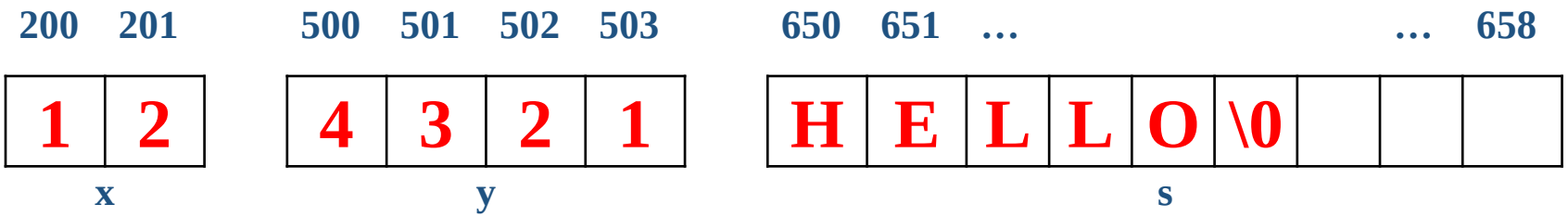
Chương IV
Con trỏ và số học địa chỉ

ĐẠI HỌC GTVT TP HỒ CHÍ MINH

Nội dung chính

4.1 Địa chỉ, phép toán &

- Địa chỉ của một biến là địa chỉ byte nhớ đầu tiên của biến đó.
- C++ cung cấp một toán tử & để lấy địa chỉ của các biến (ngoại trừ biến mảng và chuỗi kí tự). Nếu x là một biến thì &x là địa chỉ của x.
- Đối với biến kiểu mảng, thì tên mảng chính là địa chỉ của mảng, do đó không cần dùng đến toán tử &.



Biến x chiếm 2 byte nhớ, có địa chỉ là 200, biến y có địa chỉ là 500 và chiếm 4 byte nhớ. chuỗi s chiếm 6 byte nhớ tại địa chỉ 650. Các byte nhớ của một biến là liền nhau.

4.1 Địa chỉ, phép toán &

❖ Lưu ý:

```
int x;                  // khai báo biến nguyên x
cout << &x;             // in địa chỉ các biến x
int a[9];               // khai báo mảng   a
cout << a;              // in địa chỉ mảng a
cout << &a[0];          // in địa chỉ mảng a (tức địa chỉ a[0])
cout << &a[2];          // in địa chỉ phần tử a[2]
char s[9]="abcde";      // khai báo và khởi tạo chuỗi s
cout <<s;               // in ra chuỗi s  "abcdef"
cout << &s[0];          // giống cout<<s
cout << &s[2];          // in ra chuỗi "cde"
```

4.2 Con trỏ

- Con trỏ: là một kiểu dữ liệu dùng để chứa địa chỉ. Ví dụ: int*, float*, char*,…
- Biến con trỏ: là loại biến được dùng để chứa địa chỉ của biến khác khi nó trỏ đến.
  - Nếu p là con trỏ chứa địa chỉ của biến x ta gọi p trỏ tới x và x được trỏ bởi p.
  - Thông qua con trỏ ta có thể làm việc được với nội dung của những ô nhớ mà p trỏ đến.
- Để con trỏ p trỏ tới x ta phải gán địa chỉ của x cho p.
- Để làm việc với địa chỉ của các biến cần phải thông qua các biến con trỏ trỏ đến biến đó.

4.2 Con trỏ

- **Khai báo biến con trỏ**

  <Kiểu được trỏ> *<Tên biến>;

Ví dụ:

```
int *p ; // khai báo biến con trỏ p trỏ đến kiểu số nguyên.
float *p1, *p2 ; // khai báo hai con trỏ p1 và p2 trỏ đến kiểu số thực,
```

- Con trỏ NULL
  - NULL là con trỏ không trỏ vào đâu cả.
  - NULL khác với con trỏ chưa khởi tạo.

Ví dụ:

```
int n;
int *p1 = &n;
int *p2;       // unreferenced local varialbe
int *p3 = NULL;
```

4.2 Con trỏ

- **Sử dụng phép toán & và ***
  - Dùng & gán địa chỉ của biến cho con trỏ.

    Biến con trỏ = &biến;
  - Phép toán * cho lấy nội dung nơi con trỏ trỏ đến.

Ví dụ:

```
int *p, *p1, x=0, a[]={1,2,3};
p=&x //do x là biến thông thường
p1=a; //do a là biến mảng
p1=&a[0] //tương tự dòng lệnh trên
cout << *p; //in ra 0 (nội dung biến x)
```

  - & và * là 2 phép toán ngược nhau (nếu p=&x thì x=*p).

