CƠ SỞ LẬP TRÌNH

KIỂU CON TRỎ

Nội dung

- Con trỏ và địa chỉ
- Khai báo con trỏ
- Con trỏ và mảng một chiều
- Con trỏ và mảng nhiều chiều
- Mảng các con trỏ
- Con trỏ hàm
- Cấp phát bộ nhớ động

1. Con trỏ và địa chỉ

- Ví dụ:

```
float a=10.12;
```

  - Xác định một biến có tên a có kiểu float và có giá trị 10.12.
  - Máy cấp phát cho x một vùng nhớ gồm 4 byte liên tiếp.
  - Địa chỉ của biến là số thứ tự của byte đầu tiên
- Có nhiều kiểu địa chỉ khác nhau tương ứng với các kiểu biến khác nhau.

Con trỏ và địa chỉ

- Con trỏ là một biến dùng để chứa địa chỉ. Có nhiều kiểu con trỏ tương ứng với nhiều loại địa chỉ.
  - Ví dụ:
    - Con trỏ kiểu int chứa địa chỉ các biến kiểu int…
- ***a** là giá trị được lưu trong bộ nhớ có địa chỉ **a**
- &a là địa chỉ bộ nhớ chứa giá trị **a**

2. Khai báo con trỏ

- Khai báo trực tiếp

  <kiểu dữ liệu> *<tên biến con trỏ>;

  Trong đó: * là toán tử con trỏ

  - Ví dụ:

```
int *p1,m,n;
p1=&n;
*p1=10; /* ô nhớ do con trỏ p1 trỏ đến được
           gán giá trị 10 */
```

- **Chú ý**: Khi gán địa chỉ của 1 biến cho 1 biến con trỏ, mọi sự thay đổi trên nội dung ô nhớ con trỏ chỉ tới sẽ làm giá trị của biến thay đổi theo.
  - Ví dụ: `int *p2,a=10; p2=&a; *p2=*p2+3;`

    Khi đó a sẽ có giá trị 13.

Khai báo con trỏ (tt)

- Khai báo gián tiếp

```
typedef <kiểu dữ liệu> *<tên kiểu con trỏ>;

<tên kiểu con trỏ> <tên biến con trỏ>;
```

  - Ví dụ

```
typedef int *pint;
int *p1;
pint p2, p3;
```

- Kích thước của con trỏ
  - Con trỏ chỉ lưu địa chỉ nên kích thước của mọi con trỏ là như nhau:
    - Môi trường MD-DOS (16 bit): 2 bytes
    - Môi trường Windows (32 bit): 4 bytes

Con trỏ NULL

- Khái niệm
  - Con trỏ NULL là con trỏ không trỏ vào đâu cả.
  - Khác với con trỏ chưa được khởi tạo.

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

NULL

Khởi tạo kiểu con trỏ

- Khởi tạo

```
<tên biến con trỏ> = &<tên biến>;
```

  - Khi mới khai báo, biến con trỏ được đặt ở địa chỉ nào đó (không biết trước).
    - ➔ chứa giá trị không xác định
    - ➔ trỏ đến vùng nhớ không biết trước.
  - Đặt địa chỉ của biến vào con trỏ (toán tử &)

- Ví dụ

```
int a, b;
int *pa = &a, *pb;
pb = &b;
```

Sử dụng con trỏ

- Truy xuất đến ô nhớ mà con trỏ trỏ đến
  - Con trỏ chứa một số nguyên chỉ địa chỉ.
  - Sử dụng toán tử *.
- Ví dụ

```
int n=10;
int *p;
printf("\nDia chi cua n: %p",&n);
printf("\nGia tri cua n: %d",n);
p=&n; //Con tro p tro toi n
printf("\nDia chi cua con tro: %p",&p);
printf("\nGia tri cua con tro: %p",p);
printf("\nGia tri duoc tro toi la: %d",*p);
```

Sử dụng con trỏ (tt)

- Sử dụng tên con trỏ
  - Giá trị của con trỏ (địa chỉ của biến) được sử dụng trong biểu thức
  - Nếu tên con trỏ ở bên trái toán tử gán thì giá trị của biểu thức bên phải phải là địa chỉ.
  - Ví dụ: `float a,*p,*q;  p=&a;  q=p;`
    - ➔ Kết quả con trỏ q chứa địa chỉ của biến a
- Sử dụng dạng khai báo
  - Ví dụ:

```
float x,*px;
px=&x; //px trỏ tới x
```

  - Nếu con trỏ px trỏ tới biến x thì cách viết x và *px là tương đương trong mọi ngữ cảnh.

Các cách truyền đối số

- Truyền giá trị (tham trị)

```
#include <stdio.h>

void hoanvi(int x, int y);

main()
{
        int a = 5, b = 6;
        hoanvi(a, b);
        printf("a = %d, b = %d", a, b);
}
void hoanvi(int x, int y)
{
        int t = x; x = y; y = t;
}
```

Các cách truyền đối số

- Truyền địa chỉ (con trỏ)

```
#include <stdio.h>

void hoanvi(int *x, int *y);

main()
{
        int a = 2912, b = 1706;
        hoanvi(&a, &b);
        printf("a = %d, b = %d", a, b);
}
void hoanvi(int *x, int *y)
{
        int t = *x; *x = *y; *y = t;
}
```

Các cách truyền đối số

- Truyền tham chiếu (C++)

```
#include <stdio.h>

void hoanvi(int &x, int &y);

main()
{
        int a = 2912, b = 1706;
        hoanvi(a, b);
        printf("a = %d, b = %d", a, b);
}
void hoanvi(int &x, int &y)
{
        int t = x; x = y; y = t;
}
```

Một số lưu ý

- Một số lưu ý
  - Nắm rõ quy tắc sau, ví dụ int a, *pa = &a;
    - *pa và a đều chỉ nội dung của biến a.
    - pa và &a đều chỉ địa chỉ của biến a.
  - Không nên sử dụng con trỏ khi chưa được khởi tạo. Kết quả sẽ không lường trước được.
  - Ví dụ:

