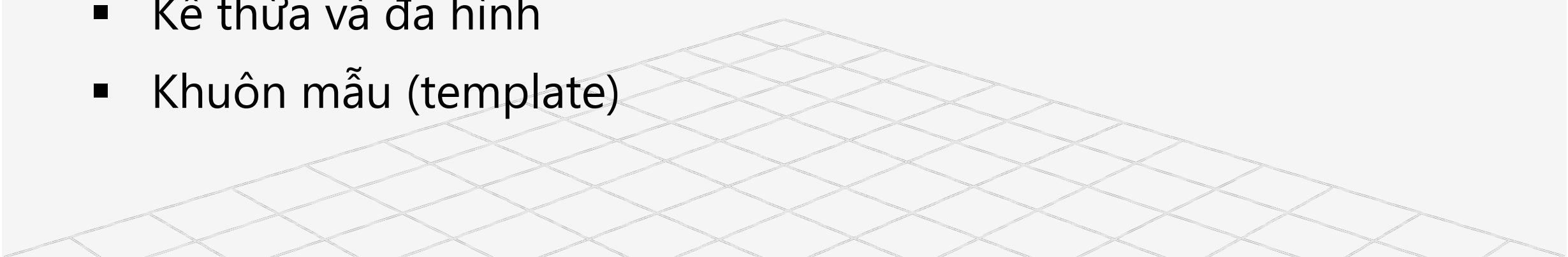


# PHẦN 2: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG (OBJECT-ORIENTED PROGRAMMING)

# NỘI DUNG

---

2

- Lịch sử các phương pháp lập trình
  - Khai báo và sử dụng lớp
  - Hàm tạo, Hàm hủy
  - Quan hệ bạn bè: hàm bạn
  - Định nghĩa toán tử trong lớp
  - Kế thừa và đa hình
  - Khuôn mẫu (template)
- 
- A decorative grid pattern consisting of thin, light gray lines forming a perspective grid that recedes towards the bottom right corner of the slide.

# HÀM TẠO (CONSTRUCTOR)

---

- Hàm tạo là một hàm thành phần đặc biệt của lớp
  - Được tự động gọi khi khai báo đối tượng
  - Có tên trùng với tên của lớp và không có kiểu
  - Được dùng để gán các giá trị khởi đầu cho các thành phần dữ liệu.
  - C++ có sẵn hàm tạo ngầm định nhưng hàm này không làm gì cả.
  - Khi định nghĩa hàm tạo tường minh thì lớp sử dụng hàm tạo tường minh.
  - Có thể khai báo chồng các hàm tạo.

# HÀM TẠO (CONSTRUCTOR)

---

- Bổ sung các hàm tạo cho lớp DIEM

```
class DIEM {  
    int x, y;  
    public:  
        DIEM();  
        DIEM(int xo, int yo);  
        void Nhap(int x1, int y1);  
        float KC();  
        void Dichuyen(int dx, int dy);  
        void Hienthi();  
};
```

- **Chú ý**

- Hàm tạo phải để dưới nhãn là public.
- Có thể có nhiều hàm tạo trong 1 lớp.
- Khi có ít nhất 1 hàm tạo thì lớp không sử dụng hàm tạo mặc định nữa