4.2 Con trỏ

Ví dụ:

```
int i, j ;                  // khai báo 2 biến nguyên i, j
int *pi, *pj ;              // khai báo 2 con trỏ nguyên pi, pj
pi = &i;                    / cho pi trỏ tới i
pj = &j;                    // cho pj trỏ tới j
cout << &i ;                // hỏi địa chỉ biến i
cout << pj ;                // hỏi địa chỉ biến j (thông qua pj)
i = 2;                      // gán i bằng 2
*pj = 5;                    // gán j bằng 5 (thông qua pj)
i++ ; cout << i ;           // tăng i và xuất i (i = 3)
(*pj)++ ; cout << j ;       // tăng j (thông qua pj) và xuất j (j = 6)
(*pi) = (*pj) * 2 + 1;      // gán lại i (thông qua pi)
cout << i ;                 // 13
```

4.3 Các phép toán với con trỏ

- **Phép toán gán** :
  - Gán con trỏ với địa chỉ một biến: p = &x ;
  - Gán con trỏ với con trỏ khác: p = q ; (sau phép toán gán này p, q chứa cùng một địa chỉ, cùng trỏ đến một nơi).

Ví dụ:

```
int i = 10 ;        // khai báo và khởi tạo biến i = 10
int *p, *q, *r ; // khai báo 3 con trỏ nguyên p, q, r
p = q = r = &i ;// cùng trỏ tới i
*p = (*q)*(*q) + 2*(*r) + 1;// i = 10*10 + 2*10 + 1
cout << i ;// 121
```

4.3 Các phép toán với con trỏ

- **Phép toán tăng giảm địa chỉ**:
  - **p ± n**: Con trỏ trỏ đến thành phần thứ n sau (trước) p. (Một đơn vị tăng giảm của con trỏ bằng kích thước của biến được trỏ).
  - **p++, p--, ++p, --p**: tương tự p+1 và p-1, có chú ý đến tăng (giảm) trước, sau.

Ví dụ: Giả sử p là con trỏ nguyên (2 byte) đang trỏ đến địa chỉ 200 thì p+1 là con trỏ trỏ đến địa chỉ 202.

Ví dụ:

```
int a[] = { 1, 2, 3, 4, 5, 6, 7 }, *p, *q;
p = a; cout << *p ; //trỏ p đến mảng a, *p = a[0]= 1
p += 5; cout << *p ; // *p = a[5] = 6 ;
q = p - 4 ; cout << *q ; // q = a[1] = 2 ;
```

4.3 Các phép toán với con trỏ

- **Hiệu của hai con trỏ**:
  - Phép toán này chỉ thực hiện được khi p và q là 2 con trỏ cùng trỏ đến các phần tử của một dãy dữ liệu nào đó trong bộ nhớ (ví dụ cùng trỏ đến 1 mảng dữ liệu).
  - Khi đó hiệu p - q là số thành phần giữa p và q (chú ý p - q không phải là hiệu của 2 địa chỉ mà là số thành phần giữa p và q).

Ví dụ: Giả sử p và q là 2 con trỏ nguyên, p có địa chỉ 200 và q có địa chỉ 208. Khi đó p - q = - 4 và q - p = 4 (4 là số thành phần nguyên từ địa chỉ 200 đến 208).

4.3 Các phép toán với con trỏ

❖ **Phép toán so sánh**:

Thông thường các phép so sánh chỉ áp dụng cho hai con trỏ trỏ đến phần tử của cùng một mảng dữ liệu nào đó.

Ví dụ :

```
float a[100], *p, *q ;
p = a ;      // p trỏ đến mảng a(tức p trỏ đến a[0])
q = &a[3] ;  // q trỏ đến phần tử thứ 3 (a[3]) của mảng
cout << (p < q) ;   // 1
cout << (p + 3 == q) ;   // 1
cout << (p > q - 1) ;   // 0
cout << (p >= q - 2) ;   // 0
for (p=a ; p < a+100; p++)
    cout << *p ;   // in toàn bộ mảng a
```

