Nhập Môn Lập Trình
Con Trỏ

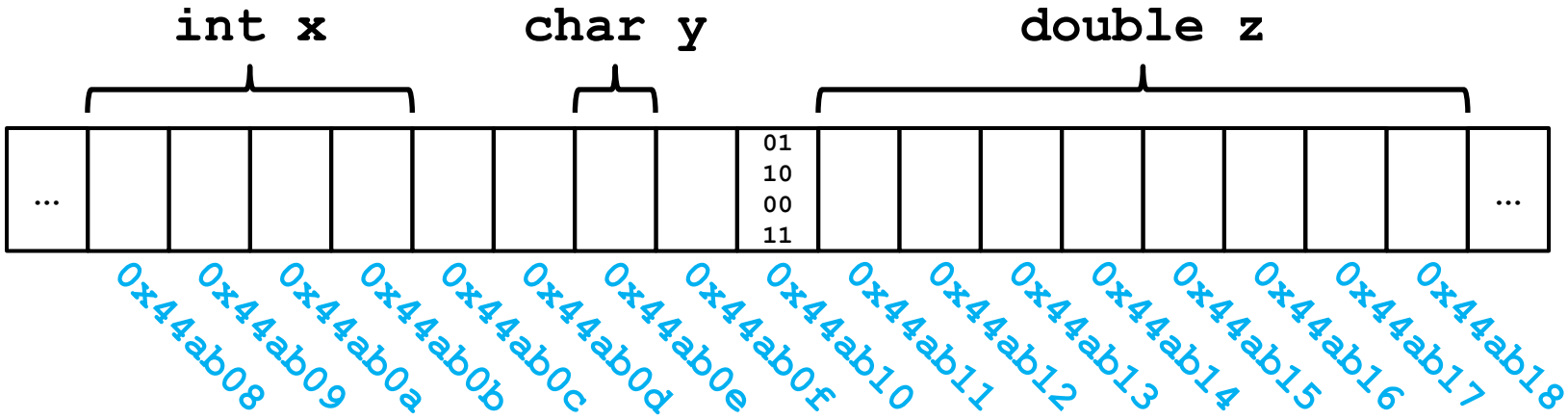
TS. Lê Nguyên Khôi
Trường Đại học Công nghệ, ĐHQGHN

Nội Dung

- Con trỏ
  - Biến kiểu con trỏ
  - Quản lý vùng nhớ
  - Toán tử con trỏ
- Mảng động
  - Khai báo & sử dụng
  - Các phép toán với con trỏ

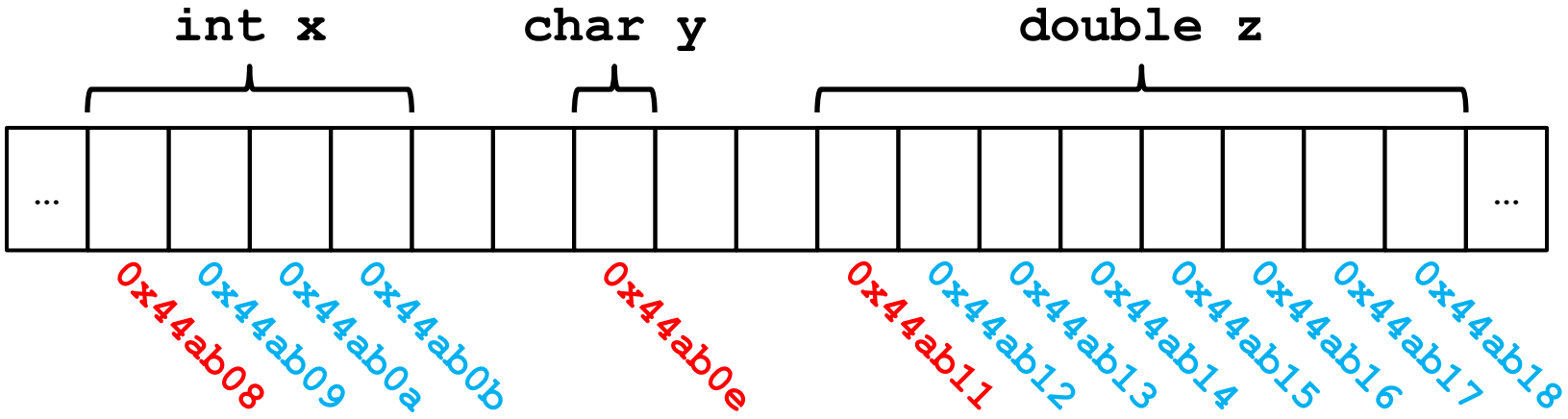
Định Nghĩa

- Con trỏ là địa chỉ vùng nhớ của một biến
- Vùng nhớ máy tính:
  - tổ chức theo byte (8 bits), mỗi byte có một địa chỉ
  - địa chỉ được đánh số liên tục (hệ 16 - **0x**.....)



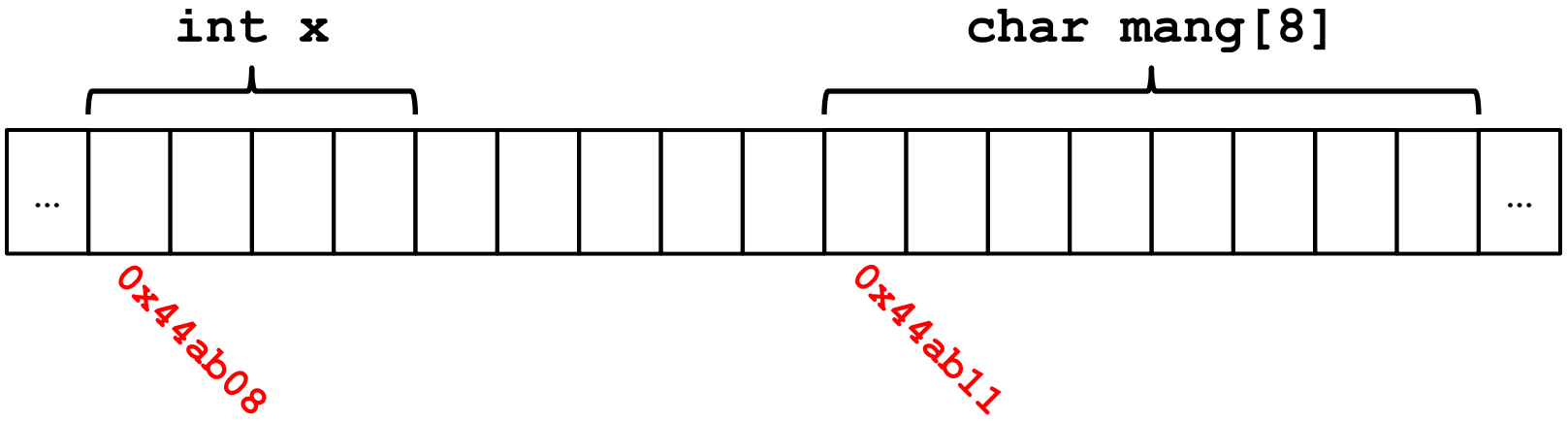
Ví Dụ

- Địa chỉ của biến là địa chỉ của byte đầu tiên
- Địa chỉ được sử dụng như tên biến, ví dụ
  - địa chỉ: 144 – tên: Đại Học Quốc Gia Hà Nội
  - địa chỉ: 136 – tên: Đại Học Sư Phạm Hà Nội
  - ĐHSPHN bao gồm 136, 138, 140, 142



