SIZE\_OF\_ARRAY(array1))<< endl ; ;

    cout <<  array\_equal(array1, SIZE\_OF\_ARRAY(array1), array2, SIZE\_OF\_ARRAY(array2))<< endl ; ;

    cout <<  array\_equal(array1, SIZE\_OF\_ARRAY(array1), array3, SIZE\_OF\_ARRAY(array3))<< endl ; ;

    cout <<  array\_equal(array1, SIZE\_OF\_ARRAY(array1), array4, SIZE\_OF\_ARRAY(array4))<< endl ; ;

    return 0;

}

1

0

1

0

### So sánh 2 mảng bằng hàm memcmp()

VD

#include <cassert>

#include <cstdbool>

#include <iostream>

#include <cstring>

using namespace std;

/\*Tạo macro SIZE\_OF\_ARRAY để lấy độ dài (số phần tử) của mảng chỉ định\*/

#define SIZE\_OF\_ARRAY(array)    (sizeof(array)/sizeof(array[0]))

bool array\_equal(const int\* array1, size\_t size1, const int\* array2, size\_t size2)

{

    assert(array1 != NULL);

    assert(array2 != NULL);

    assert(size1 != 0);

    assert(size2 != 0);

    /\*Nếu số phần tử của 2 mảng khác nhau, thì chúng sẽ khác nhau\*/

    if (size1 != size2) {

        return false;

    }

    /\*So sánh 2 vùng bộ nhớ chứa 2 mảng trên bằng hàm memcmp\*/

    return memcmp(array1, array2, size1 \* sizeof(int)) == 0;

}

int main()

{

    int array1[] = {0, 1, 2, 3, 4};

    int array2[] = {0, 1, 4};

    int array3[] = {0, 1, 2, 3, 4};

    int array4[] = {0, 1, 2, 3, 4, 5};

    cout <<  array\_equal(array1, SIZE\_OF\_ARRAY(array1), array1, SIZE\_OF\_ARRAY(array1))<< endl ; ;

    cout <<  array\_equal(array1, SIZE\_OF\_ARRAY(array1), array2, SIZE\_OF\_ARRAY(array2))<< endl ; ;

    cout <<  array\_equal(array1, SIZE\_OF\_ARRAY(array1), array3, SIZE\_OF\_ARRAY(array3))<< endl ; ;

    cout <<  array\_equal(array1, SIZE\_OF\_ARRAY(array1), array4, SIZE\_OF\_ARRAY(array4))<< endl ; ;

    return 0;

}

1

0

1

0

## Kiểm tra phần tử trùng

### Kiểm tra phần tử trùng trong mảng C++ bằng cách so sánh từng phần tử

VD

#include <iostream>

using namespace std;

/\*Tạo macro SIZE\_OF\_ARRAY để lấy độ dài (số phần tử) của mảng chỉ định\*/

#define SIZE\_OF\_ARRAY(array)  (sizeof(array)/sizeof(array[0]))

/\*Tạo hàm kiểm tra phần tử trùng trong mảng C++\*/

int array\_is\_unique(const int\* array, size\_t size){

    //flag =  1 =>  tồn tại phần tử trùng nhau

    //flag =  0 =>  không tồn tại phần tử trùng nhau

    int flag = 0;

    for (size\_t i = 0; i < size - 1; ++i) {

        for (size\_t j = i + 1; j < size; ++j) {

            if (array[i] == array[j]) {

                flag = 1;

                break;

            }

        }

    }

    return flag;

}

int main(){

    //Khởi tạo mảng cần kiểm tra phần tử trùng nhau

    int array1[] = {1,2,3,3,2,5}; //Mảng chứa phần tử trùng nhau

    int array2[] = {1,2,3,4,5,6,7}; //Mảng không chứa phần tử trùng nhau

    //Kiểm tra phần tử trùng nhau

    int check1 = array\_is\_unique(array1,SIZE\_OF\_ARRAY(array1));

    if (check1 == 1) cout << "ton tai phan tu trung nhau";

    else cout << "khong ton tai phan tu trung nhau";

    cout << endl;

    //Kiểm tra phần tử trùng nhau

    int check2 = array\_is\_unique(array2,SIZE\_OF\_ARRAY(array2));

    if (check2 == 1) cout << "ton tai phan tu trung nhau";

    else cout << "khong ton tai phan tu trung nhau";

    return 0;

}

### Kiểm tra phần tử trùng trong mảng C++ bằng cách sắp xếp mảng

#include <iostream>

using namespace std;

#include <stdlib.h>

/\*Tạo macro SIZE\_OF\_ARRAY để lấy độ dài (số phần tử) của mảng chỉ định\*/

#define SIZE\_OF\_ARRAY(array)  (sizeof(array)/sizeof(array[0]))

/\*Tạo hàm so sánh tăng dần sử dụng trong hàm qsort\*/

int compareIntAsc(const void\* a, const void\* b)

{

    int aNum = \*(int\*)a;

    int bNum = \*(int\*)b;

    return aNum - bNum;

}

/\*Tạo hàm kiểm tra phần tử trùng trong mảng C++\*/

int array\_is\_unique(const int\* array, size\_t size){

    //flag =  1 =>  tồn tại phần tử trùng nhau

    //flag =  0 =>  không tồn tại phần tử trùng nhau

    int flag = 0;

    for (size\_t i = 0; i < size - 1; ++i) {

        if (array[i] == array[i + 1]) {

            flag = 1;

            break;

        }

    }

    return flag;

}

int main(){

    //Khởi tạo mảng cần kiểm tra phần tử trùng nhau

    int array1[] = {1,2,3,3,2,5}; //Mảng chứa phần tử trùng nhau

    int array2[] = {1,2,3,4,5,6,7}; //Mảng không chứa phần tử trùng nhau

    //Sắp xếp mảng theo thứ tự tăng dần

    qsort(array1, SIZE\_OF\_ARRAY(array1), sizeof(int), compareIntAsc);

    qsort(array2, SIZE\_OF\_ARRAY(array2), sizeof(int), compareIntAsc);

    //Kiểm tra phần tử trùng nhau

    int check1 = array\_is\_unique(array1,SIZE\_OF\_ARRAY(array1));

    if (check1 == 1) cout << "ton tai phan tu trung nhau";

    else cout << "khong ton tai phan tu trung nhau";

    cout << endl;

    int check2 = array\_is\_unique(array2,SIZE\_OF\_ARRAY(array2));

    if (check2 == 1) cout << "ton tai phan tu trung nhau";

    else cout << "khong ton tai phan tu trung nhau";

    return 0;

}

ton tai phan tu trung nhau

khong ton tai phan tu trung nhau

### Lấy phần tử trùng

#include <iostream>

using namespace std;

/\*Tạo macro SIZE\_OF\_ARRAY để lấy độ dài (số phần tử) của mảng chỉ định\*/

#define SIZE\_OF\_ARRAY(array)  (sizeof(array)/sizeof(array[0]))

/\*Tạo hàm in phần tử trong mảng\*/

void show\_array(int array[], int length){

    for(short i = 0; i < length; i++)  cout << array[i] <<' ';

    cout << endl;

}

/\*Thay đổi hàm kiểm tra phần tử trùng nhau ở phương pháp một\*/

/\*Và tạo hàm lấy phần tử trùng trong mảng C++\*/

void take\_duplicate\_element(const int\* array, size\_t size){

    int result[100], count=0;

    for (size\_t i = 0; i < size - 1; ++i) {

        for (size\_t j = i + 1; j < size; ++j) {

            if (array[i] == array[j]) {

            result[count]=array[i];

              ++ count;

            }

        }

    }

    show\_array(result, count);

}

int main(){

    int array1[] = {1,2,3,3,2,5};

    int array2[] = {1,1,2,3,2,5,5,8,9,6};

    //Lấy phần tử trùng nhau

   take\_duplicate\_element(array1,SIZE\_OF\_ARRAY(array1));

   take\_duplicate\_element(array2,SIZE\_OF\_ARRAY(array2));

    return 0;

}

2 3

1 2 5

### Xoá phần tử trùng trong mảng đã được xắp xếp

VD

#include <iostream>

#include <cstdlib>

using namespace std;

/\*Tạo macro SIZE\_OF\_ARRAY để lấy độ dài (số phần tử) của mảng chỉ định\*/

#define SIZE\_OF\_ARRAY(array)    (sizeof(array)/sizeof(array[0]))

