

http://

Tuhoc.cc



# LẬP TRÌNH



# Dễ hiểu



LESSON

## 23.1

## Con trỏ C++

### Part 1



@galailaptrinh



0

## Tổng quan

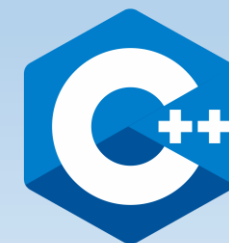
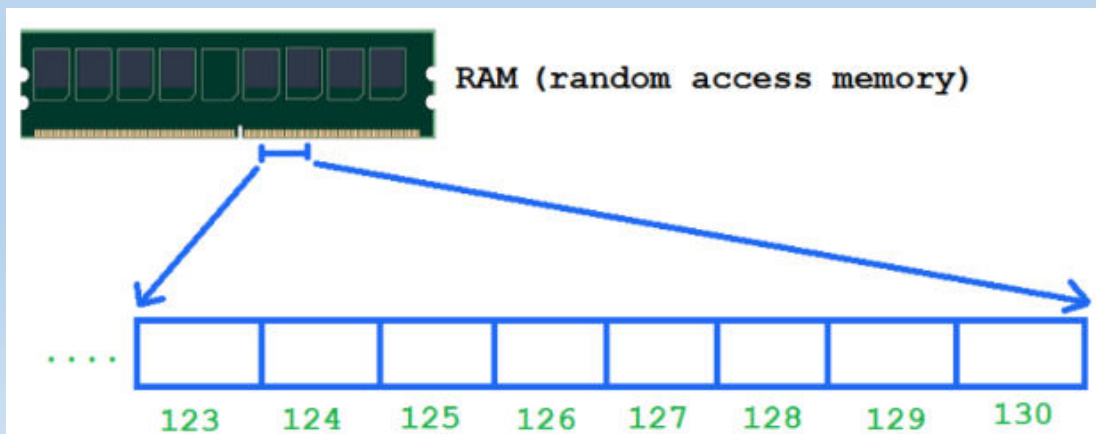
### ❑ 0. Địa chỉ của biến :

*Khi chúng ta khai báo biến:*

- ✓ *Con PC của các thím nó sẽ lấy 1 vùng nhớ trên RAM để lưu trữ các giá trị của biến*
- ✓ *Để kiểm tra xem tên vùng nhớ mà nó chiếm trên RAM các thím dùng từ khóa &*

```
//khai báo biến thông thường
int nhietDo = 78;
//xem địa chỉ ô nhớ mà biến nhietDo chiếm
cout << "địa chỉ ô nhớ của biến nhietDo : " << &nhietDo << endl;
```

địa chỉ ô nhớ của biến nhietDo : 0000052FC5FF544



1

## Con trỏ C++

❑ 1. **Khái niệm** : Con trỏ trong C++ là một biến đặc biệt . Nó chứa địa chỉ của 1 ô nhớ trên máy tính .

a. **Khai báo**: Kiểu\_dữ\_liệu \* <tên biến>;

b. **Cấp phát**: <tên biến> = new Kiểu\_dữ\_liệu;

Phải cùng kiểu dữ liệu lúc khai báo

c. **Hủy bộ nhớ**: delete <tên biến>;

```
//khai báo con trỏ
int* a;
a = new int; // cấp phát bộ nhớ
//=> Xuất trực tiếp con trỏ. Trả về địa chỉ ô nhớ mà a chiếm
cout << "địa chỉ a tro toi: " << a << endl;

//gán giá trị cho biến con trỏ
*a = 25;
cout << "gia tri cua con tro a: " << *a << endl;
```



2

## Toán tử con trỏ C++

### ❑ 2. Toán tử 1 ngôi : \* và &

```
dia chi o nho cua bien nhietDo : 000000D1B316FA54
dia chi conTroT tro toi: 000000D1B316FA54
gia tri cua con tro conTroT: 78
chomChom = 78
```

```
//2.1 Toán tử 1 ngôi : &
//khai báo biến thông thường
int nhietDo = 78;
//xem địa chỉ ô nhớ mà biến nhietDo chiếm
cout << "dia chi o nho cua bien nhietDo : " << &nhietDo << endl;

//khai báo con trỏ
int* conTroT;

//Trỏ biến conTroT vào ô nhớ của biến nhietDo
conTroT = &nhietDo;
cout << "dia chi conTroT tro toi: " << conTroT << endl;
cout << "gia tri cua con tro conTroT: " << *conTroT << endl;

//2.2 Toán tử 1 ngôi : *
//Muốn lấy giá trị của con trỏ thì dùng toán tử *
int chomChom = *conTroT; // lấy giá trị của conTroT gán vào biến chomChom
cout << "chomChom = " << chomChom << endl;
```

