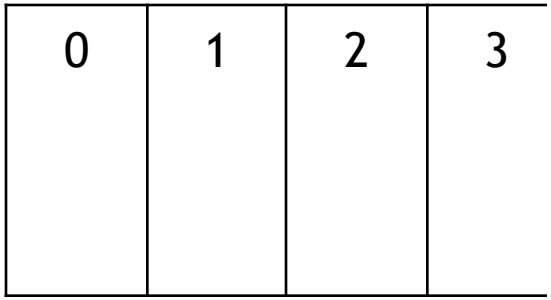
*CHƯƠNG 5
MẢNG MỘT CHIỀU

1

*

*Mảng thực chất là một biến được cấp phát bộ nhớ liên tục và bao gồm nhiều biến thành phần. *Các thành phần của mảng là tập hợp các biến có cùng kiểu

dữ liệu và cùng tên. Do đó để truy xuất các biến thành phần, ta dùng cơ chế chỉ mục.



*

* *int []a = new int[100]; //Mang so nguyen, 100 phan tu*
* *float []b = new float[50]; //Mang so thuc b, 50 phan tu*

*

Gán từng phần tử

int []a = {3, 6, 8, 1, 12};

	3	6	8	1	12
Vị trí	*0*	*1*	*2*	*3*	*4*

Gán toàn bộ phần tử có cùng giá trị

int []a = {3};

	3	3	3	3	3	3	3	3
Vị trí	*0*	*1*	*2*	*3*	*4*	*5*	*6*	

4

*

TênMảng [vị trí cần truy xuất]

static void Main(string []args)

{

int []a = {3, 6, 8, **11**, 12};

Console.Write(“Giá trị mảng tại vị trí 3 = “, a[3]);

}

Kết quả: Giá trị mảng tại vị trí 3 = 11

*Nhập

*

*Xuất (liệt kê)

*Tìm kiếm

*Đếm

*Sắp xếp

*Kiểm tra mảng thỏa điều kiện cho trước *Tách/ ghép mảng

*Chèn / xóa

*

```
static void NhapMang (int []a, int n)
{
for (int i = 0; i < n; i ++)
{
Console.Write("Nhap phan tu tai vi tri {0}: ", i);
a[i] = int.Parse(Console.ReadLine());
```

```
    }
    }
```



```
static void XuatMang (int []a, int n)
{
  for (int i = 0; i < n; i ++)
        Console.Write(a[i] + "\t");
}

static void Main (string []args)
{
   int []a;
```

```
    int n;
    Console.Write("Nhap kich thuoc mang: ");
    n = int.Parse(Console.ReadLine());
    a = new int [n];
    NhapMang (a, n);
    Console.WriteLine("Cac gia tri cua mang vua nhap: ");
    XuatMang (a,n);
}
```


*

*Sử dụng cho mảng

foreach (<KDL mảng> <tên biến> in <tên mảng>)

{

Khối lệnh;

}

Xét từng phần tử trong mảng

```
static void XuatMang(int[] a, int
n) {
      foreach(int x in a)
            Console.Write(x + "\t");
}
```


*

Sử dụng lớp Random

Phương thức	Miền giá trị phát sinh
Next()	[0…2,147,483,646]
Next(max)	[0…max -1]
Next(min, max)	[min..max -1]
NextDouble()	Số thực từ 0.0 đến 1.0

*

int songaunhien; double sothuc;

Random rd = new Random();
songaunhien = rd.Next();
Console.WriteLine(songaunhien)
; songaunhien = rd.Next(100);
Console.WriteLine(songaunhien)

```
; songaunhien = rd.Next(10,
100);
Console.WriteLine(songaunhien)
; sothuc = rd.NextDouble();
Console.WriteLine(sothuc);
```


*

```
static void PhatSinh(int[] a, int
n) {
      Random rd = new Random();
      for (int i = 0; i < n; i++)
      {
            a[i] = rd.Next(1, 100);
      }
}
```


*

Mẫu 1:

```
static void LietKeXXX(int []a, int
n) {
  for (int i = 0; i<n; i++)
        if (a[i] thỏa điều kiện)
              Xuất a[i];
}
```

Mẫu 2:

```
static void LietKeXXX(int []a, int n, int
x) {
  for (int i = 0; i<n; i++)
        if (a[i] thỏa điều kiện so với x)
              Xuất a[i];
```

```
}
```

Ví dụ 1: Liệt kê các phần tử có giá trị chẵn trong mảng

```
static void LietKeChan(int []a, int n)
{
 for (int i = 0; i<n; i++)
        if (a[i] %2 ==0)
                Console.Write(a[i] + "\t");
}
```

Ví dụ 2: Liệt kê các phần tử có giá trị lớn hơn x trong mảng

```
static void LietKeLonHonX(int []a, int n, int x) {
 for (int i = 0; i<n; i++)
        if (a[i] > x)
```

```
        Console.Write(a[i] + "\t");
}
```


