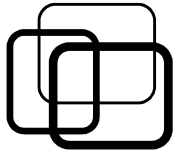
Con trỏ

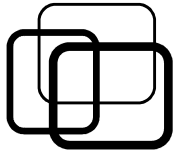
GV. Nguyễn Minh Huy

Nội dung



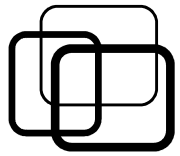
- Khái niệm con trỏ.
- Sử dụng con trỏ.
- Con trỏ vs. mảng.

Nội dung

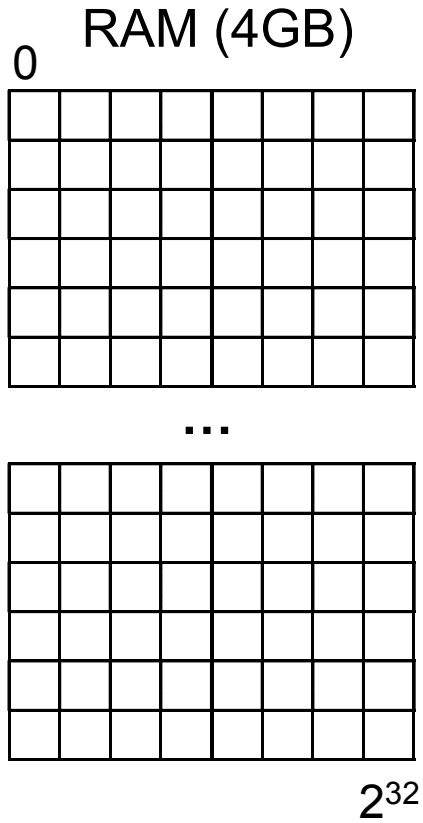


- **Khái niệm con trỏ.**
- Sử dụng con trỏ.
- Con trỏ vs. mảng.

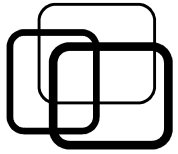
Khái niệm con trỏ



- Bộ nhớ máy tính:
  - RAM (**R**andom **A**ccess **M**emory):
    - Primary vs. Secondary memory.
  - RAM dùng để chứa:
    - Hệ điều hành.
    - Chương trình: lệnh + dữ liệu.
  - Bao gồm các ô nhớ 1 byte.
    - RAM 4GB ~ 4 tỷ ô nhớ.
  - Mỗi ô nhớ có địa chỉ đánh số từ 0.
    - RAM 4GB địa chỉ từ 0 → $2^{32} - 1$.

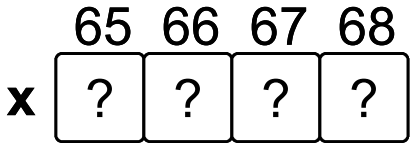


Khái niệm con trỏ



- Địa chỉ biến:
  - Điều gì xảy ra khi khai báo biến?
    - Cấp một dãy ô nhớ liên tiếp.
    - Gắn tên biến với địa chỉ ô đầu dãy.
    - Bao nhiêu ô? → kiểu dữ liệu.

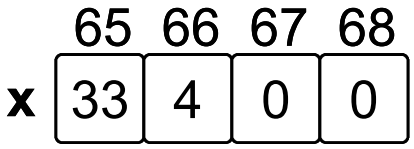
int **x**;



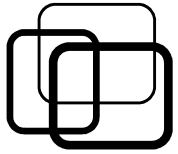
➔ Địa chỉ biến = địa chỉ ô đầu tiên.

  - Giá trị biến được lưu thế nào?
    - Chia giá trị biến thành các byte.
    - Lưu mỗi byte vào một ô nhớ.
    - Thứ tự lưu:
      - Byte thấp đến cao.
      - Ô đầu đến cuối dãy.

x = 1057;



Khái niệm con trỏ



- Kiểu địa chỉ trong C:
  - Để lưu số nguyên, thực? → kiểu int, float.
  - Để lưu địa chỉ biến? → kiểu địa chỉ.
  - Cú pháp: <kiểu dữ liệu biến> *.
    - Biến int có địa chỉ kiểu int *.
  - Toán tử **&**:
    - Công dụng: lấy địa chỉ của biến.
    - Cú pháp: **&**<Tên biến>;

