

# **Analysis and Design of Algorithms**

## **Lecture 13**

# **Generating Method**

Lecturer: Nguyen Mau Uyen

uyennm@mta.edu.vn

# Nội dung

1. Lược đồ chung
2. Bài toán chuỗi 3 ký tự
3. Liệt kê tập con của tập  $N$  phần tử
4. Bài toán tập con  $K$  phần tử
5. Hoán vị tập  $N$  phần tử

# Nội dung

## 1. Lược đồ chung

2. Bài toán chuỗi 3 ký tự
3. Liệt kê tập con của tập  $N$  phần tử
4. Bài toán tập con  $K$  phần tử
5. Hoán vị tập  $N$  phần tử

# Bài toán tổ hợp

- Có  $n$  biến  $x_1, x_2, x_3, \dots, x_n$
- Mỗi biến  $x_i$  có thể mang trị thuộc về 1 tập hợp  $P_i$   
→ Miền của bài toán là tập tích

$$P_1 \times P_2 \times P_3 \times \dots \times P_n$$

- Phép gán trị (assignment): Là một bộ trị

$$a_1, a_2, a_3, \dots, a_n$$

$$\text{Trong đó } a_1 \rightarrow a_i \in P_i$$

- Một lời giải của bài toán là 1 phép gán trị.
- Một phép gán trị được gọi là một cấu hình.

# Ví dụ 1

- **Ví dụ:** Có 3 nhân viên bảo vệ làm 3 ca sáng, chiều tối. Trong 1 ca chỉ có 1 bảo vệ. Hỏi các cách bố trí các bảo vệ?
- **Mã hóa bài toán:**  
 $\{x, y, z\}$  là tập biến có thứ tự mô tả cho 3 ca :sáng, chiều, tối theo thứ tự.  
Miền trị của 3 biến là  $\{a, b, c\}$  mô tả cho 3 bảo vệ.

## Các phép gán

<u>x</u>	<u>y</u>	<u>z</u>
a	b	c
a	c	b
b	a	c
b	c	a
c	a	b
c	b	a

✳Số lời giải là số hoán vị của tập hợp 3 phần tử này:  
 $3*2*1 = 3! = 6.$

## Ví dụ 2

- Ví dụ:** Tìm số chuỗi có độ dài 3 ký tự xyz với

$x \in \{a, b, c\},$

$y \in \{d, e\},$

$z \in \{m, n, t\}$

- Nhận xét: 3 biến có 3 miền trị khác nhau

<u>x</u>	<u>y</u>	<u>z</u>
a	d	m
a	d	n
a	d	t
a	e	m
a	e	n
a	e	t
b	d	m
b	d	n
b	d	t
b	e	m
b	e	n
b	e	t
c	d	m
c	d	n
c	d	t
c	e	m
c	e	n
c	e	t

**Số phép gán:**  
 $3 * 2 * 3 = 18$

**Tích của các  
số phần tử  
của các  
miền trị**

Độ phức tạp:  $n^m$  với  
 $n$ : số phần tử trung bình  
của mỗi miền trị,  
 $m$ : là số miền trị

# Bài toán tổ hợp

***Bài toán tổ hợp  
có độ phức tạp là  $n!$  hoặc  $n^m$***

**Làm thế nào tạo ra  
các phép gán trị ?  
→ Phương pháp sinh.**

# Lược đồ chung

- Phương pháp sinh: Từ dữ liệu ban đầu, sinh ra dữ liệu kế tiếp cho đến khi kết thúc.
- Dùng để giải quyết bài toán liệt kê của lý thuyết tổ hợp.
- Điều kiện của thuật toán sinh:
  - (1) Có thể xác định 1 thứ tự tập các cấu hình của tổ hợp (thứ tự của các phép gán trị, thường dùng thứ tự từ điển).
  - (2) Có một cấu hình cuối (điều kiện kết thúc của giải thuật).
  - (3) Có một cách để suy ra được cấu hình kế tiếp.



# Thứ tự từ điển

- $S1 = \text{"1234589"}$
- $S2 = \text{"1235789"}$
- $S1 < S2$  nếu có 1 vị trí  $i$  tại đó  
 $S1[i] < S2[i]$

Thứ tự từ điển ngược (ngược lại với thứ tự từ điển)

# Ví dụ

**Bài toán:** *Tìm số chuỗi có độ dài 3 ký tự xyz với  $x \in \{a,b,c\}$ ,  $y \in \{d,e\}$ ,  $z \in \{m,n,t\}$*

Cấu hình ban đầu: trị đầu tiên của mỗi miền trị

Cách sinh: Lấy trị kết tiếp của mỗi miền trị theo cơ chế vòng tròn

Cấu hình cuối: trị cuối cùng của mỗi miền trị

x	y	z
a	d	m
a	d	n
a	d	t
a	e	m
a	e	n
a	e	t
b	d	m
b	d	n
b	d	t
b	e	m
b	e	n
b	e	t
c	d	m
c	d	n
c	d	t
c	e	m
c	e	n
c	e	t

Dùng thứ tự từ điển để so sánh các phép gán trị.