/\*Tạo hàm in phần tử trong mảng\*/

void show\_array(int array[], int length){

    for(short i = 0; i < length; i++)  cout << array[i] <<' ';

    cout << endl;

}

/\*Tạo hàm so sánh tăng dần sử dụng trong hàm qsort\*/

int compareIntAsc(const void\* a, const void\* b)

{

    int aNum = \*(int\*)a;

    int bNum = \*(int\*)b;

    return aNum - bNum;

}

/\*Tạo hàm xoá phần tử trùng trong mảng C++\*/

size\_t array\_unique(int\* array, size\_t size)

{

    size\_t end = 0;

    for (size\_t i = 1; i < size; ++i) {

        if (array[i] != array[end]) { //Tìm thấy phần tử trùng nhau

            ++end; //Tăng dần vị trí đầu mảng để gán phần tử trùng nhau

            array[end] = array[i]; //Ghi đè phần tử trùng nhau vào vị trí đầu mảng

        }

    }

    return end + 1;

}

int main()

{

    int array1[] = { 7, 2, 6, 7, 4, 9, 8 };

    int array2[] = { 7, 2, 6, 7, 7 };

    int array3[] = { 7, 7, 7 };

    // Sắp xếp các mảng theo thứ tự tăng dần

    qsort(array1, SIZE\_OF\_ARRAY(array1), sizeof(int), compareIntAsc);

    qsort(array2, SIZE\_OF\_ARRAY(array2), sizeof(int), compareIntAsc);

    qsort(array3, SIZE\_OF\_ARRAY(array3), sizeof(int), compareIntAsc);

    //Xoá phần tử trùng nhau trong các mảng

    size\_t size1 = array\_unique(array1, SIZE\_OF\_ARRAY(array1));

    size\_t size2 = array\_unique(array2, SIZE\_OF\_ARRAY(array2));

    size\_t size3 = array\_unique(array3, SIZE\_OF\_ARRAY(array3));

    //In các phần tử không trùng nhau đã được chuyển lên đầu mảng

    show\_array(array1, size1);

    show\_array(array2, size2);

    show\_array(array3, size3);

    return 0;

}

2 4 6 7 8 9

2 6 7

7

### Xoá phần tử trùng trong mảng chưa được xắp xếp

#include <iostream>

#include <cstring>

using namespace std;

/\*Tạo macro SIZE\_OF\_ARRAY để lấy độ dài (số phần tử) của mảng chỉ định\*/

#define SIZE\_OF\_ARRAY(array)    (sizeof(array)/sizeof(array[0]))

/\*Tạo hàm in phần tử trong mảng\*/

void show\_array(int array[], int length){

    for(short i = 0; i < length; i++)  cout << array[i] <<' ';

    cout << endl;

}

/\*Tạo hàm xoá phần tử trùng trong mảng C++\*/

size\_t array\_unique(int\* array, size\_t size)

{

    for (size\_t i = 0; i < size - 1; ++i) {

        for (size\_t j = i + 1; j < size; ++j) {

            if (array[i] == array[j]) { //Tìm thấy phần tử trùng nhau

                // Ghi đè array[j] bằng phạm vi từ array[j + 1] đến cuối mảng, qua đó xoá array[j]

                memmove(&array[j], &array[j + 1], sizeof(int) \* (size - j - 1));

                --size; // Do đã xoá array[j] nên trừ độ dài mảng đi một đơn vị

                // Do có khả năng 3 phần tử trùng nhau xuất hiện liên tiếp

                // Nên chúng ta cần kiểm tra lại vị trí j bằng cách giảm giá trị của j 1 đơn vị

                --j;

            }

        }

    }

    return size;

}

int main()

{

    int array1[] = { 7, 2, 6, 7, 4, 9, 8 };

    int array2[] = { 7, 2, 6, 7, 7 };

    int array3[] = { 7, 7, 7 };

    //Xoá phần tử trùng nhau trong các mảng

    size\_t size1 = array\_unique(array1, SIZE\_OF\_ARRAY(array1));

    size\_t size2 = array\_unique(array2, SIZE\_OF\_ARRAY(array2));

    size\_t size3 = array\_unique(array3, SIZE\_OF\_ARRAY(array3));

    //in mảng sau khi đã xoá phần tử trùng nhau

    show\_array(array1, size1);

    show\_array(array2, size2);

    show\_array(array3, size3);

    return 0;

}

2 4 6 7 8 9

2 6 7

7

### Đếm số phần tử trùng nhau

#include <iostream>

using namespace std;

/\*Tạo macro SIZE\_OF\_ARRAY để lấy độ dài (số phần tử) của mảng chỉ định\*/

#define SIZE\_OF\_ARRAY(array)  (sizeof(array)/sizeof(array[0]))

/\*Tạo hàm in phần tử trong mảng\*/

void show\_array(int array[], int length){

    for(short i = 0; i < length; i++)  cout << array[i] <<' ';

    cout << endl;

}

/\*Tạo hàm đếm số phần tử giống nhau trong mảng C++\*/

int take\_duplicate\_element(const int\* array, size\_t size){

    int result[100], count=0;

    for (size\_t i = 0; i < size - 1; ++i) {

        for (size\_t j = i + 1; j < size; ++j) {

            if (array[i] == array[j]) { //Tìm thấy phần tử giống nhau

            result[count]=array[i];

            ++ count;

            }

        }

    }

    show\_array(result,count);

    return count;

}

int main(){

    int array1[] = {1,2,3,3,2,5};

    int array2[] = {1,1,2,3,2,5,5,8,9,6};

    //Lấy phần tử trùng nhau

   int check1 = take\_duplicate\_element(array1,SIZE\_OF\_ARRAY(array1));

   cout << "So phan tu giong nhau: "<<check1<<endl;

   int check2 = take\_duplicate\_element(array2,SIZE\_OF\_ARRAY(array2));

   cout << "So phan tu giong nhau: "<<check2<<endl;

    return 0;

}

2 3

So phan tu giong nhau: 2

1 2 5

So phan tu giong nhau: 3

### Đếm số phần tử khác nhau

#include <iostream>

#include <cstdlib>

using namespace std;

/\*Tạo macro SIZE\_OF\_ARRAY để lấy độ dài (số phần tử) của mảng chỉ định\*/

#define SIZE\_OF\_ARRAY(array)    (sizeof(array)/sizeof(array[0]))

/\*Tạo hàm in phần tử trong mảng\*/

void show\_array(int array[], int length){

    for(short i = 0; i < length; i++)  cout << array[i] <<' ';

    cout << endl;

}

/\*Tạo hàm so sánh tăng dần sử dụng trong hàm qsort\*/

int compareIntAsc(const void\* a, const void\* b)

{

    int aNum = \*(int\*)a;

    int bNum = \*(int\*)b;

    return aNum - bNum;

}

/\*Tạo hàm xoá phần tử trùng trong mảng C++\*/

size\_t array\_unique(int\* array, size\_t size)

{

    size\_t end = 0;

    for (size\_t i = 1; i < size; ++i) {

        if (array[i] != array[end]) { //Tìm thấy phần tử trùng nhau

            ++end; //Tăng dần vị trí đầu mảng để gán phần tử trùng nhau

            array[end] = array[i]; //Ghi đè phần tử trùng nhau vào vị trí đầu mảng

        }

    }

    show\_array(array, end+1);

    return end + 1;

}

int main()

{

    int array1[] = { 7, 2, 6, 7, 4, 9, 8 };

    int array2[] = { 7, 2, 6, 7, 7 };

    int array3[] = { 7, 7, 7 };

    // Sắp xếp các mảng theo thứ tự tăng dần

    qsort(array1, SIZE\_OF\_ARRAY(array1), sizeof(int), compareIntAsc);

    qsort(array2, SIZE\_OF\_ARRAY(array2), sizeof(int), compareIntAsc);

    qsort(array3, SIZE\_OF\_ARRAY(array3), sizeof(int), compareIntAsc);

    //Xoá phần tử trùng nhau và đếm số phần tử khác nhau trong kết quả

    size\_t size1 = array\_unique(array1, SIZE\_OF\_ARRAY(array1));

    cout << "So phan tu khac nhau: "<<size1<<endl;

    size\_t size2 = array\_unique(array2, SIZE\_OF\_ARRAY(array2));

    cout << "So phan tu khac nhau: "<<size2<<endl;

    size\_t size3 = array\_unique(array3, SIZE\_OF\_ARRAY(array3));

    cout << "So phan tu khac nhau: "<<size3<<endl;

    return 0;

