Методы сортировки массивов

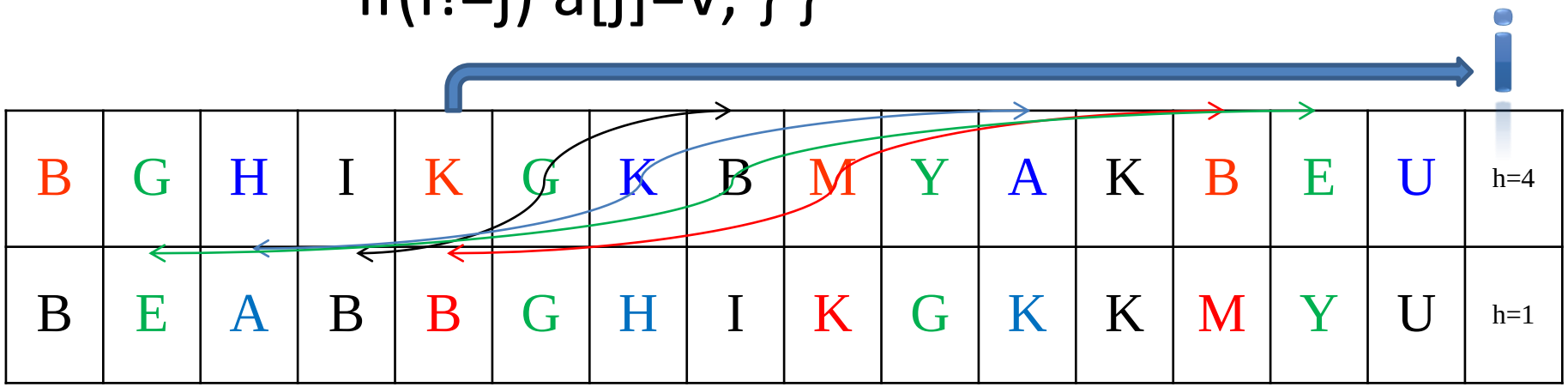
```
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;

unsigned long long start;
unsigned long long access_counter() { asm("rdtsc"); }

const int N=1000;
template <typename T> void exch(T& A, T& B)
                                        {T t=A; A=B; B=t;}
template <typename T> int compexch(T& A, T& B)
        {if(A<=B) return 0; else {exch(A,B); return 1;} }
```

```
template <typename T> void init_x(T a[]) {
                        for(int i=0;i<N;i++)a[i]=i; a[0]=N; }

template <typename T> void init_sin(T a[]) {
        double h=2.*M_PI/(N-1);
        for(int i=0;i<N;i++) a[i]=10.*sin(h*i); }

template <typename T> void TEST(T a[])  {
   for(int i=0;i<N-1;i++) if(a[i]>a[i+1]) {
   cerr << i << ' ' << a[i] << ' ' << a[i+1] << endl;
   exit(1); } }
```

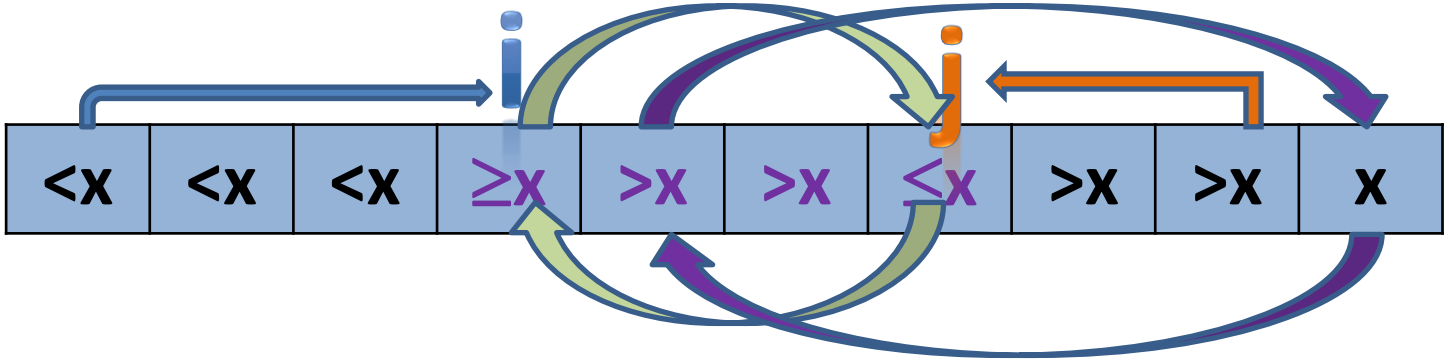
```
template <typename T> void InsertSort(T a[], int l, int r) {
int i,j;
        for(j=i=l+1;i<=r;j=++i)
        while( j>l && compexch(a[j-1],a[j]) ) j--;
}

template <typename T> void ChoiceSort(T a[], int l, int r)
{ int i,j,m;
       for(m=i=l;i<r;m=++i) {
