VIỆN ĐIỆN TỬ - VIỄN THÔNG

School of Electronics and telecommunications

KỸ THUẬT LẬP TRÌNH C/C++

MẢNG VÀ QUẢN LÝ BỘ NHỚ ĐỘNG

Giảng viên: TS. Nguyễn Thị Kim Thoa

Học kỳ: 20211

Nội dung

- Con trỏ nhiều cấp
- Mảng động nhiều chiều
- Con trỏ hàm

Con trỏ "nhiều cấp"

- Mỗi biến khi được khai báo sẽ có một địa chỉ xác định, có thể dùng con trỏ để lưu địa chỉ của biến.

  VD: int a=7; int *pa=&a; //pa là con trỏ cấp 1

- Con trỏ cũng là một biến, khi khai báo thì nó sẽ có địa chỉ như biến bình thường, có thể dùng một con trỏ khác để lưu địa chỉ của biến con trỏ.

  VD: int *ppa=&pa; //ppa là con trỏ cấp 2

- Con trỏ cấp n sẽ lưu địa chỉ của con trỏ cấp n-1

Mảng động nhiều chiều

- Con trỏ cấp 1 dùng để cấp phát động cho mảng 1 chiều

  VD: int *p=new int [10];
- Con trỏ cấp n dùng để cấp phát động cho mảng n chiều
- Ví dụ: dùng con trỏ cấp 2 để cấp phát động một mảng 2 chiều với kích thước m dòng x n cột

```
//a là mảng một chiều gồm các con trỏ cấp 1
int **a = new int*[m];
 //với mỗi con trỏ cấp 1 trong a
//ta cấp phát 1 mảng 1 chiều tương ứng
for (int i = 0; i < m; i++)
{
    //mỗi phần tử trong a là con trỏ cấp 1
    a[i] = new int[n];
}
```

Mảng động nhiều chiều

- Con trỏ cấp 1 dùng để cấp phát động cho mảng 1 chiều

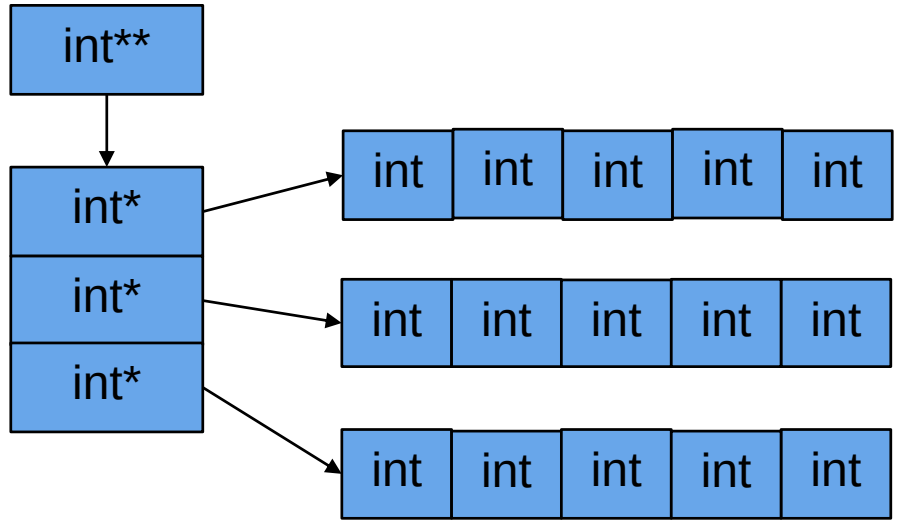
  VD: int *p=new int [10];

- Con trỏ cấp n dùng để cấp phát động cho mảng n chiều

VD: dùng con trỏ cấp 2 để cấp phát động một mảng 2 chiều với kích thước m dòng x n cột

```
//a là mảng một chiều gồm các con trỏ cấp 1
int **a = new int*[m];
//với mỗi con trỏ cấp 1 trong a
//ta cấp phát 1 mảng 1 chiều tương ứng
for (int i = 0; i < m; i++)
{
    //mỗi phần tử trong a là con trỏ cấp 1
    a[i] = new int[n];   }
```