Sử Dụng Địa Chỉ (Con Trỏ)

```
void thayDoiTruyenThamChieu(int & x) {
    x = x * 100;
    cout << "ham: x=" << x << endl;
}
void nhapMang(char mang[]) {
    for (int i = 0; i < SO_MON_HOC; i++)
        cin >> mang[i];
}
```



Kiểu Con Trỏ

- Con trỏ là “kiểu dữ liệu”
  - kiểu địa chỉ vùng nhớ
- Có thể dùng biến để lưu giá trị kiểu con trỏ
  - dữ liệu kiểu nguyên lưu trong biến kiểu **`int`**
  - dữ liệu kiểu thực lưu trong biến kiểu **`double`**
- Địa chỉ là số nguyên (hệ 16 - **`0x`**.....)
  - ví dụ: số nhà 144 thì 144 là địa chỉ
- Con trỏ không phải kiểu số nguyên
  - không dùng biến kiểu nguyên để lưu dữ liệu địa chỉ
  - dữ liệu kiểu địa chỉ phải lưu trong biến kiểu con trỏ

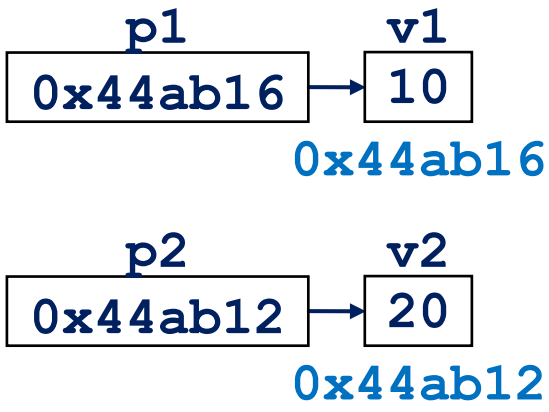
Khai Báo Con Trỏ

- Khai báo con trỏ giống biến, kiểu dữ liệu khác
  - thêm **`*`** vào trước tên biến
  - ký hiệu **`*`** phải đặt trước mỗi biến con trỏ
- Ví dụ:
  - **`int v1, v2;`**
    - **`v1`**, **`v2`** kiểu **`int`** lưu dữ liệu kiểu **`int`**
  - **`int *p1, *p2;`**
    - **`p1`**, **`p2`** kiểu con trỏ lưu địa chỉ của biến kiểu **`int`**
    - sử dụng **`p1`**, **`p2`** lưu địa chỉ của **`v1`**, **`v2`**
    - lấy địa chỉ của **`v1`**, **`v2`** lưu vào **`p1`**, **`p2`**

Toán Tử &

- Toán tử & lấy địa chỉ của biến

```
int main {
    int v1 = 10, v2 = 20;
    int *p1, *p2;
    p1 = &v1;
    cout << &v1 << v1 <<  p1;
    p2 = &v2;
    cout << &v2 << v2 <<  p2;
    return 0;
}
```

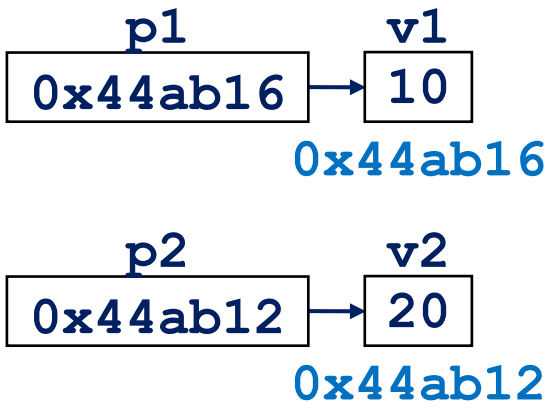



```
0x44ab16 10 0x44ab16
0x44ab12 20 0x44ab12
```

Toán Tử *

- Toán tử * lấy biến mà con trỏ đang lưu địa chỉ

```
int main {
    int v1 = 10, v2 = 20;
    int *p1, *p2;
    p1 = &v1;
    cout << &v1 << v1 << *p1;
    p2 = &v2;
    cout << &v2 << v2 << *p2;
    return 0;
}
```

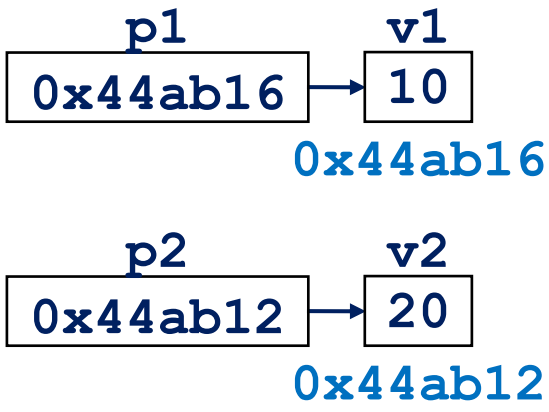



```
0x44ab16 10 10
0x44ab12 20 20
```

Toán Tử *& – &*

- **&*x** không hợp lệ, do **x** không phải con trỏ

```
int main {
    int v1 = 10, v2 = 20;
    int *p1, *p2;
    p1 = &v1;
    cout << *&v1 << &*p1;
    p2 = &v2;
    cout << *&v2 << *&p2;
    return 0;
}
```



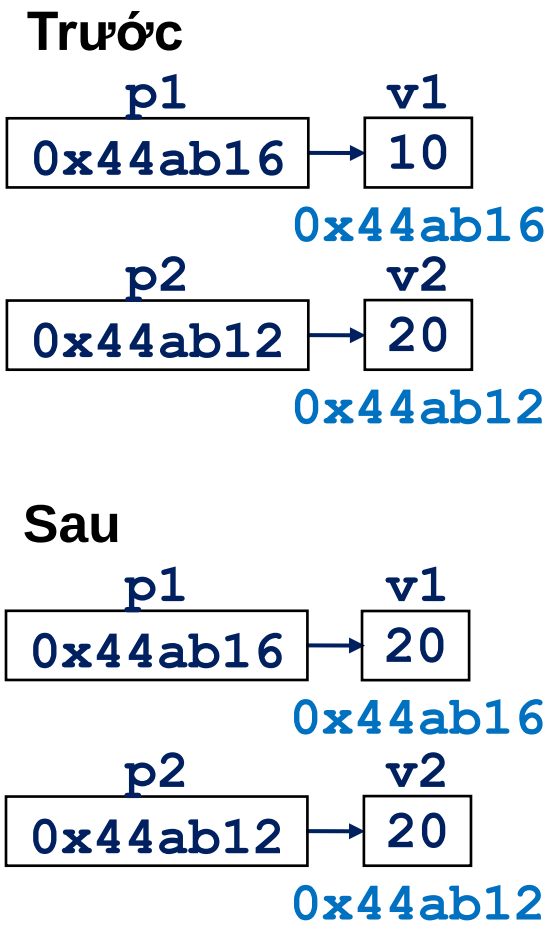

```
10 0x44ab16
20 0x44ab12
```

Toán Tử =

- Gán giá trị

```
int main {
     int v1 = 10, v2 = 20;
     int *p1 = &v1;
     int *p2 = &v2;
     cout <<*p1<<p1<<*p2<<p2;
     *p1 = *p2; // v1 = v2
     cout <<*p1<<p1<<*p2<<p2;
     return 0;
}
```

```
10 0x44ab16 20 0x44ab12
20 0x44ab16 20 0x44ab12
```

