Използване на масив като параметър на функция

```
void show(int n, int a[])
{
   for(int i=0;i<n;i++) cout << a[i] << endl;
}

int main()
{
   int a[7], b[8];
  .....
  show(7,a); show(8,b);
}
```

Пример: функция за проверка, дали отрез от масив е строго сортиран

```
bool isSorted(int a[], int m, int n)
{
   for(int i=m; i<n; i++)
     if(a[i]>=a[i+1]) return false;
  return true;
}
```

Пример: функция за проверка, дали има равни стойности в отрез от масив

```
bool isEqVal(int a[], int m, int n)
{
   for(int i=m; i<n; i++)
   for(int j=i+1; j<=n; j++)
     if(a[i]==a[j]) return true;
  return false;
}
```

Пример: функция, която намира най-голяма разлика между стойности на елементите в отрез от масив

```
int maxDif(int a[], int m, int n)
{  int max=0;
   for(int i=m; i<n; i++)
   for(int j=i+1; j<=n; j++)
     if(max < abs(a[i]-a[j])) max =abs(a[i]-a[j]);
  return max;
}
```

Търсене в сортиран масив

```
#include <iostream>
#include <algorithm>
using namespace std;
int main ()
{
  int n=9;
  int a[n] = {1,2,3,4,5,4,3,2,1};
  sort(a,a+n);
  int v=3;
  if (binary_search (a, a+n, v))
    cout << "found!"; else cout << "not found";
}
```

„Изобретяване“ на двоичното търсене

```
int f(int a[], int m, int n, int v)
{
  int i=m, j=n;
  while(i<=j)
  {
    int k=(i+j)/2;
    if(a[k]==v) return k;
    if(v<a[k]) j=k-1; else i=k+1;
  }
  return -1;
}
```

Извикване на функцията за двоично търсене

```
int main ()
{
  int n=12;
  int a[n] = {0,1,3,5,7,9,11,12,13,17,20,22};

  int k=f(a,1,11,6);

  if(k!=-1) cout << "found, k=" << k;
  else cout << "not found";
}
```

Скорост на алгоритъма за двоично търсене

Пример за размер на масива: N=1024

Разглеждаме степените на двойката:

$2^0=1$, $2^1=2$, $2^2=4$,..., $2^9=512$, $2^{10}=1014$

След стъпка 1, търсим в масив с N=512

След стъпка 2, търсим в масив с N=256

.....

След стъпка 10, търсим в масив с N=1

т.е. броят на стъпките е толкова, колкото е степента k, за която $2^k=N$

k е двоичен логаритъм от N

Скоростта на алгоритъма за двоично търсене е логаритмична

Скоростта на алгоритъма за последователно търсене е линейна

Разликата в броя операции, които трябва да извършат двата алгоритъма е значителен:

при $N=2^{10}=1\ 024$ е съответно 10 и прибл. хиляда

при $N=2^{20}= 1\ 048\ 576$ е съответно 20 и прибл. милион

Многоцифрен брояч

Моделиране на брояч на електромер или километраж на автомобил

Използваме масив за описване на разредите

a[1], a[2], ..., a[n].

Първо и на-бързо увеличаваме най-десния разред

Ако той достигне максималната си стойност – правим пренос в предния разред

Увеличаване на разред с пренос

```
int c=1;
for(int j=1;j<=n+1;j++)
{
    a[j] += c;
    if(a[j]>m) {a[j]=0;c=1;}
    else c=0;
}
```

Функция, която генерира всичките състояния

```
const int N_max=100; int a[N_max];
void gen()
{ int n, m; cin >> n >> m;  int c;
  do
  { for(int j=n;j>=1;j--) cout << a[j]; cout << endl;
    int c=1;
    for(int j=1;j<=n+1;j++)
       { a[j] += c;  if(a[j]>m) {a[j]=0;c=1;} else c=0; }
  } while(a[n+1]==0);
}
```

Предишната програма показва всичките начини за поставяне върху 3 позиции на знаците 0, 1 и 2.

Нарича се “вариации с повторение“

В общия случай: всички начини за поставяне върху N позиции на M различни знака:

0, 1, ..., M-1.