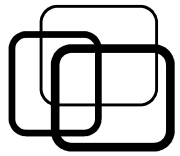
```
int     x = 1057;
float  y = 1.25;
int     *address_x = &x;
float  *address_y = &y;
```

x = 1057;

	65	66	67	68
x	33	4	0	0

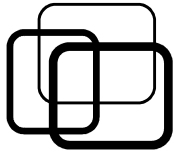
	91	92	93	94
address_x	65	0	0	0

Khái niệm con trỏ



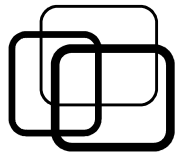
- Con trỏ trong C:
  - Là biến có kiểu địa chỉ.
  - Lưu địa chỉ của biến khác.
  - Kích thước con trỏ:
    - Bằng nhau dù khác kiểu địa chỉ.
    - Tùy thuộc vào hệ máy tính.
      - Intel 8008 (1972), 8-bit, 1 byte (256 B).
      - Intel 8086 (1978), 16-bit, 2 bytes (64 KB).
      - Intel 80386 (1985), 32-bit, 4 bytes (4 GB).
      - Intel Core (2000), 64-bit, 8 bytes (16 TB).

Nội dung



- Khái niệm con trỏ.
- **Sử dụng con trỏ.**
- Con trỏ vs. mảng.

Sử dụng con trỏ



- Khai báo con trỏ:
  - Khai báo biến có kiểu địa chỉ.
  - Cách 1:

    <Kiểu dữ liệu> *<Tên con trỏ>;

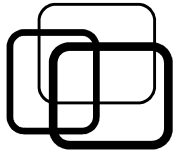
    int *p1; // Con trỏ kiểu int.
    float *p2; // Con trỏ kiểu float.
  - Cách 2:

    typedef <Kiểu dữ liệu> * <Tên thay thế>;

    <Tên thay thế> <Tên con trỏ>;

    typedef int * ConTroInt;
    typedef float * ConTroFloat;
    ConTroInt p1;
    ConTroFloat p2;

Sử dụng con trỏ



- Khởi tạo con trỏ:
  - Con trỏ vừa khai báo nhận địa chỉ nào?
  - Toán tử **&**: gán địa chỉ cho con trỏ.

    <Tên con trỏ> = **&**<Tên biến>;

    ```
    int  x;
    int  *p = &x;
    ```

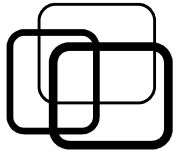
  - Con trỏ kiểu gì thì chỉ nhận địa chỉ của biến kiểu đó!!

    ```
    float  y;
    int    *q = &y;        // Sai.
    ```

  - Địa chỉ rỗng:
    - Không thuộc ô nhớ nào, khởi tạo địa chỉ mặc định.

    ```
    int  *r = NULL;      // Địa chỉ rỗng C, thư viện <stdio.h>
    int  *s = nullptr;   // Địa chỉ rỗng C++, không cần thư viện.
    ```

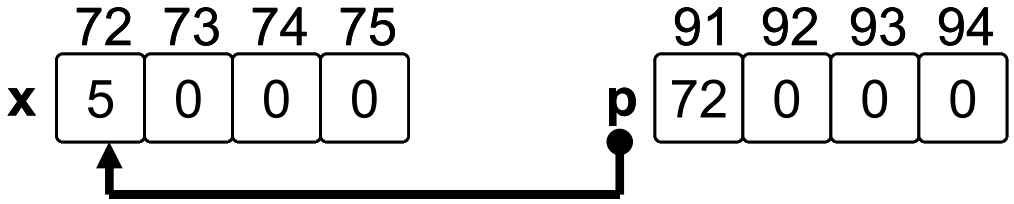
Sử dụng con trỏ



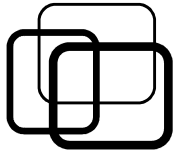
- Truy xuất nội dung vùng nhớ:
  - Toán tử *:
    - Công dụng: truy xuất nội dung vùng nhớ con trỏ giữ địa chỉ.
    - Cú pháp: <Tên biến> = *<Tên con trỏ>;

```
int  x = 5;
int  *p = &x;
int  k = *p;            // lấy giá trị của x.
printf("%d\n", p);   // Xuất địa chỉ x.
printf("%d\n", *p);  // Xuất giá trị x.
printf("%d\n", &p);  // Xuất địa chỉ p.
```