//Minh họa mảng 2 chiều gồm 3 hàng (m=3) 5 cột (n=5)



//Thu hồi vùng nhớ của mảng a

```
for  (int  i = 0; i < m; i++)
{
    delete[] a[i];
}
delete[] a;
```

Mảng động nhiều chiều

- Nếu chương trình cần sử dụng nhiều mảng 2 chiều, nên viết hàm cấp phát và thu hồi vùng nhớ để gọi khi cần

```
void CapPhat(int **&a, int m, int n)
{
    a = new int*[m];
    for (int i = 0; i < m; i++)
    {
        a[i] = new int[n];
    }
}
```

```
void ThuHoi(int **a, int m, int n)
{
    for (int i = 0; i < m; i++)
    {
        delete[] a[i];
    }
    delete[] a;
}
```

Con trỏ hàm

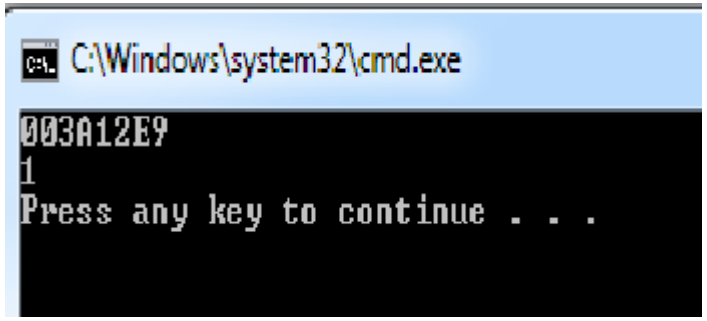
- Khi định nghĩa một hàm thì hàm cũng có địa chỉ, có thể dùng con trỏ để lưu địa chỉ của hàm.

- Con trỏ hàm là một biến lưu trữ địa chỉ của một hàm, thông qua biến đó, ta có thể gọi hàm mà nó trỏ tới.
- Khai báo con trỏ hàm phải đồng dạng với hàm cần trỏ tới:

  <kiểu trả về> (*<tên con trỏ>)(<ds tham số>);
- Chú ý: Các tham số mặc định của hàm không sử dụng được thông qua con trỏ hàm

```
int func(int a)
{
        // do something
        return a;
}
 int main()
{
        cout << func << '\n'; // in địa chỉ hàm func trong bộ nhớ
        cout << func(1) << '\n'; // đi đến địa chỉ hàm func và thực thi hàm

        return 0;
}
```



Con trỏ hàm

```
//Ví dụ: khai báo con trỏ hàm tên func
int (*func)(int, int);
//func chỉ có thể trỏ đến những hàm có dạng
int functionName(int a, int b) {...}
//Giả sử có hàm
int Max(int a, int b) {...}
//Cho func trỏ đến hàm max
func = Max;
```

```
//định nghĩa một kiểu mới có tên FuncPointer
typedef int (*FuncPointer)(int, int);
//Khai báo
FuncPointer func;
//Func cũng chỉ có thể trỏ đến
//những hàm "đồng dạng" với nó
func = Max;
```

Con trỏ hàm

```
// khai báo hàm
int funcA();
int funcB();
void funcC();
double funcD(int a);

int main()
{
        int(*fcnPtr)() = funcA(); // lỗi, không dùng dấu ngoặc đơn () sau tên hàm
        int(*fcnPtrA)() = funcA; // ok, con trỏ fcnPtrA trỏ đến hàm funcA
        fcnPtrA = funcB; // ok, fcnPtrA có thể trỏ đến một hàm khác có cùng cấu tr
        // fcnPtrA = &funcB; tương tự câu lệnh trên

        int(*fcnPtr1)() = funcA; // ok
        void(*fcnPtr2)() = funcA; // lỗi, kiểu trả về của con trỏ hàm và hàm không trùng nhau
        void(*fcnPtr3)() = funcC; // ok
        double(*fcnPtr4)(int) = funcD; // ok

        return 0;
}
```