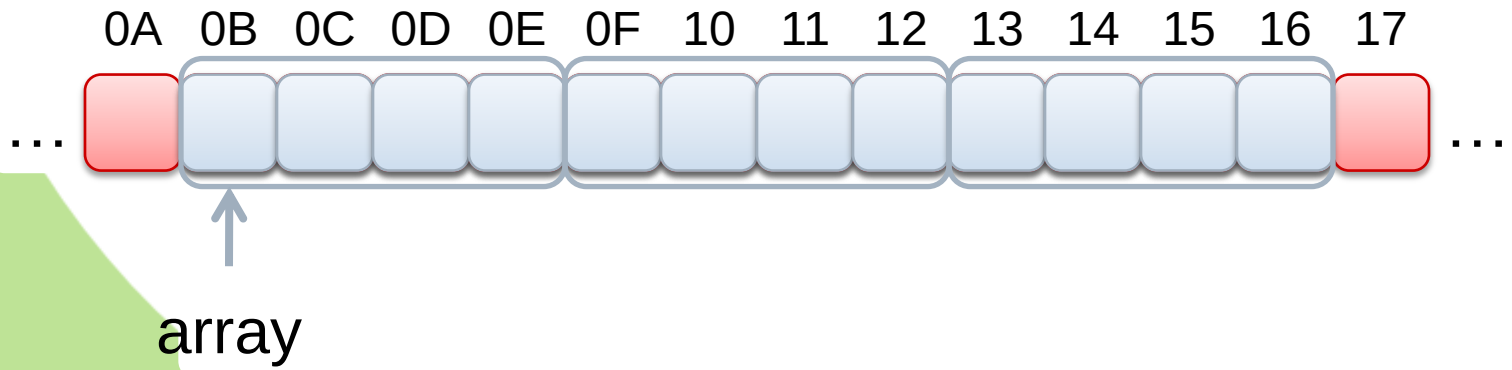
```
int *pa; *pa = 1904;
```

3. Con trỏ và mảng một chiều

Mảng một chiều

```
int array[3];
```

- Tên mảng array là một hằng con trỏ
  ➔ không thể thay đổi giá trị của hằng này.
- array là địa chỉ đầu tiên của mảng
  ➔array == &array[0]

