МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«Сибирский государственный аэрокосмический университет

имени академика М. Ф. Решетнева»

(СибГАУ)

Кафедра информатики и вычислительной техники

Отчет

по лабораторной работе №1

по дисциплине "Алгоритмы и структуры данных"

Выполнил: студент гр. БПИ12-01

Овсянников А.К. Путилин В.О.

Проверил: Тынченко В.В.

Красноярск, 2013

**Вариант 15.**

Программно реализовать следующие методы сортировки данных в оперативной памяти: корпоративная сортировка; быстрая сортировка.

Оценить быстродействие указанных методов и степень естественности их поведения.

1)

Данные результаты представлены о быстрой сортировки, так же результат можно найти в файле fast.txt

2)

Данные результаты представлены о корпоративной сортировки, так же результат можно найти в файле test1.txt

По данным графикам можно заметить, что при быстрой сортировки затрачивается намного меньше времени, чем при корпоративной сортировки.

**Программный код**

1. main.cpp

#include <iostream>//cout,cin

#include <locale.h>//руссификация вывода в консоль

using namespace std;//стандартное пространство имён

int singlesort();//первый режим

int statdata(); //второй режим

int main()//точка входа в программу

{

setlocale(LC\_ALL,"Russian");//руссификация консоли

bool flag = true;

char otv;

//цикл работы программы

while (flag)

{

cout<<"Меню:\n";

cout<<"1.Сортировка массива\n2.Накопление статистики\n3.Выход\n";

cin>>otv;

cin.clear();

cin.sync();

switch(otv)

{

case '1'://вызов первого режима программы

singlesort();

break;

case '2'://вызов второго режима программы

statdata();

break;

case '3'://выход из программы

flag=false;

break;

default: flag=true;

}

cin.clear();

cin.sync();

cout<<"\n";

}

return 0;//возврат из функции

}

1. func.cpp

#include <time.h>//работа со временем

#include <stdlib.h>//стандартная библиотека

//заполнение массива

void zap(int \*arr, int size, char metk)

{

srand(time(0));//начало отсчёта времени с нуля

unsigned long i;

switch(metk)

{

case '1'://заполнение случайными значениями

for(i=0;i<size;i++)

arr[i] = rand()%2001 -2000;

break;

case '2'://заполнение в порядке возрастания

for(i=0;i<size;i++)

arr[i] = i;

break;

case '3': //заполнение в порядке убывания

for(i=0;i<size;i++)

arr[i] = size-i-1;

break;

}

}

1. methods.cpp

#include <iostream>//cout,cin

#include <stdio.h>

#include <stdlib.h>//стандартная библиотека

#define MAXSTACK 2048//деректива препроцессора (замена MAXSTACK на 2048)

using namespace std;//стандартное пространство имён

unsigned long rev, check;//глобальные переменные

void downHeap(int\*arr, long k, long n);//корпоротивная сортировка построение пирамиды

//корпоративная сортировка

void korp\_sotr(int\*arr, long size, int\* reversion, int\* checks)

{

long i;

int temp;

rev = 0;//счётчик перестановок

check = 0;//счётчик проверок

// строим пирамиду

for(i=size/2-1; i >= 0; i--) downHeap(arr, i, size-1);

// теперь a[0]...a[size-1] пирамида

for(i=size-1; i > 0; i--)

{

rev++;

// меняем первый с последним

temp=arr[i]; arr[i]=arr[0]; arr[0]=temp;

// восстанавливаем пирамидальность a[0]...a[i-1]

downHeap(arr, 0, i-1);

}

\*reversion = rev;

\*checks = check;

}

//построение пирамиды

void downHeap(int\*arr, long k, long n)

{

// процедура просеивания следующего элемента

// До процедуры: a[k+1]...a[n] - пирамида

// После: a[k]...a[n] - пирамида

int new\_elem;

long child;

new\_elem = arr[k];

while(k <= n/2)

{ // пока у a[k] есть дети

child = 2\*k;

// выбираем большего сына

if( child < n && arr[child] < arr[child+1] )

{

check++;

child++;

}

else

{

check++;

}

if( new\_elem >= arr[child] )

{

check++;

break;

}

else

{

check++;

rev++;

arr[k] = arr[child]; // переносим сына наверх

k = child;

}

}

arr[k] = new\_elem;

}

//быстрая сортировка

void fast\_sort(int\*arr, long size, int\* reversion, int\* checks)