Sử dụng con trỏ hàm - Gọi một hàm dùng con trỏ hàm

Ví dụ 1:

```
int  (*func)(int, int);

int  Max(int  a, int  b)
 {return (a>b?a:b);}

void main(){

int  a = 5;

int  b = 9;

int  max = func(a, b);
//=> max sẽ có giá trị 9
}
```

Sử dụng con trỏ hàm - Gọi một hàm dùng con trỏ hàm

Ví dụ 2:

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

void swapNumber(int &a, int &b)
{
	int temp = a;
	a = b;
	b = temp;
}
```

```
int main()
{
	void(*ptrSwap) (int &, int &) = swapNumber;

	int a = 5, b = 10;
	cout << "Before: " << a << " " << b << endl;

	// gọi hàm tường minh
	(*ptrSwap)(a, b);
	cout << "After:  " << a << " " << b << endl;

	// hoặc gọi hàm ngầm định
	ptrSwap(a, b);
	cout << "After:  " << a << " " << b << endl;

	return 0;
}
```

Sử dụng con trỏ hàm - truyền hàm vào hàm dưới dạng đối số

Ví dụ 1

```
//Một số hàm tính đơn giản
int Cong(int a, int b) { return a + b; }
int Tru(int a, int b) { return a - b; }
int Nhan(int a, int b) { return a * b; }
int Chia(int a, int b) {
        if (b)
                return a / b;
        else
                return;
}
//Định nghĩa kiểu con trỏ hàm PhepTinh
typedef int (*PhepTinh)(int, int);
```

```
//Định nghĩa hàm tính toán tổng quát
//với tham số là 2 số nguyên và con trỏ PhepTinh
int TinhToan(int a, int b, PhepTinh tinh)
{
    //Gọi hàm thông qua con trỏ hàm
    return tinh(a, b);
}
//Một số lời gọi hàm
int tong = TinhToan(5, 9, Cong);
int hieu = TinhToan(5, 9, Tru);
int nhan= TinhToan(5, 9, Nhan);
int chia= TinhToan(5, 9, Chia);
```

Sử dụng con trỏ hàm - truyền hàm vào hàm dưới dạng đối số

Ví dụ2 : Viết chương trình thực hiện việc sắp xếp tăng, giảm mảng 1 chiều các số nguyên.

Cách làm thông thường

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

// hoán đổi giá trị hai số
void swapNumber(int &a, int &b)
{
	int temp = a;
	a = b;
	b = temp;
}
```

```
// hàm sắp xếp tăng sử dụng thuật toán selection sort
void selectionSortAsc(int *arr, int n)
{
	int i, j, min_idx;
	for (i = 0; i < n - 1; i++)
	{
		// Tìm phần tử bé nhất chưa được sắp xếp
		min_idx = i;
		for (j = i + 1; j < n; j++)
		{
			if (arr[min_idx] > arr[j])
			{
				min_idx = j;
			}
		}

		// Đổi chỗ phần tử bé nhất tìm được với phần tử đầu tiên
		swapNumber(arr[min_idx], arr[i]);
	}
}
```

Sử dụng con trỏ hàm - truyền hàm vào hàm dưới dạng đối số

```
// hàm sắp xếp giảm sử dụng thuật toán selection sort
void selectionSortDesc(int *arr, int n)
{
	int i, j, max_idx;
	for (i = 0; i < n - 1; i++)
	{
		// Tìm phần tử lớn nhất chưa được sắp xếp
		max_idx = i;
		for (j = i + 1; j < n; j++)
		{
			if (arr[max_idx] < arr[j])
			{
				max_idx = j;
			}
		}

		// Đổi chỗ phần tử lớn nhất tìm được với phần tử đầu tiên
		swapNumber(arr[max_idx], arr[i]);
	}
}
```