* Ví dụ 3: Chương trình nhập vào mảng một chiều số nguyên a, kích thước n. In ra các phần tử có giá trị lớn hơn x có trong mảng

```
static void NhapMang(int []a, int n)
{
    for(int i=0; i<n; i++)
    {
        Console.Write("Nhap phan tu tai vi tri {0}: ", i);
        a[i] = int.Parse(Console.ReadLine());
    }
}
```

```
static void XuatMang(int []a, int n)
{
    for(int i=0; i<n; i++)
        Console.Write(a[i] + "\t“);
}
```



```
static void LietKeLonHonX(int []a, int n, int x)
{
    for (int i = 0; i<n; i++)
        if (a[i] > x)
            Console.Write(a[i] + “\t”);
}
static void Main(string []args)
{
    int []a;
    int n, x;
```

```
Console.Write("Nhap vao kich thuoc mang: “);
n = int.Parse(Console.ReadLine());
NhapMang(a, n);
Console.WriteLine("Cac phan tu cua mang:« );
XuatMang(a, n);
Console.Write("Nhap gia tri x: “);
x = int.Parse(Console.ReadLine());
Console.WriteLine("Cac phan tu co gia tri lon hon {0}: ", x);
LietKeLonHonX(a, n, x);
}
```


*

1. Xuất các phần tử là bội số của 5 trong

mảng 2. Xuất các phần tử là số nguyên tố trong mảng 3. Xuất các phần tử tại vị trí lẻ trong mảng

*

Mẫu 1:

```
static int DemXXX(int []a, int
n) {
        int d = 0;
    for (int i = 0; i<n; i++)
            if (a[i] thỏa điều kiện)
                d++;
```

```
    return d;
}
```


Mẫu 2:

```
static int DemXXX(int []a, int n, int
x) {
        int d = 0;
        for (int i = 0; i<n; i++)
                if (a[i] thỏa điều kiện so với x)
```

```
            d++;
        return d;
    }
```


Ví dụ 1: Đếm các phần tử có giá trị là số nguyên tố *int*

```
DemSNT(int []a, int n)
{
    if (k % i == 0)
    {
        d++;
```

```
bool LaSNT(int k)
 {
   int d = 0;
  for (int i = 1; i <= k; i++)
```

```
    }
  }
  return (d == 2);
}
```

```
    {
      int d = 0;
      for (int i = 0; i<n; i++)
      {
          if (LaSNT(a[i])==true) {
              d++;
          }
      }
      return d;
    }
```


Ví dụ 2: Đếm các phần tử có giá trị nhỏ hơn x có trong mảng

```
int DemNhoHonX(int []a, int n, int x)
{
```

```
    int d = 0;
    for (int i = 0; i<n; i++)
        if (a[i] < x)
            d++;
    return d;
}
```


* Ví dụ 3: Chương trình nhập vào mảng một chiều số nguyên a, kích thước n. Đếm số lượng các phần tử là số