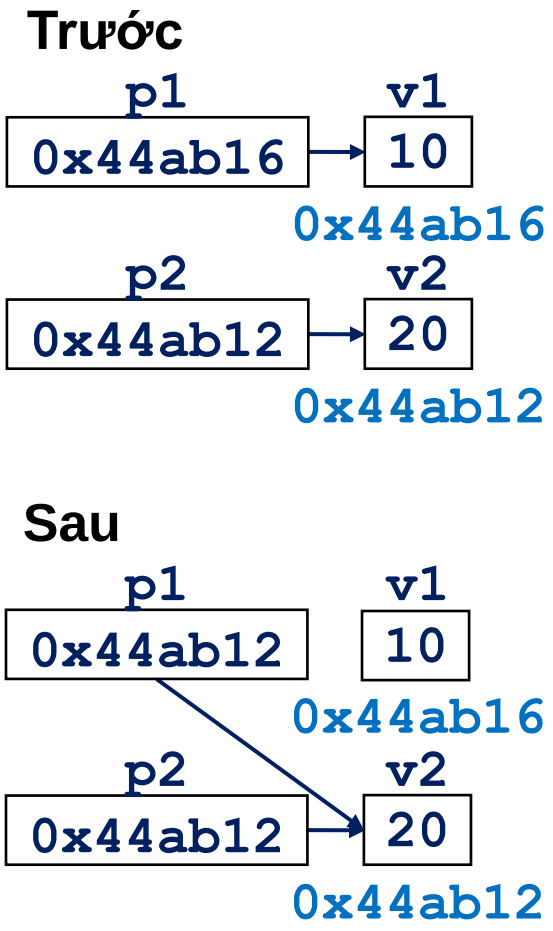


Toán Tử =

- Gán con trỏ (địa chỉ)

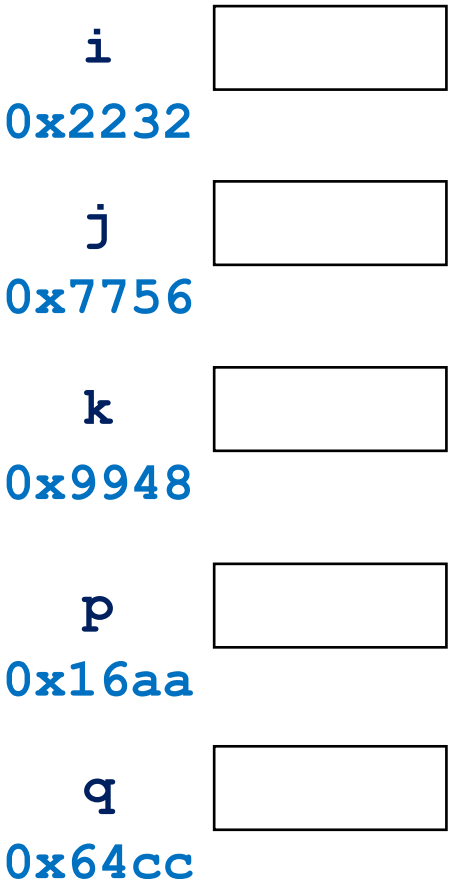
```
int main {
     int v1 = 10, v2 = 20;
     int *p1 = &v1;
     int *p2 = &v2;
     cout <<*p1<<p1<<*p2<<p2;
      p1 =  p2; // = &v2
     cout <<*p1<<p1<<*p2<<p2;
     return 0;
}
```

```
10 0x44ab16 20 0x44ab12
20 0x44ab12 20 0x44ab12
```



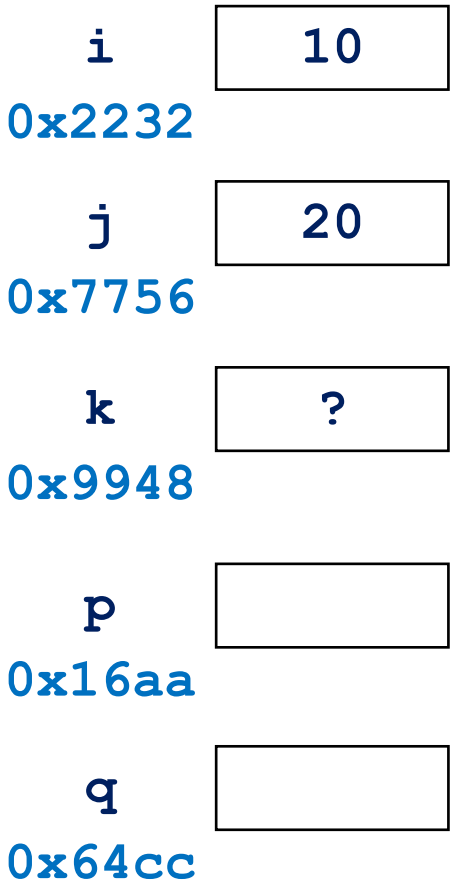
Bài Tập

```
int main {
     int i = 10, j = 20, k;
     int *p = &i;
     int *q = &j;
     *p = *p + 1;
      p = &k;
     *p = *q;
      p = q;
     *p = i;
}
```



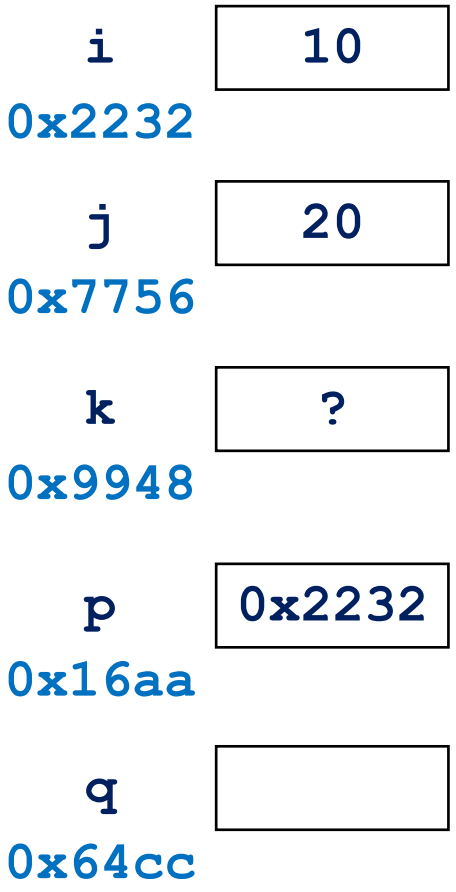
Bài Tập

```
int main {
     int i = 10, j = 20, k;
     int *p = &i;
     int *q = &j;
     *p = *p + 1;
      p = &k;
     *p = *q;
      p = q;
     *p = i;
}
```