Sử dụng con trỏ hàm - truyền hàm vào hàm dưới dạng đối số

```
/* Function to print an array */
void printArray(int arr[], int size)
{
        int i;
        for (i = 0; i < size; i++)
                cout << arr[i] << " ";
        cout << endl;
}
```

```
int main()
{
        int arr[] = { 64, 25, 12, 22, 11 };
        int n = sizeof(arr) / sizeof(int);

        // Sắp xếp tăng
        selectionSortAsc(arr, n);

        cout << "Asc array: \n";
        printArray(arr, n);

        // Sắp xếp giảm
        selectionSortDesc(arr, n);

        cout << "Desc array: \n";
        printArray(arr, n);

        return 0;
}
```

Sử dụng con trỏ hàm - truyền hàm vào hàm dưới dạng đối số

Ví dụ2 : Viết chương trình thực hiện việc sắp xếp tăng, giảm mảng 1 chiều các số nguyên.

Sử dụng con trỏ hàm

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

// hoán đổi giá trị hai số
void swapNumber(int &a, int &b)
{
        int temp = a;
        a = b;
        b = temp;
}
```

Sử dụng con trỏ hàm - truyền hàm vào hàm dưới dạng đối số

```
bool asc(int a, int b)
{
	return a > b;
}

bool desc(int a, int b)
{
	return a < b;
}
```

```
void selectionSort(int *arr, int n, bool(*comparisonFcn)(int, int))
{
	int i, j, find_idx;
	for (i = 0; i < n - 1; i++)
	{
		//Tìm phần tử lớn nhất/nhỏ nhất chưa được sắp xếp
		find_idx = i;
		for (j = i + 1; j < n; j++)
		{
			if (comparisonFcn(arr[find_idx], arr[j]))
			{
				find_idx = j;
			}
		}

		// Đổi chỗ phần tử tìm được với phần tử đầu tiên
		swapNumber(arr[find_idx], arr[i]);
	}
}
```

Sử dụng con trỏ hàm – truyền hàm vào hàm dưới dạng đối số

```
/* Function to print an array */
void printArray(int arr[], int size)
{
        int i;
        for (i = 0; i < size; i++)
                cout << arr[i] << " ";
        cout << endl;
}
```

```
int main()
{
        int arr[] = { 64, 25, 12, 22, 11 };
        int n = sizeof(arr) / sizeof(int);

        // Sắp xếp tăng
        selectionSort(arr, n, asc);

        cout << "Asc array: \n";
        printArray(arr, n);

        // Sắp xếp giảm
        selectionSort(arr, n, desc);

        cout << "Desc array: \n";
        printArray(arr, n);

        return 0;
}
```

Sử dụng con trỏ hàm - truyền hàm vào hàm dưới dạng đối số

Đối số mặc định của tham số hàm kiểu con trỏ hàm

```
// mặc định hàm được sắp xếp tăng dần nếu không truyền vào đối số thứ 3
void selectionSort(int *arr, int n, bool(*comparisonFcn)(int, int) = asc);
int main()
{
	int arr[] = { 64, 25, 12, 22, 11 };
	int n = sizeof(arr) / sizeof(int);

	// Sắp xếp tăng
	selectionSort(arr, n);

	// Sắp xếp giảm
	selectionSort(arr, n, desc);

	return 0;
}
```

Lưu ý

int MT[Rows][Columns];
khai báo như trên thì MT là mảng chứa mảng, chỉ có thể chuyển thành con trỏ tới mảng, chứ ko chuyển thành con trỏ tới con trỏ được.
ví dụ int MT[10][20]; thì MT có thể hiểu là con trỏ tới mảng 20 số nguyên. Nếu con trỏ MT có giá trị là 100 thì MT+1 là con trỏ có giá trị 100 + 4*20 = 180.

```
void inputElement (int (*arr)[20], int rows) //chỉ truyền MT[rows][20] vào được.
 {
        …….
 }
int mang1[10][20]; inputElement(mang1, 10);
int mang2[15][20]; inputElement(mang2, 15);
int mang3[10][10]; inputElement(mang3, 10); //ERROR
```