nguyên tố có trong mảng

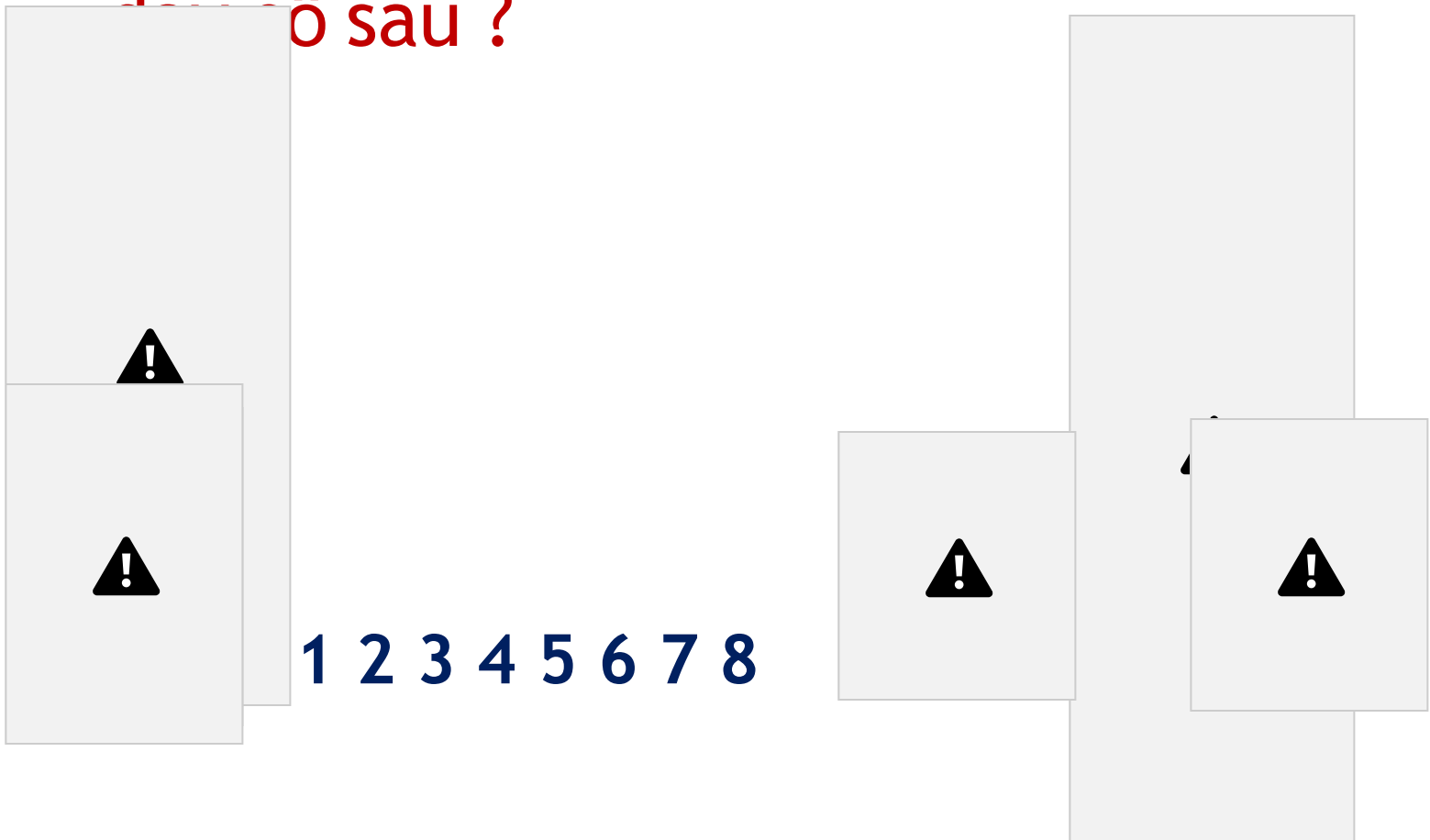
```
static void NhapMang(int []a, int n)
{
      for(int i=0; i<n; i++)
      {
            Console.Write("Nhap phan tu tai vi tri {0}: “, i);
            a[i] = int.Parse(Console.ReadLine());
      }
}
static void XuatMang(int []a, int n)
{
      for(int i=0; i<n; i++)
            Console.Write(a[i] + "\t“);
}
```

```
bool LaSNT(int k)
 {
        int d = 0;
   for (int i = 1; i <= k; i++)
       if (k % i == 0)
             d++;
   return (d == 2);
}
    int DemSNT(int []a, int
    n) {
      int d = 0;
      static void Main(string []args)
      {
          int []a;
          int n, kq;
   for (int i = 0; i<n; i++) if
      (LaSNT(a[i]) ==true)
             d++;
   return d;
}
```

```
Console.Write("Nhap vao kich thuoc mang:
“); n = int.Parse(Console.ReadLine());
NhapMang(a, n);
Console.WriteLine("Cac phan tu cua mang:");
XuatMang(a, n);
kq = DemSNT(a, n);
if(kq==0)
    Console.WriteLine("Khong co so nguyen to
    trong mang“);
else
    Console.WriteLine("So luong so nguyen to la: “ +
kq); }
```


*

Giả sử cần tìm vị trí phần tử nhỏ nhất trong dãy số sau ?



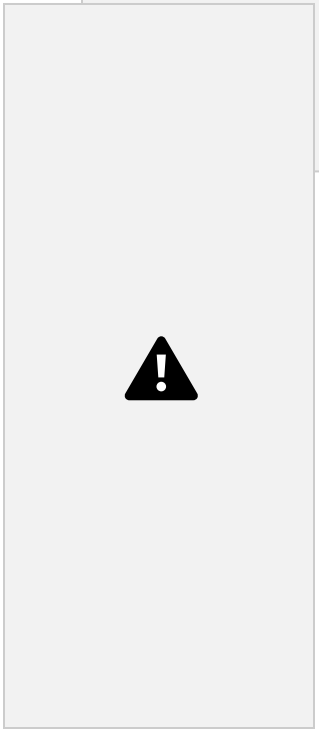
1 2 3 4 5 6 7 8

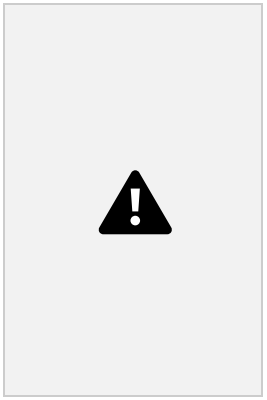
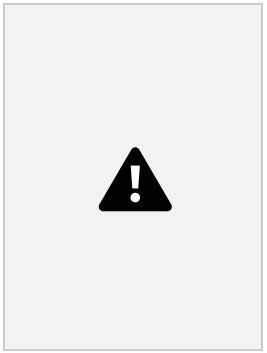
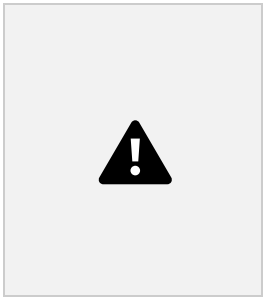
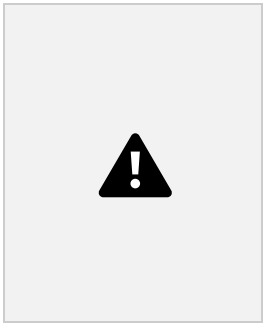
Bước 1: Giả sử vị trí phần tử nhỏ nhất là 1 (vtmin), phần tử này có giá trị 10