Biến thông thường xem địa chỉ ô nhớ dùng &

Biến con trỏ xem giá trị : dùng \*





http://

Tuhoc.cc



# LẬP TRÌNH



# Dễ hiểu



LESSON

## 23.2

## Con trỏ C++

*Con trỏ void – con trỏ null*



@galailaptrinh



3

## Con trỏ void – con trỏ null

❑ **3. Lưu ý:** *Khi khai báo con trỏ, ta đã khai báo kiểu dữ liệu mà con trỏ trỏ đến. Địa chỉ đặt vào biến con trỏ phải cùng kiểu với kiểu con trỏ*

**a. Khai báo:** Kiểu\_dữ\_liệu \***<tên biến>;**

**b. Cấp phát:** **<tên biến> = new** Kiểu\_dữ\_liệu;

**Phải cùng kiểu dữ liệu lúc khai báo**

```
//biến thường
int nguyenA;
float thucB;

//khai báo biến con trỏ
int* conTroNguyen;
float* conTroThuc;

//Gán con trỏ hợp lệ
conTroNguyen = &nguyenA;
conTroThuc = &thucB;

//không hợp lệ
conTroNguyen = &thucB;
conTroThuc =
```

(local variable) int \*conTroNguyen  
khai báo biến con trỏ  
Search Online



3

## Con trỏ void – con trỏ null

❑ 4. Con trỏ void: *Nếu ta muốn con trỏ có thể trỏ đến bất kỳ kiểu dữ liệu nào khác thì dùng **void** thay cho kiểu dữ liệu lúc khai báo con trỏ*

*a. Khai báo: **void** \* <tên biến>;*

```
//biến thường
int nguyenA;
float thucB;
```

```
//khai báo biến con trỏ:
void* giCungDuoc;
//có thể trỏ tới bất kỳ kiểu dữ liệu nào
giCungDuoc = &nguyenA;
giCungDuoc = &thucB;
```



3

## Con trỏ void – con trỏ null

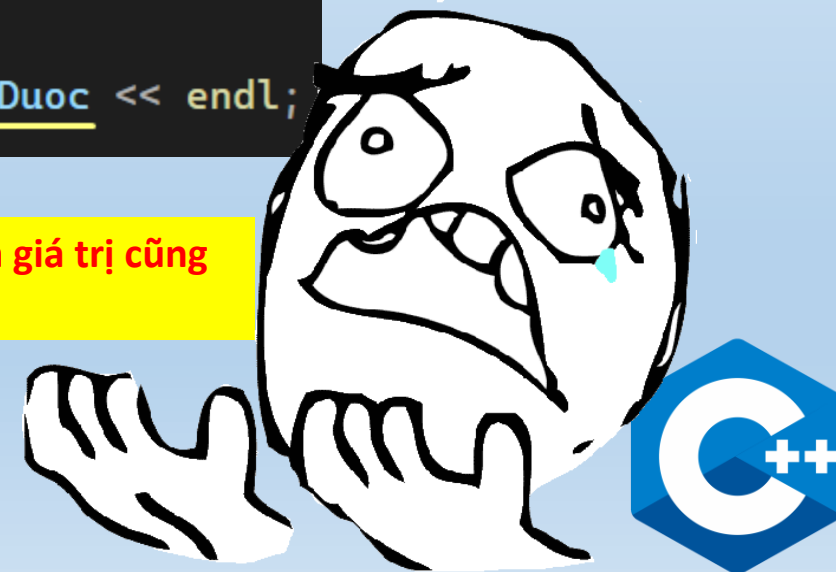
❑ 4. Con trỏ void: **C++ rất ngu**, void để trỏ tới nhiều kiểu dữ liệu, nhưng khi can thiệp đến các giá trị lại phải cần ép kiểu dữ liệu về kiểu dữ liệu nó trỏ tới :

```
//ép kiểu dữ liệu khi can thiệp giá trị
giCungDuoc = &nguyenA;
* (int*)giCungDuoc = 10;

//xuất giá trị nguyenA và giá trị tại con trỏ giCungDuoc
cout << "nguyenA = " << nguyenA << endl;
cout << "gia tri nam tren giCungDuoc = " << *(int*)giCungDuoc << endl;
```