4.3 Các phép toán với con trỏ

- **Con trỏ kiểu void:** là một loại con trỏ có thể trỏ đến các biến có kiểu dữ liệu bất kỳ.
  - Khai báo: void *<tên biến con trỏ>;
  - Ví dụ:
    - int a; float b; char c;
    - void *p;
    - p=&a; p=&b; p=&c;
- **Chuyển kiểu con trỏ:**
  - Cú pháp: **(<Kiểu>*) <Tên biến con trỏ>;**
  - Ví dụ:
    - int *p;
    - (float*)p;
- **Con trỏ kép**: (hay con trỏ cấp 2) là một loại con trỏ được dùng để chứa địa chỉ của một con trỏ khác.
  - Khai báo: <Kiểu> **<tên biến con trỏ>;

4.3 Các phép toán với con trỏ

- **Hằng con trỏ (Constant pointer)**
  - Khai báo: <Kiểu> * const <tên con trỏ>=<địa chỉ khởi tạo>;

Ví dụ:

```
int* const p;//lỗi
int a;
int* const p=&a;
```

  - Khi khai báo cần khởi tạo giá trị địa chỉ cho nó.
  - Khi đó hằng con trỏ không thể trỏ đến bất kỳ địa chỉ nào khác
  - Có thể thay đổi được giá trị tại địa chỉ khởi tạo ban đầu.

```
int a=5, b;
int *const p = &a;
p = &b;//lỗi
*p = 15;
```

4.3 Các phép toán với con trỏ

- **Con trỏ hằng (Pointer to constant)**
  - Khai báo: const <Tên kiểu> *<tên con trỏ>;

    Ví dụ:

    const int *p;
  - Có thể trỏ đến bất kỳ địa chỉ nào
  - Nhưng không thay đổi được giá trị tại địa chỉ mà nó đang trỏ đến.

```
int a, b=5;
int const *p ;
p=&a;
p = &b;
b=100;
*p = 15;//lỗi
```

4.4 Cấp phát và thu hồi bộ nhớ động

- Cấp phát bộ nhớ động với toán tử new:
  - **p = new Kiểu ;** // cấp phát 1 phần tử
  - **p = new Kiểu[n] ;** // cấp phát n phần tử

- Thu hồi bộ nhớ động với toán tử delete:
  - **delete p ;** // p là con trỏ được sử dụng trong new
  - **delete[] p ;** // p là con trỏ trỏ đến mảng

Khi gặp toán tử new, chương trình sẽ tìm trong bộ nhớ một lượng ô nhớ còn rỗi và liên tục với số lượng đủ theo yêu cầu và cho p trỏ đến địa chỉ (byte đầu tiên) của vùng nhớ này. Nếu không có vùng nhớ với số lượng như vậy thì việc cấp phát là thất bại và p = NULL

4.4 Cấp phát và thu hồi bộ nhớ động

Chú ý:

- Một con trỏ sau khi giải phóng bộ nhớ, có thể cấp phát mới

```
int *pa = new int(12) //*pa = 12
delete pa; // Giải phóng vùng nhớ vừa cấp cho pa.
int A[5] = {5, 10, 15, 20, 25};
pa = A; // Cho pa trỏ đến địa chỉ của mảng A
```

- Nhiều con trỏ cùng trỏ một địa chỉ, chỉ cần giải phóng bộ nhớ 1 con trỏ:

```
int *pa = new int(12); // *pa = 12
int *pb = pa; // pb trỏ đến cùng địa chỉ pa.
*pb += 5; // *pa = *pb = 17
delete pa; // Giải phóng cả pa lẫn pb
```

4.4 Cấp phát và thu hồi bộ nhớ động

- Một con trỏ sau khi cấp phát bộ nhớ động bằng new, cần phải phóng bộ nhớ trước khi trỏ đến một địa chỉ mới hoặc cấp phát bộ nhớ mới:

```
int *pa = new int(12);  // pa được cấp bộ nhớ
                        // và *pa = 12
*pa = new int(15);  // pa trỏ đến địa chỉ khác
                    // và *pa = 15.
// địa chỉ cũ của pa vẫn bị coi là bận
```

- Phân biệt "()" với "[]".

```
// Cấp phát bộ nhớ và khởi tạo cho một con trỏ int
int *A = new int(5);
// Cấp phát bộ nhớ cho một mảng 5 phần tử kiểu int
int *A = new int[5];
```

Truyền đối số

```
#include<iostream>
using namespace std;
void hoanvi(int *x, int *y);// Khai báo đối số bằng con trỏ

void main(){
    int a = 2912; b = 1706;
    hoanvi(&a, &b); //Truyền đối số bằng địa chỉ
    cout << “a = “ << a << “ b = “ << b;
}
void hoanvi(int *x, int *y){
    int t = *x; *x = *y; *y = t;
}
```