Bước 2: So sánh giá trị tại vtmin với tất cả giá trị tại vị trí còn lại (từ 2 đến 8), nếu có

phần tử nào nhỏ hơn phần tử tại vtmin thì

cập nhật lại vtmin

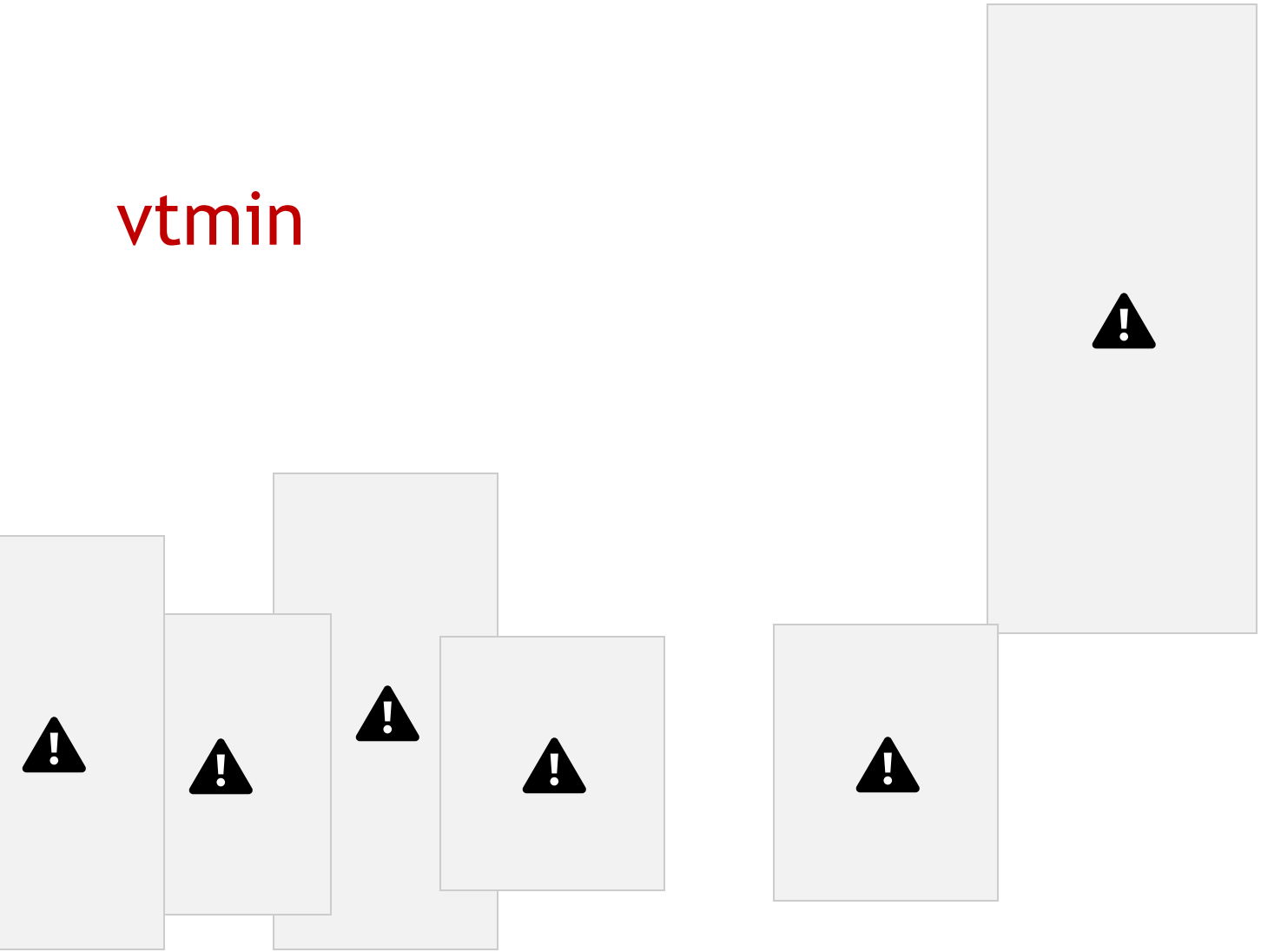
1 2 3 4 5 6 7 8 27

Bước 2: So sánh giá trị tại vtmin với tất cả giá trị tại vị trí còn lại (từ 2 đến 8), nếu có