Ngay cả truy xuất đến giá trị cũng phải ép kiểu luôn ☹






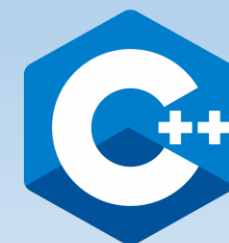
3

## Con trỏ void – con trỏ null

- ❑ 5. Con trỏ null : *Khi chúng ta **khái báo mà chưa cấp phát bộ nhớ** thì sẽ gặp lỗi này*  
*Đối với visual studio thì **chương trình sẽ không chạy** :*

```
//khái báo biến con trỏ
int* conTroxyz;
cout << "ô nhớ conTroxyz trỏ đến = " << conTroxyz << endl;
```

 (local variable) int \*conTroxyz  
 khái báo biến con trỏ  
 Search Online  
 C6001: Using uninitialized memory 'conTroxyz'.  
 Show potential fixes (Alt+Enter or Ctrl+.)



http://

Tuhoc.cc



# LẬP TRÌNH



# Dễ hiểu



LESSON

## 23.3

## Con trỏ C++

*Con trỏ và mảng 1 chiều*



@galailaptrinh

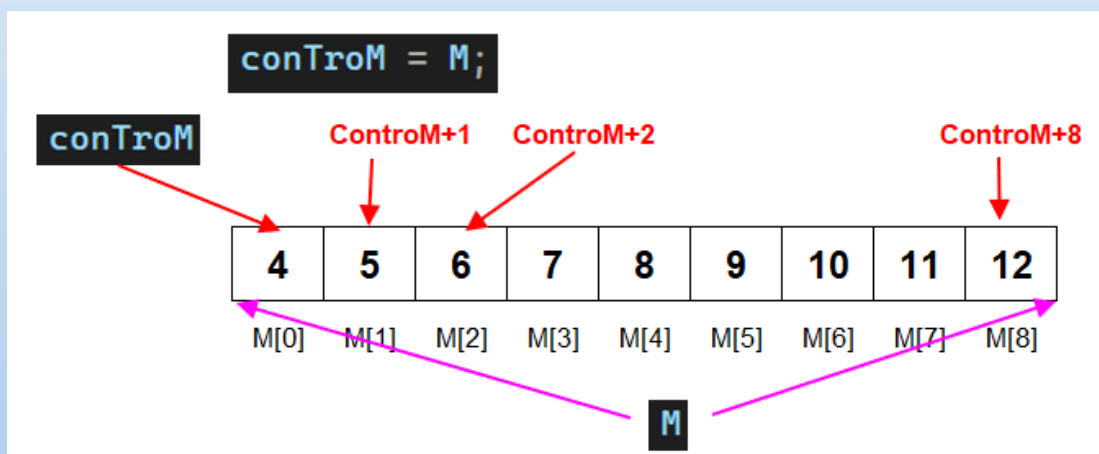


4

## Con trỏ và mảng

❑ 5. Con trỏ tương tác với mảng 1 chiều : *Con trỏ và mảng có quan hệ gần, mảng cũng quản lý theo địa chỉ ô nhớ.*

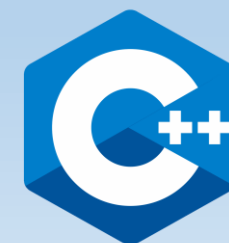
✓ Có thể **gán trực tiếp con trỏ vào mảng** mà không cần dùng toán tử &



```
//5. con trỏ và mảng
int mang[] = { 4,5,6,7,8,9,10,11,12 };
//khai báo con trỏ
int* conTroM;

//gán con trỏ vào mảng( không cần &)
//=> Trỏ conTroM tới địa chỉ ô nhớ của mang
conTroM = mang;
```

- 1, Sau khi gán, con trỏ sẽ trỏ đến ô nhớ đầu tiên
- 2, Con trỏ dịch chuyển +1, +2 ... tương ứng với các phần tử M[1], M[2].....



