1. Написати програму, що реалізує методи сортування: ***простого включення*** та ***метод швидкого сортування***.
2. Згенерувати три масиви з випадковими елементами типу Integer довжиною 100, 1000 та 10000 елементів, відповідно.
3. Відсортувати одержані масиви за збільшенням елементів, визначивши при цьому такі параметри:
    - кількість порівнянь;
    - кількість обмінів;
    - фактичний час роботи,

1. Результати порівнянь методів сортування

Метод простого включення

Кількість перевірок на і-му кроці дорівнює щонайбільше і-1, щонайменше 1, тому в середньому - і/2. Тому у середньому загальна кількість перевірок

$$C_{сер} = \sum_{i=2}^{n} \frac{i}{2} = \frac{(1+2+\ldots+n)}{2} = \frac{(n+1)n}{4} = O(n^2).$$

При n=100, C=2525.
n=1000, C=250250
n=10000, C=25002500

При цьому

$$C_{\min} = n-1; \quad C_{\max} = \frac{(n-1)n}{2}.$$

Кількість пересилань M дорівнює щонайбільше і, щонайменше 0 на і-му кроці, тобто і/2 у середньому. Тому

$$M_{\min} = 0; \quad M_{\max} = \frac{(n-1)n}{2}; \quad M_{сер} = \frac{(n-1)n}{4}.$$

При n=100, M=2475, $M_{max}=4950$
n=1000, M=249750, $M_{max}=499500$
n=10000, M=24997500

Оцінимо теоретичну кількість порівнянь для методу швидкого сортування:

Для оцінки C також застосовую апарат теорії ймовірності, проте тут оцінка буде доволі умовною, адже вважається, що масив завжди ділиться на два масиви (тобто,наприклад, викидається випадок, коли медіана вже стоїть на місці, а саме на початку, що підвищує кількість порівнянь).

$C \approx N \cdot \ln(N)$

Тоді при N=100

$C \approx 460$

Тоді при N=1000

$C \approx 6907.8$

Тоді при N=10000

$C \approx 92103.4$

М набагато складніше оцінюється апаратом теорії ймовірності, тому просто використаю емпіричну формулу :

$M \approx C/4$

Тоді при N=100

$M \approx 163$

Тоді при N=1000

$M \approx 2495$

Тоді при N=10000

$M \approx 33223$

Перевіримо відношення $\frac{M_{експ}}{M_{теор}}$ та $\frac{C_{експ}}{C_{теор}}$ для методу «Простого включення» :

при N=100 $\frac{M_{експ}}{M_{теор}} = 1$ $\frac{C_{експ}}{C_{теор}} \approx 0.996$

при N=1000 $\frac{M_{експ}}{M_{теор}} = 1$ $\frac{C_{експ}}{C_{теор}} \approx 1.021$

при N=10000 $\frac{M_{експ}}{M_{теор}} = 1$ $\frac{C_{експ}}{C_{теор}} \approx 1.004$

Як бачимо значення відношень майже не відрізняються, тобто вони не залежать від N.

Тепер перевіримо відношення $\frac{M_{експ}}{M_{теор}}$ та $\frac{C_{експ}}{C_{теор}}$ для методу швидкого сортування:

при N=100 $\frac{M_{експ}}{M_{теор}} \approx 1.16$ $\frac{C_{експ}}{C_{теор}} \approx 1.28$

при N=1000 $\frac{M_{експ}}{M_{теор}} \approx 1.075$ $\frac{C_{експ}}{C_{теор}} \approx 1.52$

при N=10000 $\frac{M_{експ}}{M_{теор}} \approx 1.15$ $\frac{C_{експ}}{C_{теор}} \approx 1.25$