}

2 4 6 7 8 9

So phan tu khac nhau: 6

2 6 7

So phan tu khac nhau: 3

7

So phan tu khac nhau: 1

## Liệt kê phần tử

### Liệt kê các phần tử duy nhất trong mảng

#include <iostream>

#include <cstdlib>

using namespace std;

/\*Tạo macro SIZE\_OF\_ARRAY để lấy độ dài (số phần tử) của mảng chỉ định\*/

#define SIZE\_OF\_ARRAY(array)    (sizeof(array)/sizeof(array[0]))

/\*Tạo hàm in phần tử trong mảng\*/

void show\_array(int array[], int length){

    for(short i = 0; i < length; i++)  cout << array[i] <<' ';

    cout << endl;

}

/\*Tạo hàm so sánh tăng dần sử dụng trong hàm qsort\*/

int compareIntAsc(const void\* a, const void\* b)

{

    int aNum = \*(int\*)a;

    int bNum = \*(int\*)b;

    return aNum - bNum;

}

/\*Tạo hàm xoá phần tử trùng lặp trong mảng đã sắp xếp\*/

void array\_unique(int\* array, size\_t size){

    size\_t end = 0;

    for (size\_t i = 1; i < size; ++i) {

        if (array[i] != array[end]) {

            ++end;

            array[end] = array[i];

        }

    }

    //Liệt kê các phần tử xuất hiện một lần trong mảng C++

    show\_array(array, end+1);

}

int main(){

    int array1[] = { 7, 2, 6, 7, 4, 9, 8 };

    int array2[] = { 7, 2, 6, 7, 7 };

    int array3[] = { 7, 7, 7 };

    // Sắp xếp các mảng theo thứ tự tăng dần

    qsort(array1, SIZE\_OF\_ARRAY(array1), sizeof(int), compareIntAsc);

    qsort(array2, SIZE\_OF\_ARRAY(array2), sizeof(int), compareIntAsc);

    qsort(array3, SIZE\_OF\_ARRAY(array3), sizeof(int), compareIntAsc);

    //Xoá phần tử trùng nhau và liệt kê các phần tử xuất hiện một lần trong kết quả

    array\_unique(array1, SIZE\_OF\_ARRAY(array1));

    array\_unique(array2, SIZE\_OF\_ARRAY(array2));

    array\_unique(array3, SIZE\_OF\_ARRAY(array3));

    return 0;

}

2 4 6 7 8 9

2 6 7

7

### Liệt kê các phần tử xuất hiện một lần trong mảng

#include <iostream>

#include <cstdlib>

using namespace std;

/\*Tạo macro SIZE\_OF\_ARRAY để lấy độ dài (số phần tử) của mảng chỉ định\*/

#define SIZE\_OF\_ARRAY(array)    (sizeof(array)/sizeof(array[0]))

/\*Tạo hàm liệt kê các phần tử xuất hiện một lần trong mảng C++\*/

void array\_unique(int\* array, size\_t size){

    for (size\_t i = 0; i < size; ++i) {

        size\_t count = 0;

        for (size\_t j = 0; j < size; ++j) {

            if (array[i] == array[j] &i!=j) {

                //Tìm thấy phần tử trùng nhau thì tăng biến đếm

               count += 1;

            }

        }

        /\*Nếu count vẫn bằng 0 thì phần tử đang kiểm tra

         chỉ xuất hiện 1 lần duy nhất trong mảng\*/

        if(count==0) cout <<array[i] <<" ";

    }

    cout << endl;

}

int main(){

    int array1[] = { 7, 2, 6, 7, 4, 9, 8 };

    int array2[] = { 7, 2, 6, 7, 7 };

    int array3[] = { 7, 7, 7 };

    //Xoá phần tử trùng nhau và liệt kê các phần tử xuất hiện một lần trong kết quả

    array\_unique(array1, SIZE\_OF\_ARRAY(array1));

    array\_unique(array2, SIZE\_OF\_ARRAY(array2));

    array\_unique(array3, SIZE\_OF\_ARRAY(array3));

    return 0;

}

2 6 4 9 8

2 6

### Liệt kê các phần tử xuất hiện nhiều hơn 1 lần trong mảng

#include <iostream>

using namespace std;

/\*Tạo macro SIZE\_OF\_ARRAY để lấy độ dài (số phần tử) của mảng chỉ định\*/

#define SIZE\_OF\_ARRAY(array)  (sizeof(array)/sizeof(array[0]))

/\*Tạo hàm in phần tử trong mảng\*/

void show\_array(int array[], int length){

    for(short i = 0; i < length; i++)  cout << array[i] <<' ';

    cout << endl;

}

/\*Thay đổi hàm kiểm tra phần tử trùng nhau ở phương pháp một\*/

/\*Và tạo hàm lấy phần tử trùng trong mảng C++\*/

void take\_duplicate\_element(const int\* array, size\_t size){

  int result[100], count=0;

    for (size\_t i = 0; i < size - 1; ++i) {

        for (size\_t j = i + 1; j < size; ++j) {

            if (array[i] == array[j]) {

            result[count]=array[i];

            ++ count;

            }

        }

    }

    //liệt kê các phần tử xuất hiện nhiều hơn 1 lần trong mảng C++

    show\_array(result, count);

}

int main(){

    int array1[] = {1,2,3,3,2,5};

    int array2[] = {1,1,2,3,2,5,5,8,9,6};

    //liệt kê các phần tử xuất hiện nhiều hơn 1 lần trong mảng C++

    take\_duplicate\_element(array1,SIZE\_OF\_ARRAY(array1));

    take\_duplicate\_element(array2,SIZE\_OF\_ARRAY(array2));

    return 0;

}

2 3

1 2 5

## Tìm kiếm phần tử

### Tìm kiếm một phần tử trong mảng

#include <iostream>

using namespace std;

/\*Tạo macro SIZE\_OF\_ARRAY để lấy độ dài (số phần tử) của mảng chỉ định\*/

#define SIZE\_OF\_ARRAY(array)  (sizeof(array)/sizeof(array[0]))

/\*Tạo hàm in phần tử trong mảng\*/

void show\_array(int array[], int length){

    for(short i = 0; i < length; i++)  cout << array[i] <<' ';

    cout << endl;

}

/\*Tạo hàm tìm vị trí phần tử trong mảng C++\*/

void find\_element\_index(const int\* array, size\_t size, int x){

    int result[100], count = 0;

    for (size\_t i = 0; i < size - 1; ++i) {

        if (array[i] == x) {

            result[count] = i;

            ++count;

        }

    }

    //liệt kê các vị trí phần tử xuất hiện trong mảng

    cout << "Phan tu "<< x <<" xuat hien tai cac vi tri sau:\n";

    show\_array(result, count);

}

int main(){

    int array[] = {1,1,2,3,2,5,5,8,9,6};

    //tìm vị trí phần tử trong mảng C++

    find\_element\_index(array,SIZE\_OF\_ARRAY(array),3);

    find\_element\_index(array,SIZE\_OF\_ARRAY(array),5);

    return 0;

}

Phan tu 3 xuat hien tai cac vi tri sau:

3

Phan tu 5 xuat hien tai cac vi tri sau:

5 6

### Tìm phần tử xuất hiện nhiều nhất trong mảng

#include <iostream>

#include <cstdlib>

#include <cassert>

using namespace std;

/\*Tạo macro SIZE\_OF\_ARRAY để lấy độ dài (số phần tử) của mảng chỉ định\*/

#define SIZE\_OF\_ARRAY(array)    (sizeof(array)/sizeof(array[0]))

/\*Tạo hàm so sánh tăng dần sử dụng trong hàm qsort\*/

int compareIntAsc(const void\* a, const void\* b)

{

    int aNum = \*(int\*)a;

    int bNum = \*(int\*)b;

    return aNum - bNum;

}

/\*Biến tấu hàm đếm số lần xuất hiện của các phần tử

và tạo hàm tìm phần tử xuất hiện nhiều nhất trong mảng C++\*/

void array\_unique(int\* array, size\_t size){

    //Khởi tạo mảng max chứa kết quả

    //max[0] : phần tử xuất hiện nhiều nhất

    //max[1] : số lần xuất hiện của phần tử đó

    int max[] ={0,0};