4

## Con trỏ và mảng

### ❑ 5. Con trỏ tương tác với mảng 1 chiều :

```
//5. con trỏ và mảng
int M[] = { 4,5,6,7,8,9,10,11,12 };
//khai báo con trỏ
int* conTroM;

//gán con trỏ vào mảng( không cần &)
//=> Trỏ conTroM tới địa chỉ ô nhớ của M
conTroM = M;

//duyet for
for (int i = 0; i < size(M);i++) {
    //lấy giá trị của ô nhớ thứ conTroM+i
    cout << *(conTroM + i) << " ";
}
```

4 5 6 7 8 9 10 11 12

Ta có thể thấy mỗi lần i chạy, con trỏ dịch chuyển, ta lấy được giá trị tại nơi con trỏ trỏ tới => Kết quả là thu được danh sách giá trị của mảng



4

## Con trỏ và mảng

### ❑ 5.5. Con trỏ tương tác với mảng 1 chiều :

*Sử dụng Con trỏ để thay đổi giá trị của mảng*

```
//5.2 thay đổi giá trị của mảng dùng con trỏ
//thay giá trị tại ô nhớ conTroM + 2
*(conTroM + 2) = 10;

//xuất lại mảng M
cout << "\nMang sau doi la: \n";
for (int i = 0; i < size(M); i++) {
    //lấy giá trị của ô nhớ thứ conTroM+i
    cout << M[i] << " ";
}
```

```
4 5 6 7 8 9 10 11 12
Mang sau doi la:
4 5 10 7 8 9 10 11 12
```





5

## Mảng 1 chiều con trỏ

❑ **6. Mảng các phần tử là con trỏ :** *Ta có thể khai báo 1 mảng, mà các phần tử toàn là các con trỏ 😊 ( Các con trỏ quản lý các ô nhớ trên ram )*

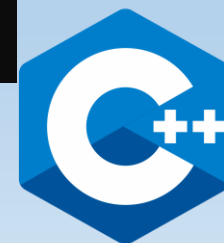
```
//khai báo mảng con trỏ
int* mangConTro[4]; //mảng có 4 phần tử, mỗi phần tử đều là con trỏ

//duyet mảng, cấp phát bộ nhớ
//lưu ý chưa cấp phát sẽ lỗi con trỏ null :)

for (int i = 0; i < size(mangConTro);i++) {
    mangConTro[i] = new int;
}

//xuất các giá trị của mảng (địa chỉ các ô nhớ)
for (int i = 0; i < size(mangConTro);i++) {
    cout << mangConTro[i] <<endl;
}
```

```
0000020AC9415200
0000020AC9415240
0000020AC9415730
0000020AC9415770
```



http://

Tuhoc.cc



# LẬP TRÌNH



# Dễ hiểu



LESSON

## 23.4

### Con trỏ cấp 2 C++

*Con trỏ và mảng 2 chiều*



@galailaptrinh



5

## Mảng 2 chiều và con trỏ cấp 2

❑ **7. Mảng 2 chiều con trỏ :** *Giống với mảng 2 chiều thông thường, ta có thể khai báo 1 ma trận, có  $n$  dòng  $m$  cột – (lưu ý các phần tử này toàn bộ là các con trỏ)*

	Cột 0	Cột 1	Cột 2	Cột 3
Dòng 0	M[0][0] 7	M[0][1] 2	M[0][2] 9	M[0][3] 0
Dòng 1	M[1][0] 9	M[1][1] 5	M[1][2] 4	M[1][3] 1
Dòng 2	M[2][0] 8	M[2][1] 0	M[2][2] 3	M[2][3] 6

Tên mảng —————  
 Chỉ số cột  
 Chỉ số dòng

```
const int dong = 3;
const int cot = 4;

int** capHai = new int* [dong];
for (int i = 0; i < dong; i++)
{
    //di chuyển con trỏ bên trong để cấp phát
    *(capHai + i) = new int[cot];
}
```

**3 dòng**  
**4 phần tử**



5

## Mảng 2 chiều và con trỏ cấp 2

### ❑ 7.2 Xuất các địa chỉ ô nhớ trong ma trận con trỏ :

```
//7.2 xuất địa chỉ các ô nhớ trong mảng con trỏ cấp 2
for (int i = 0; i < dong; i++) {
    for (int j = 0; j < cot; j++) {
        //khi sử dụng ngoặc vuông:
        //sẽ truy xuất đến giá trị trên ô nhớ tại i, j
        //vì vậy cần dùng thêm toán tử &
        cout << &capHai[i][j] << " " ;
    }
    cout << endl;
}
```

```
const int dong = 3;
const int cot = 4;
```

```
000001DFFBA65D20 000001DFFBA65D24 000001DFFBA65D28 000001DFFBA65D2C
000001DFFBA654B0 000001DFFBA654B4 000001DFFBA654B8 000001DFFBA654BC
000001DFFBA65190 000001DFFBA65194 000001DFFBA65198 000001DFFBA6519C
```