              for(j=i+1;j<=r;j++) if(a[j]<a[m]) m=j;
              exch(a[i],a[m]); }
}
```

```
template <typename T> void BubbleSort(T a[], int , int r) {
int j,k;
        do { for(j=k=r;j>l;j--) if( compexch(a[j-1],a[j]) ) k=j; l=k; }
        while(l<r);
}

template <typename T> void ShakerSort(T a[], int l=0, int
    r=N-1) {
int j,k;
    do { for(j=k=r;j>l;j--) if(compexch(a[j-1],a[j])) k=j; l=k;
        for(j=k=l+1;j<=r;j++) if(compexch(a[j-1],a[j])) k=j; r=k-1;}
    while(l<r);
}
```

```
template <typename T> void ShellSort(T a[], int l, int r)
{ int i,j,h;
   for(h=1; h<=(r-l)/3; h=3*h+1);
   for(; h>0; h/=3)
        for(i=l+h; i<=r; i++) {
              T v=a[j=i];
               while( j>=l+h && v<a[j-h] ) a[j]=a[j-=h];
               if(i!=j) a[j]=v; } }
```



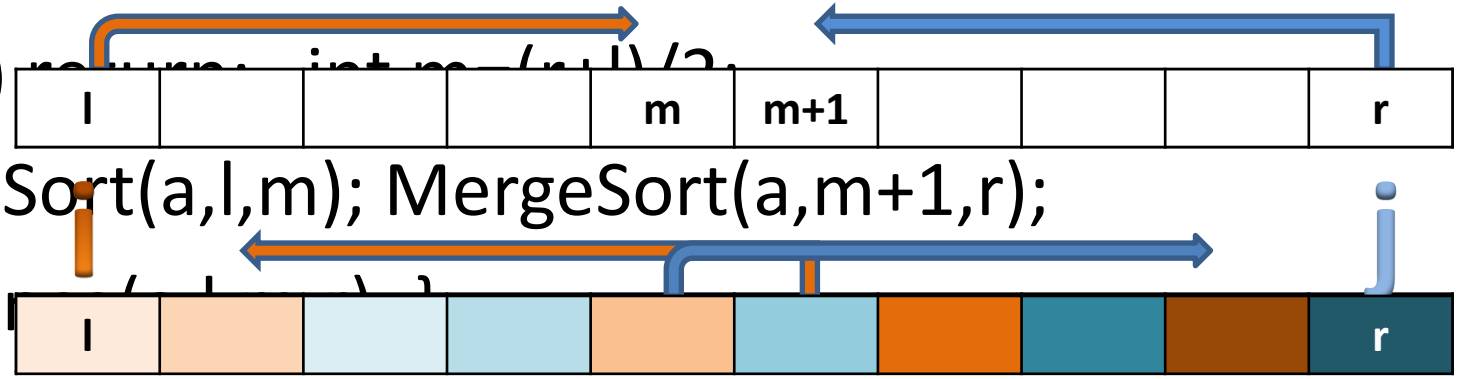
```
template <typename T> void QuickSort(T a[], int l, int r)
{ int i=l-1,j=r; T x=a[r];
   for(;;) {   while(a[++i]<x);
               while(x<a[--j]) if(j==l) break;
               if(i>=j)break; exch(a[i],a[j]); }
   exch(a[i],a[r]);
   if(l<i-1) QuickSort(a,l,i-1);
   if(r>i+1) QuickSort(a,i+1,r); }
```