Ví dụ:

$adm < adn$

# Lược đồ chung

**Procedure Generate**

**Begin**

**c = InitialConfigure; //cấu hình ban đầu**

**Process (c); // xử lý cấu hình đang có**

**if c=LastConfigure then Stop:=true**

**else stop := false;**

**while (not stop) do**

**Begin**

**//Sinh cấu hình kế tiếp từ cấu hình đang có**

**c=getNextConfigure(c);**

**Process (c); // xử lý cấu hình này**

**if c= LastConfigure then stop = true;**

**End;**

**End;**

# Nội dung

1. Lược đồ chung
- 2. Bài toán chuỗi 3 ký tự**
3. Liệt kê tập con của tập  $N$  phần tử
4. Bài toán tập con  $K$  phần tử
5. Hoán vị tập  $N$  phần tử

# Bài toán

**Bài toán:** *Tìm số  
chuỗi có độ dài 3 ký  
tự xyz với  
 $x \in \{a, b, c\},$   
 $y \in \{d, e\},$   
 $z \in \{m, n, t\}$*

# Cài đặt

```
[ ]===== 3KYTU.CPP =====
#include <conio.h>
#include <string.h>
// Khai bao cau truc mien tri va cac thuat toan tren mien tri
struct DOMAIN
{ char values[27]; // tap tri
  int n;          // so phan tu
  int cur;        // vi tri hien hanh
  int serviced;   // da lay tri chua?
};
void Init (DOMAIN &D, char* S)
{ strcpy(D.values,S);
  D.n= strlen(S);
  D.cur=0;
  D.serviced=0;
}
char getValue (DOMAIN &D) // Lay 1 tri trong domain
{ char x= D.values[D.cur++];
  if (D.cur==D.n) D.cur=0; // cap nhat vi tri va trang thai
  if (D.serviced==0) D.serviced=1;
  return x;
}
int isCircular(DOMAIN D) // kiem tra da het 1 vong tri chua?
{ return D.cur==0 && D.serviced==1;
}
```

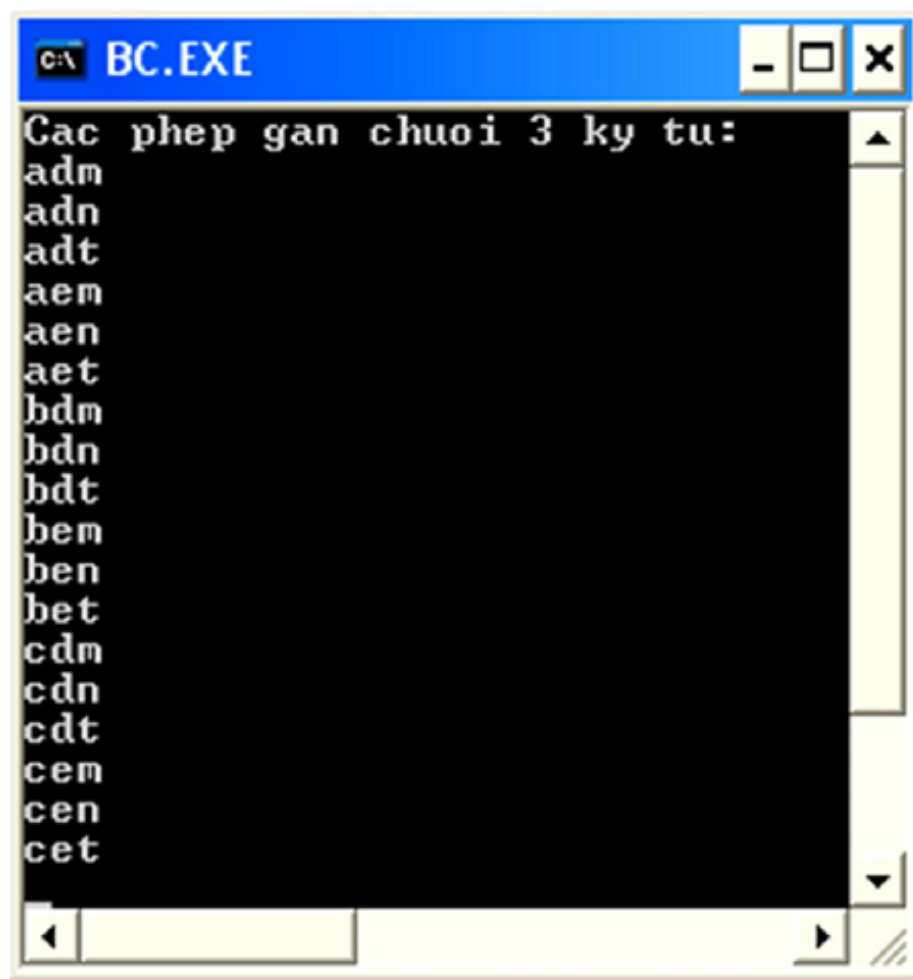
```

[1]===== 3KYTU.CPP =====
void Generate()
{
    char vars[4]; // 3 bien la 3 ky tu
    DOMAIN D0,D1,D2; // 3 domain cho 3 ky tu
    Init (D0,"abc"); // Khoi tao 3 domain
    Init (D1,"de");
    Init (D2, "mnt");
    // Lay cau hinh ban dau
    vars[0]= getValue(D0);
    vars[1]= getValue(D1);
    vars[2]= getValue(D2);
    vars[3]=0; // khoa chuoi
    // Xu ly cau hinh ban dau
    puts(vars);
    // Kiem tra ngung
    int Stop= isCircular(D0) && isCircular(D1) && isCircular(D2);
    while(!Stop)
    {
        // Lay cau hinh ke tiep
        if (isCircular(D1) && isCircular(D2)) vars[0]= getValue(D0);
        if (isCircular(D2)) vars[1]= getValue(D1);
        vars[2]= getValue(D2);
        // Xu ly cau hinh
        puts(vars);
        // Kiem tra ngung
        Stop= isCircular(D0) && isCircular(D1) && isCircular(D2);
    }
}

void main()
{
    clrscr();
    puts("Cac phep gan chuoi 3 ky tu:");
    Generate();
    getch();
}