5

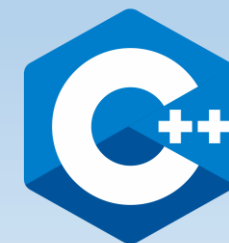
## Mảng 2 chiều và con trỏ cấp 2

### ❑ 7.3 Gán và xuất các giá trị trên các ô nhớ trong mảng con trỏ cấp 2 :

```
//7.3 gán giá trị vào các ô nhớ trong mảng con trỏ cấp 2
for (int i = 0; i < dong; i++) {
    for (int j = 0; j < cot; j++) {
        capHai[i][j] = i+j;
    }
}

//7.4 duyệt mảng xuất các giá trị trên ma trận
for (int i = 0; i < dong; i++) {
    for (int j = 0; j < cot; j++) {
        cout << capHai[i][j] << " ";
    }
    cout << endl;
}
```

0	1	2	3
1	2	3	4
2	3	4	5





5

## Mảng 2 chiều và con trỏ cấp 2

### ❑ 7.4 Hủy bộ nhớ :

```
const int dong = 3;
const int cot = 4;

int** capHai = new int* [dong];
for (int i = 0; i < dong; i++)
{
    //di chuyển con trỏ bên trong để cấp phát
    capHai[i] = new int[cot];
}
```

Chỗ nào có cấp phát thì cần hủy , để không bị chiếm dụng bộ nhớ trên thanh RAM

```
//7.5 hủy con trỏ
for (int i = 0; i < dong; i++) {
    delete *(capHai+i);
}
delete capHai;
```



2

## Bài tập C++ 23 -24

### □ Bài tập C++ 23

- ✓ 1. *Viết hàm (function) hoán vị 2 biến  $a, b$  là số thực bằng cách sử dụng con trỏ*
- ✓ 2. *Viết chương trình nhập 2 số thực  $a, b$ . Sử dụng hàm trên để đổi chỗ  $a, b$*

### □ Bài tập C++ 24

- ✓ 1. *Viết chương trình nhập vào một mảng 1 chiều  $M$ , gồm  $n$  phần tử nguyên ngẫu nhiên  $[0,100]$ .*
- ✓ 2. *Sắp xếp mảng  $M$  theo chiều giảm dần, lưu ý: yêu cầu sử dụng tên mảng như con trỏ và sử dụng con trỏ*

```
Mang ngau nhien vua tao ra:  
25 94 71 5 25 52 65 79 60 21  
Mang sau sx giam la:  
94 79 71 65 60 52 25 25 21 5
```



http://

Tuhoc.cc



# LẬP TRÌNH



# Dễ hiểu



LESSON

## 23.5

## Giải bài tập C++ 23

*Con trỏ C++*



@galailaptrinh



5

## Giải Bài tập 23

```

/*
Bài tập C++ 23
1. Viết hàm (function) hoán vị 2 biến a, b là số thực bằng cách sử dụng con trỏ
2. Viết chương trình nhập 2 số thực a, b. Sử dụng hàm trên để đổi chỗ a, b
*/

#include <iostream>
using namespace std;

void hoanDoi(double* a, double* b) {
    double temp = *a; // biến tạm = giá trị của a
    *a = *b; // gán giá trị a = giá trị nằm trên ô nhớ b
    *b = temp; // gán giá trị b = biến tạm
}

int main()
{
    double a, b;
    cout << "Moi nhap a: ";
    cin >> a;
    cout << "Moi nhap b: ";
    cin >> b;
    //gọi hàm hoán đổi
    hoanDoi(&a, &b);
    cout << "sau hoan doi a= " << a << " va gia tri b= " << b << endl;
}