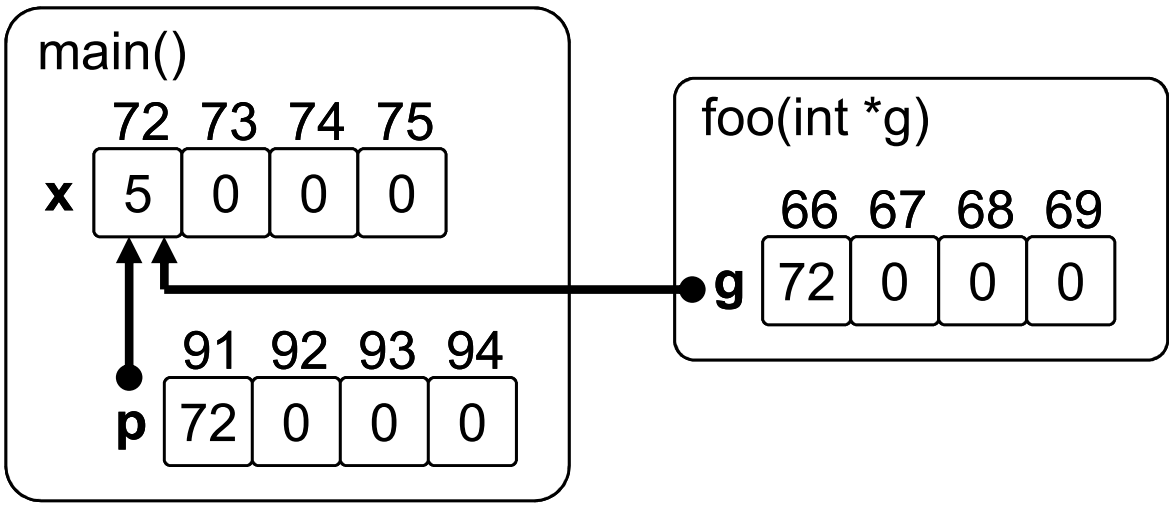
  ➔ Con trỏ **"trỏ"** đến vùng nhớ nó giữ địa chỉ!!



Sử dụng con trỏ

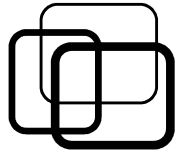


- Truyền tham số con trỏ:
  - Truyền tham trị:
    - Truyền bản sao con trỏ vào hàm.
    - Giá trị con trỏ KHÔNG thay đổi.
    - Nội dung vùng nhớ con trỏ "trỏ" đến CÓ THỂ bị thay đổi.

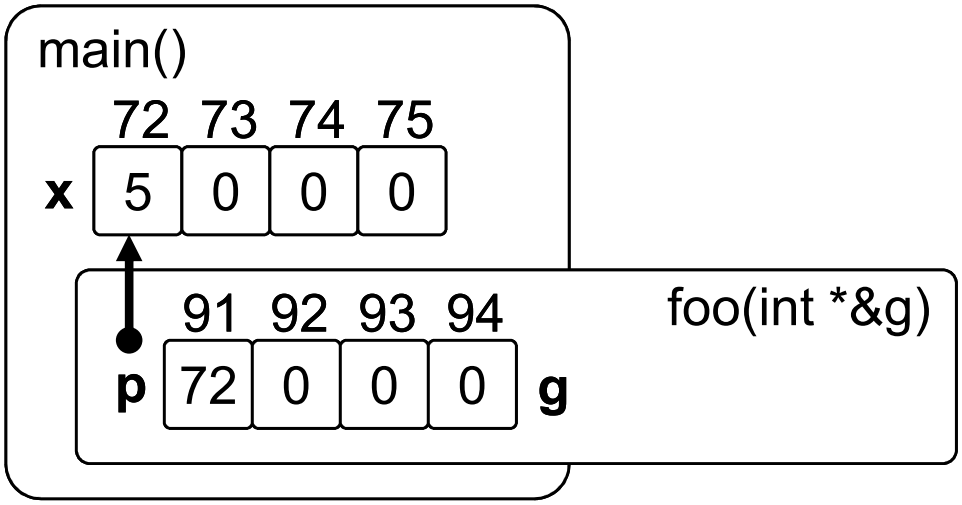



```
void foo( int *g )
{
        *g = *g + 1;
        g = g + 1;
}

int main()
{
        int  x = 5;
        int  *p = &x;

        foo(p);
        foo(&x);
        // Giá trị x đổi.
}
```