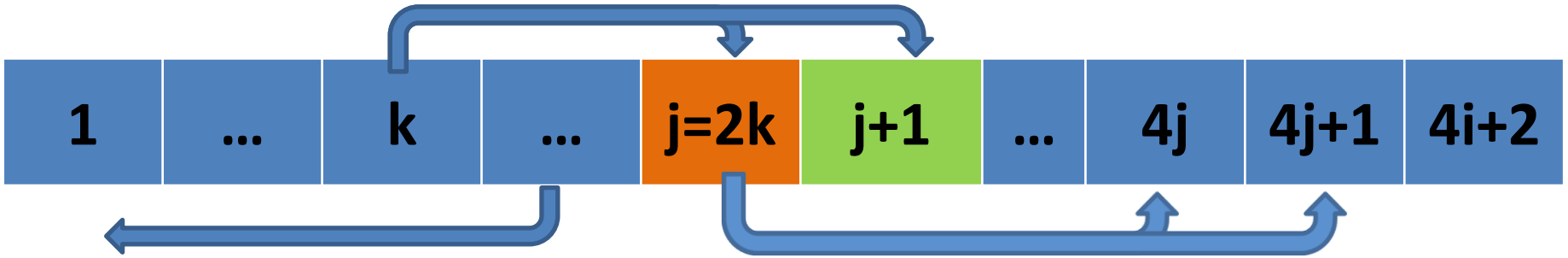
```
template <typename T> void Quick3Sort(T a[], int l, int r)
{  exch(a[(l+r)/2],a[r-1]); compexch(a[l],a[r]);
   if(!compexch(a[l],a[r-1])) compexch(a[r-1],a[r]);
int i=l-1,j=r-1; T x=a[j];
   for(;;) {   while(a[++i]<x);
               while(x<a[--j]) if(j==l) break;
               if(i>=j)break; exch(a[i],a[j]); }
   exch(a[i],a[r-1]);
   if(l<i-1)Quick3Sort(a,l,i-1);
   if(r>i+1)Quick3Sort(a,i+1,r); }
```

```
template <typename T> void QuickISort(T a[], int l, int r)
{ const int M=10; if(r<=l) return;
    if(r-l<=M) InsertSort(a,l,r);
    else {
      exch(a[(l+r)/2],a[r-1]); compexch(a[l],a[r]);
      if(!compexch(a[l],a[r-1])) compexch(a[r-1],a[r]);
      int i=l-1,j=r-1; T x=a[j];
      for(;;) { while(a[++i]<x); while(x<a[--j]) if(j==l) break;
                  if(i>=j)break; exch(a[i],a[j]); }
      exch(a[i],a[r-1]);
      QuickISort(a,l,i-1); QuickISort(a,i+1,r); } }
```

```
template <typename T>
                    void Mergence(T a[], int l, int m, int r) {
int i,j; static T aux[N];
   for(i=m+1;i>l;i--) aux[i-1]=a[i-1];
   for(j=m;j<r;j++) aux[r+m-j]=a[j+1];
   for(int k=l;k<=r;k++)