```

# Minh họa



```
C:\> BC.EXE
Gac phép gan chuo i 3 ky tu:
adm
adn
adt
aem
aen
aet
bdm
bdn
bdt
bem
ben
bet
cdm
cdn
cdt
cem
cen
cet
```



# Nội dung

1. Lược đồ chung
2. Bài toán chuỗi 3 ký tự
- 3. Liệt kê tập con của tập  $N$  phần tử**
4. Bài toán tập con  $K$  phần tử
5. Hoán vị tập  $N$  phần tử

# Bài toán

- Mã hóa tập biến: Tập biến gồm  $n$  biến ký tự theo thứ tự các phần tử  $\rightarrow$  mảng  $n$  ký tự.
- Miền trị của mỗi biến  $\{ '0', '1' \}$ . ' $0$ ' mô tả cho tình huống phần tử này **không có** trong tập con, ' $1$ ': mô tả cho tình huống phần tử này **có mặt** trong tập con.
- Với tập cha là 4 phần tử  $X = \{ a, b, c, d \}$ , có thể dùng mảng "0111" mô tả cho tập con  $\{ b, c, d \}$ .  
 $\rightarrow$  Mỗi tập con được biểu diễn là một chuỗi (xâu) nhị phân.
- Trạng thái khởi tạo: "0000" mang ý nghĩa tập trống.
- Trạng thái kết thúc: "1111" mang ý nghĩa là tập cha.

# Ví dụ

- Với tập cha gồm 4 phần tử, có  $2^4$  tập con b với các biểu diễn:

<u>vars</u>	<u>p(b)</u>	<u>vars</u>	<u>p(b)</u>
0000	0	1000	8
0001	1	1001	9
0010	2	1010	10
0011	3	1011	11
0100	4	1100	12
0101	5	1101	13
0110	6	1110	14
0111	7	1111	15

# Cộng 1 đơn vị

0000

0001

0001

0010

0011

0100

0111

1000

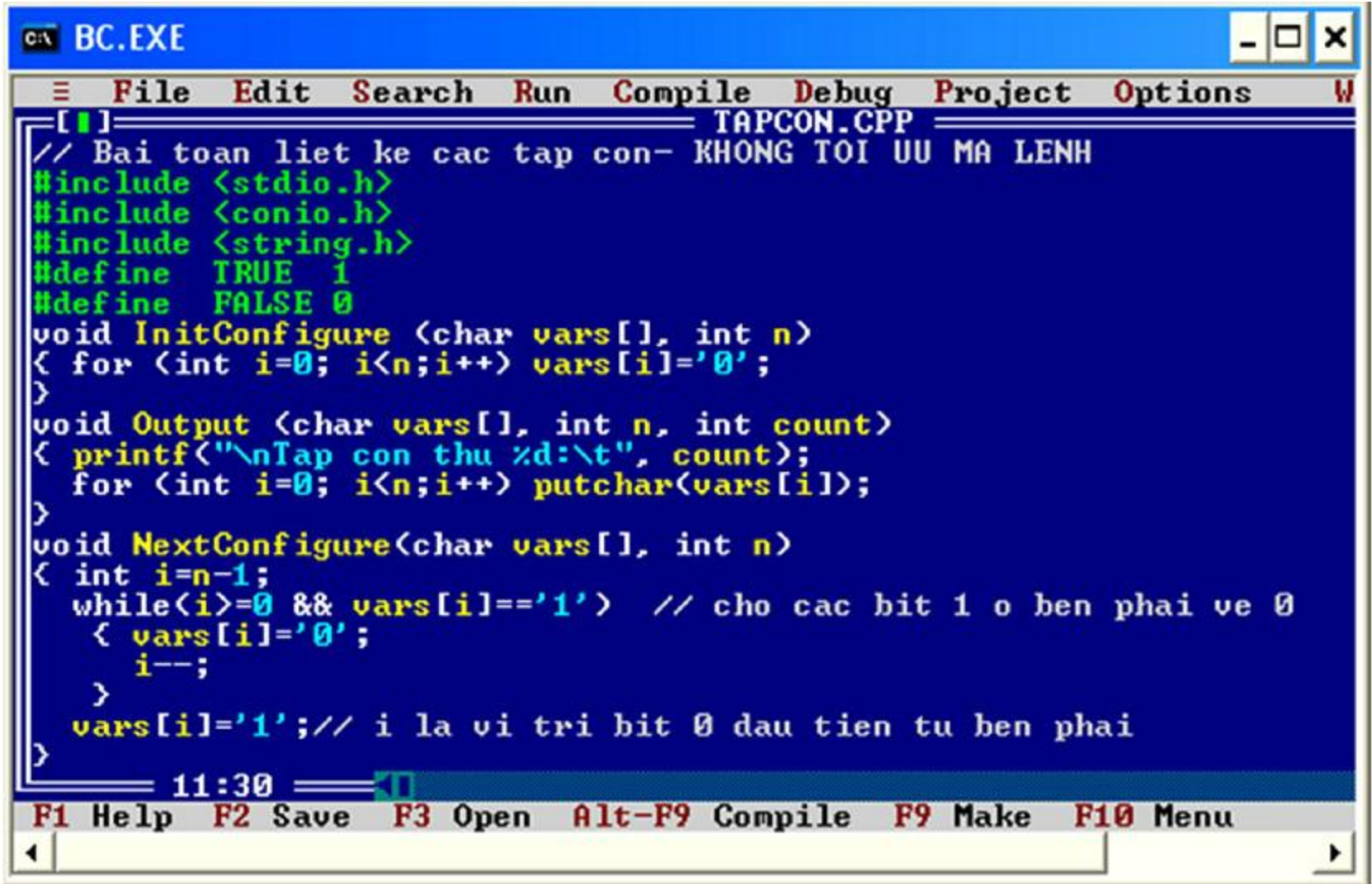
- Gọi **i** : vị trí bit **0** đầu tiên từ bên phải.
- Cho các bit **1** bên phải vị trí **i** thành **0**
- Cho bit **i** mang trị **1**