# VÍ DỤ LỚP CRECTANGLE

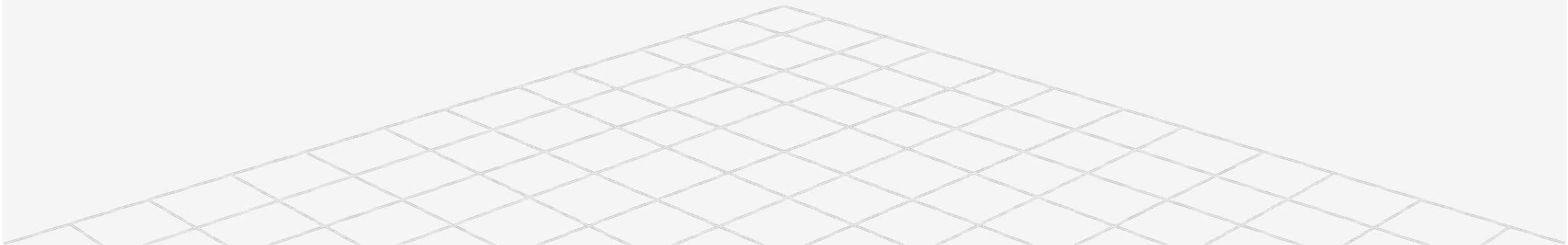
```
1 // example: class constructor
2 #include <iostream>
3 using namespace std;
4
5 class CRectangle {
6     int width, height;
7 public:
8     CRectangle (int,int);
9     int area () {return (width*height);}
10 };
11
12 CRectangle::CRectangle (int a, int b) {
13     width = a;
14     height = b;
15 }
16
17 int main () {
18     CRectangle rect (3,4);
19     CRectangle rectb (5,6);
20     cout << "rect area: " << rect.area() << endl;
21     cout << "rectb area: " << rectb.area() << endl;
22     return 0;
23 }
```

```
rect area: 12
rectb area: 30
```

# HÀM HỦY (DESTRUCTOR)

---

- Hàm hủy là một hàm thành phần của lớp
  - Có tên trùng với tên lớp, nhưng có thêm dấu ~ phía trước tên lớp.
  - Được gọi tự động khi đối tượng được giải phóng
  - Dùng để “dọn dẹp” vùng nhớ đã cấp phát cho các biến con trỏ
  - Chỉ có ý nghĩa khi lớp có dữ liệu con trỏ.
  - Mỗi lớp chỉ có duy nhất 1 hàm hủy



# HÀM HỦY (DESTRUCTOR)

---

- Bổ sung hàm huỷ cho lớp DIEM

```
class DIEM {  
    int x, y;  
public:  
    DIEM();  
    DIEM(int xo, int yo);  
    void Nhap(int x1, int y1);  
    float KC();  
    void Dichuyen(int dx, int dy);  
    void Hienthi();  
    ~DIEM(); // ham huy  
};
```

# HÀM TẠO SAO CHÉP (COPY CONSTRUCTOR)

---

- Khi thực hiện phép gán đối tượng  $A = B$  thì
  - Các thành phần dữ liệu thường được sao chép.
  - Các thành phần dữ liệu con trỏ chỉ sao chép địa chỉ con trỏ chứ không sao chép vùng dữ liệu mà nó trỏ tới.
- C++ cung cấp một hàm thành phần để điều khiển việc sao chép đối tượng trong lớp.
  - Có dạng `ten_lop(ten_lop &bien)`
  - Ví dụ `DIEM(DIEM &A)`
  - Được gọi tự động khi gán các đối tượng.
  - Chỉ có ý nghĩa khi lớp có dữ liệu con trỏ.



# VÍ DỤ 6-1

---

- Xây dựng lớp VECTOR
  - Dữ liệu
    - $n$  - số chiều của vector
    - $*v$  - chứa các giá trị thành phần của vector
  - Hàm thành phần
    - Hàm tạo
    - Hàm tạo sao chép
    - Hàm nhập giá trị của vector
    - Hàm tìm thành phần max của vector
    - Hàm hiển thị vector

## VÍ DỤ 6-1

```
1  #include <iostream>
2  #include <conio.h>
3  #include <math.h>
4  using namespace std;
5  // dinh nghia lop vector
6  class VECTOR{
7  public:
8      int n;
9      float *v;
10     VECTOR();
11     VECTOR(int n);
12     VECTOR(int n, float *v1);
13     VECTOR(VECTOR &a);
14     float Timmax();
15     void Hienthi();
16     ~VECTOR();
17 };
```

```
18 //trien khai cac nam thann phan
19 VECTOR::VECTOR() {
20     n = 0;
21     v = NULL;
22 }
23 VECTOR::VECTOR(int n) {
24     this->n = n;
25     v = new float [n];
26     for (int i = 0; i < n; i++) v[i] = 0;
27 }
28 VECTOR::VECTOR(int n, float *v1) {
29     this->n = n;
30     this->v = new float [n];
31     for (int i = 0; i < n; i++) {
32         v[i] = v1[i];
33     }
34 }
35 VECTOR::VECTOR(VECTOR &a) {
36     this->n = a.n ;
37     v = new float [n];
38     for (int i = 0; i < n; i++) {
39         v[i] = a.v[i];
40     }
41 }
```