Sử dụng con trỏ

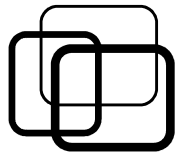


- Truyền tham số con trỏ:
  - Truyền tham chiếu (C++):
    - Truyền bản gốc con trỏ vào hàm.
    - Giá trị con trỏ CÓ THỂ thay đổi.
    - Nội dung vùng nhớ con trỏ "trỏ" đến CÓ THỂ bị thay đổi.




```
void foo( int *&g )
{
        *g = *g + 1;
        g = g + 1
}

int main()
{
        int  x = 5;
        int  *p = &x;

        foo(p);
        foo(&x);   // Sai
        // Giá trị x đổi.
        // Giá trị p đổi.
}
```

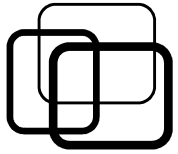
Sử dụng con trỏ



- Con trỏ cấu trúc:
  - Giữ địa chỉ biến cấu trúc.
  - Khai báo:
    - Cách 1: struct <Kiểu cấu trúc> *<Tên con trỏ>;
    - Cách 2: **typedef** struct <Kiểu cấu trúc> * <Tên thay thế>;
      <Tên thay thế> <Tên con trỏ>;

```
struct PhanSo
{
   int tu, mau;
};
typedef struct PhanSo * ConTroPhanSo;

struct PhanSo     *p;
ConTroPhanSo   q;
```

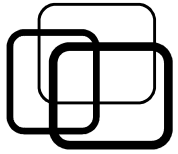
Sử dụng con trỏ



- Con trỏ cấu trúc:
  - Truy xuất thành phần:
    - Cách 1: **(***<Tên con trỏ>**)**.<Tên thành phần>;
    - Cách 2: <Tên con trỏ>**->**<Tên thành phần>;

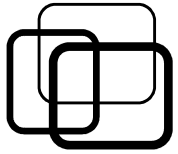
```
struct PhanSo      p;
struct PhanSo      *q = &p;

(*q).tu = 1;
q->mau = 2;
```

Nội dung



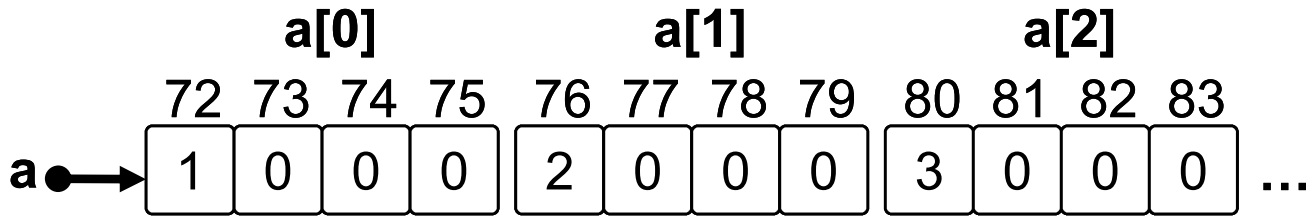
- Khái niệm con trỏ.
- Sử dụng con trỏ.
- **Con trỏ vs. mảng.**

Con trỏ vs. mảng

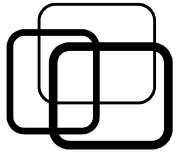


- Mảng trong C:
  - Là một con trỏ.
  - Giữ địa chỉ phần tử đầu tiên.

```
int main()
{
        int  a[ 10 ] = { 1, 2, 3 };
        printf("%d\n", a);
        printf("%d\n", &a[0]);      // a == &a[0].
}
```