**i = n-1;**

**while (i >= 0 && vars[i] == '1') vars[i--] = '0';**

**vars[i] = '1';**

# Cài đặt



```
BC.EXE
File Edit Search Run Compile Debug Project Options
TAPCON.CPP
// Bai toan liet ke cac tap con- KHONG TOI UU MA LENH
#include <stdio.h>
#include <conio.h>
#include <string.h>
#define TRUE 1
#define FALSE 0
void InitConfigure (char vars[], int n)
{ for (int i=0; i<n;i++) vars[i]='0';
}
void Output (char vars[], int n, int count)
{ printf("\nTap con thu %d:\t", count);
  for (int i=0; i<n;i++) putchar(vars[i]);
}
void NextConfigure(char vars[], int n)
{ int i=n-1;
  while(i>=0 && vars[i]=='1') // cho cac bit 1 o ben phai ve 0
  { vars[i]='0';
    i--;
  }
  vars[i]='1'; // i la vi tri bit 0 dau tien tu ben phai
}
11:30
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu
```

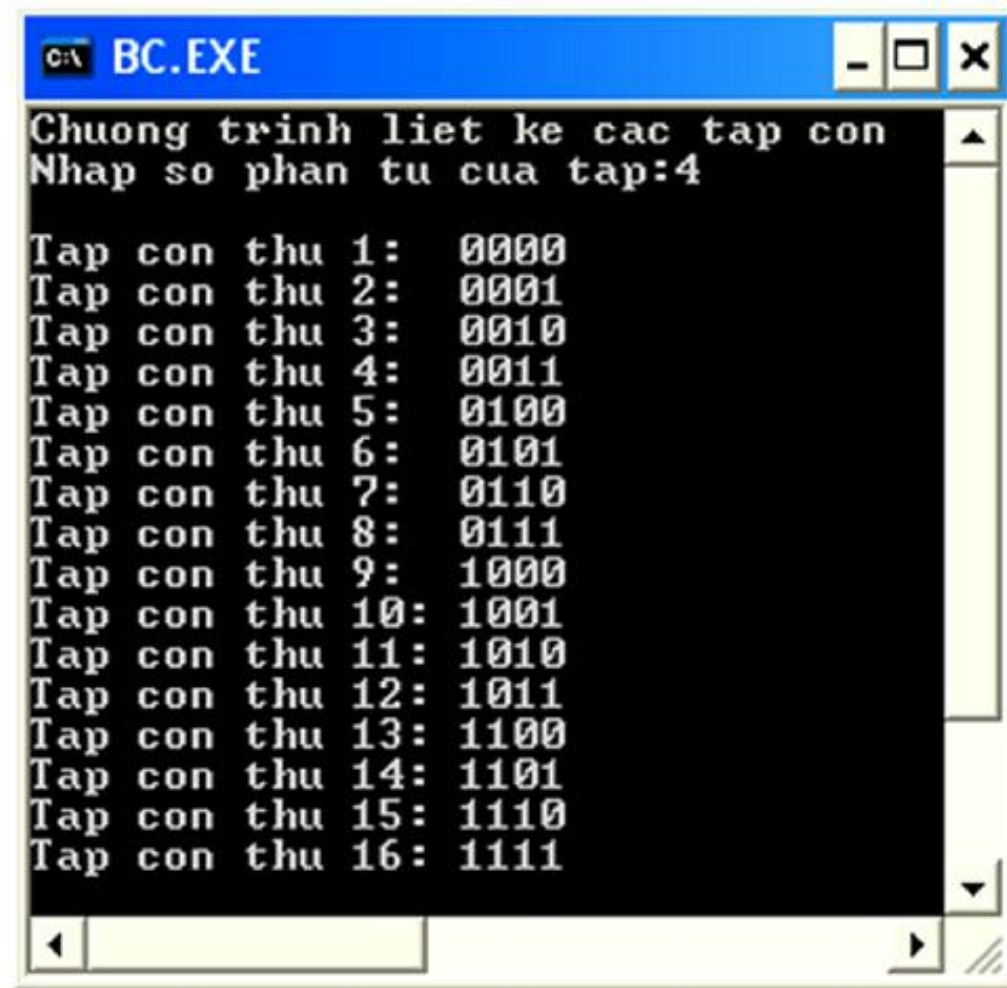
# Cài đặt

```
[1] TAPCON.CPP
int LastConfigure(char * vars, int n)
{ for (int i=0;i<n;i++) if (vars[i]=='0') return 0;
  return 1;
}
void Generate(char vars[],int n)
{ int Stop=FALSE;
  InitConfigure (vars,n);
  int count=1;
  Output<vars,n,count>;
  while (!Stop)
  { NextConfigure<vars,n>;
    count++;
    Output<vars,n,count>;
    Stop=LastConfigure<vars,n>;
  }
}
void main()
{ clrscr();
  int n; // so phan tu
  puts("Chuong trinh liet ke cac tap con");
  printf("Nhap so phan tu cua tap:");
  scanf("%d",&n);
  char* vars= new char[n];
  Generate<vars,n>;
  getch();
}
```