                    a[k]=aux[j]<aux[i]?aux[j--]:aux[i++]; }
template <typename T> void MergeSort(T a[], int l, int r)
{  if(r<=l) return; int m=(r+l)/2;
   MergeSort(a,l,m); MergeSort(a,m+1,r);
   Merge(a,l,m,r); }
```



```
template <typename T> void fixDown(T a[],int k,int N)
{  int j; while((j=2*k)<=N) {
                    if(j<N && a[j]<a[j+1]) j++;
                    if(!compexch(a[j],a[k])) break;
                    k=j; } }
template <typename T> void HeapSort(T a[],int l,int r)
{  int N=r-l+1; T* pq=a+l-1;
   for(int k=N/2; k>=1; k--) fixDown(pq,k,N);
   while(N>1) { exch(pq[1],pq[N]); fixDown(pq,1,--N); } }
```



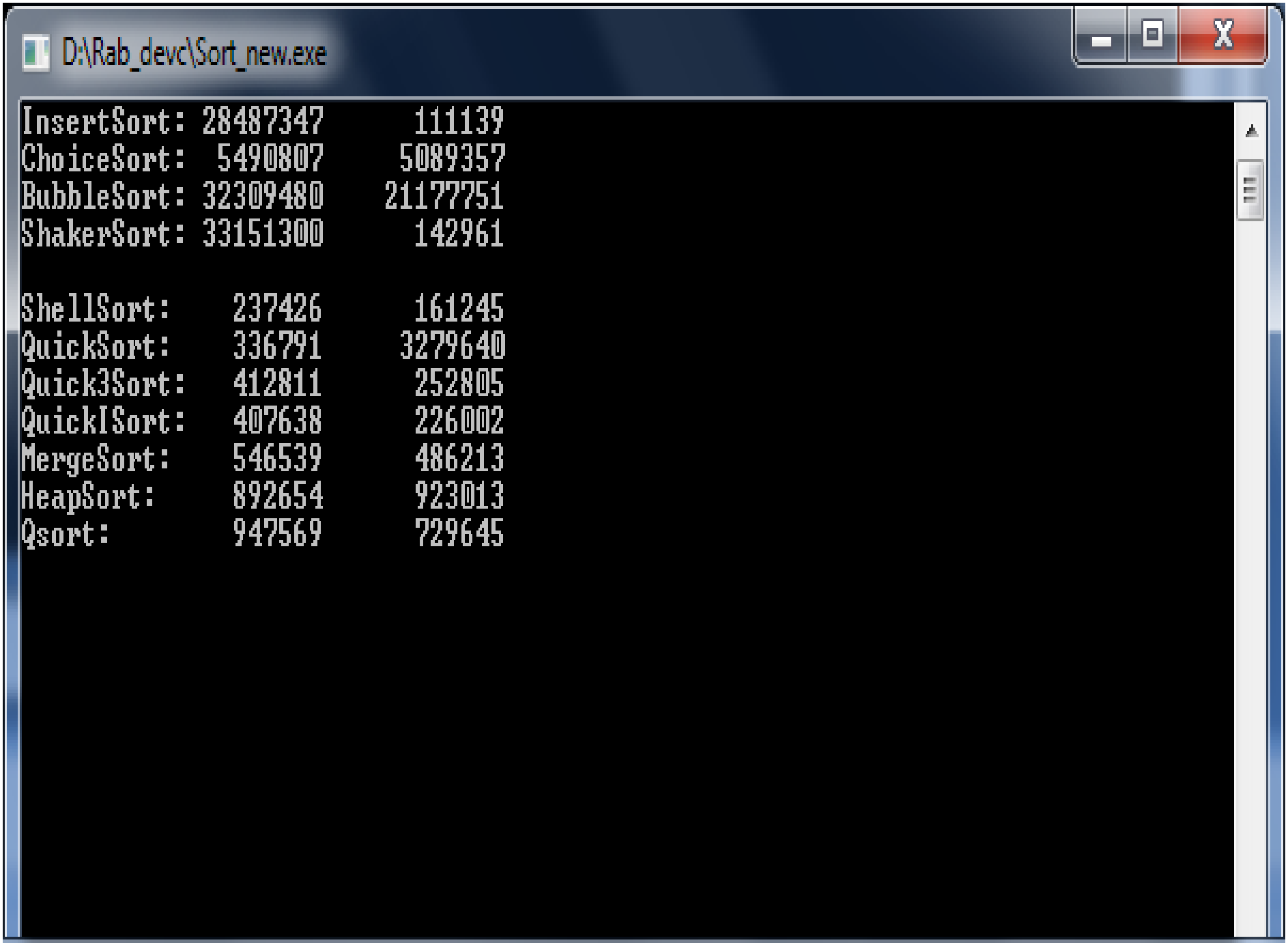
Библиотечная функция qsort

```
void qsort(void* a, size_t N, size_t sizeof(Type),
   int (*comp)(const void*, const void*));

int comp(const void* i, const void* j) {
   double r = *(double*)i - *(double*)j;
   return r>0?1:(r<0?-1:0); }
```

```
int main() {
double a[N];
char* Name[] = { "InsertSort: ", "ChoiceSort: ",
   "BubbleSort: ", "ShakerSort: ", "\nShellSort:  ",
   "QuickSort:   ", "Quick3Sort: ", "QuickISort: ",
   "MergeSort:  ", "HeapSort:   "};