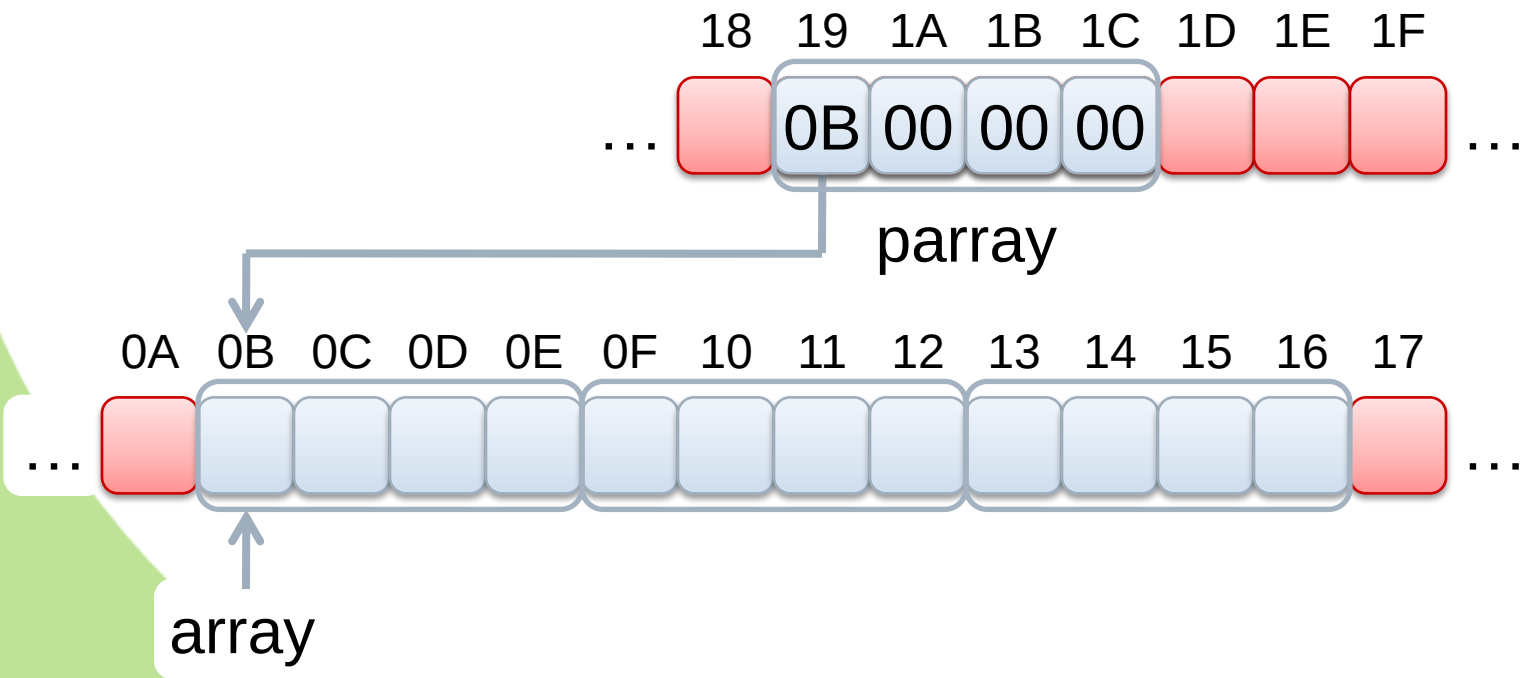


Con trỏ và mảng một chiều

Con trỏ đến mảng một chiều

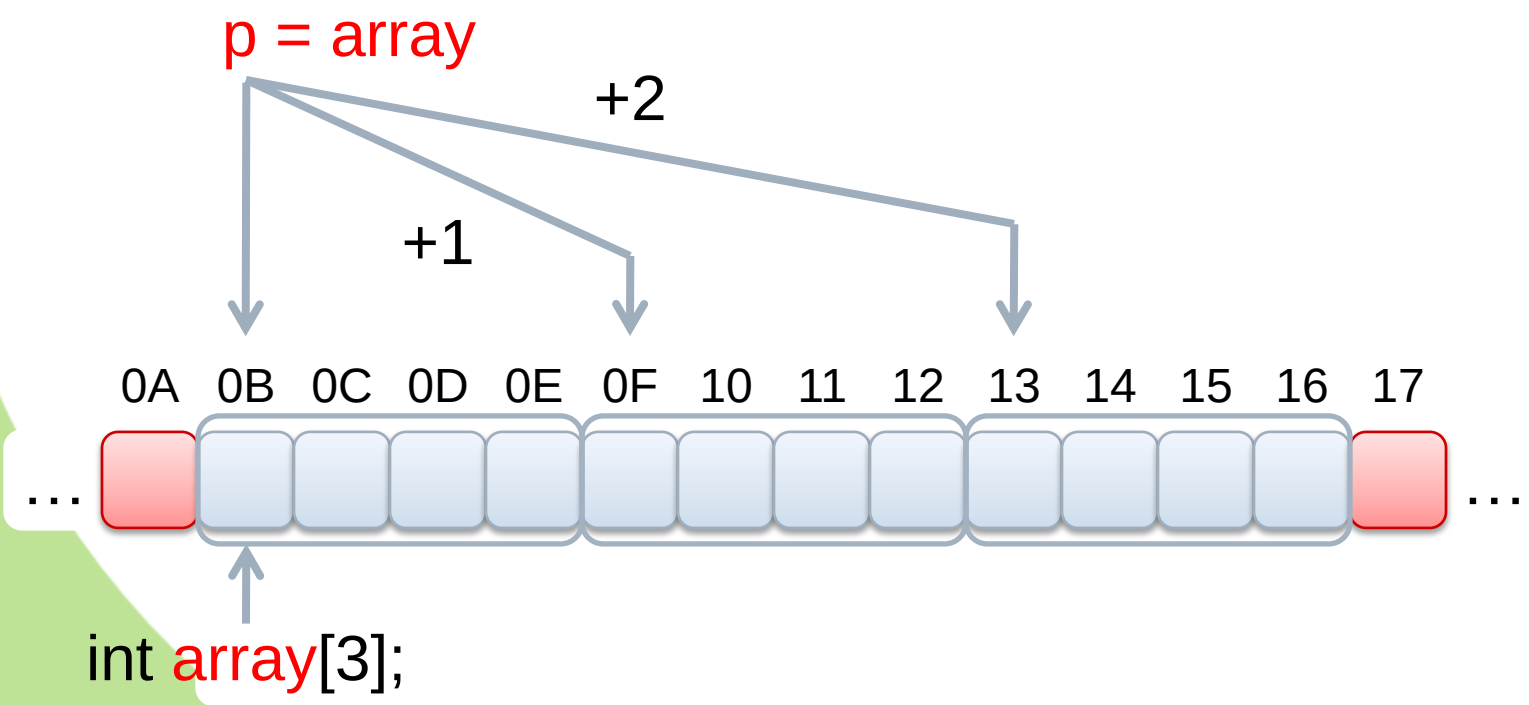
```
int array[3], *parray;

parray = array;          // Cách 1
parray = &array[0];      // Cách 2
```



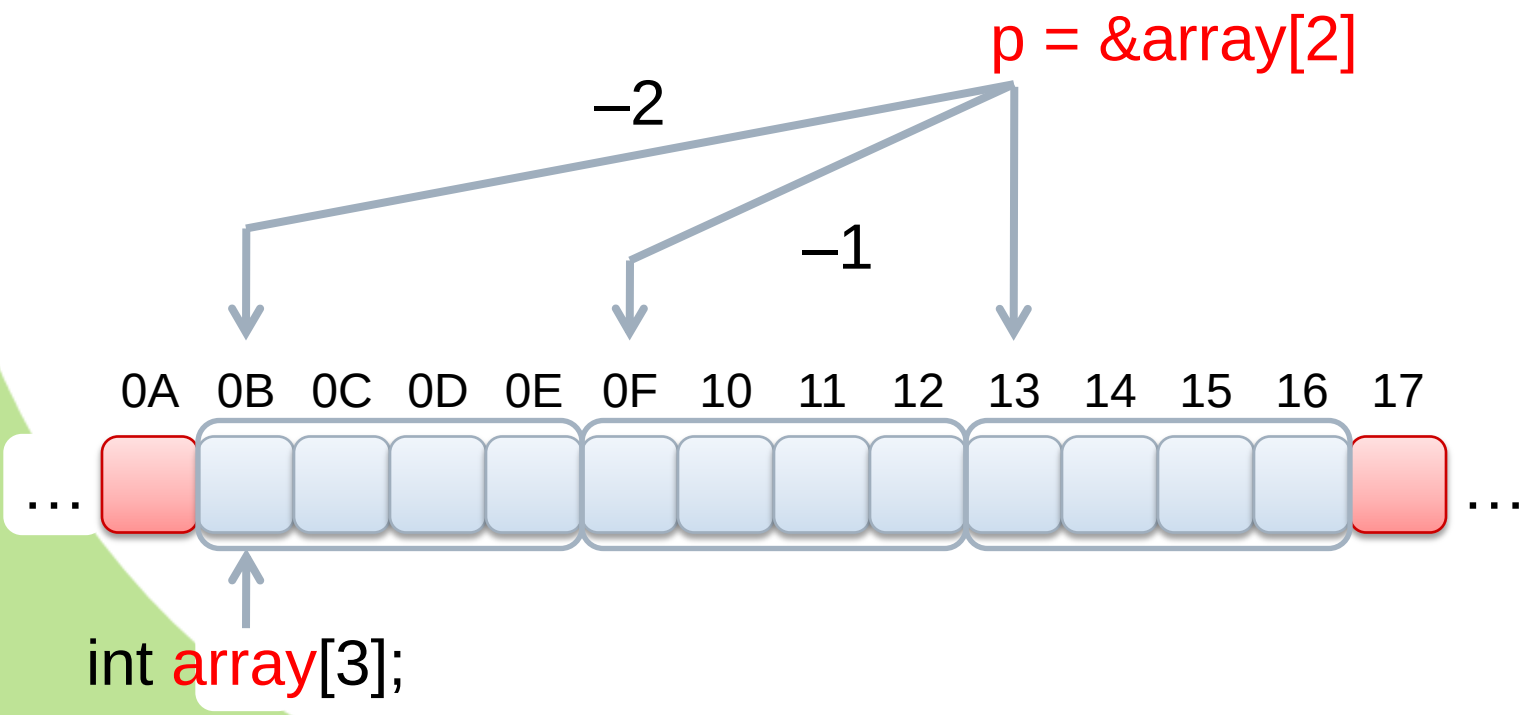
Phép toán số học trên con trỏ

- Phép cộng (tăng)
  - + n ⇔ + n * sizeof(<kiểu dữ liệu>)
  - Có thể sử dụng toán tử gộp += hoặc ++