## VÍ DỤ 6-1

```
42 float VECTOR::Timmax() {
43     float max = v[0];
44     for (int i = 1; i < n; i++) {
45         if (v[i] > max) max = v[i];
46     }
47     return max;
48 }
49 void VECTOR::Hienthi() {
50     cout << "So chieu: " << n << endl;
51     cout << "Cac thanh phan: " << endl;
52     cout << "    [" << v[0];
53     for (int i = 1; i < n; i++) cout << ", " << v[i];
54     cout << "]" << endl;
55 }
56 VECTOR::~~VECTOR() {
57     delete v;
58 }
```

```
59 // chương trình chính
60 int main() {
61     float v[] = {3,4,2,1,4,6};
62     VECTOR A(6, v), B;
63     B = A;
64     B.Hienthi();
65     cout << "\nPhan tu max nhat: " << B.Timmax();
66     cout << endl;
67     getch();
68     return 0;
69 }
```

# BÀI TẬP

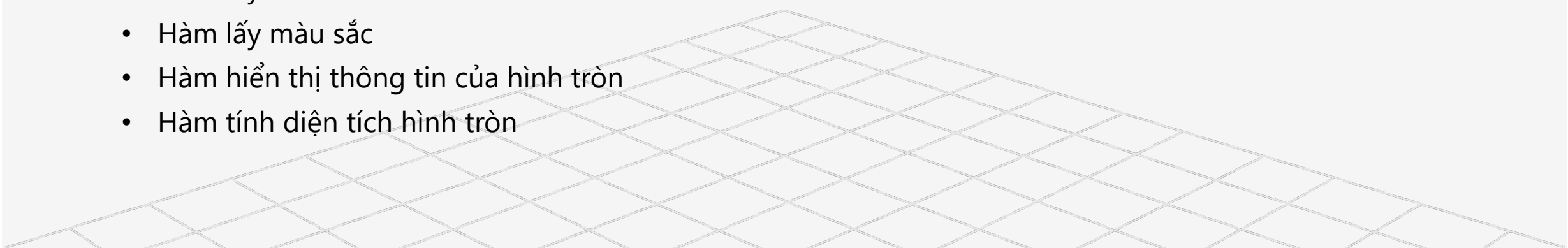
---

**Bài 6-1:** Hãy xây dựng lớp **Hình tròn** với các thông tin như sau:

Thành phần dữ liệu:

- Bán kính (double) ; Màu sắc (string)

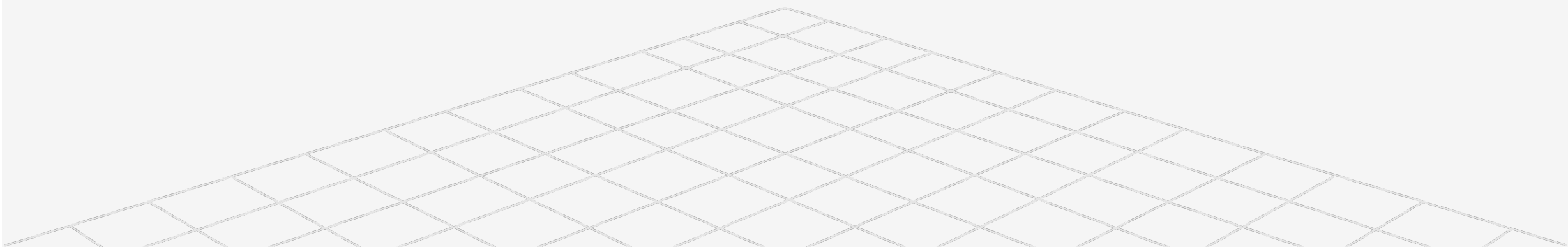
Hàm thành phần:

- Các hàm tạo
  - Hàm hủy
  - Hàm thiết lập bán kính
  - Hàm thiết lập màu sắc
  - Hàm lấy bán kính
  - Hàm lấy màu sắc
  - Hàm hiển thị thông tin của hình tròn
  - Hàm tính diện tích hình tròn
- 