    //Sử dụng vòng lặp for để lọc ra các phần tử xuất hiện nhiều hơn 1 lần

    //So sánh số lần xuất hiện và thay đổi max khi cần.

    int count =1;

    for (size\_t i = size-1; i > 0; --i) {

        if (array[i] == array[i-1]) ++count; //Thấy phần tử trùng nhau thì tiếp tục đếm

        else{

             //So sánh số lần xuất hiện với max[1]

             if (max[1] < count){

                  //Nếu tìm thấy phần tử xuất hiện nhiều hơn thì gán phần tử vào max[0]

                  //Và gán số lần xuất hiện vào max[1]

                  max[0] = array[i];

                  max[1] = count;

             }

             count = 1;

        }

    }

    cout << "Phan tu "<<max[0]<<" xuat hien nhieu nhat trong mang voi "<<max[1]<<" lan";

}

int main(){

    int array1[] = { 7, 2, 6, 7, 4, 9, 8 };

    // Sắp xếp các mảng theo thứ tự tăng dần

    qsort(array1, SIZE\_OF\_ARRAY(array1), sizeof(int), compareIntAsc);

    //Đếm số lần xuất hiện của các phần tử trong mảng C++

    array\_unique(array1, SIZE\_OF\_ARRAY(array1));

    return 0;

}

Phan tu 7 xuat hien nhieu nhat trong mang voi 2 lan

# Mảng 2 chiều

## Khai báo

type name[y][x];

Trong đó type là kiểu dữ liệu, name là tên mảng 2 chiều, y là số lượng mảng 1 chiều có trong mảng 2 chiều, còn x là độ dài(số phần tử con) của các mảng 1 chiều đó.

Lưu ý là chúng ta bắt buộc phải chỉ định các giá trị y và x khi khai báo mảng 2 chiều, để chương trình có thể tạo ra vùng có kích thước tương đương để lưu mảng 2 chiều này trong bộ nhớ máy tính.

## Khởi tạo

type name[y][x]= {{y1, x1}, {y2, x2}, {y3, x3}, ...};

type name[][x]= {{y1, x1}, {y2, x2}, {y3, x3}, ...};

type name[][y]= {

{y1, x1},

{y2, x2},

{y3, x3},

...

};

## Truy cập phần tử

name[y][x]

VD

#include <iostream>

using namespace std;

int main(){

    int bangdiem[][3] = {

        {7, 9, 8} ,

        {8, 6, 7} ,

        {5, 7, 6} ,

        {4, 9, 5} ,

        {5, 8, 7} ,

        {6, 9, 3}

    } ;

    //Lấy giá trị phần tử

    cout << "Phan tu [1][2]: "<< bangdiem[1][2] <<endl;

    cout << "Phan tu [3][2]: "<< bangdiem[3][2] <<endl;

    //Thay đổi giá trị phần tử

    bangdiem[3][2] = 10;

    cout << "Phan tu [3][2]: "<< bangdiem[3][2];

}

Phan tu [1][2]: 7

Phan tu [3][2]: 5

Phan tu [3][2]: 10

## Nhập xuất

### Nhập

#include <iostream>

using namespace std;

int main(){

    /\*Nhập mảng 2 chiều trong C++\*/

    int y,x;

    cout << ">>Nhap do dai (so mang 1 chieu) trong mang 2 chieu: ";

    cin >> y;

    cout << ">>Nhap do dai (so phan tu) trong mang 1 chieu: ";

    cin >>  x;

    int array[y][x];

    cout << (">>Nhap phan tu:\n");

    /\*Sử dụng vòng lặp lồng để nhập mảng 2 chiều trong C++\*/

    for (short i = 0; i < y; i++) {

        for (short j = 0; j < x; j++) cin >> array[i][j];

    }

    cout << (">>Mang vua nhap:\n");

    /\*Sử dụng vòng lặp lồng để xuất mảng 2 chiều trong C++\*/

    for (short i = 0; i < y; i++) {

        for (short j = 0; j < x; j++) cout << array[i][j] <<endl;

        cout << endl;

    }

}

>>Nhap do dai (so mang 1 chieu) trong mang 2 chieu: 3

>>Nhap do dai (so phan tu) trong mang 1 chieu: 2

>>Nhap phan tu:

1 2

3 4

5 6

>>Mang vua nhap:

1 2

3 4

5 6

### Xuất

#include <iostream>

using namespace std;

int main() {

  int a[][3] = {{7, 9, 8 }, {8, 6, 7}} ;

  /\* in từng phần tử trong mảng 2 chiều \*/

  cout << "a[0][0] =" <<a[0][0]<< endl;

  cout << "a[0][1] =" <<a[0][1]<< endl;

  cout << "a[0][2] =" <<a[0][2]<< endl;

  cout << "a[1][0] =" <<a[1][0]<< endl;

  cout << "a[1][1] =" <<a[1][1]<< endl;

  cout << "a[1][2] =" <<a[1][2]<< endl;

  return 0;

}

a[0][0] = 7

a[0][1] = 9

a[0][2] = 8

a[1][0] = 8

a[1][1] = 6

a[1][2] = 7

### In toàn bộ phần tử trong mảng 2 chiều

#include <iostream>

using namespace std;

int main() {

  int i;

    int a[][3] = {

        {7, 9, 8} ,

        {8, 6, 7} ,

        {5, 7, 6} ,

        {4, 9, 5} ,

        {5, 8, 7} ,

        {6, 9, 3}

    } ;

    //Lấy độ dài của mảng 2 chiều

    int y = sizeof(a) / sizeof(a[0]);

    int x = sizeof(a[0]) / sizeof(a[0][0]);

    /\*Sử dụng vòng lặp lồng để in toàn bộ phần tử trong mảng 2 chiều\*/

    for (short i = 0; i < y; i++) {

        for (short j = 0; j < x; j++) cout << a[i][j] <<' ';

        cout <<endl;

    }

  return 0;

}

7 9 8

8 6 7

5 7 6

4 9 5

5 8 7

6 9 3

## Mảng chuỗi 2 chiều

### Khởi tạo

char name[y][x]= {"string1", "string2", "string3", ...};

VD

char s1[5][4] = {"ABC", "DE", "FGH", "abc", "xyz"};

char s1[][4] = {"ABC", "DE", "FGH", "abc", "xyz"};

char s[] = { 'A', 'B', 'C', '\0' };

### Truy cập phần tử

VD

#include <iostream>

using namespace std;

int main(){

    char fruits[][10] = {"Apple", "Kiwi", "Lemon", "Melon", "Orange"};

    // Truy cập vào một chuỗi ký tự cụ thể

     cout << "fruits[0]= "<< fruits[0] <<endl;

     cout << "fruits[2]= "<< fruits[2] <<endl;

    //Lấy độ dài của mảng chuỗi 2 chiều

    int y = sizeof(fruits) / sizeof(fruits[0]);

    cout << ("\nIn toan bo chuoi\n");

    // In ra toàn bộ các chuỗi ký tự trong mảng chuỗi 2 chiều

    for(short i=0; i<y; i++){

        cout <<  fruits[i] << endl;

    }

    return 0;

}

fruits[0]= Apple

fruits[2]= Lemon

In toan bo chuoi

Apple

Kiwi

Lemon

Melon

Orange

VD

#include <iostream>

using namespace std;

int main(){

    char fruits[][10] = {"Apple", "Kiwi", "Lemon", "Melon", "Orange"};

    //Lấy độ dài của mảng chuỗi 2 chiều

    int y = sizeof(fruits) / sizeof(fruits[0]);

    // In ra các ký tự của chuỗi ký tự thứ 3 trong mảng

    for(short i=0; i<y; i++){

        cout <<  fruits[i][3] <<endl;

    }

    return 0;

}

l

i

o

o

n

## Lấy kích thước

sizeof a;

VD

#include <iostream>

using namespace std;

int main(){

    int bangdiem[][3] = {{7, 9, 8 }, {8, 6, 7}} ;

    int size = sizeof bangdiem;

    cout << size;

}

//24

## Lấy độ dài

VD

#include <iostream>

using namespace std;

int main(){

    int a[][3] = {{7, 9, 8 }, {8, 6, 7}} ;

     //Lấy độ dài của mảng 2 chiều

    int y = sizeof(a) / sizeof(a[0]);

    int x = sizeof(a[0]) / sizeof(a[0][0]);

    cout << "Do dai mang 2 chieu: y = "<<y<<" , x = " <<x;