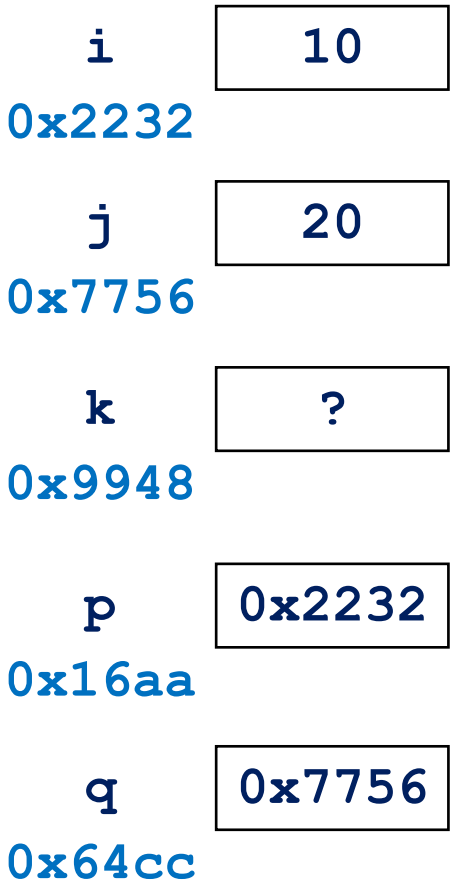
Bài Tập

```
int main {
     int i = 10, j = 20, k;
     int *p = &i;
     int *q = &j;
     *p = *p + 1;
      p = &k;
     *p = *q;
      p = q;
     *p = i;
}
```



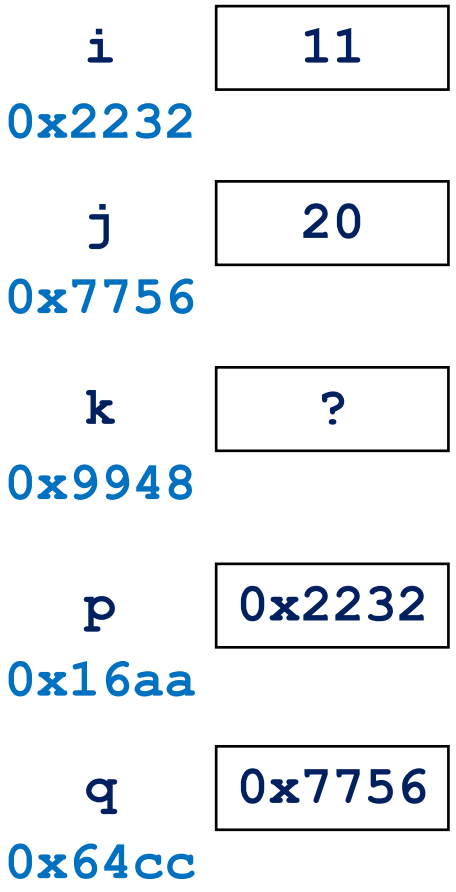
Bài Tập

```
int main {
     int i = 10, j = 20, k;
     int *p = &i;
     int *q = &j;
     *p = *p + 1;
      p = &k;
     *p = *q;
      p = q;
     *p = i;
}
```



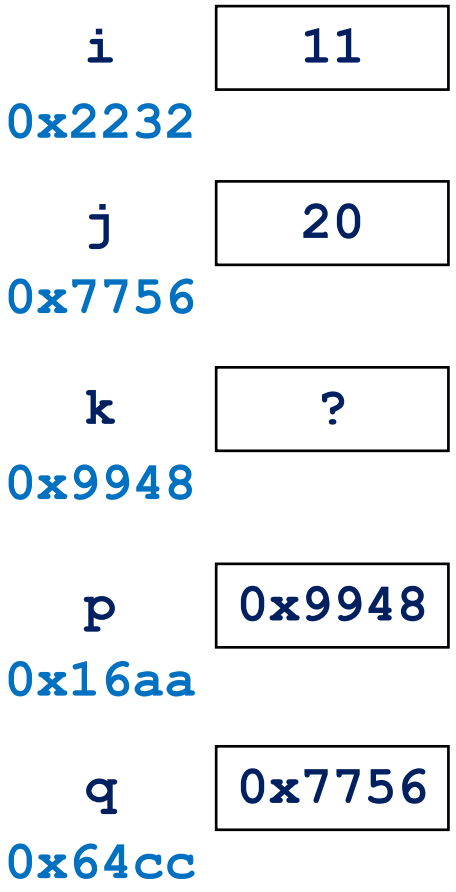
Bài Tập

```
int main {
     int i = 10, j = 20, k;
     int *p = &i;
     int *q = &j;
     *p = *p + 1;
      p = &k;
     *p = *q;
      p = q;
     *p = i;
}
```



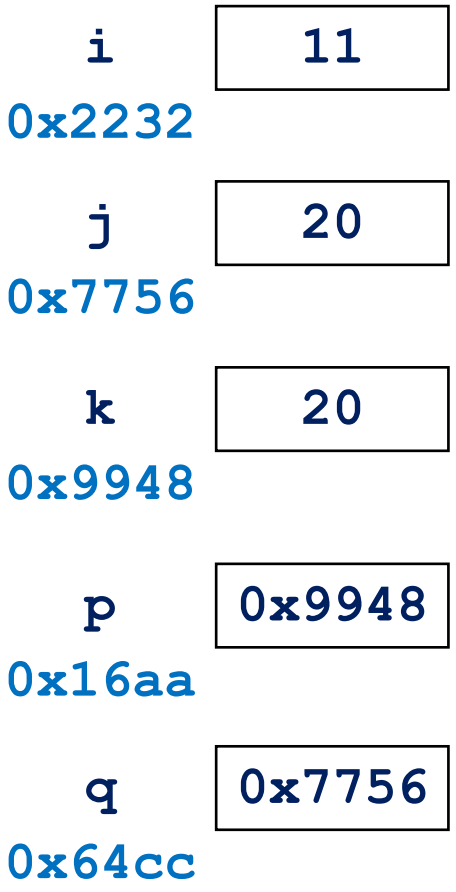
Bài Tập

```
int main {
     int i = 10, j = 20, k;
     int *p = &i;
     int *q = &j;
     *p = *p + 1;
      p = &k;
     *p = *q;
      p = q;
     *p = i;
}
```