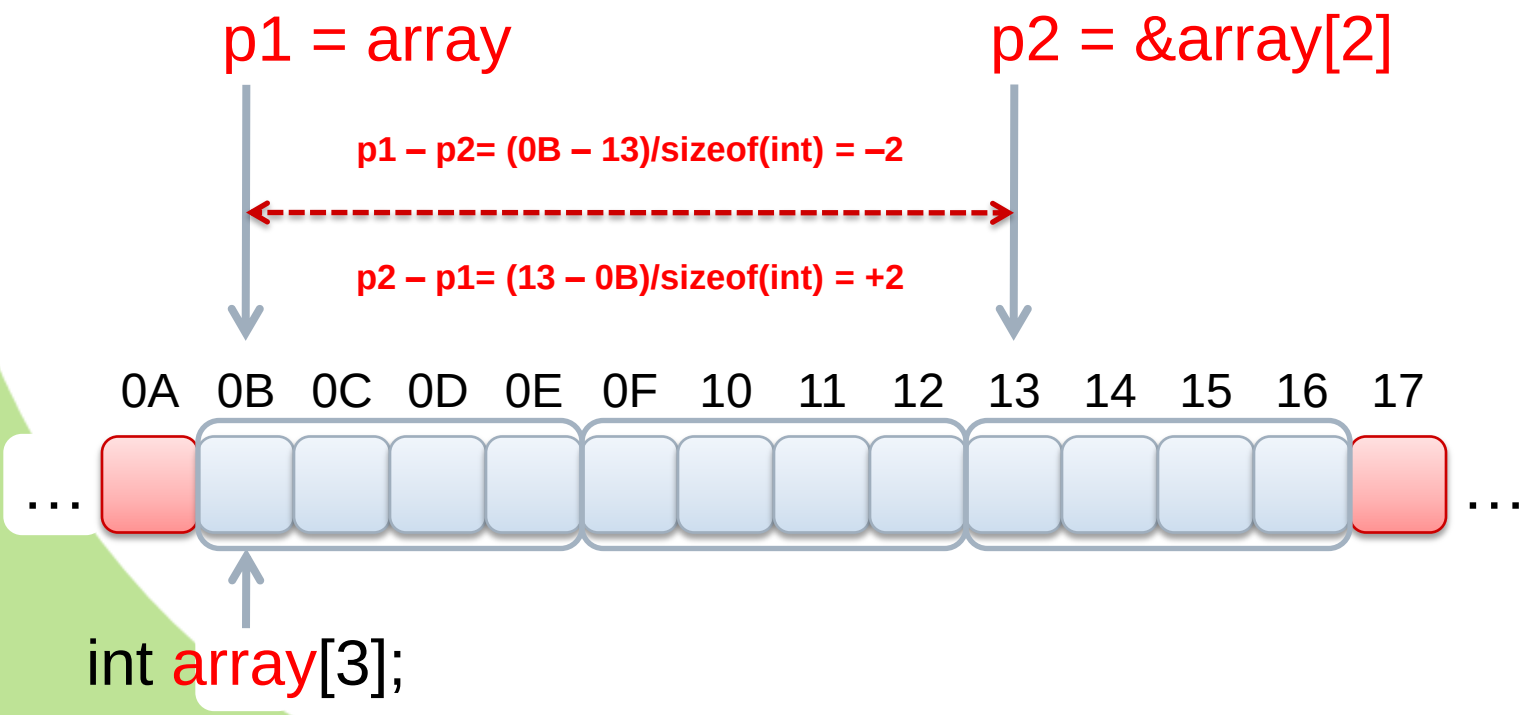
Phép toán số học trên con trỏ

- Phép trừ (giảm)
  - – n ⇔ – n * sizeof(<kiểu dữ liệu>)
  - Có thể sử dụng toán tử gộp –= hoặc – –



Phép toán số học trên con trỏ

- Phép toán tính khoảng cách giữa 2 con trỏ
  - <kiểu dữ liệu> *p1, *p2;
  - p1 – p2 cho ta khoảng cách (theo số phần tử) giữa hai con trỏ (cùng kiểu)

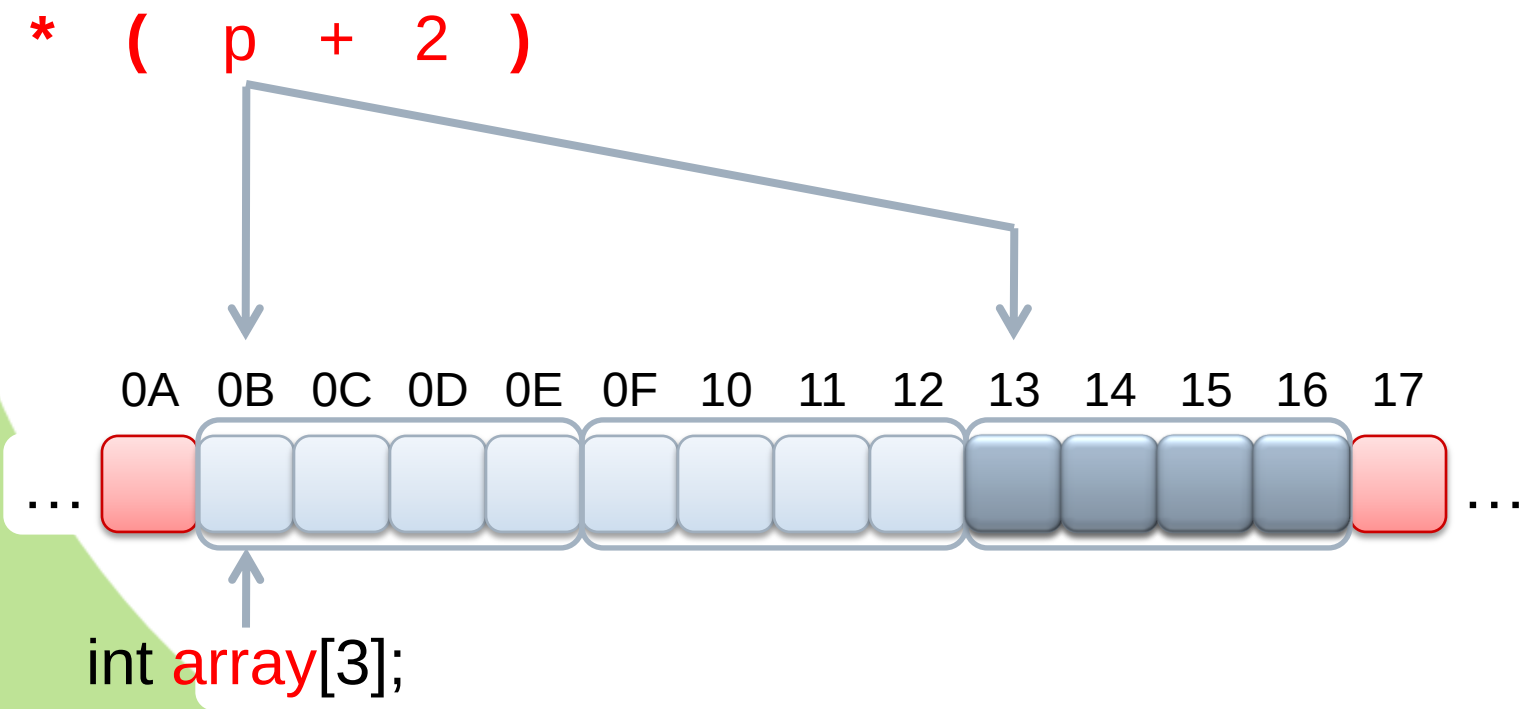


Phép toán số học trên con trỏ

- Các phép toán khác
  - Phép so sánh: So sánh địa chỉ giữa hai con trỏ (thứ tự ô nhớ)
    - == !=
    - > >=
    - < <=
  - Không thể thực hiện các phép toán: * / %