# HÀM BẠN (FRIEND)

---

- Hàm tự do là bạn của 1 lớp
- Hàm thành phần của 1 lớp là bạn của lớp khác
- Một hàm là bạn của nhiều lớp
- Tất cả các hàm thành phần là bạn của 1 lớp



# HÀM BẠN (FRIEND)

---

4

**Đặt vấn đề: Với lớp DIEM, cần viết hàm so sánh hai điểm A và B với nhau.**



- Viết hàm compare có dạng int **compare(DIEM B)** là 1 hàm thành phần trong lớp để so sánh đối tượng hiện thời với đối tượng B.
- Viết hàm **compare(DIEM A, DIEM B)** là một hàm tự do nhưng được khai báo là bạn của lớp DIEM.
- Hàm tự do là hàm bạn của 1 lớp thì có quyền truy xuất các thuộc tính **private** và các thuộc tính **protected** của lớp.

# HÀM TỰ DO LÀ BẠN CỦA 1 LỚP

5

```
1 class DIEM {  
2   int x , y;  
3   Diem(int x, int y);  
4   void display();  
5   // ham tu do compare la ham ban cua lop  
6   friend int compare(DIEM A, DIEM B);  
7 };  
8 // ham tu do  
9 int compare(DIEM A, DIEM B) {  
10  // duoc phép truy xuất vào thành phần dữ liệu  
11  if ((A.x == B.x)&&(A.y == B.y)) return 1;  
12  else return 0;  
13 }
```

Khai báo hàm bạn

Triển khai hàm tự do,  
ngoài khai báo lớp



## VÍ DỤ 6-2

```
1  #include <iostream>
2  using namespace std;
3  class DIEM {
4  int x , y;
5  public:
6  // ham thanh phan cua lop
7  DIEM(int x, int y) {
8      this->x = x;
9      this->y = y;
10 }
11 void display() {
12     cout << x << " " << y << endl;
13 }
14 private:
15 // ham tu do compare la ham ban cua lop
16 friend int compare(DIEM A, DIEM B);
17 };
```

Chương trình minh  
họa việc khai báo và  
sử dụng hàm bạn  
đầy đủ



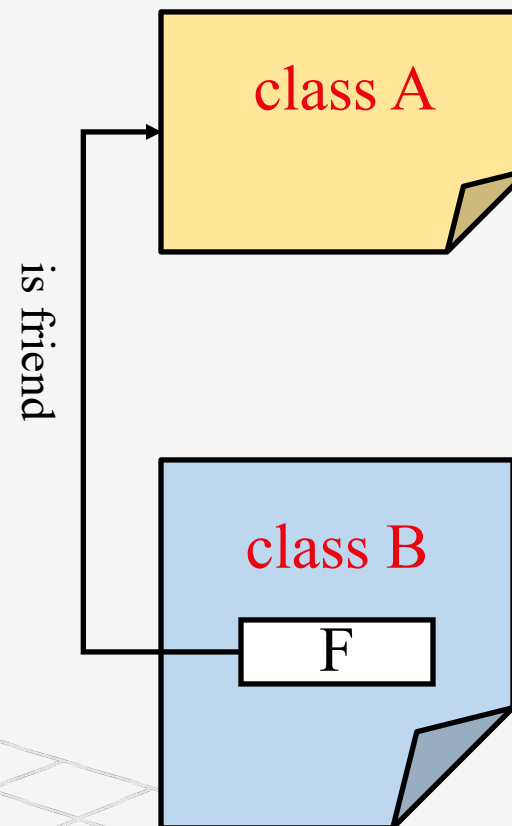
## VÍ DỤ 6-2

```
15 // ham tu do compare la ham ban cua lop
16 friend int compare(DIEM A, DIEM B);
17 };
18 // ham tu do
19 int compare(DIEM A, DIEM B) {
20     // duoc phép truy xuất vào thành phần dữ liệu
21     if ((A.x == B.x)&&(A.y == B.y)) return 1;
22     else return 0;
23 }
24
25 main() {
26     DIEM X(2, 3);
27     DIEM Y(2, 3);
28     if (compare(X, Y) == 1) cout << "Hai diem trung nhau !";
29     else cout << "Hai diem khac nhau !";
30     return 0;
31 }
```