{

rev = 0;

check = 0;

long i, j; // указатели, участвующие в разделении

long lb, ub; // границы сортируемого в цикле фрагмента

long lbstack[MAXSTACK], ubstack[MAXSTACK]; // стек запросов

// каждый запрос задается парой значений,

// а именно: левой(lbstack) и правой(ubstack)

// границами промежутка

long stackpos = 1; // текущая позиция стека

long ppos; // середина массива

double pivot; // опорный элемент

int temp;

lbstack[1] = 0;

ubstack[1] = size-1;

do {

// Взять границы lb и ub текущего массива из стека.

lb = lbstack[ stackpos ];

ub = ubstack[ stackpos ];

stackpos--;

do {

// Шаг 1. Разделение по элементу pivot

ppos = ( lb + ub ) >> 1;

i = lb;

j = ub;

pivot = arr[ppos];

do {

while ( arr[i] < pivot ) i++;

while ( pivot < arr[j] ) j--;

if ( i <= j ) {

check++;

temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

rev++;

i++;

j--;

}

} while ( i <= j );

// Сейчас указатель i указывает на начало правого подмассива,

// j - на конец левого (см. иллюстрацию выше), lb ? j ? i ? ub.

// Возможен случай, когда указатель i или j выходит за границу массива

// Шаги 2, 3. Отправляем большую часть в стек и двигаем lb,ub

if ( i < ppos )

{ // правая часть больше

check++;

if ( i < ub )

{ // если в ней больше 1 элемента - нужно

check++;

stackpos++; // сортировать, запрос в стек

lbstack[ stackpos ] = i;

ubstack[ stackpos ] = ub;

}

ub = j; // следующая итерация разделения

// будет работать с левой частью

}

else

{ // левая часть больше

if ( j > lb )

{

check++;

stackpos++;

lbstack[ stackpos ] = lb;

ubstack[ stackpos ] = j;

}

lb = i;

}

} while ( lb < ub ); // пока в меньшей части более 1 элемента

} while ( stackpos != 0 ); // пока есть запросы в стеке

\*reversion = rev;

\*checks = check;

}

1. singlesort.cpp

#include <iostream>//cout,cin

#include <stdlib.h>//стандартная библиотека

#include <time.h>//работа со временем

#include <locale.h>//руссификация консоли

#include <ctime>//clock()

using namespace std;///стандартное пространство имён

void korp\_sotr(int\*arr, long size, int\* reversion, int\* checks);//корпоративная сортировка

void fast\_sort(int\*arr, long size, int\* reversion, int\* checks);//быстрая сортировка

int write\_mas(int\*arr, int nlen);//вывод массива

//первый режим

int singlesort()

{

setlocale(LC\_ALL,"Russia");//руссификация консоли

srand(time(0));//начало отсчёта времени с нуля

int RangeStart,RangeEnd, i, reversion, checks;

long n;

char otv;

bool flag = true;

unsigned int start\_time, end\_time;

double serch\_time;

//размер массива

printf("Введите размер массива\n");

cin>>n;

int \*mas = new int[n];//динамический массив

//диапозон щначений

cout<<"Укажите диапозон значений массива\n";

cin>>RangeStart>>RangeEnd;

//заполнение массива

for(i=0;i<n;i++)

if(RangeStart>=0)

mas[i] = rand()%(RangeEnd - RangeStart) + RangeStart;

else

mas[i] = rand()%(RangeEnd - RangeStart + 1) + RangeStart;

//выбор метода сортировки

while(flag)

{

cout<<"Выберите метод сортировки\n";

cout<<"1.Корпоративный\n2.быстрый\n";

cin>>otv;

cin.clear();

cin.sync();

flag = false;

switch(otv)

{

case '1'://корпоративная сортировка

printf("Вывод исходного массива\n");

write\_mas(mas, n);//вывод массива

start\_time = clock();

korp\_sotr(mas, n, &reversion, &checks);//корпоративная сортировка

end\_time = clock();

printf("Вывод отсортированного массива\n");

write\_mas(mas, n); //вывод массива

//время работы сортировки

cout<<"Время пирамидальной сортировки: ";

serch\_time = (end\_time-start\_time)/1000.0;

cout<<serch\_time<<" сек\n";

break;

case '2'://быстрая сортировка

printf("Вывод исходного массива\n");

write\_mas(mas, n);//вывод массива

start\_time = clock();

fast\_sort(mas, n, &reversion, &checks); //быстрая сортировка

end\_time = clock();

printf("Вывод отсортированного массива\n");

write\_mas(mas, n);//вывод массива

//время работы сортировки

cout<<"Время быстрой сортировки: ";

serch\_time = (end\_time-start\_time)/1000;

cout<<(serch\_time)<<" сек\n";

break;

default: flag = true;