Con trỏ và mảng một chiều

- Truy xuất đến phần tử thứ n của mảng (không sử dụng biến mảng)
  - array[n] == p[n] == *(p + n)



Con trỏ và mảng một chiều (tt)

- Ví dụ: Nhập vào một dãy các số nguyên, tính tổng các số dương trong dãy.

```
void nhapmang(int *a,int &n)
{ int i;
  printf("Nhap so phan tu cua mang: ");
  scanf("%d",&n);
  for(i=0;i<n;i++)
    {    printf("a[%d]= ",i);
         scanf("%d",a+i);    }
}
void xuatmang(int *a,int n)
{    int i;
     printf("\nMang vua nhap la: ");
     for(i=0;i<n;i++)
     printf("%d ",*(a+i));
}
```


Con trỏ và mảng một chiều (tt)

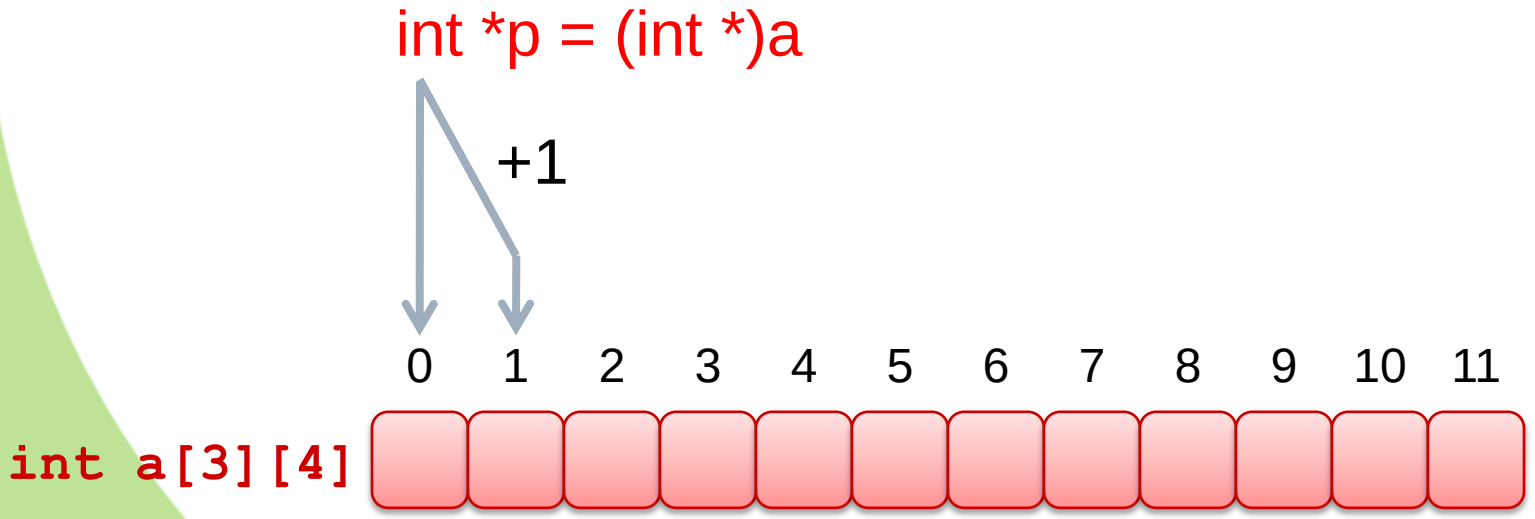
```
long tong(int *a,int n)
{
     int i;
     long t; t=0;
     for(i=0;i<n;i++)
       if (*(a+i)>0) t=t+*(a+i);
     return t;
}
int main()
{
       int a[20],n,i;
       nhapmang(a,n);
       xuatmang(a,n);
       printf("\nTong cac phan tu duong la: %ld",
                      tong(a,n));
       getch();
}
```


Con trỏ và mảng một chiều (tt)

- Lưu ý
  - Không thực hiện các phép toán nhân, chia, lấy phần dư.
  - Tăng/giảm con trỏ n đơn vị có nghĩa là tăng/giảm giá trị của nó n*sizeof(<kiểu dữ liệu mà nó trỏ đến>)
  - Không thể tăng/giảm biến mảng. Hãy gán một con trỏ đến địa chỉ đầu của mảng và tăng/giảm nó.
  - Đối số mảng một chiều truyền cho hàm là địa chỉ phần tử đầu tiên của mảng.

4. Con trỏ và mảng 2 chiều

- Hướng tiếp cận
  - Các phần tử tạo thành mảng 1 chiều
  - Sử dụng con trỏ int * để duyệt mảng 1 chiều