phần tử nào nhỏ hơn phần tử tại vtmin thì

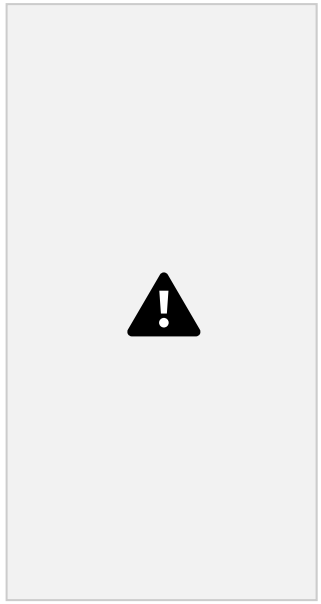
cập nhật lại

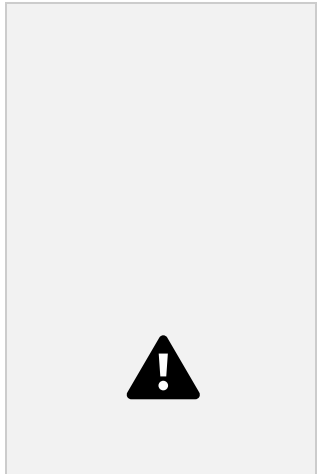
vtmin



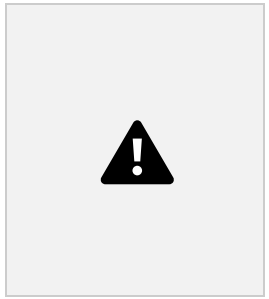
Bước 2: So sánh giá trị tại vtmin với tất cả giá trị tại vị trí còn lại (từ 2 đến 8), nếu có phần tử nào nhỏ hơn phần tử tại vtmin thì

cập nhật lại vtmin

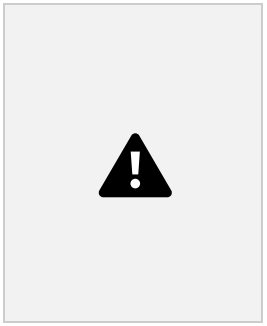




1 2



3 4

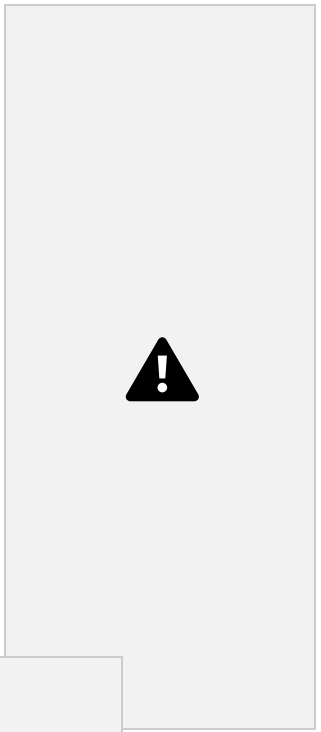


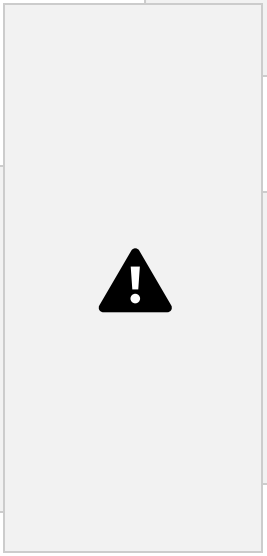
5 6 7 8

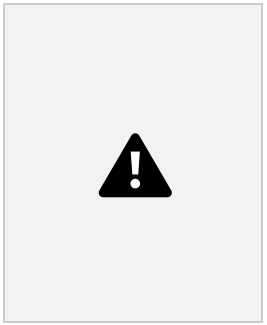
Bước 2: So sánh giá trị tại vtmin với tất cả giá trị tại vị trí còn lại (từ 2 đến 8), nếu có