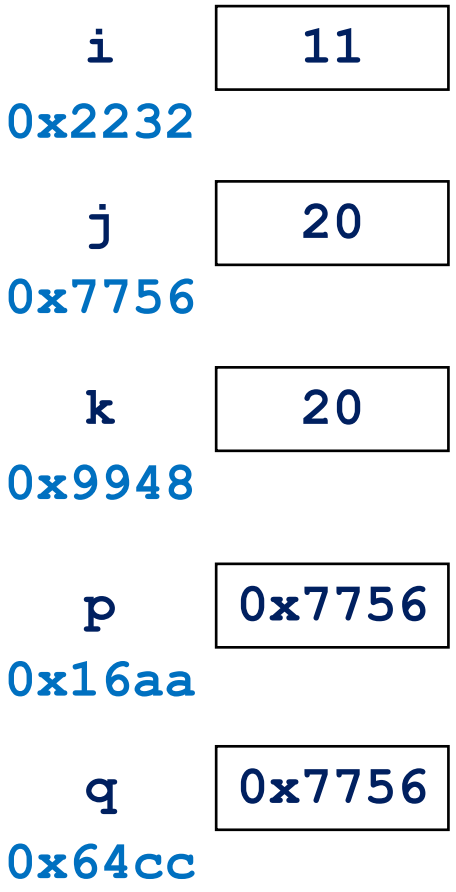
Bài Tập

```
int main {
     int i = 10, j = 20, k;
     int *p = &i;
     int *q = &j;
     *p = *p + 1;
      p = &k;
     *p = *q;
      p = q;
     *p = i;
}
```



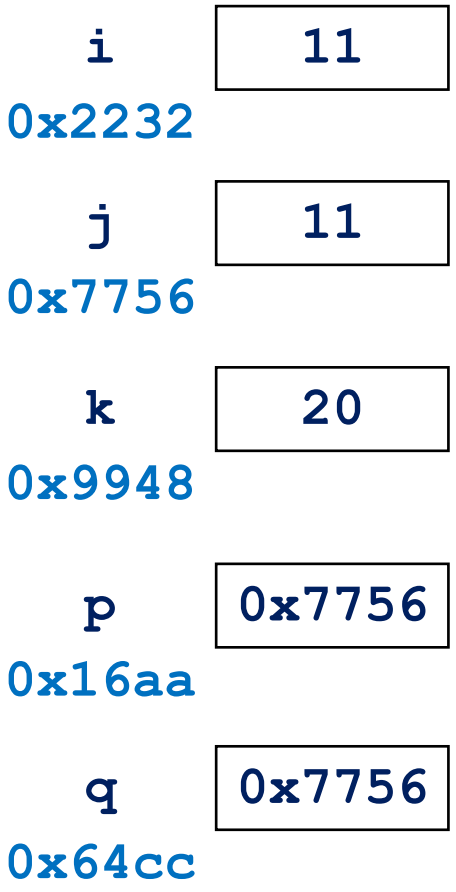
Bài Tập

```
int main {
     int i = 10, j = 20, k;
     int *p = &i;
     int *q = &j;
     *p = *p + 1;
      p = &k;
     *p = *q;
      p = q;
     *p = i;
}
```



Bài Tập

```
int main {
     int i = 10, j = 20, k;
     int *p = &i;
     int *q = &j;
     *p = *p + 1;
      p = &k;
     *p = *q;
      p = q;
     *p = i;
}
```



Toán Tử **new**

- Sử dụng toán tử **new** tạo một vùng nhớ mới, vùng nhớ này không có tên

```
int *p = new int;
```

  - tạo một vùng nhớ mới (không có tên) dùng để lưu dữ liệu **int**, gán địa chỉ vùng nhớ cho con trỏ **p**, giá trị tại vùng nhớ này không xác định

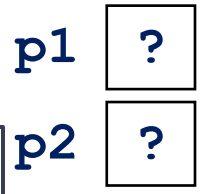
```
int *p = new int(10);
```

  - tạo một vùng nhớ mới (không có tên) dùng để lưu dữ liệu **int**, gán địa chỉ vùng nhớ cho con trỏ **p**, giá trị tại vùng nhớ này được khởi tạo là **10**

Bài Tập

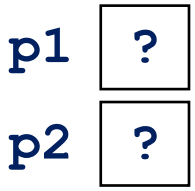
```
int main
{
    int *p1, *p2;
    p1 = new int;
    *p1 = 10;
    p2 = p1;
    *p2 = 20;
    p1 = new int;
    *p1 = 30;
}
```

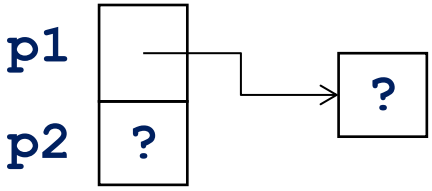
Bài Tập




```
int main
{
    int *p1, *p2;
    p1 = new int;
    *p1 = 10;
    p2 = p1;
    *p2 = 20;
    p1 = new int;
    *p1 = 30;
}
```

Bài Tập

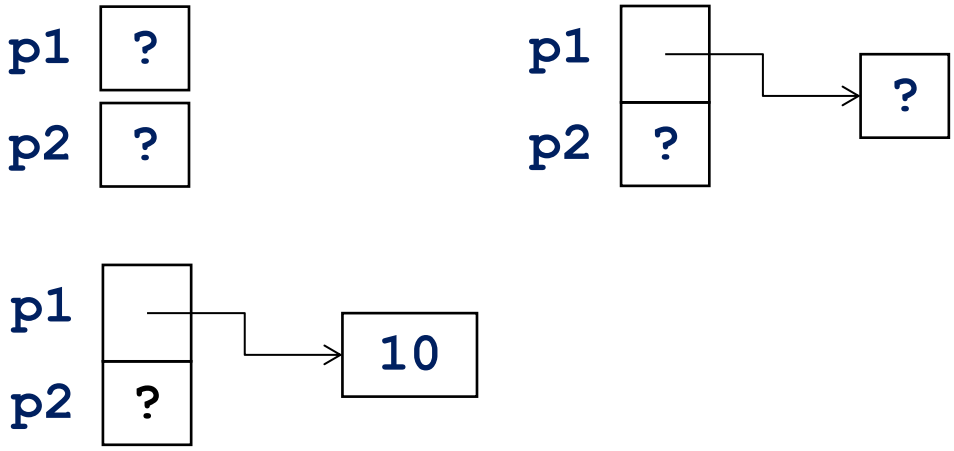





```
int main
{
    int *p1, *p2;
    p1 = new int;
    *p1 = 10;
    p2 = p1;
    *p2 = 20;
    p1 = new int;
    *p1 = 30;
}
```