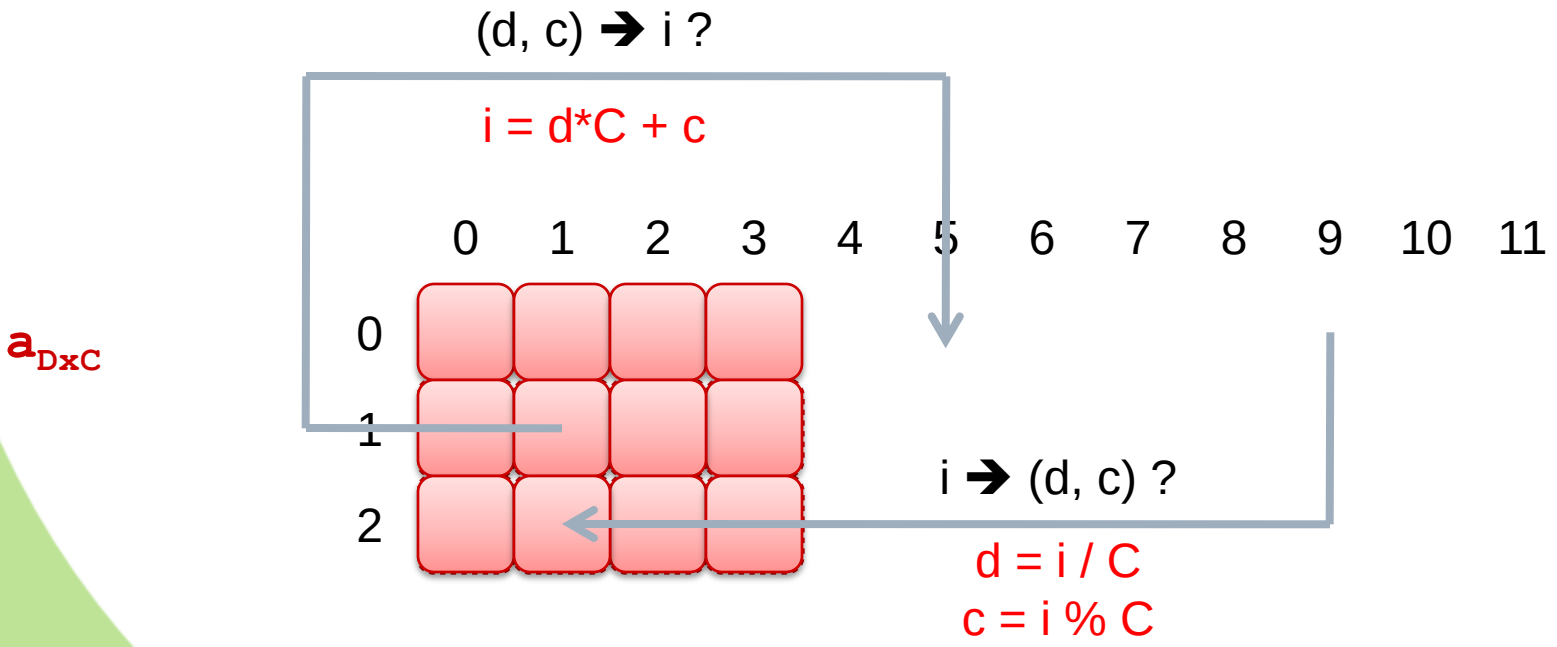
Con trỏ và mảng 2 chiều (tt)

- Nhập / Xuất theo chỉ số mảng 1 chiều

```
main()
{
        int a[D][C], i,m,n;
        int *p = (int *)a;
        printf("Nhap so dong, so cot: ");
        scanf("%d%d",&m,&n);
        for (i = 0; i < m*n; i++)
        {       printf("Nhap phan tu thu %d: ", i);
                scanf("%d", p + i);
        }
        for (i = 0; i < m*n; i++)
                {  printf("%5d ", *(p + i));
                   if ((i+1)%n==0) printf("\n");
                }
        getch();
}
```

Con trỏ và mảng 2 chiều (tt)

- Liên hệ giữa chỉ số mảng 1 chiều và chỉ số mảng 2 chiều?



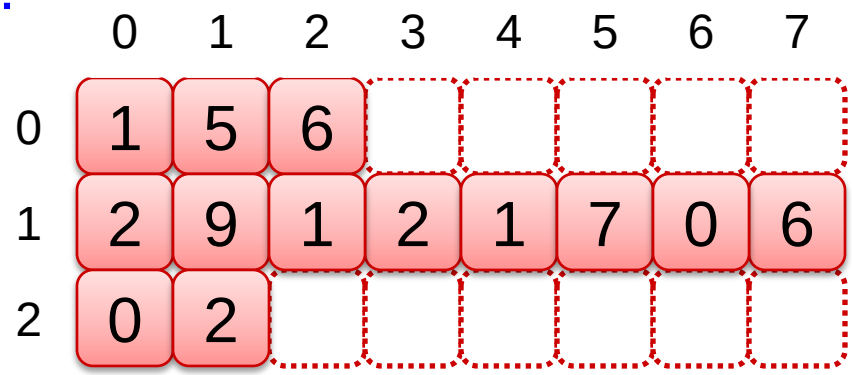
Con trỏ và mảng 2 chiều (tt)

- Nhập / Xuất theo chỉ số mảng 2 chiều

```
#define D 100
#define C 100
int a[D][C], i, d, c,m,n;
int *p = (int *)a;
main()
{
 printf("Nhap so dong, so cot: "); scanf("%d%d",&m,&n);
 for (i = 0; i < m*n; i++)
 {    printf("Nhap a[%d][%d]: ", i / n, i % n);
      scanf("%d", p + i);  }
 for (d = 0; d < m; d++)
 {    for (c = 0; c < n; c++)
      printf("%5d", *(p + d * n + c));// hoac *p++
      printf("\n");
 }
}
```