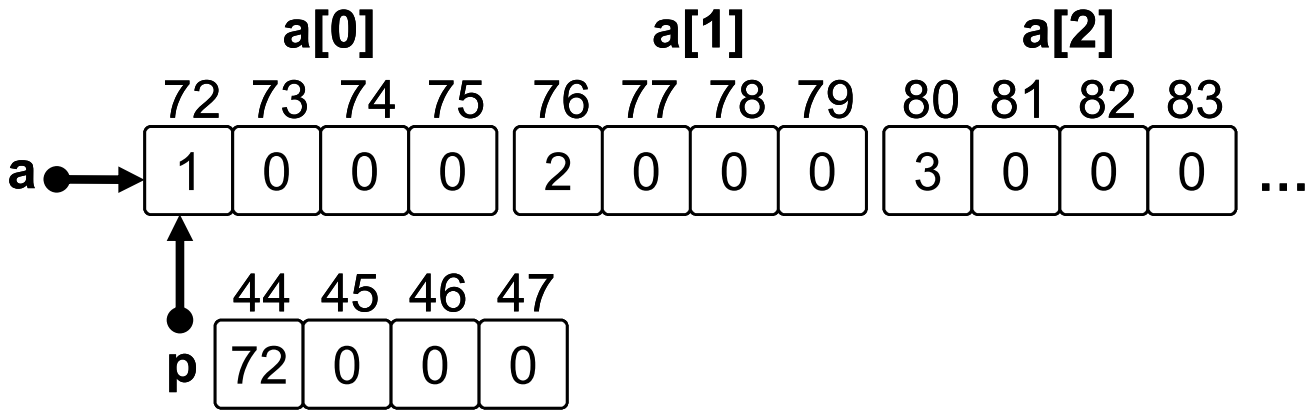
Con trỏ vs. mảng



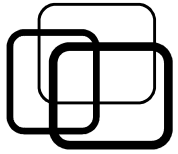
- Con trỏ đến phần tử mảng:
  - Truy xuất mảng gián tiếp.
  - Xét đoạn chương trình sau:

```
int  a[100] = { 1, 2, 3 };
int  *p = a;              // p = &a[0]
*p = *p + 1;
printf("%d\n", *p);
```

  - Cơ chế hoạt động:

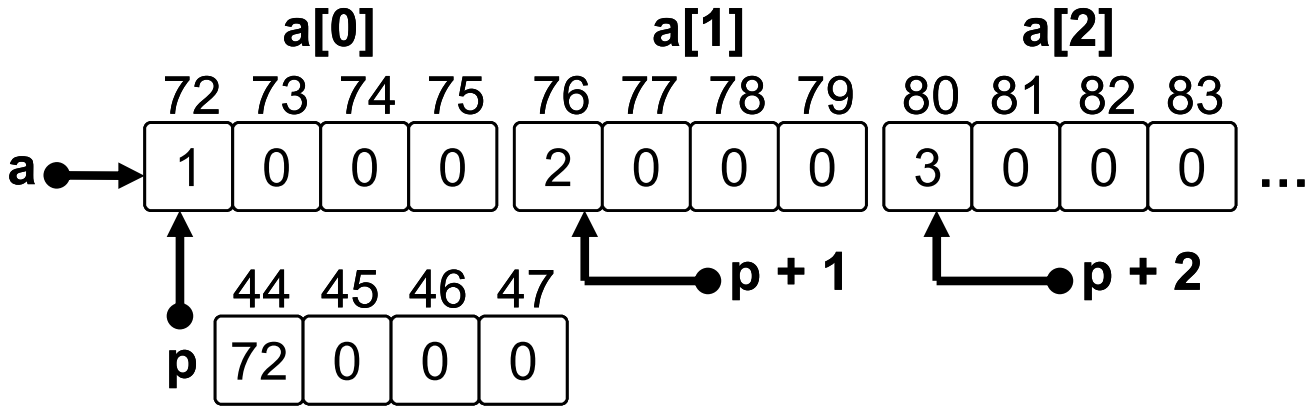


Con trỏ vs. mảng

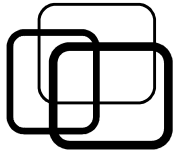


- Phép toán tăng, giảm con trỏ:
  - Giá trị con trỏ tăng giảm theo kích thước kiểu dữ liệu.
  - Công thức:
    - <Con trỏ> +/- k = <Địa chỉ> +/- k * sizeof(<Kiểu dữ liệu>).

```
int  a[100] = { 1, 2, 3 };
int  *p = a;                          // p = &a[0]
printf("%d\n", *(p + 1) );
printf("%d\n", *(p + 2) );
```