void (*fp[])(double*,int,int) = { InsertSort,
   ChoiceSort, BubbleSort, ShakerSort, ShellSort,
    QuickSort, Quick3Sort, QuickISort, MergeSort,
   HeapSort};
```

```
for(int i=0;i<10;i++) {
    init_sin(a);
    start=access_counter();
    fp[i](a,0,N-1);
    cout << Name[i] << setw(8) << access_counter()-start;
    TEST(a);
    init_x(a);
    start=access_counter();
    fp[i](a,0,N-1);
    cout << '\t' << setw(8) << access_counter()-start << endl;
}
```

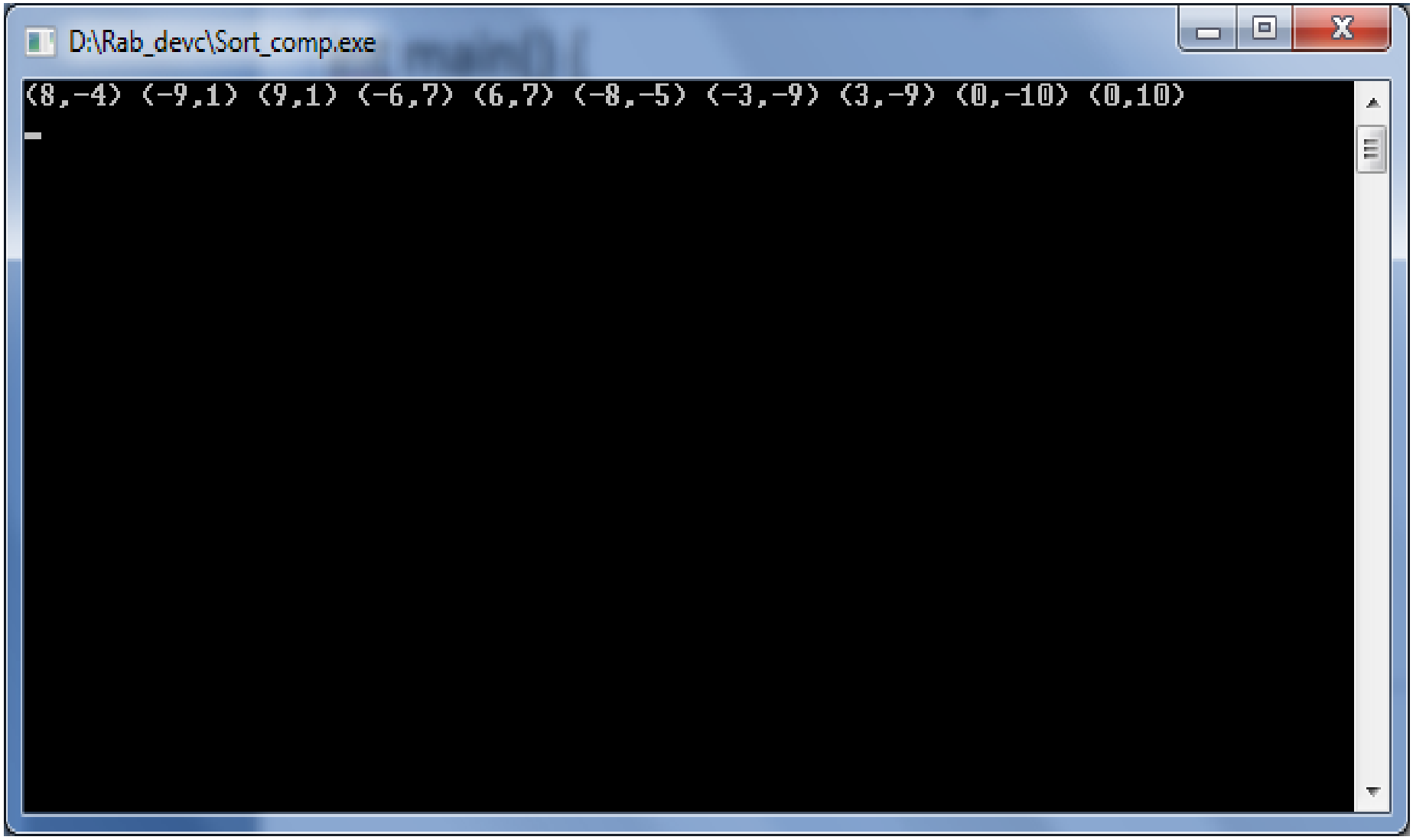
```
 init_sin(a);
    start=access_counter();
    qsort(a,N,sizeof(double),comp);
    cout << "Qsort:     " << setw(8) << access_counter()-start;
    TEST(a);
    init_x(a);
    start=access_counter();
    qsort(a,N,sizeof(double),comp);
    cout << '\t' << setw(8) << access_counter()-start << endl;

    return 0;
}
```


D:\Rab_devc\Sort_new.exe
InsertSort: 28487347 111139
ChoiceSort: 5490807 5089357
BubbleSort: 32309480 21177751
ShakerSort: 33151300 142961
ShellSort: 237426 161245
QuickSort: 336791 3279640
Quick3Sort: 412811 252805
QuickISort: 407638 226002
MergeSort: 546539 486213
HeapSort: 892654 923013
Qsort: 947569 729645

