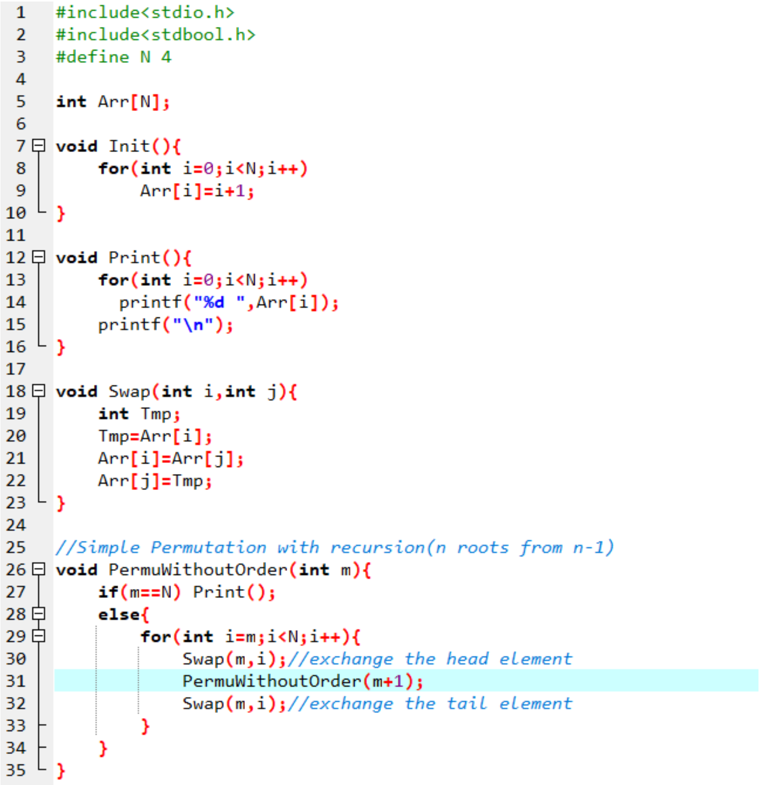
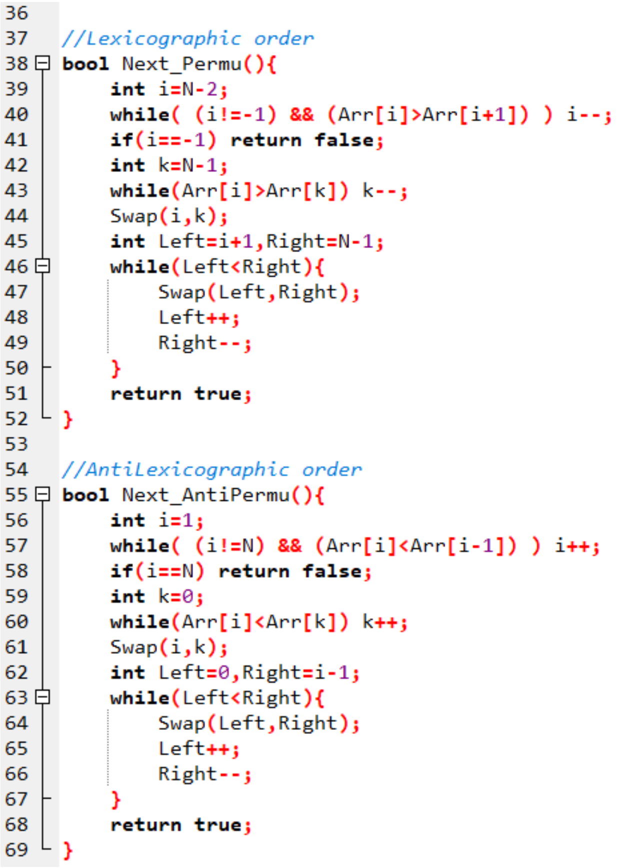
Комбинаторика Перестановка(простое, лексикограф. и антилексикограф.)

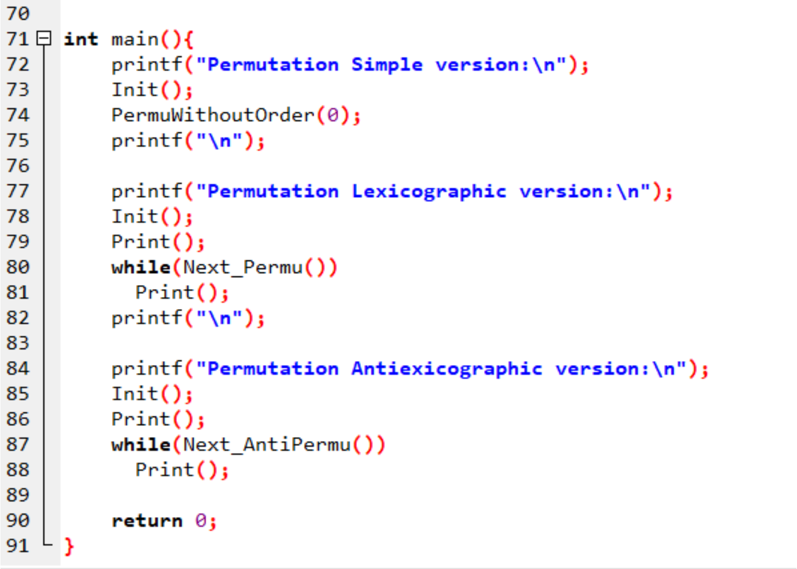
Code by Xumingchuan



```c
#include<stdio.h>
#include<stdbool.h>
#define N 4

int Arr[N];

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

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

void Swap(int i,int j){
    int Tmp;
    Tmp=Arr[i];
    Arr[i]=Arr[j];
    Arr[j]=Tmp;
}

//Simple Permutation with recursion(n roots from n-1)
void PermuWithoutOrder(int m){
    if(m==N) Print();
    else{
        for(int i=m;i<N;i++){
            Swap(m,i);//exchange the head element
            PermuWithoutOrder(m+1);
            Swap(m,i);//exchange the tail element
        }
    }
}
```