phần tử nào nhỏ hơn phần tử tại vtmin thì

cập nhật lại vtmin





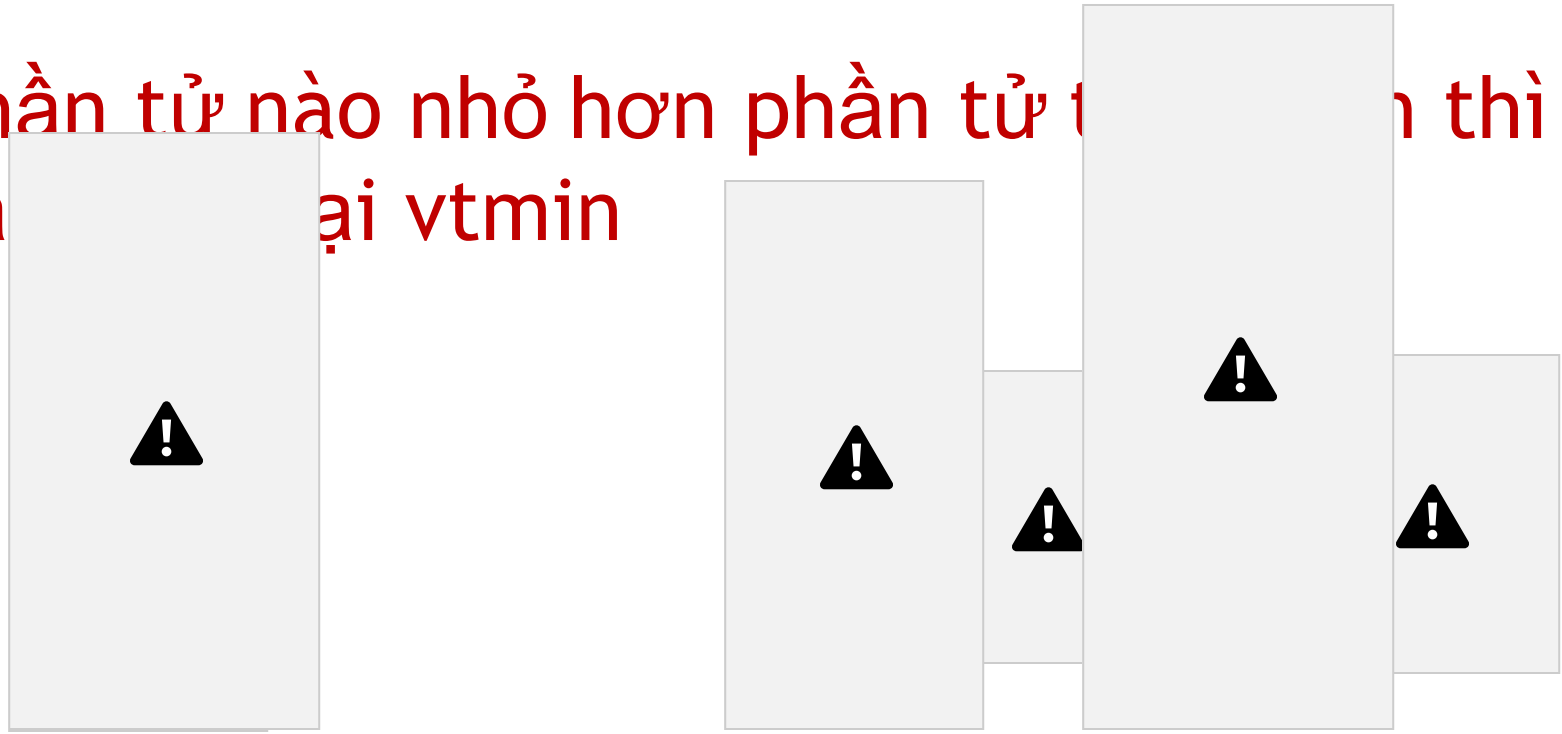
1 2 3 4 5 6 7 8 30

Bước 2: So sánh giá trị tại vtmin với tất cả

giá trị tại vị trí còn lại (từ 2 đến 8), nếu có

phần tử nào nhỏ hơn phần tử t n thì

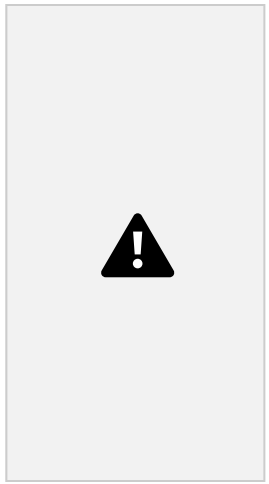
cậ ại vtmin

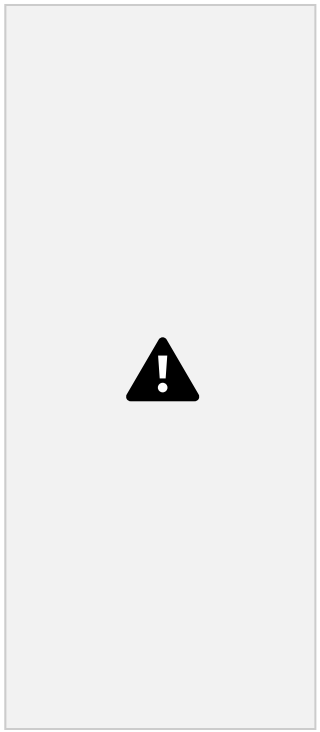


1 2 3 4 5 6 7 8

Bước 2: So sánh giá trị tại vtmin với tất cả giá trị tại vị trí còn lại (từ 2 đến 8), nếu có phần tử nào nhỏ hơn phần tử tại vtmin thì