```
template <typename T> class complex {
      T Re,Im;
public:
     complex(T r=0, T i=0) { Re=r; Im=i; }
     friend ostream& operator<< (ostream& a, complex<T> const& b)
                    {  a << '(' << b.Re << ',' << b.Im << ')'; return a;  }
     bool operator<(complex<T> b) {
          T m1=Re*Re+Im*Im, m2=b.Re*b.Re+b.Im*b.Im;
          if(m1==m2) if(Re==b.Re) return Im<b.Im; else return Re<b.Re;
          else return m1<m2; }
     bool operator==(complex<T> b) { return Re==b.Re && Im==b.Im; }
};
```

```
template <typename Item> int compexch(Item& A, Item& B) {
          if(A<B || A==B) return 0; else {exch(A,B); return 1;} }
```

```
template <typename Item> void print(Item a[]) {
        for(int i=0;i<N;i++) cout << a[i] << (i==N-1?'\n':' '); }

template <class T>
void init(complex<T> a[]) { double h=2.*M_PI/(N-1);
for(int i=0;i<N;i++)
a[i]=complex<T>(T(10.*sin(h*i)),T(i==9?-0.:10.*cos(h*i)));}

int main() {
complex<int> *a;
   init(a = new complex<int>[N=10]);
   QuickSort(a,0,N-1);
   print( a );
   return 0;
}
```


D:\Rab_devc\Sort_comp.exe
(8,-4) (-9,1) (9,1) (-6,7) (6,7) (-8,-5) (-3,-9) (3,-9) (0,-10) (0,10)