САА – Упражнение 4

Сравнителен анализ на алгоритми за сортиране

Дадени са програмните реализации на следните алгоритми за сортиране:

- Сортиране чрез пряко вмъкване (вмъкването на елемента на подходяща позиция става чрез последователно търсене);

```
void sort_vmykvane(int n, t_element A[])
{
	int i,j;
	t_element p;
	for(i=1; i<n; i++)
	{
		p=A[i];
		j=i-1;
		while((A[j]>p)&&(j>=0))
		{
			A[j+1]=A[j];
			j--;
		}
		A[j+1]=p;
	}
}
```

- Сортиране чрез пряко вмъкване (вмъкването на елемента на подходяща позиция става чрез двоично търсене);

```
int DvoichnoTyrsene(int l, int r, t_element x, t_element A[])
{
	int m;
	while(l<=r)
	{
		m=(l+r)/2;
		if (x<A[m])
			r=m-1;
		else
			l=m+1;
	}
	return l;
}

void sort_vmykvane_d(int n, t_element A[])
{
	int i,j, j1;
	t_element p;
	for(i=1; i<n; i++)
	{
		p=A[i];
		j1=DvoichnoTyrsene(0,i-1,p,A);
		for(j=i-1; j>=j1; j--)
			A[j+1]=A[j];
```

```
                A[j1]=p;
        }
}
```

- Сортиране чрез пряка размяна (метод на мехурчето);

```
void razmiana(t_element & x, t_element & y)
{
        t_element p;
        p=x;
        x=y;
        y=p;
}

void mehurche(int n, t_element A[])
{
        int j, i;
        for(i=0; i<n-1; i++)
        {
                for(j=n-1; j>=i+1; j--)
                {
                        if (A[j]<A[j-1])
                                razmiana(A[j],A[j-1]);
                }
        }
}
```

- Сортиране чрез алгоритъм на Шел.

```
void Shellsort(t_element A[], int n)
{
        int delta, j, i;
        t_element R;
        bool flag;
        delta = n/2;
        while(delta>0)
        {
                for(i=delta; i<n; i++)
                {
                        R=A[i];
                        j=i;
                        flag=true;
                        while((j-delta>=0)&&(flag))
                        {
                                if (R<A[j-delta])
                                {
                                        A[j]=A[j-delta];
                                        j=j-delta;
                                }
                                else
                                {
                                        flag=false;
                                }
                        }
                        A[j]=R;
```

```
                }
                delta=delta/2;
        }
}
```

Задачи:

1. Реализирайте конзолно приложение, което сравнява времето за изпълнение на всеки от описаните по-горе четири алгоритъма за сортиране и алгоритмите за бързо сортиране и цифрова сортировка върху цели числа. За целта използвайте масив от случайно генерирани цели числа с
   А) 1000 елемента;
   Б) 2000 елемента;
   В) 5000 елемента;
   Г) 10000 елемента;
   Д) 20000 елемента;
   Е) 50000 елемента.

   За отчитане на времето за изпълнение на всеки от алгоритмите използвайте функцията GetTickCount(), която връща като резултат времето в милисекунди от началото на стартиране на системата (библиотека windows.h) .
   В главната функция преди и след извикване на съответната функция за сортиране извикайте функцията GetTickCount() и разликата от двете стойности определете като време за изпълнение на съответната функция за сортиране.
   За генериране на масива от случайни числа можете да използвате следните функции (библиотеки <stdlib.h> и <time.h>):

   ```
   srand( (unsigned)time( NULL ) );
   rand ();
   ```

2. Реализирайте конзолно приложение, което сравнява времето за изпълнение на всеки от описаните по-горе четири алгоритъма за сортиране и алгоритмъма за бързо сортиране върху реални числа. За целта използвайте масив от случайно генерирани реални числа с
   А) 1000 елемента;
   Б) 2000 елемента;
   В) 5000 елемента;
   Г) 10000 елемента;
   Д) 20000 елемента;
   Е) 50000 елемента.

3. Подредете алгоритмите по възходящ ред според времето, необходимо за тяхното изпълнение. Анализирайте резултатите, направете изводи.