پیش‌نیاز: برای آشنایی با طرز کار تابع sort، توصیه می‌شود، فصل 6 کتاب «طراحی الگوریتم با رویکردی خلاقانه» قسمت مرتب‌سازی را مطالعه کنید.

در این بخش، با چند تابع پراستفاده، آشنا می‌شوید.

برای استفاده از این توابع، باید تابع کتابخانه ای algorithm را include کنید:

```
#include<algorithm>
```

Sort

با استفاده از این تابع می‌توانید آرایه‌ای از متغیرهای قابل مقایسه را مرتب کنید. (آرایه‌ای از int, char, double, …) این تابع دو آرگومان می‌گیرد. آرگومان اول pointer ای به اولین خانه‌ی قسمتی از آرایه است که شما می‌خواهید آن(قسمت) را مرتب کنید. آرگومان دوم pointer ای به خانه‌ی بعد از آخرین خانه‌ی قسمتی از آرایه است که شما می‌خواهید آن را مرتب کنید.

برای روشن شدن مطلب به مثال زیر توجه کنید:

```
sort(arr, arr+n);
```

آرگومان اول (arr)، pointer ای به اولین خانه‌ی آرایه‌ای است، که شما می‌خواهید آن را مرتب کنید. (اشاره‌گر به خانه‌ی اول آرایه). آرگومان دوم (arr+n)، pointer ای به بعد-از-آخرین خانه‌ی آرایه‌ای است که شما می‌خواهید آن را مرتب کنید (اشاره‌گر به خانه‌ی بعد از خانه‌ی n-ام).

به طور کلی برای مرتب کردن، خانه‌ی i تا j آرایه‌ی arr می‌نویسیم:

```
sort(arr+i-1, arr+j);
```

برای مثال، خط زیر، خانه های 3-ام تا (n-2)-ام آرایه‌ی a را مرتب می‌کند:

```
sort(a+2, a+n-2);
```

next and prev permutation

فرض کنید در آرایه A جایگشتی از N عدد دلخواه داشته باشیم. در آرایه B نیز، جایگشتی از همان اعداد داشته باشیم.

جایگشتA از جایگشت B کوچکتر است، اگر (0<=K<N) K وجود داشته باشد که:

i<K → a[i]=b[i]

a[k]<b[k]

جایگشت بعدی جایگشت A، کوچکترین جایگشتی است که از جایگشت A بزرگتر است.

همچنین، جایگشت قبلی جایگشت A، بزرگترین جایگشتی است که از جایگشت A کوچکتر است.

با این تعاریف، بزرگترین جایگشتی که می‌توان از n عدد دلخواه داشت، جایگشتی با ترتیب نزولی این n عدد؛ و کوچکترین جایگشتی که می‌توان از n عدد دلخواه داشت، جایگشتی با ترتیب صعودی این n عدد است.

در C++ توابعی وجود دارند، که با گرفتن جایگشتی از n عدد دلخواه، جایگشت بعدی و قبلی آن را می‌سازند.

ورودی این توابع یک آرایه است که با جایگشتی از n عدد دلخواه پر شده است:

```
next_permutation(temp, temp+n);
prev_permutation(temp, temp+n);
```

آرگومان‌های این توابع، pointer ای به خانه‌ی اول آرایه و خانه‌ی بعد-از-آخر آرایه است.

این توابع علاوه بر اینکه جایگشت بعدی(next) یا قبلی(prev) را روی آرایه‌ی فرستاده شده برای تابع، می‌سازد، مقدار true یا false را نیز باز می‌گرداند. در صورتی که جایگشت بعدی(یا قبلی) برای آرایه‌ی فرستاده شده وجود نداشت، (یعنی به بزرگترین(کوچکترین) جایگشت این n عدد رسیده باشیم) تابع مقدار false و در غیر این‌صورت true باز می‌گرداند. بنابراین، تکه‌کدهای زیر همه‌ی جایگشت‌های اعداد 1 تا n را در خروجی چاپ می‌کند:

```
for(int i=0; i<n; i++)
        temp[i]=i+1;
do
{
        for(int i=0; i<n; i++)
                cout<<temp[i]<<" ";
        cout<<endl;
} while(next_permutation(temp, temp+n));
```

```
for(int i=0; i<n; i++)
        temp[i]=n-i;
do
{
        for(int i=0; i<n; i++)
                cout<<temp[i]<<" ";
        cout<<endl;
} while(prev_permutation(temp, temp+n));
```

تکه کد اول، جایگشت‌ها را از کوچک به بزرگ؛ و تکه‌کد دوم جایگشت‌ها از بزرگ به کوچک، در خروجی چاپ می‌کند.

همانند تابع sort**، می‌توانید توابع بالا را برای قسمت خاصی از آرایه، فراخوانی کنید. برای مثال، برای خانه‌ی** 5**–ام آرایه تا خانه‌ی** n-5 **ام:**

```
next_permutation(temp+4, temp+n-5);
```