cập nhật lại vtmin



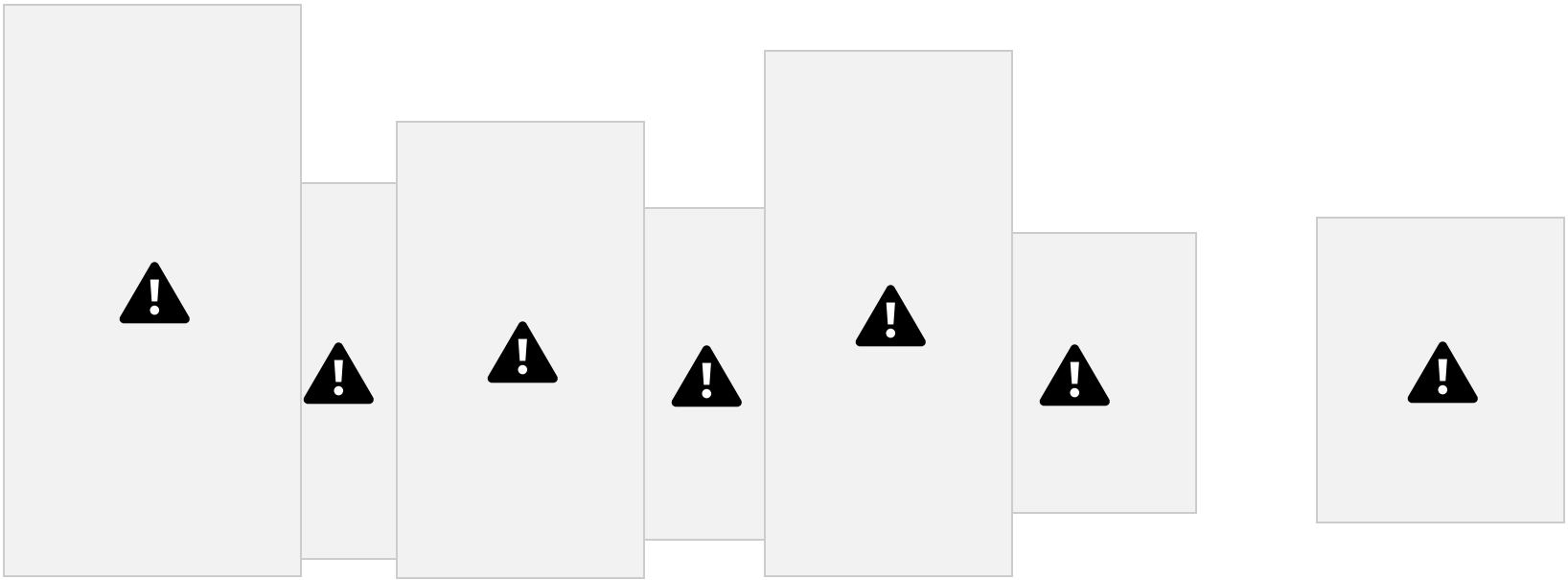





1 2 3 4 5 6
7 8

Bước 2: So sánh giá trị tại vtmin với tất cả giá trị tại vị trí còn lại (từ 2 đến 8), nếu có

phần tử nào nhỏ hơn phần tử tại vtmin thì
cập nhật lại vtmin



*

```
static int TimVTMin(int []a, int n)
{
int vtmin = 0;
for (int i = 1; i < n; i++)
{
if (a[i] < a[vtmin])
vtmin = i;
}
return vtmin;
```

```
}
```

34

*

```
static int TimMax(int []a, int n)
{
int max = a[0];
for (int i = 1; i < n; i++)
{
if (a[i] > max)
```

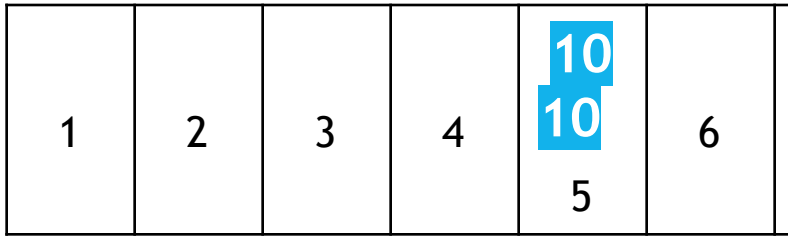
```
max = a[i];
}
return max;
}
```


*

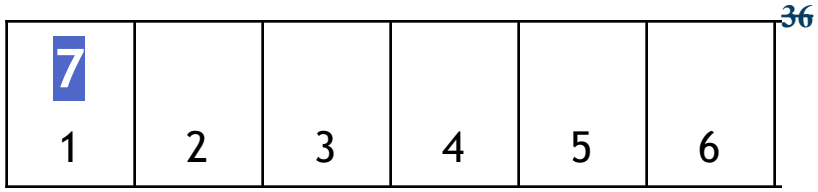
*Ý tưởng

Lần lượt so sánh **x** với phần tử thứ nhất, thứ hai, ... của mảng **a** cho đến khi gặp được phần tử cần tìm, hoặc đã tìm hết mảng mà không thấy **x**

*Minh họa tìm x =10



*Minh họa tìm x =25



*

(nếu x không xuất hiện trong mảng trả về -1)

```
static int TimVTX(int []a, int n, int x)
{
  for (int i = 0; i < n; i++)
    {
        if (a[i] == x)
            return i;
    }
   return -1;
```

}

*

*__TH1__:

kiểm tra tồn tại một phần tử trong mảng thỏa điều kiện nào đó cho trước tìm phần tử thỏa điều kiện để kết luận.