Ví dụ : Sắp xếp dãy số

```
#include<iostream>
using namespace std;
int main() {
        int *dau, *p, *q, n, tam;          //  dau sẽ là số đầu tiên của dãy
        cout << "Cho biet so luong phan tu cua day: ";
        cin >> n ;
        dau = new int[n] ;    // Cấp cho dau n phần tử số nguyên
        for (p = dau; p<dau+n; p++)   {  // nhập dãy
            cout << "So thu " << p-dau+1 << ": " ; cin >> *p ;
        }
        for (p=dau; p<dau+n-1; p++) // Sắp xếp dãy
            for (q=p+1; q<dau+n; q++)
                if (*q < *p)  {
                    tam = *p; *p = *q; *q = tam;
                }
        for (p=dau; p<dau+n; p++) cout << *p <<"\t";      // Xuất dãy
        delete dau;  // Thu hồi bộ nhớ
}
```

4.5 Con trỏ và mảng, chuỗi

- **Con trỏ và mảng 1 chiều :**
  - **Con trỏ trỏ đến phần tử mảng:**

Việc cho con trỏ trỏ đến mảng cũng tương tự trỏ đến các biến khác, tức gán địa chỉ của mảng (chính là tên mảng) cho con trỏ.

Nếu ta có câu lệnh **p=a**; thì:

p trỏ đến a[0]

p+1 trỏ đến a[1]

.....

p+i trỏ đến a[i]

4.5 Con trỏ và mảng, chuỗi

Ví dụ: In toàn bộ mảng thông qua con trỏ.

Cho : int a[5] = {1, 2, 3, 4, 5}, *p, i;

Cách 1:

```
p = a;
for (i=0; i<5; i++) cout << *(p+i) <<"\t";
// p không thay đổi
```

Cách 2:

```
for (p=a; p<a+5; p++) cout << *p <<"\t";
// thay đổi p
```

4.5 Con trỏ và mảng, chuỗi

- **Tên mảng dùng như con trỏ:**

C/C++ xem tên mảng 1 chiều chính là một hằng con trỏ (a[0]=2,a[5]=7//ok; nhưng a++, a--//lỗi), địa chỉ của mảng chính là địa chỉ của phần tử đầu tiên.

Giả sử a là tên mảng

a tương đương với &a[0]

a+i tương đương với &a[i]

***(a+i) tương đương với a[i]**

Ví dụ

```
#include <iostream>
using namespace std;
int main() {
        int a[10], n = 10, *pa;
        pa = a;  // hoặc pa = &a[0];
        for (int i = 0; i<n; i++)
                cin >> a[i];
                // cin >> pa[i];
                // cin >> *(a + i);
                // cin >> *(pa + i);
                // cin >> *(a++);  //Lỗi vì a là hằng con trỏ
                // cin >> *(pa++);
}
```

⇨&a[i] ⇔ (a + i) ⇔ (pa + i) ⇔ &pa[i]

Ví dụ

```
void xuat(int a[], int n) {// xuat(int *a, int n)
        for (int i = 0; i<n; i++)
            cout << *(a++);  // Đúng a không phải hằng con trỏ
} //in ra các số 1,2,3,4,5 (toàn bộ mảng)

int main() {
        int a[5]={1,2,3,4,5}, n = 5;
        xuat(a, n);
        for (int i = 0; i<n; i++)
            cout << *(a++);  // Lỗi, a hằng con trỏ
}
```

⇨ Đối số mảng truyền cho hàm không phải hằng con trỏ.

4.5 Con trỏ và mảng 2 chiều

Tên mảng dùng như con trỏ

- Một mảng 2 chiều có thể hiểu như là mảng 1 chiều của mảng, tức là mỗi phần tử của mảng lại là mảng 1 chiều.
- Ví dụ: int a[2][3];
- Thì mảng 2 chiều a được xem gồm 2 phần tử, mỗi phần tử là một mảng gồm 3 số nguyên int.
- a có phần tử thứ 0 là a[0], a[0] là mảng 1 chiều gồm các phần tử: a[0][0], a[0][1], a[0][2].
- a có phần tử thứ 1 là a[1], a[1] là mảng 1 chiều gồm các phần tử: a[1][0], a[1][1], a[1][2].

