Комбинаторика Сочетание и Размещение

Code by Xumingchuan

```c
#include <stdio.h>
#include <stdbool.h>

// m     m
//C     A
// n     n
const int n=5,m=3;
const int num=100;
int a[num];
int M=m;
int Binary[n];

void Init(){
    for(int i=0;i<n;i++)
    a[i]=i+1;
}

void InitBinary(int n,int r){
    for(int i=0;i<r;i++)
        Binary[i]=1;
    for(int i=r;i<n;i++)
        Binary[i]=0;
}

void Print(){
    for(int i=0;i<m;i++)
        printf("%d ",a[i]);
    printf("\n");
}

void PrintPermu(int *b,int End){
    for(int i=0;i<=End;i++)
      printf("%d ",b[i]);
    printf("\n");
}

void SwapBinary(int i,int j){
    int Tmp=Binary[i];
    Binary[i]=Binary[j];
    Binary[j]=Tmp;
}

```

```c
void Swap(int *c,int *d){
    int Tmp=*c;
    *c=*d;
    *d=Tmp;
}

//递归：n位排列来源于n-1位排列
void Permu(int *b,int Beg,int End){
    if(Beg>=End) PrintPermu(b,End);
    else{
        for(int i=Beg;i<=End;i++){
            Swap(&b[Beg],&b[i]);//交换头
            Permu(b,Beg+1,End);
            Swap(&b[Beg],&b[i]);//交换末尾
        }
    }
}

//1 - Combination with recursion
void combination(int n,int m,int flag){
    if(m<1) {
        if(flag=='A'){
            int Arr[M];
            for(int i=0;i<M;i++)
            Arr[i]=a[i];
            Permu(Arr,0,M-1);
            printf("\n");
            return;
        }
        else{
            Print();
            return;
        }
    }

    for(int i=n;i>=m;i--){
        a[m-1]=i;//chhose the last one
        combination(i-1,m-1,flag);
    }
}
```

```
83
84  //2 - Combination with Binary
85  void CombiBinary(int n,int m,int flag){
86      if(m>n) return;
87      InitBinary(n,m);
88      bool bFind=true;
89      while(bFind){
90          if(flag=='A'){
91              int Arr[m];
92              int index=0;//Matrix of Permutation
93              for(int i=0;i<n;i++)
94              if(Binary[i]){
95                  Arr[index]=a[i];
96                  index++;
97              }
98          Permu(Arr,0,m-1);
99          printf("\n");
100         }
101         else{
102             for(int i=0;i<n;i++)
103             if(Binary[i])
104                 printf("%d ",a[i]);
105             printf("\n");
106         }
107
108         bFind=false;
109
110         for(int i=0;i<n-1;i++)
111           if(Binary[i]==1&&Binary[i+1]==0){
112             SwapBinary(i,i+1);
113             bFind=true;
114             if(Binary[0]==0){
115                for(int k=0,j=0;k<i;k++)
116                    if(Binary[k]){
117                     SwapBinary(k,j);
118                     j++;
119                    }
120             }
121             break;
122           }
123     }
124 }
125
126 int main() {
127     if(m>n) printf("input error!");
128     char A='A';
129     char C='C';
130     printf("Combination with recursion without order:\n");
131     Init();
132     combination(n,m,C);
133     printf("\n");
134     printf("Combination with Binary without order:\n");
135     Init();
136     CombiBinary(n,m,C);
137
138     printf("Combination with recursion with order:\n");
139     Init();
140     combination(n,m,A);
141     printf("\n");
142     printf("Combination with Binary with order:\n");
143     Init();
144     CombiBinary(n,m,A);
145     return 0;
146 }
147
```

ВЫВОД:

```
Combination with recursion without order:
3 4 5
2 4 5
1 4 5
2 3 5
1 3 5
1 2 5
2 3 4
1 3 4
1 2 4
1 2 3

Combination with Binary without order:
1 2 3
1 2 4
1 3 4
2 3 4
1 2 5
1 3 5
2 3 5
1 4 5
2 4 5
3 4 5
```

Combination with recursion with order: ***(Первые две столбца)***

Combination with Binary with order: ***(Вторые две столбца)***

```
3 4 5   1 2 5   1 2 3   1 3 5
3 5 4   1 5 2   1 3 2   1 5 3
4 3 5   2 1 5   2 1 3   3 1 5
4 5 3   2 5 1   2 3 1   3 5 1
5 4 3   5 2 1   3 2 1   5 3 1
5 3 4   5 1 2   3 1 2   5 1 3

2 4 5   2 3 4   1 2 4   2 3 5
2 5 4   2 4 3   1 4 2   2 5 3
4 2 5   3 2 4   2 1 4   3 2 5
4 5 2   3 4 2   2 4 1   3 5 2
5 4 2   4 3 2   4 2 1   5 3 2
5 2 4   4 2 3   4 1 2   5 2 3

1 4 5   1 3 4   1 3 4   1 4 5
1 5 4   1 4 3   1 4 3   1 5 4
4 1 5   3 1 4   3 1 4   4 1 5
4 5 1   3 4 1   3 4 1   4 5 1
5 4 1   4 3 1   4 3 1   5 4 1
5 1 4   4 1 3   4 1 3   5 1 4

2 3 5   1 2 4   2 3 4   2 4 5
2 5 3   1 4 2   2 4 3   2 5 4
3 2 5   2 1 4   3 2 4   4 2 5
3 5 2   2 4 1   3 4 2   4 5 2
5 3 2   4 2 1   4 3 2   5 4 2
5 2 3   4 1 2   4 2 3   5 2 4

1 3 5   1 2 3   1 2 5   3 4 5
1 5 3   1 3 2   1 5 2   3 5 4
3 1 5   2 1 3   2 1 5   4 3 5
3 5 1   2 3 1   2 5 1   4 5 3
5 3 1   3 2 1   5 2 1   5 4 3
5 1 3   3 1 2   5 1 2   5 3 4
```

Permu( ) – Простое порождение перестановок (это для того, чтобы порождает размещение перестановками каждого элемента сочетания)

Combination( ) – Простое сочетание (выбор) **m** из **n** чисел (1-ой способ)

CombiBinary( ) – Сочетание, используя двоичной системой. (2-ой способ)То есть, Мы определяем вспомогательный массив **Binary []**. В этом массиве 1 представляет позицию, представленную выбранным числом, а 0 представляет невыбранное число. Инициализируем этот массив так, чтобы все 1 (m штук) были перед массивом (например, выберим 3 числа из 5, мы инициализируем **Binary []** в виде 1 1 1 0 0) . При этом продолжим обмениваться значениями. Обмен останавливается до появления массива (в данном случае 0 0 1 1 1). С помощью этих массивов мы можем генерироваться сочетание лексикографической порядка.

Замечание: В функции **Combination( )** и **CombiBinary( )** входятся параметр **flag,** который определяет, сочетание или размещение порождают эти функции.Если **flag ==‘A’** , функция порождает размещение. Если **flag == ‘ C’,** функция порождает сочетание.

Вывод состоит из 4 части:

Сочетание 1-ым способом

Сочетание 2-ым способом

Размещение 1-ым способом

Размещение 2-ым способом