Thêm dòng:  
Stop = LastConfigure(vars,n);

Thêm dòng: delete[] vars;

# Minh họa



```
C:\ BC.EXE
Chương trình liệt kê các tap con
Nhập số phần tử của tap:4

Tap con thu 1: 0000
Tap con thu 2: 0001
Tap con thu 3: 0010
Tap con thu 4: 0011
Tap con thu 5: 0100
Tap con thu 6: 0101
Tap con thu 7: 0110
Tap con thu 8: 0111
Tap con thu 9: 1000
Tap con thu 10: 1001
Tap con thu 11: 1010
Tap con thu 12: 1011
Tap con thu 13: 1100
Tap con thu 14: 1101
Tap con thu 15: 1110
Tap con thu 16: 1111
```

# Nội dung

1. Lược đồ chung
2. Bài toán chuỗi 3 ký tự
3. Liệt kê tập con của tập  $N$  phần tử
- 4. Bài toán tập con  $K$  phần tử**
5. Hoán vị tập  $N$  phần tử



# Bài toán

- Liệt kê các tập con k phần tử của tập n phần tử.
- Ví dụ: Các tập con 3 phần tử của tập

**{ 1,2,3,4,5 }** là:

{ 1,2,3 }

{ 2,3,4 }

{ 1,2,4 }

{ 2,3,5 }

{ 1,2,5 }

{ 2,4,5 }

{ 1,3,4 }

{ 3,4,5 }

{ 1,3,5 }

{ 1,4,5 }

$$\begin{aligned}C_5^3 &= 5! / (3! * (5-3)!) \\&= 5! / (3! * 2!) \\&= 4 * 5 / 2 = 10\end{aligned}$$

**Tổ hợp n chập k**

# Bài toán

- Ánh xạ tập hợp bất kỳ  $n$  phần tử vào tập  $X = \{1, 2, \dots, n\}$
- Một tập con  $k$  phần tử của  $X$  là một bộ có thứ tự  $a_1 a_2 a_3 \dots a_k$  với
$$1 \leq a_1 < a_2 < a_3 < \dots < a_k \leq n$$

# Ý tưởng

- Tập con đầu:  $\{ 1, 2, 3, \dots, k \}$

Ví dụ  $\{ 1, 2, 3 \}$  với  $k=3, n=5$

- Tập con cuối:  $\{ (n-k+1), (n-k+2), \dots, n \}$

Ví dụ:  $\{ 3, 4, 5 \}$  với  $k=3, n=5$

# Cách sinh tập con kế tiếp từ tập con đã có $a_1 a_2 a_3 a_4 \dots a_k$ , chỉ số ở đây đi từ 1

(1) Tìm vị trí đầu tiên từ bên phải 1 vị trí  $i$  sao cho  $a[i] \neq n-k+i$

$i=k$ ;

while ( $a[i]==n-k+i$ )  $i--$  ;

(2) Thay  $a[i]$  bằng  $a[i] + 1$

$a[i] = a[i] + 1$ ;

(3) Thay các trị sau  $i$  ( $a[j]$ ) bằng các trị  $a[i]+j-i, \dots$

for ( $j=i+1; j \leq k; j++$ )

$a[j] = a[i] + j - i$ ;

$n=8, k=6$

{ 1, 2, 5, 6, 7, 8 }  
( $i=6, n-k+i=8$ )

Tìm vị trí đầu tiên khác với nhóm trị ở cuối tập cha theo thứ tự

{ 1, 3, 5, 6, 7, 8 }

{ 1, 3, 4, 5, 6, 7 }

# Cài đặt

```
[1]===== TOHOPN_K.CPP =====
//TOHOPN_K.CPP ** To hop n chap kphan tu
#include <stdio.h>
#include <conio.h>
void Init ( int* result, int k)
{ for (int i=1; i<=k; i++) result[i]=i;
}
void Process (int* result, int n)
{ printf("\n");
  for (int i=1; i<=n; i++) printf("%d,", result[i]);
}
void NextCombination( int*result , int n, int k , int & Stop)
{ int i= k;
  while ( i>0 && result[i]==n-k+i) i--;
  if (i>0)
  { result[i]++;
    for (int j=i+1; j<=k; j++) result[j] = result[i] +j-i;
  }
  else Stop = 1;
}
```