```

```

Moi nhap a: 8
Moi nhap b: 9
sau hoan doi a= 9 va gia tri b= 8

```

*Không cần thiết phải delete con trỏ trong chương trình trên.  
Con trỏ chỉ trỏ đến vùng nhớ của biến a và b,  
nó không tạo ra bất kỳ vùng nhớ mới nên không cần phải giải phóng.*





http://

Tuhoc.cc



# LẬP TRÌNH



# Dễ hiểu



LESSON

## 23.6

## Giải bài tập C++ 24

### Con trỏ C++



@galailaptrinh





5

Giải Bài tập 24

```
#include <iostream>
#include<random>
using namespace std;

//hàm nhập mảng
//cần thêm dấu & vì có thay đổi giá trị của biến sau khi thoát hàm
//xem lại bài 18, truyền tham biến và truyền tham trị
void Nhap(int* &M, int n)
{
    M = new int[n];
    //duyet từng phần tử của mảng
    //gán phần tử ngẫu nhiên
    random_device rd;
    mt19937 gen(rd());
    uniform_int_distribution<> dis(0, 100);

    for (int i = 0; i < n; i++) {
        //sử dụng M+i để di chuyển con trỏ
        *(M+i) = dis(gen); //gán giá trị thứ i
    }
}

//xuất mảng
void Xuat(int* &M, int n) {
    for (int i = 0; i < n; i++) {
        //sử dụng M+i để di chuyển con trỏ
        cout << *(M + i) << " ";
    }
}
```

```
//hàm hoán đổi 2 số nguyên
void hoanDoi(int* a, int* b) {
    int temp = *a; // biến tạm = giá trị của a
    *a = *b; // gán giá trị a = giá trị nằm trên ô nhớ b
    *b = temp; // gán giá trị b = biến tạm
}

//sắp xếp
void SapXep(int* &M, int n) {
    //chạy 2 vòng for để so sánh số trước và sau:
    //thuật toán sắp xếp nổi bọt (bubble sort)
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            //cout << i << "-" << j << "\t";
            //kiểm tra số trước < số liền kề thì hoán đổi
            if (*(M + i) < *(M + j)) {
                hoanDoi(M + i, M + j);
            }
        }
        //cout << endl;
    }
}
```



5

## Giải Bài tập 24

```
int main()
{
    //khai báo con trỏ
    int* M;
    int n = 10;
    //gọi hàm nhập mảng
    Nhap(M, n);
    cout << "Mang ngau nhien vua tao ra: \n";
    Xuat(M, n);

    cout << endl;
    //gọi hàm sắp xếp
    SapXep(M, n);

    //xuất mảng sau sx
    cout << "Mang sau sx giam la: \n";
    Xuat(M, n);
}
```

```
Mang ngau nhien vua tao ra:
25 94 71 5 25 52 65 79 60 21
Mang sau sx giam la:
94 79 71 65 60 52 25 25 21 5
```



5

## Giải Bài tập 24

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

0-1	1	2	3	4	5	6	7	8	9	10
Hoán đổi	2	1	3	4	5	6	7	8	9	10
0-2	2	1	3	4	5	6	7	8	9	10
Hoán đổi	3	1	2	4	5	6	7	8	9	10
0-3	4	1	2	3	5	6	7	8	9	10
0-4	5	1	2	3	4	6	7	8	9	10
0-5	6	1	2	3	4	5	7	8	9	10
0-6	7	1	2	3	4	5	6	8	9	10
0-7	8	1	2	3	4	5	6	7	9	10
0-8	9	1	2	3	4	5	6	7	8	10
0-9	10	1	2	3	4	5	6	7	8	9

1-2	10	2	1	3	4	5	6	7	8	9
1-3	10	3	1	2	4	5	6	7	8	9
1-4	10	4	1	2	3	5	6	7	8	9
1-5	10	5	1	2	3	4	6	7	8	9
1-6	10	6	1	2	3	4	5	7	8	9
1-7	10	7	1	2	3	4	5	6	8	9
1-8	10	8	1	2	3	4	5	6	7	9
1-9	10	9	1	2	3	4	5	6	7	8

```
void SapXep(int*& M, int n) {
    //chạy 2 vòng để so sánh số trước và số sau
    //thuật toán sắp xếp nổi bọt( bubble sort)
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            //kiểm tra xem số trước < số liền kề hay không
            if (*(M + i) < *(M + j)) {
                hoanDoi(M + i, M + j);
            }
        }
    }
}
```

0-1	0-2	0-3	0-4	0-5	0-6	0-7	0-8	0-9
1-2	1-3	1-4	1-5	1-6	1-7	1-8	1-9	
2-3	2-4	2-5	2-6	2-7	2-8	2-9		
3-4	3-5	3-6	3-7	3-8	3-9			
4-5	4-6	4-7	4-8	4-9				
5-6	5-7	5-8	5-9					
6-7	6-8	6-9						
7-8	7-9							
8-9								