}

//Do dai mang 2 chieu: y = 2 , x = 3

VD độ dài chuỗi

#include <iostream>

using namespace std;

int main(){

    char fruits[][10] = {"Apple", "Kiwi", "Lemon", "Melon", "Orange"};

     //Lấy độ dài của mảng 2 chiều

    int y = sizeof(fruits) / sizeof(fruits[0]);

    int x = sizeof(fruits[0]) / sizeof(fruits[0][0]);

    cout << "Do dai mang 2 chieu: y = "<<y<<" , x = " <<x;

}

//Do dai mang 2 chieu: y = 5 , x = 10

## Lấy số phần tử mảng 2 chiều

VD

#include <iostream>

using namespace std;

int main(){

    int bangdiem[][3] = {

        {7, 9, 8} ,

        {8, 6, 7} ,

        {5, 7, 6} ,

        {4, 9, 5} ,

        {5, 8, 7} ,

        {6, 9, 3}

    } ;

     //Lấy độ dài của mảng 2 chiều

    int y = sizeof(bangdiem) / sizeof(bangdiem[0]);

    int x = sizeof(bangdiem[0]) / sizeof(bangdiem[0][0]);

    cout << "So phan tu cua mang 2 chieu: "<<y \* x;

}

//So phan tu cua mang 2 chieu: 18

## Con trỏ mảng 2 chiều

### Cấp phát bộ nhớ động cho mảng hai chiều

Trong các bài trước, chúng ta đã học cách xử lý các mảng 2 chiều với kích thước và độ dài cố định, nghĩa là số hàng và số cột trong mảng sẽ luôn luôn cố định, và mảng cũng chiếm một vùng bộ nhớ trong máy tính với một số byte nhất định. Các mảng như vậy được lưu trong bộ nhớ tĩnh, và chúng ta không thể thay đổi kích thước của mảng kiểu này sau khi tạo chúng.

Tuy nhiên khi sử dụng mảng 2 chiều trong C++, kích thước của mảng không phải là lúc nào cũng cần phải cố định. Cũng có những lúc chúng ta muốn xóa đi một số hàng, số cột không cần thiết, và cũng có khi chúng ta muốn thêm phần tử vào mảng bằng cách thêm hàng và cột. Khi đó, thay vì mảng có kích thước cố định thì chúng ta cần phải dùng tới mảng có kích thước tự động thay đổi theo số hàng, số cột và số phần tử trong mảng.

Để có thể tự động thay đổi kích thước mảng như vậy, chúng ta cần cấp phát bộ nhớ động cho mảng hai chiều trong C++, thông qua một hàm có sẵn là hàm malloc mảng 2 chiều trong C++ . Và để thao tác với bộ nhớ động được tạo ra bằng hàm này, chúng ta sẽ cần dùng tới con trỏ trong mảng 2 chiều.

Chú ý là khi sử dụng hàm malloc mảng 2 chiều trong C++ này, chúng ta cần phải giải phóng bộ nhớ sau khi đã sử dụng xong bộ nhớ động được cấp phát. Chúng ta sẽ sử dụng tới một hàm có sẵn là hàm free() để thực hiện việc này.

Hai hàm trên đều được tích hợp trong header file 「stdlib.h」, do đó chúng ta phải ghi tên file này vào đầu chương trình để có thể sử dụng chúng.

### Khai báo mảng 2 chiều trong C++ dưới dạng con trỏ trong mảng 2 chiều

VD

#include <iostream>

using namespace std;

#include <stdlib.h>

int main() {

    int y , x;

    cout << ("Nhap so hang va so cot cua mang 2 chieu: ");

    cin >> y;

    cin >>  x;

    //Khai báo con trỏ của con trỏ

    int \*\*nums1;

    /\*Cấp phát bộ nhớ động cho 1 mảng mẹ gồm y phần tử,

    để chứa địa chỉ tương ứng của y hàng trong mảng 2 chiều cần tạo\*/

    /\*Gán địa chỉ mảng mẹ vào con trỏ nums1\*/

    nums1 = (int\*\*)malloc(sizeof(int \*) \* y);

    /\*Cấp phát bộ nhớ động cho y mảng con, mỗi mảng gồm x phần tử,

    để chứa x phần tử tương ứng trong từng hàng của mảng 2 chiều cần tạo\*/

    /\*Gán địa chỉ của các mảng này vào phần tử của mảng mẹ\*/

    for(int i = 0; i < y; i++) {

        nums1[i] = (int\*)malloc(sizeof(int) \* x);

    }

    /\*Truy cập và gán giá trị lần lượt vào các phần tử của y mảng con đã tạo ở trên,

    thông qua đia chỉ của chúng được gán trong con trỏ\*/

    cout << ("\nNhap phan tu: \n");

    for(int i = 0; i < y; i++) {

        for(int j = 0; j < x; j++) {

            cin >>  nums1[i][j];//Gán giá trị nhập từ bàn phím vào phần tử

        }

    }

    //In và kiểm tra mảng 2 chiều vừa khai báo

    cout << ("Mang 2 chieu vua nhap\n");

    for(int i = 0; i < y; i++) {

        for(int j = 0; j < x; j++) {

           cout <<  nums1[i][j];

        }

        cout << ("\n");

    }

    // Giải phóng các bộ nhớ đã dùng để lưu các phần tử sau khi đã tạo mảng

    for(int i = 0; i < y; i++) {

        free(nums1[i]);

    }

    // Giải phóng bộ nhớ đã dùng để lưu địa chỉ mảng sau khi đã tạo mảng

    free(nums1);

    return 0;

}

Nhap so hang va so cot cua mang 2 chieu: 4 3

Nhap phan tu:

7 8 9

2 3 1

0 9 2

8 5 4

Mang 2 chieu vua nhap

7 8 9

2 3 1

0 9 2

8 5 4

### Khai báo mảng 2 chiều trong C++ dưới dạng con trỏ trong mảng 1 chiều

#include <iostream>

using namespace std;

#include <stdlib.h>

int main() {

    int y , x;

    cout << ("Nhap so hang va so cot cua mang 2 chieu: ");

    cin >> y;

    cin >>  x;

    int \*nums2;

    //Cấp phát bộ nhớ động cho 1 mảng gồm y×x phần tử

    //Gán địa chỉ mảng trên vào con trỏ nums2

    nums2 = (int\*)malloc(sizeof(int) \* x \* y);

    for(int i = 0; i < y; i++) {

        for(int j = 0; j < x; j++) {

             cin >>  nums2[i \* x + j];//Gán giá trị nhập từ bàn phím vào phần tử

        }

    }

    //In và kiểm tra mảng 2 chiều vừa khai báo

    cout << ("Mang 2 chieu vua nhap\n");

     for(int i = 0; i < y; i++) {

        for(int j = 0; j < x; j++) {

               if(j < x -1) {

                    cout << nums2[i \* x + j] << " " ;

               } else {

                    cout << nums2[i \* x + j] <<endl;

               }

        }

    }

    // Giải phóng bộ nhớ

    free(nums2);

    return 0;

}

Nhap so hang va so cot cua mang 2 chieu: 2 3

1 2 3

2 3 4

Mang 2 chieu vua nhap

1 2 3

2 3 4

### Sử dụng con trỏ mảng 2 chiều trong hàm

#include <iostream>

using namespace std;

#include <stdlib.h>

void init\_2d\_array(int \*arr, int x, int y) {

     for(int i = 0; i < y; i++) {

          for(int j = 0; j < x; j++) {

               arr[i \* x + j] = i \* x + j; // Khởi tạo giá trị trong mảng

          }

     }

}

int main() {

     int x = 3, y = 5;

     int \*nums;

    //Cấp phát bộ nhớ động cho 1 mảng gồm y×x phần tử

    //Gán địa chỉ mảng trên vào con trỏ nums

     nums = (int\*)malloc(sizeof(int) \* x \* y);

     init\_2d\_array(nums, 5, 3);

     for(int i = 0; i < y; i++) {

          for(int j = 0; j < x; j++) {

               if(j < x - 1) {

                    cout << nums[i \* x + j] <<" ";

               } else {

                    cout << nums[i \* x + j] << endl;

               }

          }

     }

     // Giải phóng bộ nhớ

     free(nums);

     return 0;

}

0 1 2

3 4 5

6 7 8

9 10 11

12 13 14

### Sử dụng con trỏ mảng 2 chiều trong struct

#include <iostream>

using namespace std;