*__TH2__: **kiểm tra tất cả** các phần tử thỏa điều kiện nào đó cho trước tìm phần tử không thỏa điều kiện để kết luận mảng không thỏa điều kiện.

Mẫu TH1:

```
static bool KiemTraTonTaiXXX(int []a, int
n) {
        for (int i = 0; i<n; i++)
                if (a[i] thỏa điều kiện)
                        return true;
```

```
        return false;
}
```

Mẫu TH2:

```
static bool KiemTraXXX(int []a, int n)
{
        for (int i = 0; i<n; i++)
                if (a[i] không thỏa điều kiện)
                        return false;
        return true;
 }
```


Ví dụ 1: Kiểm tra xem mảng có tồn tại số lẻ không?

```
static bool KiemTraTonTaiLe(int []a, int n)
{
    for (int i = 0; i < n; i++)
    {
        if (a[i] % 2 != 0)
            return true;
    }
    return false;
}
```

<u>Ví dụ 2:</u> Kiểm tra xem mảng có toàn giá trị âm không? *(true: có/ false: không)*

```
static bool KiemTraToanAm(int []a, int
n) {
    for (int i = 0; i<n; i++)
     {
          if (a[i] >= 0)
               return false;
    }
```

return true;

}

*

Mẫu tính tổng:

static int

TongXXX(int []a, int n) {

int s = 0;

```
for (int i = 0; i<n; i++)
{
if (a[i] thỏa điều kiện)
s += a[i];
}
return s;
}
```


Mẫu tính trung bình:

```
static float TrungBinhXXX(int []a, int n) {
int s = 0;
int d = 0;
```

```
for (int i = 0; i<n; i++)
{
if (a[i] thỏa điều kiện)
{
s += giatri;
d ++;
}
}
if (d==0)
return 0;
return (float) s / d;
}
```


Ví dụ 1: Tính tổng các phần tử có giá trị lẻ trong

mảng *static int TongLe(int []a, int n)*

```
{
        int s = 0;
       for (int i = 0; i<n; i++)
        {
            if (a[i] %2!=0)
                s += a[i];
        }
        return s;
}
```

Ví dụ 2: Tính giá trị trung bình các phn tử có giá trị âm trong mảng

```
static float TrungBinhAm(int []a, int n)
{
long s = 0;
int d = 0;
for (int i = 0; i<n; i++)
{
if (a[i] < 0)
{
s += a[i];
d++;
}
}
```

```
if (d == 0)
return 0;
return (float)s / d;
}
```

45

Mẫu * SẮP XẾP phương thức sắp thứ tự tăng:

```
static void SapTang(int []a, int n)
{
for (int i = 0; i < n-1; i ++)
{
for(int j = i+1; j < n; j ++)
if (a[i] > a[j])
HoanVi(ref a[i], ref a[j]);
```

```
}
}
static void HoanVi(ref int a, ref int b)
{
int tam = a;
a = b;
b = tam;
}
```


* Cho mảng sau:

*

12	5	7	9	21	38
0	*1*	*2*	*3*	*4*	*5*

*Hãy trình bày từng bước chèn 111 vào vị trí 3 của mảng

12	5	7	9	21	38	

0	1	2	3	4	5	

*

*Hãy viết hàm chèn phần tử có giá trị x vào vị trí k cho trước trong mảng a kích thước n theo mẫu sau:

static void ChenX(int []a, ref int n, int x, int

k); 48

*

Hãy viết hàm chèn phần tử có giá trị x vào sau phần tử có giá trị nhỏ nhất có trong

mảng a, kích thước n (giả sử mảng không có giá trị trùng nhau)

* Cho mảng sau:
*

12	5	7	9	21	38
0	*1*	*2*	*3*	*4*	*5*

* Hãy trình bày từng bước xóa phần tử tại vị trí

3 trong mảng

12	5	7	9	21	38
0	1	2	3	4	5

*

*Hãy viết hàm xóa phần tử tại vị trí k cho trước trong mảng a kích thước n theo mẫu sau:

static ***void XoaTaiVTk(int []a, ref int n, int***

k); 51

*

Hãy viết hàm xóa phần tử x *(nếu có)* trong

mảng a, kích thước n (giả sử mảng không có giá trị trùng nhau)