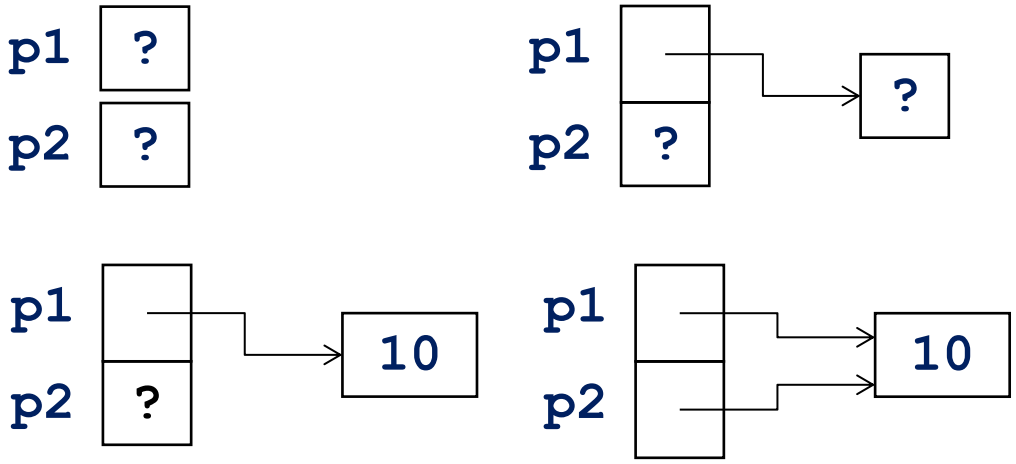
Bài Tập

```
int main
{
    int *p1, *p2;
    p1 = new int;
    *p1 = 10;
    p2 = p1;
    *p2 = 20;
    p1 = new int;
    *p1 = 30;
}
```



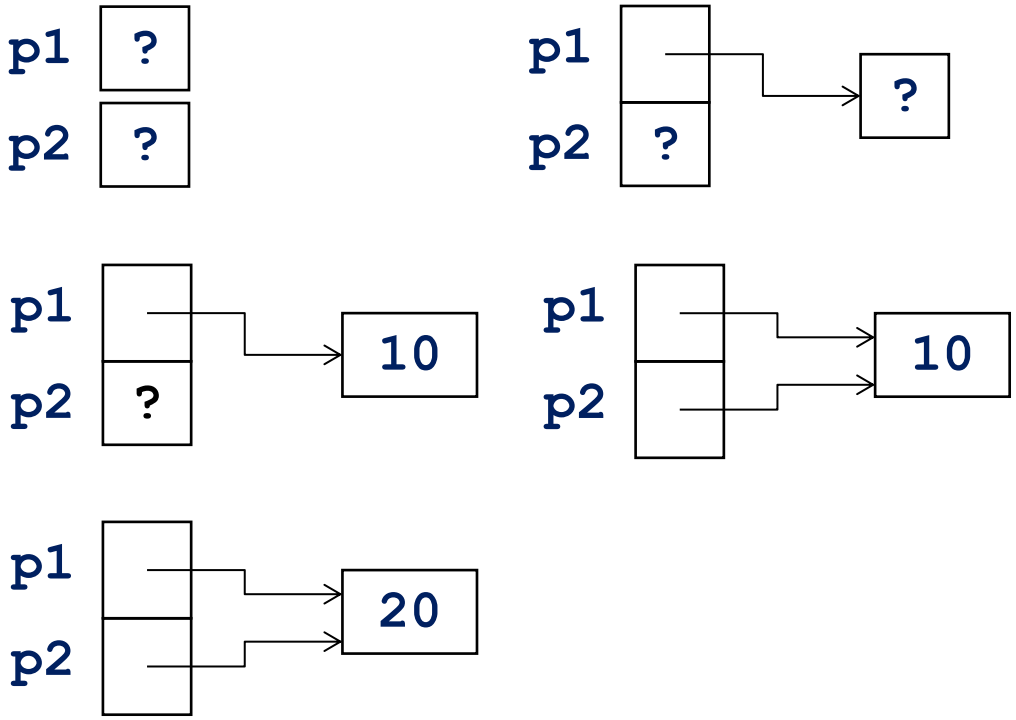
Bài Tập

```
int main
{
    int *p1, *p2;
    p1 = new int;
    *p1 = 10;
    p2 = p1;
    *p2 = 20;
    p1 = new int;
    *p1 = 30;
}
```



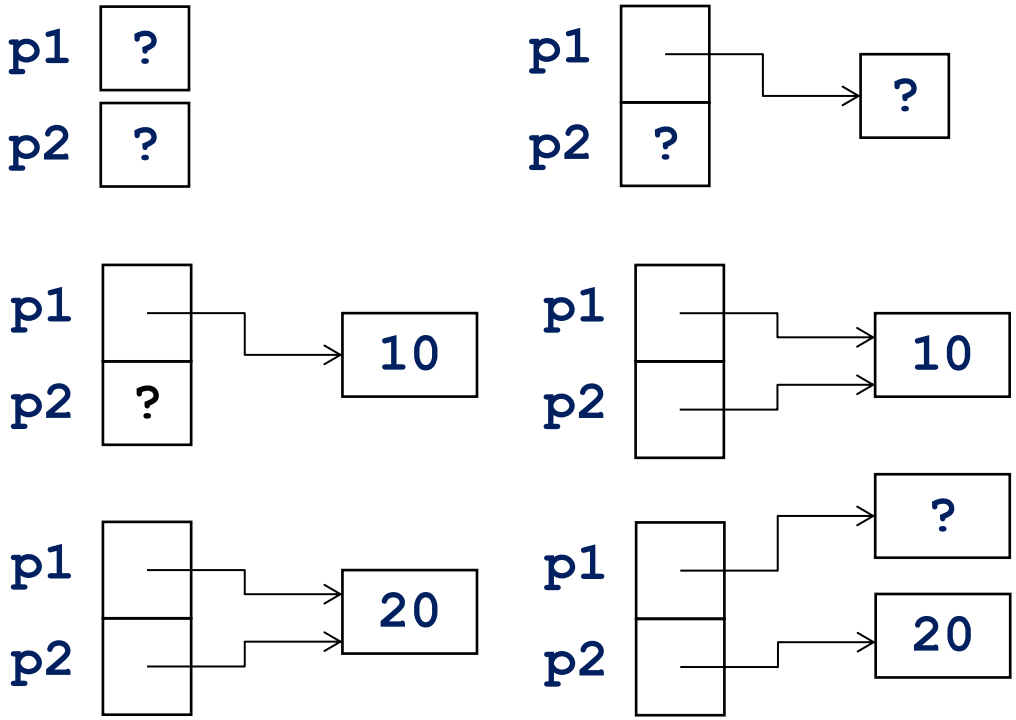
Bài Tập

```
int main
{
    int *p1, *p2;
    p1 = new int;
    *p1 = 10;
    p2 = p1;
    *p2 = 20;
    p1 = new int;
    *p1 = 30;
}
```