#include <stdlib.h>

typedef struct str{

    int x;

    int y;

    int \*arr;

} number;

void init\_2d\_array\_struct(number \*num) {

    for(int i = 0; i < num->y; i++) {

        for(int j = 0; j < num->x; j++) {

            num->arr[i \* num->x + j] = i \* num->x + j; // Khởi tạo struct

        }

    }

}

int main() {

    // Khởi tạo và khai báo struct

    number num;

    num.x = 3;

    num.y = 5;

    // Cấp phát bộ nhớ động cho 1 mảng gồm y×x phần tử

    num.arr = (int\*)malloc(sizeof(int) \* num.x \* num.y);

    init\_2d\_array\_struct(&num);

    for(int i = 0; i < num.y; i++) {

        for(int j = 0; j < num.x; j++) {

            if(j < num.x - 1) {

                cout << num.arr[i \* num.x + j] << " ";

            } else {

                cout << num.arr[i \* num.x + j] << endl;

            }

        }

    }

    // Giải phóng bộ nhớ

    free(num.arr);

    return 0;

}

0 1 2

3 4 5

6 7 8

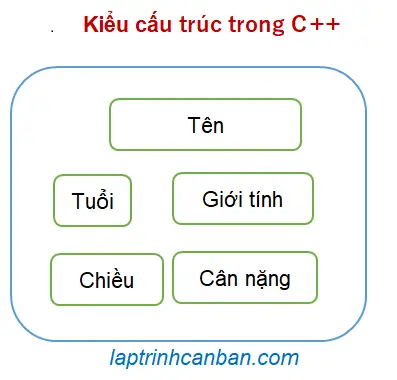
9 10 11

12 13 14

# Kiểu cấu trúc

## Tổng quan

Kiểu cấu trúc trong C++, hay còn gọi là kiểu struct trong C++ là một tập hợp các thuộc tính liên quan tới cùng một đối tượng. Ví dụ điển hình của kiểu cấu trúc là tập hợp các thuộc tính liên quan tới một người như tên, tuổi và giới tính.

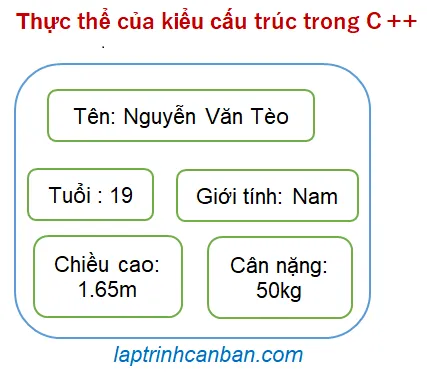


Các thuộc tính tạo nên một cấu trúc được gọi là các thành viên (tiếng Anh: member) của cấu trúc đó. Trong ví dụ trên, “tên”, “giới tính”, “tuổi”, “chiều cao”, “cân nặng”, v.v. chính là các thành viên trong cấu trúc.

Lưu ý: chúng ta không thể sử dụng hàm làm thành viên của một cấu trúc được.

Từ một cấu trúc, chúng ta có thể tạo ra vô vàn các thực thể (tiếng Anh: instance) có cấu trúc giống nhau. Từ một thực thể cụ thể, chúng ta có thể gọi các thành viên trong thực thể đó, cũng như gán các giá trị và sử dụng chúng.

Ví dụ, với cấu trúc [người] ở trên, chúng ta có thể tạo ra một thực thể là [Nguyễn Văn Tèo], với các thành viên có giá trị cụ thể như sau:



## Khai báo

struct name{

type1 member1;

type2 member2;

type3 member3;

...

};

VD

struct people1{

    int old; // tuổi

    char \*name; // tên

    int height; // chiều cao

    char \*sex; // giới tính

};

## Khởi tạo

struct struct\_name instance\_name;

VD

struct people1{

    int old; // tuổi

    char \*name; // tên

    int height; // chiều cao

    char \*sex; // giới tính

};

int main(){

    struct people1 BaKien;

}

Khởi tạo với giá trị ban đầu:

struct struct\_name instance\_name = {member1, member2, member3, ...};

int main(){

    struct people1 BaKien = {40, "BaKien",160, "male" };

}

## Định nghĩa kiểu dữ liệu mới với typedef

### Khai báo

typedef struct old\_struct\_name{

type1 member1;

type2 member2;

type3 member3;

...

}new\_struct\_name;

Tuy nhiên trong thực tế thì chúng ta hay lược bỏ đi old\_struct\_name và khai báo cấu trúc với cú pháp sau đây:

typedef struct{

type1 member1;

type2 member2;

type3 member3;

...

}struct\_name;

### Khởi tạo

struct\_name instance\_name;

struct\_name instance\_name = {member1, member2, member3, ...};

VD

typedef struct{

    int old; // tuổi

    char \*name; // tên

    int height; // chiều cao

    char \*sex; // giới tính

}people2;

int main(){

    people2 ChiPheo;

    people3 ChiPheo = {20, "ChiPheo",180, "male" };

}

## Truy cập vào thành viên

Chúng ta có 2 phương pháp để truy cập vào thành viên trong một thực thể như sau:

* Sử dụng dấu chấm . trong các trường hợp sử dụng biến thông thường
* Sử dụng dấu mũi tên -> trong trường hợp sử dụng biến con trỏ.

VD

int main(){

    struct people1 BaKien;

    BaKien.old = 40;

    BaKien.name = "BaKien";

    BaKien.height = 160;

    BaKien.sex = "male";

}

## Lấy kích thước

Sử dụng sizeof ();

VD

#include <iostream>

using namespace std;

typedef struct {

    char name[20];

    char sex;

    int age;

    double height;

    double weight;

} person\_t;

int main(){

  person\_t p;

  cout << "name="<< sizeof(p.name)<<" sex="<< sizeof(p.sex)<<" age="<< sizeof(p.age)

        <<" height="<< sizeof(p.height)<<" weight="<< sizeof(p.weight)<<endl;

  cout << "person\_t= "<< sizeof(p);

}

name=20 sex=1 age=4 height=8 weight=8

person\_t= 48

## Con trỏ cấu trúc

### Truy cập tới thành viên

Sau khi tạo ra một con trỏ cấu trúc lưu giữ địa chỉ của một thực thể tạo ra từ cấu trúc đó, để truy cập vào các thành viên của thực thể thì thay vì sử dụng dấu chấm như thông thường thì chúng ta sẽ sử dụng tới dấu mũi tên ->, với cú pháp sau đây:

instance\_pointer -> member

Trong đó instance\_pointer là con trỏ cấu trúc dùng để lưu địa chỉ của thực thể instance, và member là tên thành viên cần truy cập của thực thể đó.

#include <iostream>

using namespace std;

//Khai báo cấu trúc

typedef struct person {

    string name;

    char sex;

    int age;

    string add;

    string job;

} person2;

int main() {

    //Khai báo thực thể kiyoshi, và con trỏ tới thực thể này

    person2 kiyoshi, \*p1;

    // Gán địa chỉ của thực thể vào con trỏ

    p1 = &kiyoshi;

    //Truy cập và gán giá trị vào thành viên trong con trỏ cấu trúc

    p1->name = "Kiyoshi";

    p1->sex = 'M';

    p1->age = 20;

    p1->add = "Tokyo";

    p1->job = "BrSE";

    cout << kiyoshi.name<<' '<< kiyoshi.sex<<' '<< kiyoshi.age

        <<' '<< kiyoshi.add<<' '<< kiyoshi.job<<endl;

     //Khai báo thực thể honda, và con trỏ tới thực thể này

    person2 honda, \*p2;

    // Gán địa chỉ của thực thể vào con trỏ

    p2 = &honda;

    //Gán con trỏ của thực thể kiyoshi vào con trỏ của thực thể honda

    //Qua đó tiến hành copy thực thể kiyoshi vào thực thể honda

    \*p2 = \*p1;

    cout << honda.name<<' '<< honda.sex<<' '<< honda.age

        <<' '<< honda.add<<' '<< honda.job;

    return 0;

}

Kiyoshi M 20 Tokyo BrSE

Kiyoshi M 20 Tokyo BrSE

### Sử dụng con trỏ cấu trúc trong hàm

VD

#include <iostream>

using namespace std;