2. Лістинг програми

```
#include <stdlib.h>
#include <conio.h>
#include <math.h>
#include <iostream>
#include <fstream>
#include <time.h>
#include <cstring>

#define eps 0.00001

using namespace std;
unsigned long moves, compares;

ofstream fout("sort.txt");

//Quick sort
void QuickSort(int * a, int onep, int twop) {
  int m,i,j;
  int x;
  i = onep;
  j = twop;
  m=(onep+twop)/2;
  x = a[m];

  while (i<=j) {
    while (a[i]<x){
              i++;
              compares++;
     }
    while (x<a[j]){
              j--;
              compares++;
     }
    if (i<=j) {
              if (i<j) {
              swap(a[i], a[j]);
              moves+=2;
              }
     i++; j--;
    }
  }

  if (onep<j) QuickSort(a,onep,j);
  if (i<twop) QuickSort(a,i,twop);
}

//Sort with insert
void ins(int * a, int N){
     int i,c,k;
     for(i=1; i<N; i++){
              k=i-1;
              c=a[i];
              while((k>=0) and(c<a[k])){
                       a[k+1]=a[k];
                       k--;
                       compares++;
              }
     a[k+1]=c;
     moves++;
     }
}
```

```
//Main part of the program
int main(){
  int * a;
  int * b;
  int i,j,N;
  clock_t start, finish;
  double  duration;
  bool sorted = true;
  do{
     cout<<"\n(Input '0' to finish) N=";
     cin>>N;
     if (N<=0) return 1;
     a = new int[N];
     b = new int[N];
     cout<<"\nRandom elements:\n\n"<<endl;
     srand(time(0));
     moves=0;
     compares=0;
     start = clock();
     j=0;
     while(duration = (double)(finish - start)/ CLOCKS_PER_SEC<eps){
              for(i=0; i<N; i++) a[i] = rand()%100+1;
        for(i=0; i<N; i++) cout<<a[i]<<"  ";
        cout<<endl;
        for (i = 0; i<N; i++) b[i]=a[i];
              QuickSort(b,0,N-1);
              finish = clock();
              j++;
     }
     cout<<"\n\nQuick\n"<<endl;
     for(i=0; i<N; i++) cout<<b[i]<<"  ";
     cout<< "\nTime to sort  "<<j<<"  times array of " << N << " elements QuickSort:";
     duration = (double)(finish - start) / CLOCKS_PER_SEC;
     cout << duration << " seconds" << endl;
     fout << endl << "Time to sort array of " << N << " elements QuickSort: "<< duration
<< endl;
     cout <<compares<<" - compares; "<< moves << " moves \n";
     fout <<compares<<" - compares; "<< moves << " moves \n";
     srand(time(0));
     moves=0;
     compares=0;
     start = clock();

     cout<<"\n\nSorting with insert\n"<<endl;
     j=0;
     while(duration = (double)(finish - start)<eps){
              for(i=0; i<N; i++)
        a[i] = rand()%100+1;
        for (i = 0; i<N; i++) b[i]=a[i];
              ins(b,N);
              finish = clock();
              j++;
     }
     for(i=0; i<N; i++){
        cout<<b[i]<<"  ";
     }
     cout<< "\nTime to sort  "<<j<<"  times   array of " << N << " elements Ins Sort:";
     duration = (double)(finish - start) / CLOCKS_PER_SEC;
     cout << duration << " seconds" << endl;
     fout << endl << "Time to sort array of " << N << " elements InsSort: "<< duration <<
endl;
     cout <<compares<<" - compares; "<< moves << " moves \n";
     fout <<compares<<" - compares; "<< moves << " moves \n";
```

```
  } while(N>0);

  return 0;
}
```

3. Результати роботи

```
(Input '0' to finish) N=100

Random elements:


12  44  12  25  98  39  31  18  43  2  21  9  72  6  32  13  28  50  36  51  92
 11  82  90  92  12  80  86  30  25  41  94  70  53  52  48  98  95  2  1  63  3
4  21  76  11  17  78  35  64  94  99  27  76  72  69  75  51  55  23  100  79
75  40  7  34  5  82  47  33  81  56  13  34  27  57  90  55  68  41  88  57  71
  43  15  72  41  12  85  73  74  51  91  92  29  93  22  4  26  79  83


Quick

1  2  2  4  5  6  7  9  11  11  12  12  12  12  13  13  15  17  18  21  21  22
23  25  25  26  27  27  28  29  30  31  32  33  34  34  34  35  36  39  40  41
41  41  43  43  44  47  48  50  51  51  51  52  53  55  55  56  57  57  63  64
68  69  70  71  72  72  72  73  74  75  75  76  76  78  79  79  80  81  82  82
83  85  86  88  90  90  91  92  92  92  93  94  94  95  98  98  99  100
Time to sort  1  times array of 100 elements QuickSort:0.028 seconds
563 - compares; 308 moves


Sorting with insert

1  2  3  3  4  4  5  6  7  8  8  8  10  13  13  13  14  14  15  16  17  18  20
20  20  21  21  23  23  23  24  24  24  27  27  28  32  33  33  37  37  38  39
39  42  43  43  44  45  46  46  51  53  53  54  54  55  57  61  63  63  64  65
66  67  67  67  68  70  73  73  73  74  75  76  76  78  78  79  80  81  85  85
85  87  89  90  92  92  92  93  94  94  97  97  98  98  98  99  99
Time to sort  1  times   array of 100 elements Ins Sort:0.001 seconds
2370 - compares; 99 moves

(Input '0' to finish) N=0

--------------------------------
Process exited after 3.685 seconds with return value 1
Press any key to continue . . .
```

4. Висновки

Під час роботи я навчився сортувати масиви простими і більш досконалими методами, оцінювати різні параметри ефективності різних методів (кількість перестановок, кількість порівнянь, час роботи).

Порівнюючи два методи можна відзначити, що швидке сортування займає набагато менше часу (на масивах великих розмірів) і потребує меншої кількості порівнянь та перестановок.

При малих розмірах масивів метод "простого включення" не дуже поступається у швидкодії методу швидкого сортування і враховуючи простішу його реалізацію можна сказати що цей метод доречніший (для малих розмірів масивів), особливо якщо масив майже відсортований.