# HÀM THÀNH PHẦN CỦA 1 LỚP LÀ BẠN CỦA 1 LỚP KHÁC

8

- Giả sử có hai lớp A và B.
- Trong lớp B có 1 hàm thành phần F cần truy xuất các thành phần **private** và **protected** của A.
- Để truy xuất được thì trong định nghĩa lớp A hàm F phải được khai báo là bạn của lớp A.



# HÀM THÀNH PHẦN CỦA 1 LỚP LÀ BẠN CỦA 1 LỚP KHÁC

---

## ■ Khai báo:

```
class A;  
class B {  
    // các thành phần dữ liệu của B  
    // các hàm thành phần của B  
    int F(int, A);  
};
```

```
class A {  
    // các thành phần dữ liệu của A  
    // các hàm thành phần của A  
    friend int B::F(int, A);  
};
```

## VÍ DỤ 6-3

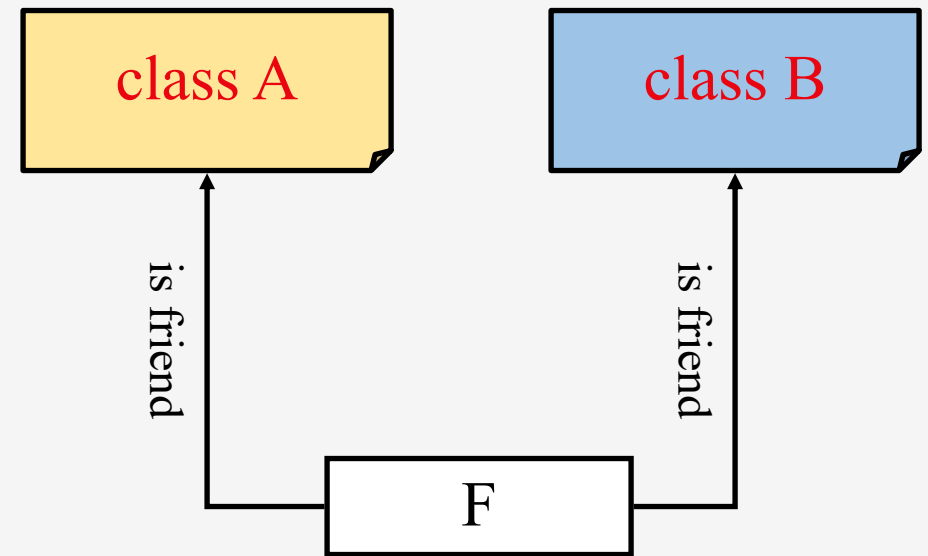
```
1 // ví dụ hàm của một lớp
2 // là bạn của lớp khác
3 #include <iostream>
4 using namespace std;
5 class A;
6 class B {
7     int x;
8     public:
9         B(int x0) { x = x0; }
10        int Cong(int , A);
11};
12 class A {
13     int y;
14     public:
15         A(int y0) { y = y0; }
16         friend int B::Cong(int , A);
17};
```

```
18 int B::Cong(int h, A a) {
19     return h*(x+a.y);
20 }
21 main() {
22     A a(10);
23     B b(20);
24     cout<<"Ket qua: "<<b.Cong(2,a);
25     return 0;
26 }
```

# HÀM BẠN CỦA NHIỀU LỚP

---

- Một hàm có thể là bạn của nhiều lớp khác nhau
- Trong khai báo lớp phải khai báo bạn bè cho hàm bạn
- Hàm bạn chỉ được khai báo 1 chiều trong khai báo lớp.