//Khai báo cấu trúc biểu diễn số phức

typedef struct {

    double re;    /\* Phần thực \*/

    double im;    /\* Phần ảo \*/

} complex\_t;

//Tạo hàm tính tổng các cấu trúc thôg qua con trỏ cấu trúc

void addComplexPtr(complex\_t \*a, complex\_t \*b, complex\_t \*c)

{

    (\*c).re = (\*a).re + (\*b).re;

    (\*c).im = (\*a).im + (\*b).im;

}

//Tạo hàm nhận và in thực thể của cấu trúc

void printComplex(complex\_t c)

{

     cout <<c.re<< " + "<< c.im;

}

int main(){

    //Khởi tạo các thực thể của cấu trúc complex\_t

    complex\_t x = {1.2, 3.4};

    complex\_t y = {5.6, 7.8};

    complex\_t z;

    //Khởi tạo các con trỏ chứa địa chỉ các thực thể của cấu trúc

    complex\_t \*p1 = &x;

    complex\_t \*p2 = &y;

    complex\_t \*p3 = &z;

    addComplexPtr(p1, p2, p3);//addComplexPtr(&x, &y, &z);

    printComplex(z);

    return 0;

}

6.8 + 11.2

## Mảng cấu trúc

Mảng cấu trúc trong C++ có đặc tính của cả kiểu mảng và kiểu cấu trúc, do đó chúng ta có thể truy cập vào các thực thể trong mảng cấu trúc thông qua index, cũng như là truy cập vào các thành viên của từng thực thể theo cách thông thường.

### Khai báo

VD

//Khai báo kiểu cấu trúc

typedef struct {

    char name[20];

    char sex;

    int age;

    double height;

    double weight;

} person\_t;

main(){

    //Khai báo mảng cấu trúc chứa 3 thực thể của kiểu cấu trúc

    person\_t p[3];

}

### Khởi tạo

struct\_name array\_name[length] = {value1, value2, value3,..};

VD

#include <iostream>

using namespace std;

#define PERSON\_NUM 5

typedef struct {

    char name[20];

    char sex;

    int age;

    double height;

    double weight;

} person\_t;

main(){

    person\_t p[PERSON\_NUM] = {{"Bob",      'M', 19, 165.4, 72.5},

                              {"Alice",    'F', 19, 161.7, 44.2},

                              {"Tom",      'M', 20, 175.2, 66.3},

                              {"Stefany",  'F', 18, 159.3, 48.5},

                              {"Leonardo", 'M', 19, 172.8, 67.2}};

    return 0;

}

### Truy cập vào phần tử và thành viên

VD

#include <iostream>

using namespace std;

//Định nghĩa số thực thể trong mảng cấu trúc

#define PERSON\_NUM 5

//Khai báo kiểu cấu trúc

typedef struct {

    char name[20];

    char sex;

    int age;

    double height;

    double weight;

} person\_t;

main(){

    //Khởi tạo mảng cấu trúc với 5 thực thể được tạo ra từ cấu trúc

    person\_t p[PERSON\_NUM] = {{"Bob",      'M', 19, 165.4, 72.5},

                              {"Alice",    'F', 19, 161.7, 44.2},

                              {"Tom",      'M', 20, 175.2, 66.3},

                              {"Stefany",  'F', 18, 159.3, 48.5},

                              {"Leonardo", 'M', 19, 172.8, 67.2}};

    //Khai báo các biến tổng và trung bình của chiều cao và cân nặng

    double height\_sum, weight\_sum, height\_ave, weight\_ave;

    height\_sum = weight\_sum = 0.0;

    //Sử dụng vòng lặp để tính tổng chiều cao và cân nặng từ các thực thể

    for (short i = 0; i < PERSON\_NUM; i++) {

        height\_sum += p[i].height;

        weight\_sum += p[i].weight;

    }

    //Từ tổng chiều cao và cân nặng tính ra giá trị trung bình

    height\_ave = height\_sum / PERSON\_NUM;

    weight\_ave = weight\_sum / PERSON\_NUM;

    cout << "Chieu cao trung binh = "<< height\_ave<<endl;

    cout << "Can nang trung binh = "<< weight\_ave;

    return 0;

}

Chieu cao trung binh = 166.88

Can nang trung binh = 59.74

## Cấu trúc và hàm

### Truyền cấu trúc cho hàm

VD

#include <iostream>

using namespace std;

//Khai báo cấu trúc biểu diễn số phức

typedef struct {

    double re;    /\* Phần thực \*/

    double im;    /\* Phần ảo \*/

} complex\_t;

//Tạo hàm nhận và in thực thể của cấu trúc

void printComplex(complex\_t num)

{

    cout <<num.re<< " + "<< num.im;

}

int main(){

    //Tạo một thực thể từ cấu trúc

    complex\_t num = {1.2, 3.4};

    //Truyền thực thể vào hàm

    printComplex(num);

}

1.2 + 3.4

### Trả về cấu trúc từ hàm

#include <iostream>

using namespace std;

typedef struct {

    double re;    /\* Phần thực \*/

    double im;    /\* Phần ảo \*/

} complex\_t;

//Tạo hàm nhận, tính toán và trả về thực thể của cấu trúc

complex\_t addComplex(complex\_t a, complex\_t b){

    complex\_t c;

    c.re = a.re + b.re;

    c.im = a.im + b.im;

    return c;

}

//Tạo hàm nhận và in thực thể của cấu trúc

void printComplex(complex\_t c){

    cout <<c.re<< " + "<< c.im;

}

int main(){

    complex\_t a = {1.2, 3.4};

    complex\_t b = {5.6, 7.8};

    complex\_t c;

    c = addComplex(a, b);

    printComplex(c);

    return 0;

}

6.8 + 11.2

## Sao chép cấu trúc

### Cách 1: sử dụng toán tử =

#include <iostream>

using namespace std;

typedef struct {

    char name[20];

    char sex[10];

    int age;

    double height;

    double weight;

} person\_1;

int main()

{

    person\_1 p1 = {"Tom", "M", 19, 175.2, 69.5};

    person\_1 p2;

    p2 = p1; /\*Sao chép struct p1 sang p2 \*/

    cout << "p1: "<< p1.name << ' ' << p1.sex << ' ' << p1.age << ' ' << p1.height << ' ' << p1.weight<<endl;

    cout << "p2: "<< p2.name << ' ' << p2.sex << ' ' << p2.age << ' ' << p2.height << ' ' << p2.weight<<endl;

    return 0;

}

### Cách 2: Sử dụng con trỏ

VD

#include <iostream>

using namespace std;

typedef struct person {

    string name;

    char sex;

    int age;

    string add;

    string job;

} person2;

int main() {

    person2 kiyoshi = {"Kiyoshi", 'm', 30, "Tokyo", "BrSE"};

    cout << "p1: "<< kiyoshi.name << ' ' << kiyoshi.age << ' ' << kiyoshi.add << ' ' << kiyoshi.job <<endl;

    //Khai báo con trỏ nguồn và gán địa chỉ thực thể nguồn

    person2 \*p1;

    p1 = &kiyoshi;

    //Khai báo con trỏ đích và gán địa chỉ thực thể đích

    person2 kiyoshi2, \*p2;

    p2 = &kiyoshi2;

    //Gán con trỏ nguồn vào con trỏ đích

    //Qua đó sao chép thực thể nguồn vào thực thể đích

    \*p2 = \*p1;

    cout << "p2: "<< kiyoshi2.name << ' ' << kiyoshi2.age << ' ' << kiyoshi2.add << ' ' << kiyoshi2.job <<endl;

    return 0;

}

p1: Kiyoshi 30 Tokyo BrSE

p2: Kiyoshi 30 Tokyo BrSE

## So sánh 2 struct

### Cách 1: So sánh các thành viên với nhau

#include <iostream>

#include <cstring>

using namespace std;

//Khai báo struct person

typedef struct person {

    char name[100];

    char sex;

    int age;

    char add[100];

    char job[100];

} person2;

//Tạo hàm so sánh 2 struct trong C++

int compare\_struct(person2 \*p1, person2 \*p2) {

    // return 1: Khác nhau

    // return 0: Giống nhau

    //So sánh các thành viên thuộc kiểu số

    if(p1->sex != p2->sex || p1->age != p2->age) {

        return 1;

    }

    //So sánh các thành viên thuộc kiểu chuỗi bằng hàm strcmp

    if(strcmp(p1->name, p2->name) != 0) {

        return 1;