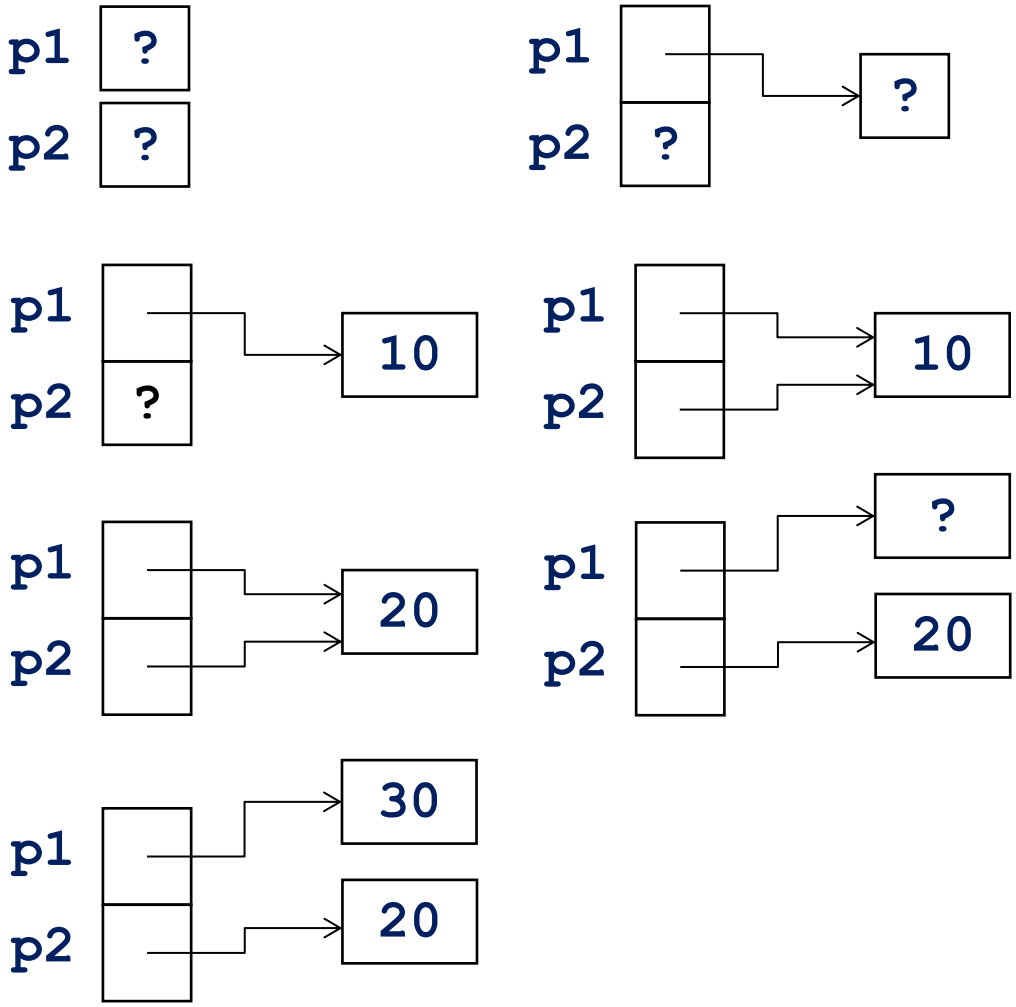
Bài Tập

```
int main
{
      int *p1, *p2;
      p1 = new int;
      *p1 = 10;
      p2 = p1;
      *p2 = 20;
      p1 = new int;
      *p1 = 30;
}
```



Bài Tập

```
int main
{
      int *p1, *p2;
      p1 = new int;
      *p1 = 10;
      p2 = p1;
      *p2 = 20;
      p1 = new int;
      *p1 = 30;
}
```



Toán Tử **delete**

- Sử dụng toán tử **delete** để giải phóng vùng nhớ được tạo ra bởi toán tử **new**
- Lưu ý: chỉ giải phóng vùng nhớ tạo ra bởi **new**, không phải xóa biến con trỏ **p**

```
int main()
{
    int *p = new int;
    … … …
    delete p;
    p = NULL;
    … … …
    return 0;
}
```

Toán Tử **delete**

- Vùng nhớ động có thể được trả về bởi hàm
- Vùng nhớ động chỉ được giải phóng khi sử dụng **delete**, do đó vùng nhớ động tạo ra trong hàm không bị xóa sau khi kết thúc hàm
- Chỉ các biến khai báo trong hàm bị xóa, trong đó có biến con trỏ **p**

```
int* f(int *q)
{
    int *p = new int;
    … … …
    return p;
}
```

Mảng Động

- Biến mảng thực chất là biến con trỏ
- Mảng thông thường (**`int a [10]`**)
  - độ dài mảng cố định
  - không thể thay đổi độ dài sau khi khai báo
  - có thể coi là con trỏ hằng (con trỏ tĩnh)
- Mảng động (**`int *p = new int [10]`**)
  - độ dài mảng có thể thay đổi sau khi khai báo
  - sử dụng lại toán tử **`new`**

```
int *p = new int [10];
p = new int [50];
```

Mảng Động – Mảng Tĩnh

```
int a [10];
int *p = new int [10];
```

- **a** và **p** đều là biến con trỏ, nhưng **a** là hằng
  - có thể gán (được phép)

    ```
    p = a;
    ```

    - **p** chỉ tới địa chỉ mà **a** đang chỉ tới
  - không được phép (lỗi dịch)

    ```
    a = p;
    ```

    - **a** là hằng, không thay đổi giá trị của **a**

Mảng Động

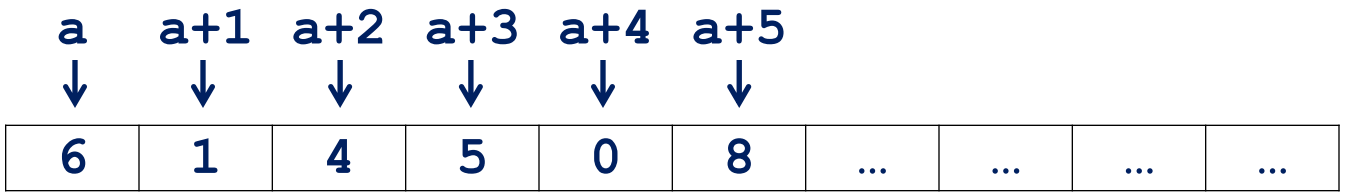
- Hạn chế của mảng thông thường:
  - phải khai báo độ dài trước
  - độ dài mảng có thể không biết tới khi chạy chương trình
- Phải ước lượng độ dài lớn nhất
  - lãng phí bộ nhớ
- Mảng động
  - có thể tăng và giảm khi cần thiết
  - thực hiện sao chép dữ liệu sang mảng mới
  - xóa mảng cũ

Giải Phóng Mảng Động

- Cũng được thực hiện khi chạy
  - giải phóng tất cả bộ nhớ cho mảng động
  - `[]` thông báo giải phóng bộ nhớ cho mảng
  - vẫn chỉ tới vùng nhớ đó