Con trỏ vs. mảng

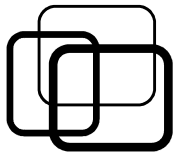


- Toán tử [ ]:
  - Truy xuất nội dung vùng nhớ con trỏ giữ địa chỉ.
  - Công thức:

    <Con trỏ>[<Chỉ số>] ~ * (<Con trỏ> + <Chỉ số>)

```
int  a[100] = { 1, 2, 3 };
int  *p = a;

a[2] = 5;
*(a + 2) = 5;
*(p + 2) = 5;
p[2] = 5;
```

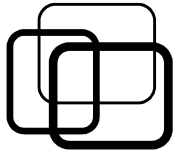
Con trỏ vs. mảng



- Truyền tham số mảng:
  - Không phải truyền tất cả mảng.
  - Chỉ truyền địa chỉ phần tử đầu tiên.
  - ➔ Truyền con trỏ đến phần tử đầu tiên.

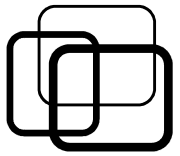
```
void xuatMang(int a[ ], int n) {    // Tương tự int *a
        for (int i = 0; i < n; i++)
                printf("%d ", *(a++) );   // Tương tự a[ i ]…
}

int main() {
        int  a[100];
        xuatMang(a, 100);
        for (int i = 0; i < n; i++)
                printf("%d ", *(a++) );   // Sai, vì sao?
}
```

Tóm tắt



- Khái niệm con trỏ:
  - Biến có kiểu là địa chỉ.
- Sử dụng con trỏ:
  - Khai báo: <Kiểu dữ liệu> *.
  - Khởi tạo: toán tử & lấy địa chỉ biến.
  - Toán tử *: truy xuất nội dung vùng nhớ.
- Con trỏ vs. mảng:
  - Mảng trong C là một con trỏ.
  - Con trỏ có thể "trỏ" đến vùng nhớ mảng.
  - Toán tử [ ]: truy nội dung vùng nhớ mảng.

Bài tập



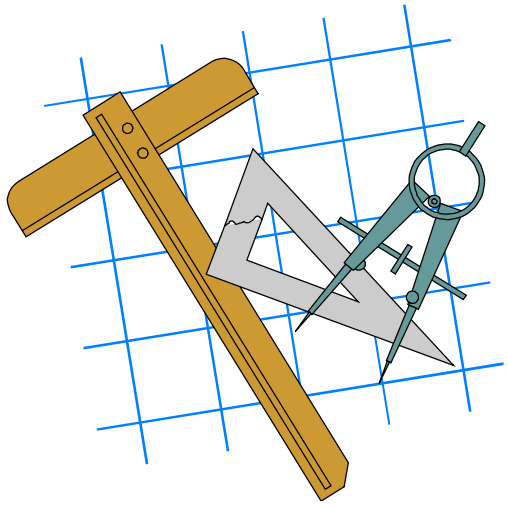
Bài tập 2.1:

Cho đoạn chương trình sau:

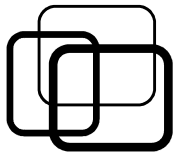
```
int main()
{
        int     *x, y = 2;
        float  *z = &y;

        *x = *z + y;
        printf("%d", y);
}
```

a) Đoạn chương trình trên có lỗi gì?

b) Hãy sửa lại cho đúng.