# Cài đặt

```
void Generate (int *result, int n, int k)
{
    int Stop=0;
    Init(result, k);
    Process(result, k);
    while (!Stop)
    {
        NextCombination(result, n, k, Stop);
        if (!Stop) Process(result, k);
    }
}

void main()
{
    int n, k, *result;
    printf("Nhap so phan tu cua tap n="); scanf("%d", &n);
    printf("Nhap so phan tu cua tap con k="); scanf("%d", &k);
    result = new int [k+1];
    Generate(result, n, k);
    getch();
}
```

Thêm dòng: delete[] result;

# Nội dung

1. Lược đồ chung
2. Bài toán chuỗi 3 ký tự
3. Liệt kê tập con của tập  $N$  phần tử
4. Bài toán tập con  $K$  phần tử
- 5. Hoán vị tập  $N$  phần tử**

# Bài toán

- Cho tập  $X = \{ 1, 2, 3, \dots, n \}$ . Hãy liệt kê tất cả các hoán vị của tập này.
- Một hoán vị của  $X$  là một bộ  $A = (a_1, a_2, \dots, a_n)$  với  $a_i \neq a_j$  nếu  $i \neq j$
- Định nghĩa 1 thứ tự:

$A = (a_1, a_2, \dots, a_{k-1}, \mathbf{a_k}, \dots a_n)$  là hoán vị trước của

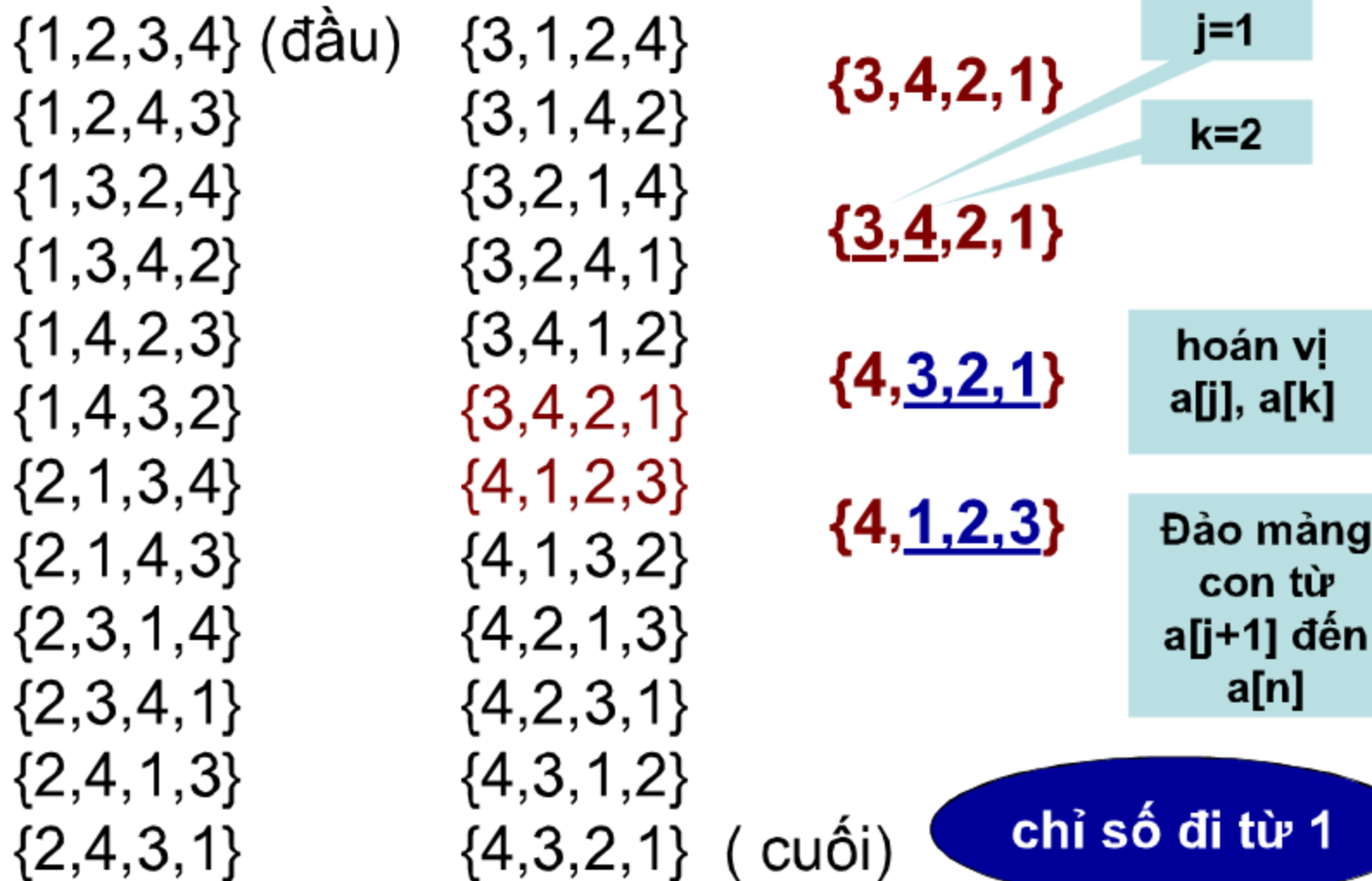
$A' = (a'_1, a'_2, \dots, a'_{k-1}, \mathbf{a'_k}, \dots a'_n)$  nếu tìm được vị trí  $k$  sao cho  $\mathbf{a_k < a'_k}$