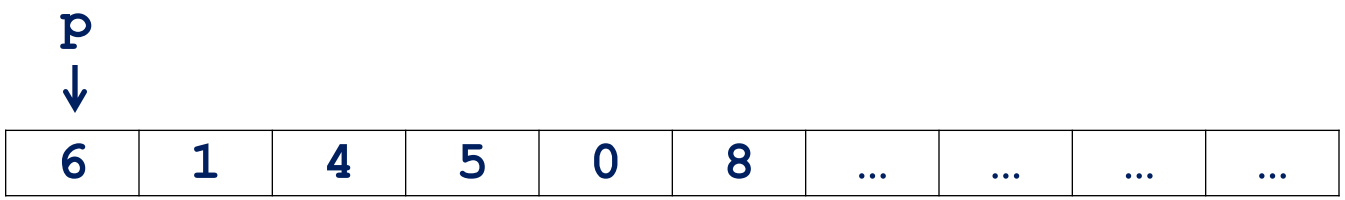
```
int main()
{
    int *p = new int[10];
    … … …
    delete [] p;
    p = NULL;
    … … …
    return 0;
}
```

Bài Tập




```
int a[10] = {6,1,4,5,0,8};
int *p = a;
cout << a[0] << *p << "\n";
p++;
cout << *p << p[2] << "\n";
p++; a[2] = 0;
cout << p[1] << *p << "\n";
p -= 2;
cout << p[3] << p[1] << "\n";
```

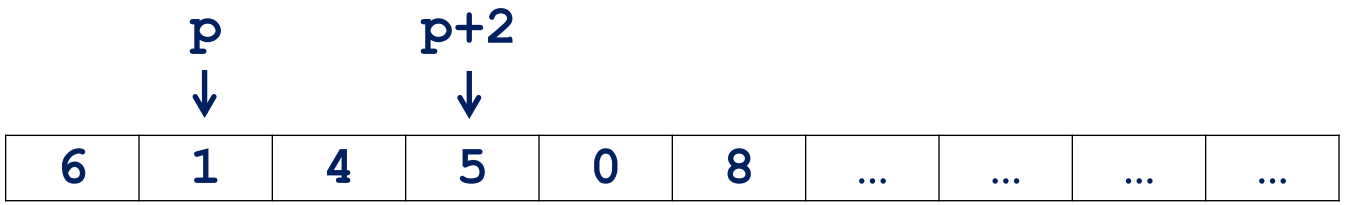
Bài Tập




```
int a[10] = {6,1,4,5,0,8};
int *p = a;
cout << a[0] << *p << "\n";
p++;
cout << *p << p[2] << "\n";
p++; a[2] = 0;
cout << p[1] << *p << "\n";
p -= 2;
cout << p[3] << p[1] << "\n";
```

```
6 6
```

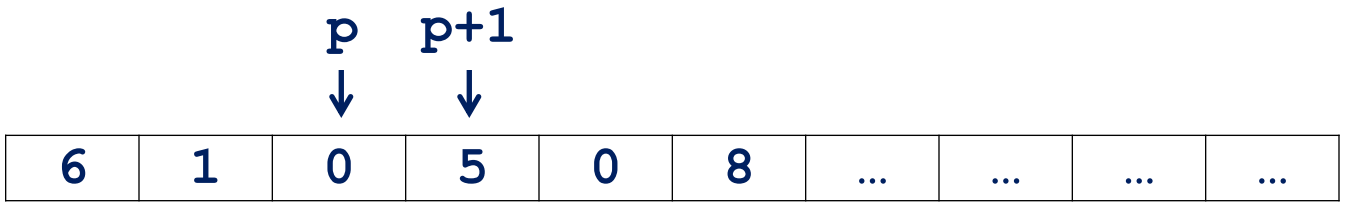
Bài Tập




```
int a[10] = {6,1,4,5,0,8};
int *p = a;
cout << a[0] << *p << "\n";
p++;
cout << *p << p[2] << "\n";
p++; a[2] = 0;
cout << p[1] << *p << "\n";
p -= 2;
cout << p[3] << p[1] << "\n";
```

```
6 6
1 5
```

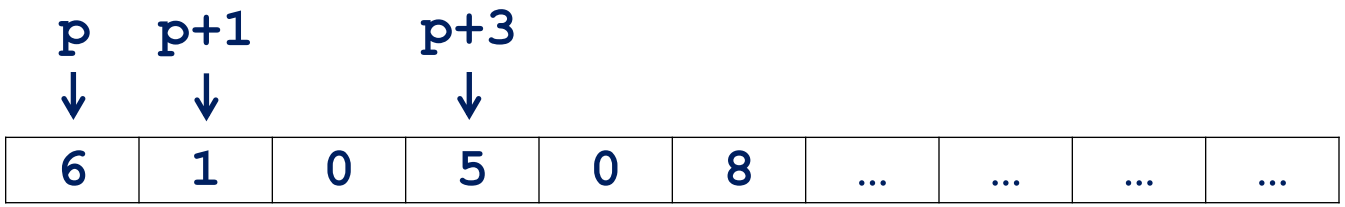
Bài Tập




```
int a[10] = {6,1,4,5,0,8};
int *p = a;
cout << a[0] << *p << "\n";
p++;
cout << *p << p[2] << "\n";
p++; a[2] = 0;
cout << p[1] << *p << "\n";
p -= 2;
cout << p[3] << p[1] << "\n";
```

```
6 6
1 5
5 0
```

Bài Tập




```
int a[10] = {6,1,4,5,0,8};
int *p = a;
cout << a[0] << *p << "\n";
p++;
cout << *p << p[2] << "\n";
p++; a[2] = 0;
cout << p[1] << *p << "\n";
p -= 2;
cout << p[3] << p[1] << "\n";
```

```
6 6
1 5
5 0
5 1
```

Mảng Động Nhiều Chiều

```
int main() {
    int col, row;
    cin >> col >> row;
    int **mang = new int* [row];
    for (int i = 0; i < row; i++) {
        mang[i] = new int[col];
        for (int j = 0; j < col; j++)
            mang[i][j] = 0;
    }
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++)
            cout << mang[i][j] << " ";
        cout << endl;
    }
    return 0;
}
```

Mảng Động Nhiều Chiều

```
int** copy(int **m, int _row, int _col) {
    int **mangCopy = new int*[_row];
    for (int i = 0; i < _row; i++) {
        mangCopy[i] = new int[_col];
        for (int j = 0; j < _col; j++)
            mangCopy[i][j] = m[i][j];
    }
    return mangCopy;
}
int main() {
    … … … … …
    int **mang1 = new int*[… … …];
    … … … … …
    int **mang2 = copy(mang1, row, col);
    return 0;
}
```

Mảng Động Nhiều Chiều

```
int** transpose(int **m, int _row, int _col) {
    int **mangT = new int*[_col];
    for (int i = 0; i < _col; i++) {
        mangT[i] = new int[_row];
        for (int j = 0; j < _row; j++)
            mangT[i][j] = m[j][i];
    }
    return mangT;
}
int main() {
    … … … … …
    int **mang1 = new int*[… … …];
    … … … … …
    int **mang2 = transpose(mang1, row, col);
    return 0;
}
```