Множества

Разполагайки с N предмета, интересуваме се от всички начини на постявянето им в контейнер, като не вземаме предвид реда на поставянето им. Например при N=3 начините са 8:

{},

{1}, {2}, {3},

{1 2}, {2 3}, {1 3},

{1 2 3}

Генериране на множества

Чрез мноцифрен брояч, като всеки разряд има 2 стойности: 0 и 1, и номерът на разряда е номер на предмета. Пример за N=3

000 – {}

001 – {1}

010 – {2}

011 – {1 2}

100 – {3}

101 – {1 3}

110 – {2 3}

111 – {1 2 3}

```
int n, m=1; cin >> n; int c;
do
{ cout << "{ ";
  for(int j=n;j>=1;j--) if(a[j]==1) cout << j << " ";
  cout << "}" << endl;
  int c=1;
  for(int j=1;j<=n+1;j++)
   {a[j] += c;  if(a[j]>m) {a[j]=0;c=1;} else c=0;}
} while(a[n+1]==0);
```

Пермутации

Пермутация на числата 1, 2, 3, ..., N е всяко тяхно разместване. Например при N=3, всички пермутации са 6:

123 132 213 231 312 321

Генериране на пермутации

```
#include<iostream>
#include<algorithm>
using namespace std;
int a[1001];
int main()
{   int n; cin>>n;
    for(int i=1;i<=n;i++) a[i]=i;
    do
    {
      for(int i=1;i<=n;i++) cout<<a[i]<<" "; cout<<endl;
    }
    while(next_permutation(a+1,a+n+1));
}
```