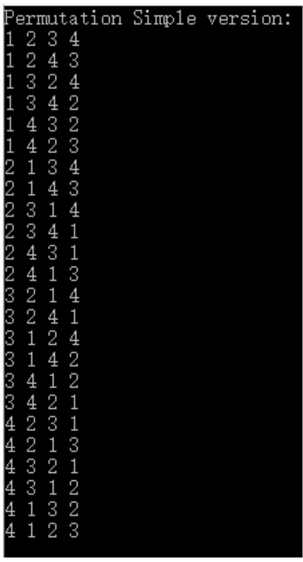


```c

//Lexicographic order
bool Next_Permu(){
    int i=N-2;
    while( (i!=-1) && (Arr[i]>Arr[i+1]) ) i--;
    if(i==-1) return false;
    int k=N-1;
    while(Arr[i]>Arr[k]) k--;
    Swap(i,k);
    int Left=i+1,Right=N-1;
    while(Left<Right){
        Swap(Left,Right);
        Left++;
        Right--;
    }
    return true;
}

//AntiLexicographic order
bool Next_AntiPermu(){
    int i=1;
    while( (i!=N) && (Arr[i]<Arr[i-1]) ) i++;
    if(i==N) return false;
    int k=0;
    while(Arr[i]<Arr[k]) k++;
    Swap(i,k);
    int Left=0,Right=i-1;
    while(Left<Right){
        Swap(Left,Right);
        Left++;
        Right--;
    }
    return true;
}
```

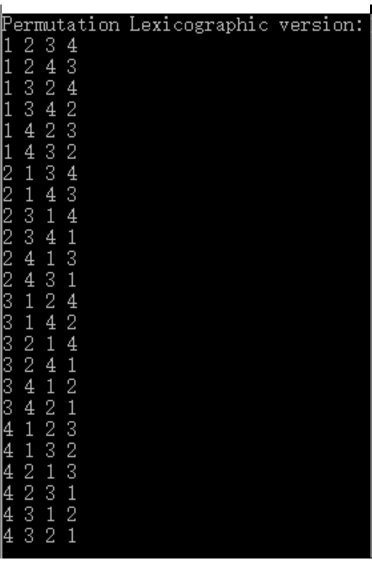


```
int main(){
    printf("Permutation Simple version:\n");
    Init();
    PermuWithoutOrder(0);
    printf("\n");

    printf("Permutation Lexicographic version:\n");
    Init();
    Print();
    while(Next_Permu())
      Print();
    printf("\n");

    printf("Permutation Antiexicographic version:\n");
    Init();
    Print();
    while(Next_AntiPermu())
      Print();

    return 0;
}
```

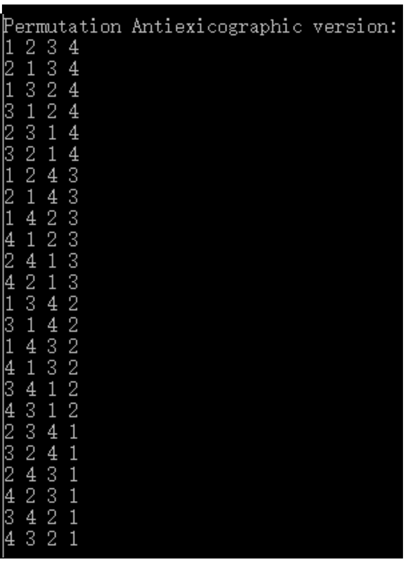
ВЫВОД:



```
Permutation Simple version:
1 2 3 4
1 2 4 3
1 3 2 4
1 3 4 2
1 4 3 2
1 4 2 3
2 1 3 4
2 1 4 3
2 3 1 4
2 3 4 1
2 4 3 1
2 4 1 3
3 2 1 4
3 2 4 1
3 1 2 4
3 1 4 2
3 4 1 2
3 4 2 1
4 2 3 1
4 2 1 3
4 3 2 1
4 3 1 2
4 1 3 2
4 1 2 3
```



```
Permutation Lexicographic version:
1 2 3 4
1 2 4 3
1 3 2 4
1 3 4 2
1 4 2 3
1 4 3 2
2 1 3 4
2 1 4 3
2 3 1 4
2 3 4 1
2 4 1 3
2 4 3 1
3 1 2 4
3 1 4 2
3 2 1 4
3 2 4 1
3 4 1 2
3 4 2 1
4 1 2 3
4 1 3 2
4 2 1 3
4 2 3 1
4 3 1 2
4 3 2 1
```



```
Permutation Antiexicographic version:
1 2 3 4
2 1 3 4
1 3 2 4
3 1 2 4
2 3 1 4
3 2 1 4
1 2 4 3
2 1 4 3
1 4 2 3
4 1 2 3
2 4 1 3
4 2 1 3
1 3 4 2
3 1 4 2
1 4 3 2
4 1 3 2
3 4 1 2
4 3 1 2
2 3 4 1
3 2 4 1
2 4 3 1
4 2 3 1
3 4 2 1
4 3 2 1
```