    }

    if(strcmp(p1->add, p2->add) != 0) {

        return 1;

    }

    if(strcmp(p1->job, p2->job) != 0) {

        return 1;

    }

    return 0;

}

int main() {

    person2 kiyoshi = {"kiyoshi", 'm', 30, "Tokyo", "BrSE"};

    //Tạo kiyoshi2 giống kiyoshi

    //bằng cách copy struct kiyoshi sang kiyoshi2 thông qua con trỏ

    person2 \*p1;

    p1 = &kiyoshi;

    person2 kiyoshi2, \*p2;

    p2 = &kiyoshi2;

    \*p2 = \*p1;

    //Tạo thực thể honda khác kiyoshi bằng cách khởi tạo giá trị mới

    person2 honda = {"Honda", 'm', 30, "Tokyo", "BrSE"};

    // So sánh struct kiyoshi và kiyoshi2

    if(compare\_struct(&kiyoshi, &kiyoshi2) == 0) {

        cout << "struct kiyoshi giong struct kiyoshi2\n";

    } else {

        cout << "struct kiyoshi khac struct kiyoshi2\n";

    }

    // So sánh struct kiyoshi và honda

    if(compare\_struct(&kiyoshi, &honda) == 0) {

        cout << "struct kiyoshi giong struct honda\n";

    } else {

        cout << "struct kiyoshi khac struct honda\n";

    }

    return 0;

}

struct kiyoshi giong struct kiyoshi2

struct kiyoshi khac struct honda

### Cách 2: Sử dụng hàm memcmp()

#include <iostream>

#include <cstring>

using namespace std;

//Khai báo struct person

typedef struct person {

    char name[100];

    char sex;

    int age;

    char add[100];

    char job[100];

} person2;

int main() {

    person2 kiyoshi = {"kiyoshi", 'm', 30, "Tokyo", "BrSE"};

    //Tạo kiyoshi2 giống kiyoshi

    //bằng cách copy struct kiyoshi sang kiyoshi2 thông qua con trỏ

    person2 \*p1;

    p1 = &kiyoshi;

    person2 kiyoshi2, \*p2;

    p2 = &kiyoshi2;

    \*p2 = \*p1;

    //Tạo thực thể honda khác kiyoshi bằng cách khởi tạo giá trị mới

    person2 honda = {"Honda", 'm', 30, "Tokyo", "BrSE"};

    // So sánh struct kiyoshi và kiyoshi2

    if(memcmp(&kiyoshi, &kiyoshi2, sizeof(person2)) == 0) {

        cout << "struct kiyoshi giong struct kiyoshi2\n";

    } else {

        cout << "struct kiyoshi khac struct kiyoshi2\n";

    }

    // So sánh struct kiyoshi và honda

    if(memcmp(&kiyoshi, &honda, sizeof(person2)) == 0) {

        cout << "struct kiyoshi giong struct honda\n";

    } else {

        cout << "struct kiyoshi khac struct honda\n";

    }

    return 0;

}

struct kiyoshi giong struct kiyoshi2

struct kiyoshi khac struct honda

## Xắp xếp theo thành viên

VD

#include <iostream>

#include <cstring>

using namespace std;

/\*Định nghĩa macro SWAP để hoán đổi phần tử trong mảng chỉ định\*/

#define SWAP(type,x,y) do{type tmp = x; x = y; y = tmp;}while(0)

#define PERSON\_NUM 5

//Khai báo kiểu cấu trúc

typedef struct {

    char name[20];

    char sex;

    int age;

    double height;

    double weight;

} person\_t;

//Tạo hàm in kiểu cấu trúc

void show\_struct(person\_t array[], int n){

    for (short i = 0; i < PERSON\_NUM; i++) {

      cout << array[i].name<<' '<< array[i].sex<<' '

      << array[i].age<<' '<< array[i].height<<' '<< array[i].weight<<endl;

   }

    cout <<endl;

}

//Tạo hàm sắp xếp struct tăng dần theo chiều cao

void asc\_order\_height(person\_t array[], int n){

    //Tạo vòng lặp để sắp xếp struct

    for(int i = 0; i < n - 1; i++){

        for(int j = i + 1; j < n; j++){

            //Nếu tìm ra phần tử có thành viên height lớn hơn

            //thì hoán đổi với số đang xét

            if (array[i].height > array[j].height) SWAP(person\_t,array[i],array[j]);

        }

    }

}

//Tạo hàm sắp xếp struct giảm dần theo chiều cao

void desc\_order\_height(person\_t array[], int n){

    //Tạo vòng lặp để sắp xếp struct

    for(int i = 0; i < n - 1; i++){

        for(int j = i + 1; j < n; j++){

            //Nếu tìm ra phần tử có thành viên height nhỏ hơn

            // thì hoán đổi với số đang xét

            if (array[i].height < array[j].height) SWAP(person\_t,array[i],array[j]);

        }

    }

}

// Tạo hàm sắp xếp struct tăng dần theo tên

void asc\_order\_name(person\_t array[], int n){

    //Tạo vòng lặp để sắp xếp struct

    for(int i = 0; i < n - 1; i++){

        for(int j = i + 1; j < n; j++){

            //Nếu tìm ra phần tử có thành viên name lớn hơn

            // thì hoán đổi với số đang xét

            if (strcmp(array[i].name, array[j].name) > 0) SWAP(person\_t,array[i],array[j]);

        }

    }

}

// Tạo hàm sắp xếp struct giảm dần theo tên

void desc\_order\_name(person\_t array[], int n){

    //Tạo vòng lặp để sắp xếp struct

    for(int i = 0; i < n - 1; i++){

        for(int j = i + 1; j < n; j++){

            //Nếu tìm ra phần tử có thành viên name nhỏ hơn

            // thì hoán đổi với số đang xét

            if (strcmp(array[i].name, array[j].name) < 0) SWAP(person\_t,array[i],array[j]);

        }

    }

}

int main(){

    //Khởi tạo mảng cấu trúc với 5 thực thể được tạo ra từ cấu trúc

    person\_t p[PERSON\_NUM] = {{"Bob", 'M', 19, 165.4, 72.5},

                              {"Ali", 'F', 19, 161.7, 44.2},

                              {"Tom", 'M', 20, 175.2, 66.3},

                              {"Ste", 'F', 18, 159.3, 48.5},

                              {"Leo", 'M', 19, 172.8, 67.2}};

    cout << "Mang struct ban dau:\n";

    show\_struct(p,PERSON\_NUM);

    cout << "\nSap xep theo chieu cao tang dan:\n";

    asc\_order\_height(p,PERSON\_NUM);

    show\_struct(p,PERSON\_NUM);

    cout << "\nSap xep theo chieu cao giam dan:\n";

    desc\_order\_height(p,PERSON\_NUM);

    show\_struct(p,PERSON\_NUM);

    cout << "\nSap xep theo ten tang dan:\n";

    asc\_order\_name(p,PERSON\_NUM);

    show\_struct(p,PERSON\_NUM);

    cout << "\nSap xep theo ten giam dan:\n";

    desc\_order\_name(p,PERSON\_NUM);

    show\_struct(p,PERSON\_NUM);

}

Mang struct ban dau:

Bob M 19 165.4 72.5

Ali F 19 161.7 44.2

Tom M 20 175.2 66.3

Ste F 18 159.3 48.5

Leo M 19 172.8 67.2

Sap xep theo chieu cao tang dan:

Tom M 20 175.2 66.3

Leo M 19 172.8 67.2

Bob M 19 165.4 72.5

Ali F 19 161.7 44.2

Ste F 18 159.3 48.5

Sap xep theo chieu cao giam dan:

Ste F 18 159.3 48.5

Ali F 19 161.7 44.2

Bob M 19 165.4 72.5

Leo M 19 172.8 67.2

Tom M 20 175.2 66.3

Sap xep theo ten tang dan:

Ali F 19 161.7 44.2

Bob M 19 165.4 72.5

Leo M 19 172.8 67.2

Ste F 18 159.3 48.5

Tom M 20 175.2 66.3

Sap xep theo ten tang dan:

Tom M 20 175.2 66.3

Ste F 18 159.3 48.5

Leo M 19 172.8 67.2

Bob M 19 165.4 72.5

Ali F 19 161.7 44.2

# Tham khảo

Lập trình căn bản:

<https://laptrinhcanban.com/cpp/lap-trinh-cpp-co-ban/>