Броят на пермутации на N елемента е

$N! = 1 * 2 * 3 * ... * N$

Например

$3! = 6$

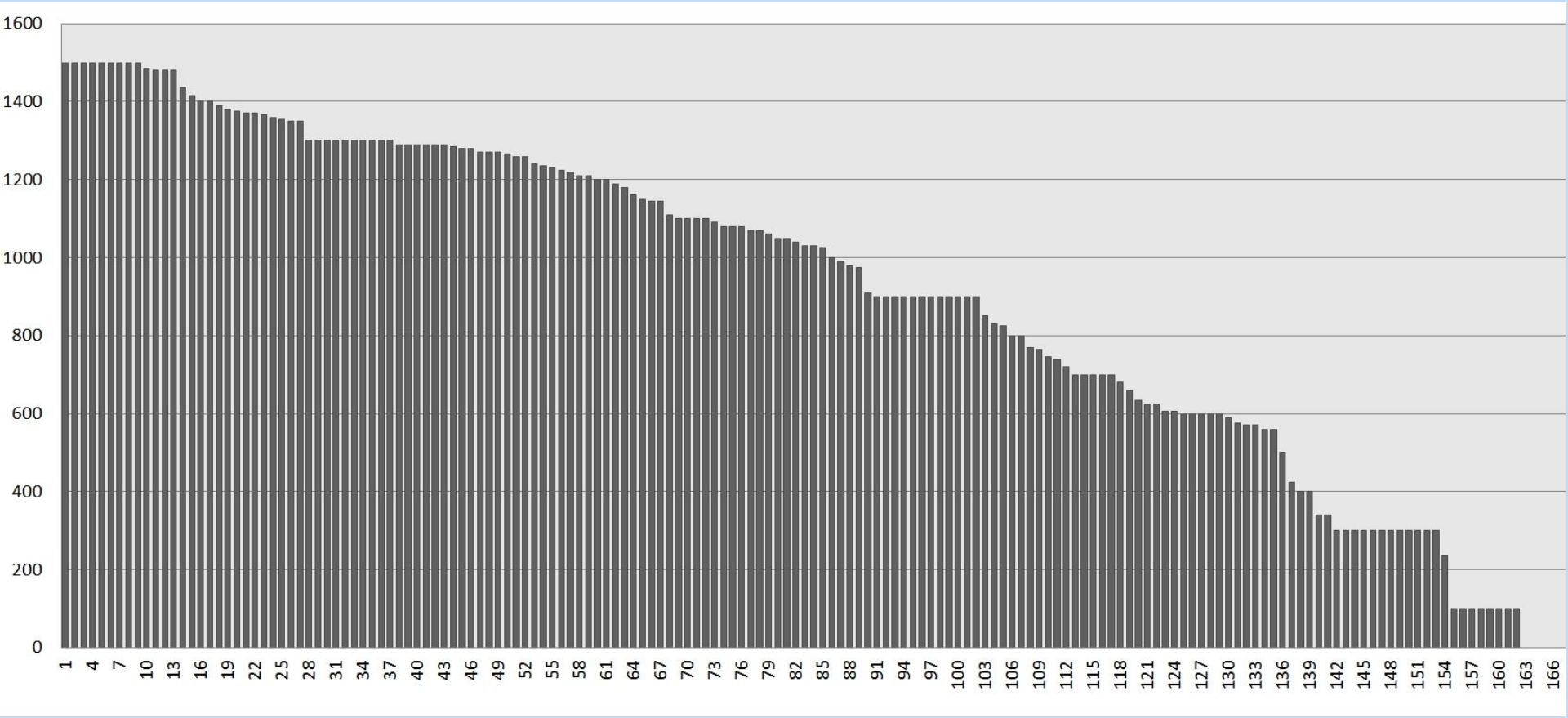
$10! = 3628800$

Време за работа на програмата

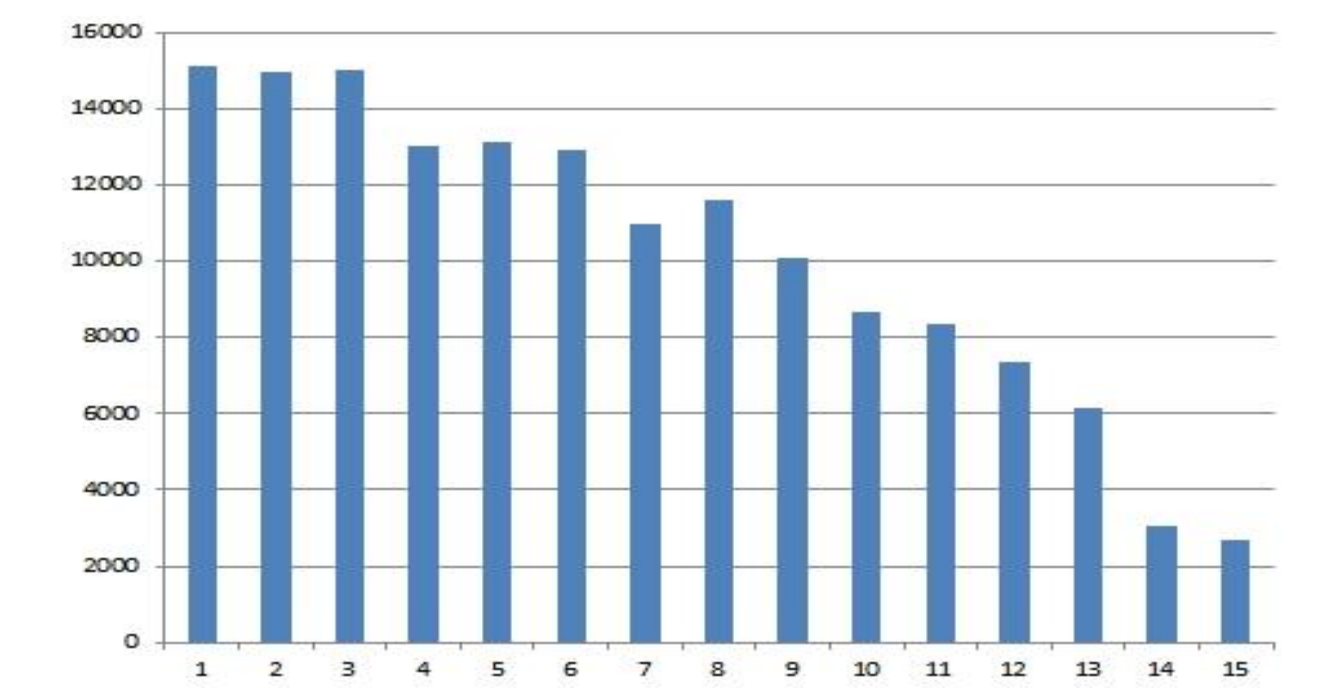
```
 int n; cin>>n;
for(int i=1;i<=n;i++) a[i]=i;
int t1=clock();
do
{
  //for(int i=1;i<=n;i++) cout<<a[i]<<" ";
  // cout<<endl;
}
while(next_permutation(a+1,a+n+1));
int t2=clock();
cout << (t2-t1)/1000.0 << " sec" << endl;
```

Резултати от задачите за самостоятелна работа

Анонимна подредба по точки



Успех на участниците



Трудност на задачите