**Алгоритм экспериментального исследования алгоритмов сортировки**

**(лабораторная работа № 5)**

**Именованные константы**

KolRazm – число исследуемых размеров таблиц

MaxTbl – максимальный размер массива для таблицы (должен быть кратным KolRazm)

Step = MaxTbl **div** KolRazm – шаг изменения размера исследуемой таблицы

Beg = Step – начальный размер таблицы

MaxExp = 1000 – число экспериментов

KolAlg – число исследуемых алгоритмов

//*Обнуление результатов*

**for** i:=1 **to** KolAlg **do**

**for** j:=1 **to** KolRazm **do**

**begin**

Rez[i,j].Srav:=0;

Rez[i,j].Peres:=0;

Rez[i,j].Vrem:=0;

**end**;

n:=Beg; Razm:=1;

**while** n <= MaxTbl **do**

**begin**

//*заполнение таблиц данными*

**for** i:=1 **to** n **do** Tbl[i]:=i; //*заполнение таблицы числами*

**for** NExp:=1 **to** MaxExp **do** //*проведение эксперимента*

**begin**

//*порождение случайной перестановки*

Randomize;

**for** i:=n **downto** 2 **do**

**begin**

temp:=Tbl[i];

j:=random(i-1)+1;

Tbl[i]:=Tbl[j];

Tbl[j]:=temp;

**end**;

//*последовательно выполняются исследуемые алгоритмы*

{1} Vstavki(n,RezSort); //*простая вставками*

Rez[1,Razm].Srav:=Rez[1,Razm].Srav+RezSort.Srav;

Rez[1,Razm].Peres:=Rez[1,Razm].Peres+RezSort.Peres;

Rez[1,Razm].Vrem:=Rez[1,Razm].Vrem+RezSort.Vrem;

{2} Bubble(n,RezSort); //*пузырьковая*

Rez[2,Razm].Srav:=Rez[2,Razm].Srav+RezSort.Srav;

Rez[2,Razm].Peres:=Rez[2,Razm].Peres+RezSort.Peres;

Rez[2,Razm].Vrem:=Rez[2,Razm].Vrem+RezSort.Vrem;

{3} Select(n,RezSort); //*простая выбором*

Rez[3,Razm].Srav:=Rez[3,Razm].Srav+RezSort.Srav;

Rez[3,Razm].Peres:=Rez[3,Razm].Peres+RezSort.Peres;

Rez[3,Razm].Vrem:=Rez[3,Razm].Vrem+RezSort.Vrem;

{4} Shell(n,RezSort); //*Шелла*

Rez[4,Razm].Srav:=Rez[4,Razm].Srav+RezSort.Srav;

Rez[4,Razm].Peres:=Rez[4,Razm].Peres+RezSort.Peres;

Rez[4,Razm].Vrem:=Rez[4,Razm].Vrem+RezSort.Vrem;

{5} QuickRec(n,RezSort); //*быстрая рекурсивный*

Rez[5,Razm].Srav:=Rez[5,Razm].Srav+RezSort.Srav;

Rez[5,Razm].Peres:=Rez[5,Razm].Peres+RezSort.Peres;

Rez[5,Razm].Vrem:=Rez[5,Razm].Vrem+RezSort.Vrem;

{6} QuickIter(n,RezSort); //*быстрая итерационный*

Rez[6,Razm].Srav:=Rez[6,Razm].Srav+RezSort.Srav;

Rez[6,Razm].Peres:=Rez[6,Razm].Peres+RezSort.Peres;

Rez[6,Razm].Vrem:=Rez[6,Razm].Vrem+RezSort.Vrem;

{7} Piram(n,RezSort); //*пирамидальная*

Rez[7,Razm].Srav:=Rez[7,Razm].Srav+RezSort.Srav;

Rez[7,Razm].Peres:=Rez[7,Razm].Peres+RezSort.Peres;

Rez[7,Razm].Vrem:=Rez[7,Razm].Vrem+RezSort.Vrem;

**end**;

//*подсчет среднего числа операций*

**for** i:=1 **to** KolAlg **do**

**begin**

Rez[i,Razm].Srav:=Round(Rez[i,Razm].Srav/MaxExp);

Rez[i,Razm].Peres:=Round(Rez[i,Razm].Peres/MaxExp);

Rez[i,Razm].Vrem:=Round(Rez[i,Razm].Vrem/MaxExp);

**end**;

n:=n+Step; //*увеличение размера исследуемой таблицы*

Razm:=Razm+1; //*номер столбца результатов*

**end**;

**Примечания**

Процедуры сортировки возвращают результат как выходной параметр типа

TRezSort = **record** Srav, Peres : real; Vrem : Int64; **end**;

В начале каждой процедуры сортировки создается копия исходной перестановки, которая затем и сотируется.

Элемент таблицы результатов исследования (массив Rez) имеет тип TRezSort.