4.5 Con trỏ và mảng 2 chiều

- Ta có quy tắc sau:

  a[i] tương đương với **&a[i][0]**

  a[i]+j tương đương với **&a[i][j]**

  ***(a[i]+j)** tương đương với **a[i][j]**

- Hoặc

  ***(a+i)+j** tương đương với **&a[i][j]**

  ***(*(a+i)+j)** tương đương với **a[i][j]**

- Ví dụ:

```
for(int i=1;i<3;i++)
      {
        for(int j=1;j<4;j++)
              cout<<*(*(a+i)+j) <<" ";
        cout<<endl;
      }
```

Con trỏ chỉ đến phần tử mảng

- Xét khai báo

```
int a[2][3];
int (*p)[3];// p là con trỏ kiểu int[3]
```

Nếu p=a;

***(p+i)+j** tương đương với **&a[i][j]**

***(*(p+i)+j)** tương đương với **a[i][j]**

4.5 Con trỏ và mảng 2 chiều

Ví dụ sau đây cho phép nhập và in một mảng 2 chiều m*n (m dòng, n cột) thông qua con trỏ p. Nhập liên tiếp m*n số vào mảng và in thành ma trận m dòng, n cột.

VD1

```
float a[2][3], *p;
int i, j;
p = (float*) a; //p=&a[0][0]
for (i=0; i<2*3; i++)
      cin >> *(p+i);
for (i=0; i<2; i++){
     for (j=0; j<3; j++)
          cout <<a[i][j]<< '\t';
     cout << endl;
}
```

VD2

```
float a[2][5], *p;
int n=2, m=3;
p = (float*) a; //p=&a[0][0]
for (i=0; i<n*m; i++)
      cin >> *(p+i);
for (i=0; i<2; i++){
    for (j=0; j<3; j++)
         cout <<a[i][j]<< '\t';
    cout << endl;
}   //Kết quả khác với ví dụ bên
```

Con trỏ và mảng 2 chiều

Ví dụ:

```
float a[2][5], *p = (float *) a;
int n=2, m=3, i, j;

for (i = 0 ; i < n; i++)
      for (j = 0 ; j < m; j++) {
          cout <<"a["<< i<<"]["  << j<< "]=";
          cin>>*(p + i * 5 + j);//do có 5 cột
      }
for (i = 0 ; i < n; i++) {
      cout<< endl; // xuong dong
      for (j = 0 ; j < m; j++)
           cout<<a[i][j]<<‘t’;  //cout << *(p + i *5 + j) << '\t';
}
```

Con trỏ và chuỗi

- Con trỏ và chuỗi:
  - Một con trỏ kí tự có thể xem như một biến chuỗi kí tự, trong đó chuỗi chính là tất cả các kí tự kể từ byte con trỏ trỏ đến cho đến byte '\0' gặp đầu tiên.
  - Các hàm trên chuỗi vẫn được sử dụng như khi ta khai báo nó dưới dạng mảng kí tự.
  - Ngoài ra khác với mảng kí tự, ta được phép sử dụng phép gán cho 2 chuỗi dưới dạng con trỏ.
  - Khi khai báo chuỗi dạng con trỏ nó vẫn chưa có bộ nhớ cụ thể, vì vậy thông thường kèm theo khai báo ta cần phải xin cấp phát bộ nhớ cho chuỗi với độ dài cần thiết.

Ví dụ

```
#include<iostream>
#include<cstring>
using namespace std;
int main()
{
	char *s="Ky Thuat Lap Trinh", *t;
	t = new char[33];  //Cấp phát bộ nhớ động cho t
	//strncpy(t, s, 8);  //Copy nội dung 8 kí tự đầu của s sang t
	for (int i=0;i<8;i++)
		*(t+i)=*(s+i);
	*(t+8)='\0';
	cout<<t;
}
```

4.6 Mảng con trỏ

- Mảng con trỏ:

  Là mảng trong đó các phần tử của nó là một con trỏ trỏ đến một mảng nào đó. Nói cách khác một mảng con trỏ cho phép quản lý nhiều mảng dữ liệu cùng kiểu.

- Cách khai báo:

  Kiểu *a[size];

Ví dụ:

```
int *a[10];
```

/ khai báo một mảng chứa 10 con trỏ. Mỗi con trỏ a[i] chứa địa chỉ của một mảng nguyên nào đó. */*

Ví dụ

```
#include<iostream>
using namespace std;
int main(){
     int a[4]={1,2,3,4}, b[4]={4,5,6}, c[4]={9,8,7,3}, i, j;
     int *p[3]={a, b, c}; // mảng p là các con trỏ trỏ đến a, b, c
    for(i=0; i<3; i++) {
          for(j=0; j<4; p[i]++, j++)
               cout<<*p[i]<<" ";
              cout<<endl;
          }
}
```


Thank You !