- Ví dụ : 1234**5**67 là hoán vị trước của  
1234**6**57

- Đây chính là thứ tự từ điển.
- Độ phức tạp  $n!$



# Các hoán vị của $X=\{1,2,3,4\}$



# Ý tưởng

- trạng thái trước {1,3,4,2} trạng thái sau: {1,4,2,3}

## Giải thuật

- Tìm chỉ số lớn nhất  $j$  mà  $a_j < a_{j+1}$  từ phía phải vì đây là phần tử sẽ bị hoán vị.

1 **3** 4 2 (  $j=2$  )  $\rightarrow j=n-1$ ; while ( $a[j]>a[j+1]$ )  $j--$ ;

- Tìm vị trí đầu tiên  $k$  đi ngược từ cuối tập trị với  $a[k] > a[j]$

1 **3** 4 2 (  $k=3$  )  $\rightarrow k=n$ ; while ( $a[j]>a[k]$ )  $k--$ ;

- Hoán vị  $a[j]$  với  $a[k]$

1 **4** **3** 2

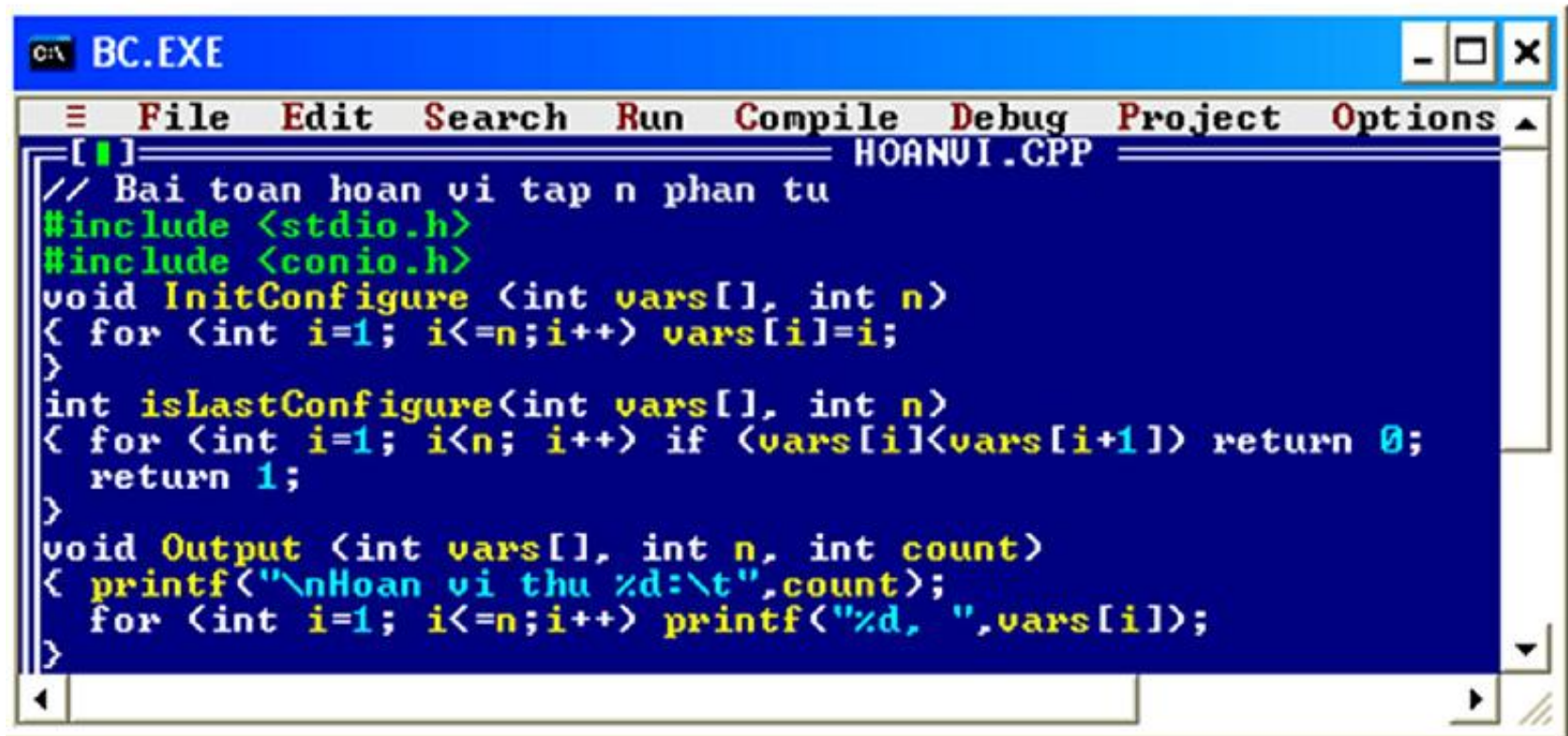
- Lật ngược đoạn  $a_{j+1} \dots a_n$

1 **4** 2 **3** trạng thái kế tiếp

# Cài đặt

{1,2,3,4,5,..., n}

{n,..., 5,4,3,2,1}



The screenshot shows a window titled "BC.EXE" with a menu bar (File, Edit, Search, Run, Compile, Debug, Project, Options) and a file named "HOANVI.CPP". The code is written in C++ and implements a permutation problem. It includes `<stdio.h>` and `<conio.h>`. The `InitConfigure` function initializes an array `vars` from 1 to `n`. The `isLastConfigure` function checks if the current permutation is the last one by comparing `vars[i]` with `vars[i+1]`. The `Output` function prints the current permutation and the count. The code is as follows:

```
[ ] HOANVI.CPP
// Bai toan hoan vi tap n phan tu
#include <stdio.h>
#include <conio.h>
void InitConfigure (int vars[], int n)
{ for (int i=1; i<=n;i++) vars[i]=i;
}
int isLastConfigure(int vars[], int n)
{ for (int i=1; i<n; i++) if (vars[i]<vars[i+1]) return 0;
  return 1;
}
void Output (int vars[], int n, int count)
{ printf("\nHoan vi thu %d:\t",count);
  for (int i=1; i<=n;i++) printf("%d, ",vars[i]);
}
```

Hoan vi thu 8: 2, 1, 4, 3,

# Cài đặt

```
BC.EXE
File Edit Search Run Compile Debug
[ ]===== HOANVI.C
void NextConfigure(int vars[], int n)
{
    int j=n-1;
    while(j>0 && vars[j]>vars[j+1]) j-- ;
    int k=n;