5. Mảng con trỏ

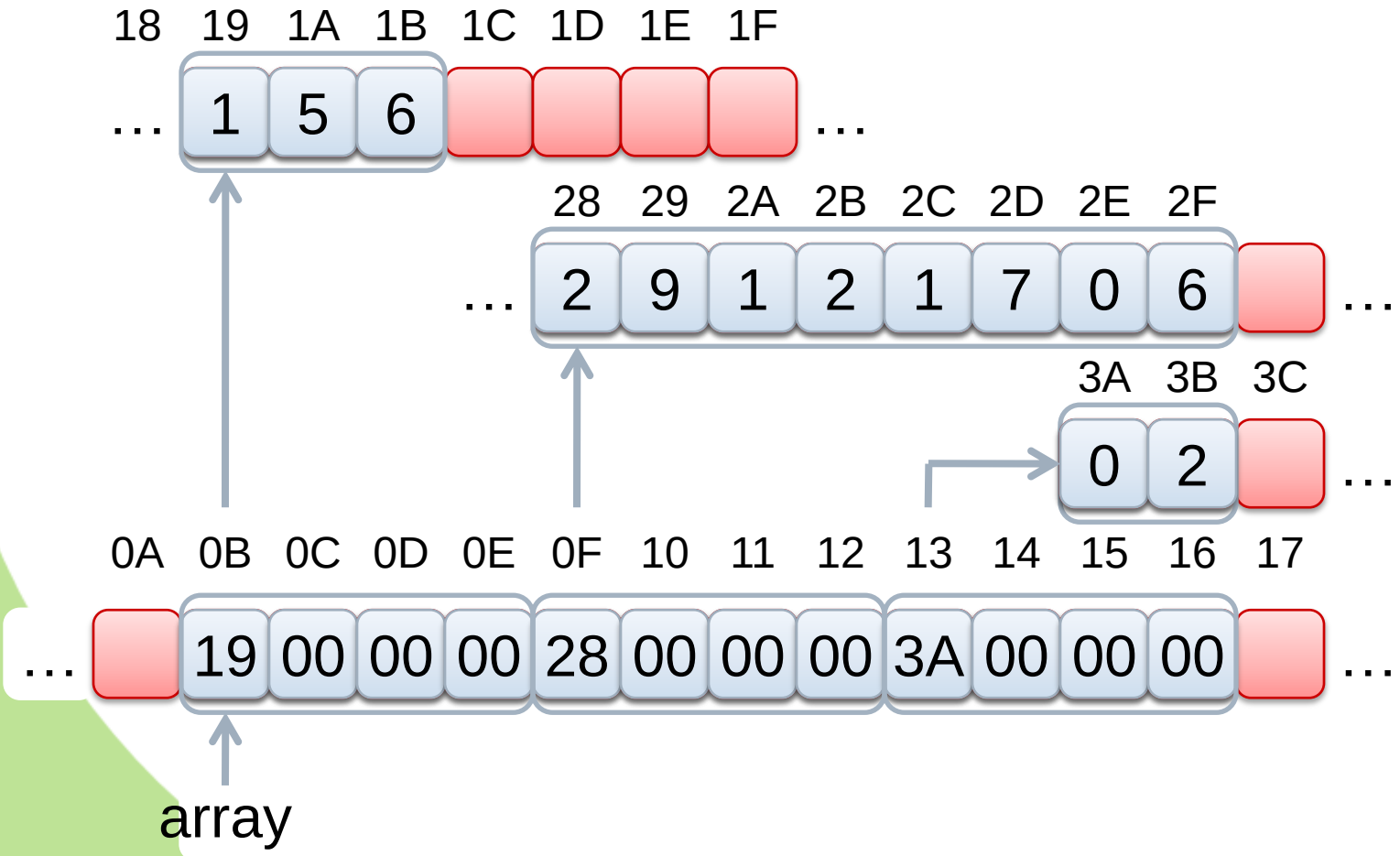
- Bài toán
  - Sử dụng cấu trúc dữ liệu nào để lưu trữ thông tin sau?



- Giải pháp?
  - Cách 1: Mảng 2 chiều 3x8 (tốn bộ nhớ)

Mảng con trỏ (tt)

- Cách 2: Mảng 1 chiều các con trỏ



Mảng con trỏ (tt)

- Cú pháp

```
<Kiểu dữ liệu> *Tên mảng con trỏ [số phần tử];
```

- Ví dụ: Chương trình giải bài toán mở đầu

```
int a[200],n,m;
int sl[20]; //Chua so luong phan tu cua moi mang
int *p[20]; //Mang cac con tro
void nhap()
{ int i,j=0;
  printf("Nhap so mang: "); scanf("%d",&m);
  for (i=0;i<m;i++)
  {  printf("Nhap so phan tu cua mang: "); scanf("%d",&n);
     p[i]=a+j;
     for(int k=0;k<n;k++)
         { printf("a[%d]= ",k);
           scanf("%d",a+j);    j++; }
     sl[i]=n;
  }
}
```

Mảng con trỏ (tt)

```
void xuat()
{     int i,j;
      for(i=0;i<m;i++)
          {
              for (j=0;j<sl[i];j++)
                  printf("%d ",*((p[i])+j));
              printf("\n");
          }
}
main()
{
       nhap();
       xuat();
       getch();
}
```

6. Con trỏ hàm (tự đọc)

- Khai báo tường minh

```
<kiểu trả về> (* <tên biến con trỏ>)(ds tham số);
```

- Ví dụ

```
// Con trỏ đến hàm nhận đối số int, trả về int
int (*ptof1)(int x);

// Con trỏ đến hàm nhận 2 đối số double, không trả về
void (*ptof2)(double x, double y);

// Con trỏ đến hàm nhận đối số mảng, trả về char
char (*ptof3)(char *p[]);

// Con trỏ đến không nhận đối số và không trả về
void (*ptof4)();
```

Con trỏ hàm (tt)

- Khai báo không tường minh (thông qua kiểu)

```
typedef <kiểu trả về> (* <tên kiểu>)(ds tham số);

<tên kiểu> <tên biến con trỏ>;
```

- Ví dụ

```
int (*pt1)(int, int);    // Tường minh

typedef int (*PhepToan)(int, int);

PhepToan pt2, pt3;       // Không tường minh
```

Con trỏ hàm (tt)

- Gán giá trị cho con trỏ hàm

```
<biến con trỏ hàm> = <tên hàm>;

<biến con trỏ hàm> = &<tên hàm>;
```

  - Hàm được gán phải cùng dạng (vào, ra)

- Ví dụ

```
int Cong(int x, int y);                  // Hàm
int Tru(int x, int y);                   // Hàm
int (*tinhtoan)(int x, int y);           // Con trỏ hàm

tinhtoan = Cong;  // Dạng ngắn gọn
tinhtoan = &Tru;  // Dạng sử dụng địa chỉ
tinhtoan = NULL;  // Không trỏ đến đâu cả
```

Con trỏ hàm (tt)

- So sánh con trỏ hàm

```
if  (tinhtoan != NULL)
{
      if (tinhtoan == &Cong)
            printf("Con trỏ đến hàm Cong.");
      else
            if (tinhtoan == &Tru)
                  printf("Con trỏ đến hàm Tru.");
            else
                  printf("Con trỏ đến hàm khác.");
}
else
      printf("Con trỏ chưa được khởi tạo!");
```

Con trỏ hàm (tt)

- Gọi hàm thông qua con trỏ hàm
  - Sử dụng toán tử lấy nội dung "*" (chính quy) nhưng trường hợp này có thể bỏ

```
int Cong(int x, int y);
int Tru(int x, int y);

int (*tinhtoan)(int, int);

tinhtoan = Cong;
int kq1 = (*tinhtoan)(1, 2);   // Chính quy
int kq2 = tinhtoan(1, 2);      // Ngắn gọn
```

Con trỏ hàm (tt)

- Truyền tham số là con trỏ hàm

```
int Cong(int x, int y);
int Tru(int x, int y);
int TinhToan(int x, int y, int (*pheptoan)(int, int))
{
        int kq = (*pheptoan)(x, y);    // Gọi hàm
        return kq;
}

main()
{
        int (*pheptoan)(int, int) = &Cong;
        int kq1 = TinhToan(1, 2, pheptoan);
        int kq2 = TinhToan(1, 2, &Tru);
}
```

Con trỏ hàm (tt)