Bài tập

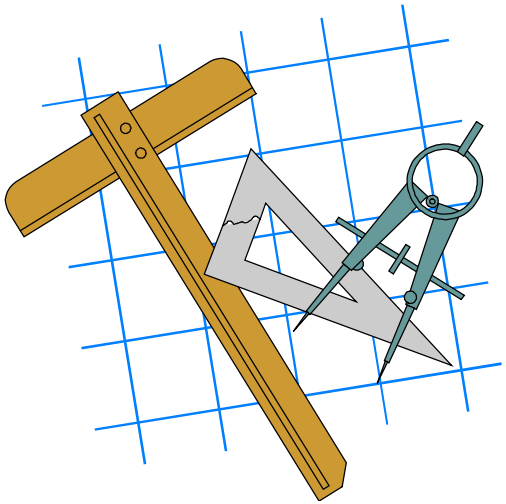


Bài tập 2.2:

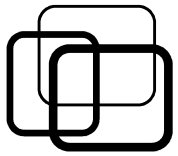
Cho đoạn chương trình sau:

```
int main()
{
        double   m[100];
        double   *p1, *p2;

        p1 = m;
        p2 = &m[6];
}
```

Hãy cho biết ô nhớ mà p1 và p2 trỏ đến cách nhau bao nhiêu byte?
(quy ước: ô nhớ 1 cách ô nhớ 0 là 1 byte)



Bài tập



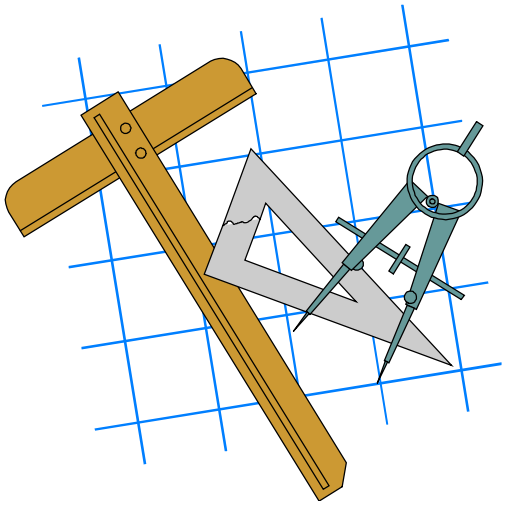
- **Bài tập 2.3:**

  Hãy giải thích sự khác nhau giữa 3 hàm sau:

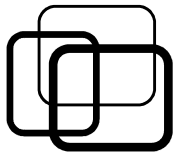
```
void swap1( int x, int y )
{
        int temp = x;
        x = y;
        y = temp;
}
```

```
void swap2( int &x, int &y )
{
        int temp = x;
        x = y;
        y = temp;
}
```

```
void swap3( int *x, int *y )
{
        int temp = *x;
        *x = *y;
        *y = temp;
}
```



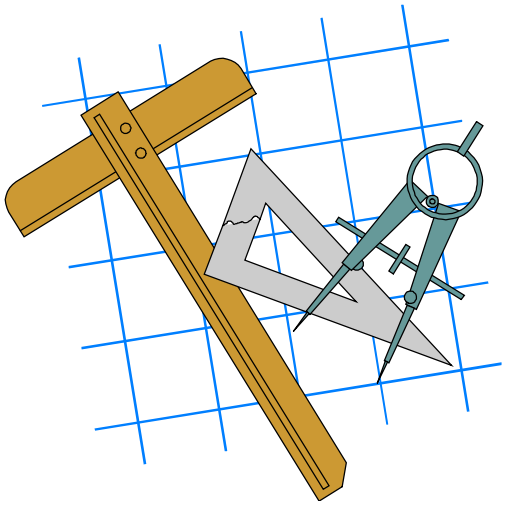
Bài tập



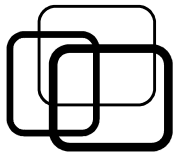
Bài tập 2.4:

Hãy giải thích kết quả xuất ra màn hình của đoạn chương trình sau:

```
#include <stdio.h>

int main()
{
        int     x = 1023;
        char  *p = (char *)&x;

        printf("%d %d %d %d\n", p[0], p[1], p[2], p[3]);
}
```



Bài tập



Bài tập 2.5:

Viết chương trình C sử dụng con trỏ để thực hiện:

a) Nhập mảng N phân số từ bàn phím.

b) Trích ra mảng các phân số âm.

c) Xuất kết quả ra màn hình.