    while (vars[j]>vars[k]) k--;
    int t= vars[j];
    vars[j]=vars[k];
    vars[k]=t;
    int left=j+1, right= n;
    while (left<right)
    {
        t=vars[left];
        vars[left]=vars[right];
        vars[right]=t;
        left++; right--;
    }
}
```

Tìm chỉ số lớn nhất  $j$  mà  $a_j < a_{j+1}$  từ phía phải vì đây là phần tử sẽ bị hoán vị.

Tìm vị trí đầu tiên  $k$  đi ngược từ cuối tập trị với  $a[k] > a[j]$

Hoán vị  $a[j], a[k]$

Đảo ngược nhóm trị  $a[j+1], \dots, a[n]$

# Cài đặt

```
[ ] HOANVI.CPP =
void Generate<int vars[],int n>
{
    InitConfigure <vars,n>;
    int count=1;
    Output<vars,n,count>;
    int Stop=isLastConfigure<vars,n>;
    while (!Stop)
    { NextConfigure<vars,n>;
      count++;
      Output<vars,n,count>;
      Stop=isLastConfigure<vars,n>;
    }
}

void main<>
{ clrscr();
  int n; // so phan tu
  puts("Chương trình hoan vi tap n phan tu");
  printf("Nhap so phan tu cua tap:");
  scanf("%d",&n);
  int* vars= new int[n+1];
  Generate<vars,n>;
  getch();
}
```

Thêm dòng: delete[ ] vars;



# Kết quả

```
BC.EXE
Chương trình hoán vị tập n phần tử
Nhập số phần tử của tập:4

Hoán vị thứ 1: 1, 2, 3, 4,
Hoán vị thứ 2: 1, 2, 4, 3,
Hoán vị thứ 3: 1, 3, 2, 4,
Hoán vị thứ 4: 1, 3, 4, 2,
Hoán vị thứ 5: 1, 4, 2, 3,
Hoán vị thứ 6: 1, 4, 3, 2,
Hoán vị thứ 7: 2, 1, 3, 4,
Hoán vị thứ 8: 2, 1, 4, 3,
Hoán vị thứ 9: 2, 3, 1, 4,
Hoán vị thứ 10: 2, 3, 4, 1,
Hoán vị thứ 11: 2, 4, 1, 3,
Hoán vị thứ 12: 2, 4, 3, 1,
Hoán vị thứ 13: 3, 1, 2, 4,
Hoán vị thứ 14: 3, 1, 4, 2,
Hoán vị thứ 15: 3, 2, 1, 4,
Hoán vị thứ 16: 3, 2, 4, 1,
Hoán vị thứ 17: 3, 4, 1, 2,
Hoán vị thứ 18: 3, 4, 2, 1,
Hoán vị thứ 19: 4, 1, 2, 3,
Hoán vị thứ 20: 4, 1, 3, 2,
Hoán vị thứ 21: 4, 2, 1, 3,
Hoán vị thứ 22: 4, 2, 3, 1,
Hoán vị thứ 23: 4, 3, 1, 2,
Hoán vị thứ 24: 4, 3, 2, 1,
```

# Bài tập

1. Liệt kê các tập con của tập  $\{a,b,c,d,e,f\}$ .
2. Liệt kê các tập con 3, 4, 5 phần tử từ tập 6 phần tử.
3. Liệt kê các hoán vị của tập 3,4,5 phần tử