- Trả về con trỏ hàm

```
int (*LayPhepToan(char code))(int, int)
{
        if (code == '+')
                return &Cong;
        return &Tru;
}

main()
{
        int (*pheptoan)(int, int) = NULL;
        pheptoan = LayPhepToan('+');
        int kq2 = pheptoan(1, 2, &Tru);
}
```

Con trỏ hàm (tt)

- Trả về con trỏ hàm (khai báo kiểu)

```
typedef (*PhepToan)(int, int);
PhepToan LayPhepToan(char code)
{
        if (code == '+')
                return &Cong;
        return &Tru;
}

main()
{
        PhepToan pheptoan = NULL;
        pheptoan = LayPhepToan('+');
        int kq2 = pheptoan(1, 2, &Tru);
}
```

Con trỏ hàm (tt)

Mảng con trỏ hàm

```
typedef (*PhepToan)(int, int);
main()
{
        int (*array1[2])(int, int);    // tường minh
        PhepToan array2[2];            // kô tường minh

        array1[0] = array2[1] = &Cong;
        array1[1] = array2[0] = &Tru;

        printf("%d\n", (*array1[0])(1, 2));
        printf("%d\n", array1[1](1, 2));
        printf("%d\n", array2[0](1, 2));
        printf("%d\n", array2[1](1, 2));
}
```

Con trỏ hàm (tt)

- Lưu ý
  - Không được quên dấu () khi khai báo con trỏ hàm
    - int (*PhepToan)(int x, int y);
    - int *PhepToan(int x, int y);
  - Có thể bỏ tên biến tham số trong khai báo con trỏ hàm
    - int (*PhepToan)(int x, int y);
    - int (*PhepToan)(int, int);

7. Cấp phát bộ nhớ động

- Biến tĩnh:
  - Ví dụ: int, float, mảng…
  - Là các biến được xác định khi mô tả kiểu, địa chỉ của các biến được xác định ngay khi thực hiện chương trình;
  - Thời gian tồn tại cùng với thời gian tồn tại của khối chương trình chứa khai báo
- Biến động:
  - Được tạo ra lúc chạy chương trình theo yêu cầu
  - Biến động không có tên
  - Truy nhập thông qua các biến con trỏ (là biến tĩnh, chứa địa chỉ của các biến động)

Cấp phát bộ nhớ động

- Cấp phát bộ nhớ (trong thư viện stdlib.h,alloc.h)

void *malloc(Số ô nhớ cần cấp phát)

Cấp phát vùng nhớ có kích thước được chỉ ra.

void *calloc(n,sizeof(object))

Cấp phát vùng nhớ có kích thước là n*sizeof(object)

Dùng khi cấp phát bộ nhớ cho kiểu dữ liệu không phải kiểu cơ sở

- Ví dụ:

```
int a, *pa, *pb;
pa = (int*)malloc(sizeof(int));
pb= (int*)calloc(10, sizeof(int));
```

- **Lưu ý**: Khi sử dụng hàm malloc() hay calloc(), ta phải ép kiểu vì nguyên mẫu các hàm này trả về con trỏ kiểu void.

Cấp phát bộ nhớ động (tt)

Cấp phát lại vùng nhớ cho biến con trỏ

```
void *realloc(void *poiter, kích thước mới)
```

- *Ý nghĩa*:
  - Cấp phát lại 1 vùng nhớ cho con trỏ *pointer* quản lý, vùng nhớ này có kích thước mới là kích thước mới được chỉ ra; khi cấp phát lại thì nội dung vùng nhớ trước đó vẫn tồn tại.
  - Kết quả trả về của hàm là địa chỉ đầu tiên của vùng nhớ mới. Địa chỉ này có thể khác với địa chỉ được chỉ ra khi cấp phát ban đầu.
- *Ví dụ*:

```
int a, *pa;
pa=(int*)malloc(sizeof(int));
pa = realloc(pa, 6); /* Cấp phát lại vùng nhớ có kích thước 6 byte*/
```

Cấp phát bộ nhớ động (tt)

- Giải phóng vùng nhớ đã cấp phát

```
void free(void *pointer)
```

  - *Ý nghĩa*: Giải phóng vùng nhớ được quản lý bởi con trỏ pointer.
  - *Ví dụ*:

Ở ví dụ trên, sau khi thực hiện xong, ta giải phóng vùng nhớ cho 2 biến con trỏ pa và pb:

```
free(pa);
free(pb);
```

Ví dụ cấp phát bộ nhớ động

- Nhập từ bàn phím một số nguyên n, đưa ra màn hình n số nguyên tố đầu tiên

```
int ngto(long a)
{ //Kiem tra so nguyen to
}
int main()
{   long *prime=NULL, x=0;
    int i=0, found=0, n=0,j;
    printf("Ban muon tim bao nhieu so nguyen to? ");
    scanf("%d",&n);
    prime=(long *)malloc(n*sizeof(long));
    if (prime==NULL)
      {
            printf("Khong du bo nho!");
            return 0;
      }
```


Ví dụ cấp phát bộ nhớ động (tt)

```
*prime=2;  //Cac so nguyen to dau tien da biet
*(prime+1)=3;
*(prime+2)=5;
x=5; j=3;
do
      {
          x+=2; //Gia tri tiep theo de kiem tra
          if (ngto(x))
             {          *(prime+j)=x;
                        j++;
             }
      }
 while (j<n);
printf("%d so nguyen to dau tien la: \n",n);
for (i=0;i<n;i++) printf("%ld ",*(prime+i));
getch();
}
```


Bài tập thực hành

- **Bài 1**. Viết chương trình khai báo mảng hai chiều có 12x12 phần tử kiểu char. Gán ký tự 'X' cho mọi phần tử của mảng này. Sử dụng con trỏ đến mảng để in giá trị các phần tử mảng lên màn hình ở dạng lưới.
- **Bài 2**. In giá trị của con trỏ và giá trị của biến mà nó trỏ tới.
- **Bài 3**. Sử dụng con trỏ để làm lại các bài tập về mảng một chiều.
  - Nhập, xuất mảng
  - Tính tổng các phần tử âm, nguyên tố, chính phương…

Bài tập thực hành (tt)

- **Bài 4.** Viết chương trình nhập số nguyên dương n gồm k chữ số ($0 < k \leq 9$) , sắp xếp các chữ số của n theo thứ tự tăng dần.
  Ví dụ:
  - Nhập n = 1536
  - Kết quả sau khi sắp xếp: 1356.
- **Bài 5.** Sử dụng con trỏ để làm lại các bài tập về mảng hai chiều.
  - Nhập, xuất mảng
  - Tính tổng các phần tử trên đường chéo chính…
  - Đưa ra các số nguyên tố, chính phương, số đẹp trong mảng