# HÀM BẠN CỦA NHIỀU LỚP

2

## ■ Khai báo

```
class B;  
class A {  
    // các khai báo của A  
    friend void F(A,B);  
};  
  
class B {  
    // các khai báo của B  
    friend void F(A,B);  
};
```

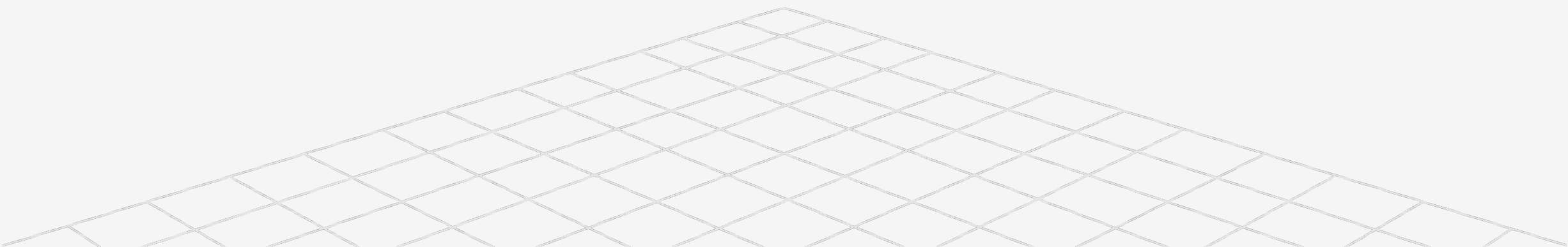
```
void F(A ..., B ....)  
{  
    // nội dung hàm F  
    // có thể truy xuất các thành  
    // phần private và protected  
    // của cả lớp A và B  
}
```

## VÍ DỤ 6-4

---

3

- Tự thiết kế 1 tình huống gồm 1 lớp  $A()$ , và một hàm tự do  $F()$ , khai báo hàm  $F()$  là bạn bè của  $A$ .
- Nêu tình huống và giải thích sự cần thiết phải khai báo  $F$  là bạn bè.



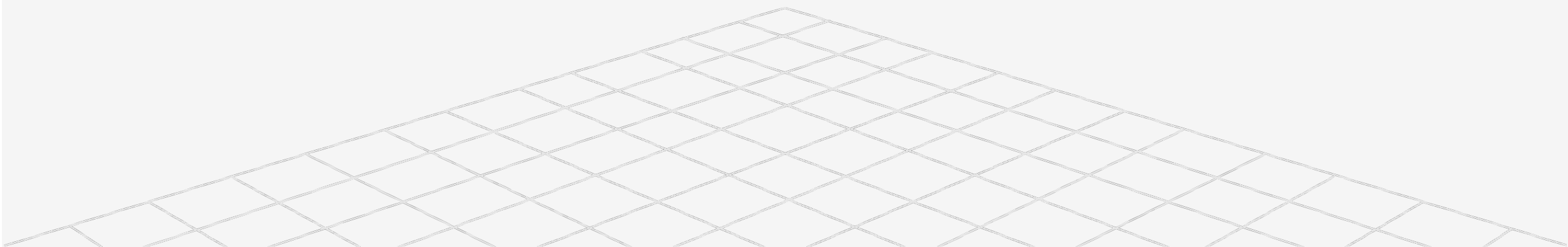
# BÀI TẬP

---

## Bài 6-2.

Bổ sung thêm thuộc tính tâm hình tròn vào lớp **Hình tròn** đã xây dựng trong bài 6-1. Tâm hình tròn là một điểm thuộc lớp DIEM.

Viết hàm bạn **vitrituongdoi(Hinhtron a, Hinhtron b)** để xét vị trí tương đối giữa hai hình tròn (giao nhau hoặc không giao nhau)





# BÀI TẬP

---

**Bài 6-3:** Hãy xây dựng lớp Time gồm các thành phần dữ liệu và các hàm thành phần như hình dưới đây. Chú ý hàm nextSecond() sẽ cộng thêm 1 giây vào thời gian hiện tại theo quy luật cộng của thời gian; hàm print() hiển thị thời gian dưới dạng: hh:mm:ss.

Time
-hour:int = 0 -minute:int = 0 -second:int = 0
+Time(h:int, m:int, s:int) +getHour():int +getMinute():int +getSecond():int +setHour(h:int):void +setMinute(m:int):void +setSecond(s:int):void +setTime(h:int, m:int, s:int) +print():void +nextSecond():void

---

QUESTIONS ?