}

cout<<"Количество перестановок: "<<reversion<<"\n";

cout<<"Количество проверок: "<<checks<<"\n";

}

delete[] mas;//освобождение памяти

return 0;

}

//вывод массива

int write\_mas(int\*arr, int nlen)

{

int i,k = 12;

char file\_name\_out[45];

FILE \*out;

puts("Введите имя файл для вывода массива");

gets(file\_name\_out);

//проверка на возможность открыть файл

if((out = fopen(file\_name\_out,"a+"))==0)

{

fprintf(stderr,"Невозможно открыть файл %s\n",file\_name\_out);

return 1;

}

//вывод массива в файл

for (i = 0; i<nlen; i++)

{

fprintf(out,"%d, ",arr[i]);

if(i == k)

{

fprintf(out,"\n");

k+=12;

}

}

fprintf(out,"\n\n\n");

fclose(out);//закрытие файла

return 0;

}

1. statdata.cpp

#include <iostream>//cout,cin

#include <stdlib.h>//стандартная библиотека

#include <time.h>//работа со временем

#include <locale.h>//руссификация консоли

#include <ctime>//clock()

using namespace std;//стандартное пространство имён

void korp\_sotr(int\*arr, long size, int\* reversion, int\* checks);//корпоративная сортировка

void fast\_sort(int\*arr, long size, int\* reversion, int\* checks);//быстрая сортировка

void zap(int \*arr, int size, char metk);//заполнение массива

int write\_stat(int size, double time, char\* file\_name); //вывод в файл результатов

//второй режим

int statdata()

{

setlocale(LC\_ALL,"Russia");//руссификация

srand(time(0));//начало отсчёта времени с нуля

bool flag1 = true, flag = true, flag2 = true;

char otv, otv\_sp, otv\_method;

int RangeStart,RangeEnd, step, reversion, checks, n;

unsigned int start\_time, end\_time;

double serch\_time;

char str[45];

//способ заполнения массива

while(flag1)

{

cout<<"Введите способ заполнения массива\n";

cout<<"1.Случайными значениями\n2.В порядке возрастания\n3.Натуральные в порядке убывания\n";

cin>>otv\_sp;

cin.clear();

cin.sync();

if ((otv\_sp == '1')||(otv\_sp == '2')||(otv\_sp == '3'))

flag1 = false;

}

//диапозон значений

cout<<"Укажите диапозон размеров массива\n";

cin>>RangeStart>>RangeEnd;

//шаг изменения размеров массива

cout<<"Укажите шаг изменения размеров массива\n";

cin>>step;

//метод сортировки

while(flag)

{

cout<<"Выберите метод сортировки\n";

cout<<"1.Корпоративный\n2.быстрый\n";

cin>>otv\_method;

cin.clear();

cin.sync();

if ((otv\_method == '1')||(otv\_method == '2'))

flag = false;

}

puts("Введите имя файл для вывода результотов программы");

gets(str);

n = RangeStart;

while (flag2)

{

int \*mas = new int[n];//выделение памяти динамическому массиву

zap(mas, n, otv\_sp);//заполнение массива

switch(otv\_method)

{

case '1'://корпоративная сортировка

start\_time = clock();

korp\_sotr(mas, n, &reversion, &checks);

end\_time = clock();

break;

case '2'://быстрая сортировка

start\_time = clock();

fast\_sort(mas, n, &reversion, &checks);

end\_time = clock();

break;

}

delete[] mas; //освобождение памяти от массива

serch\_time = (end\_time-start\_time)/\*/1000.0\*/;//время сортировки

write\_stat(n,serch\_time,str);//запись результатов

n+=step;

if(n>=RangeEnd)

flag2 = false;

}

return 0;

}

//вывод в файл

int write\_stat(int size, double time, char\* file\_name\_out)

{

FILE \*out;

//проверка на возможность открытия файла

if((out = fopen(file\_name\_out,"a+"))==0)

{

fprintf(stderr,"Невозможно открыть файл %s\n",file\_name\_out);

return 1;

}

//вывод в файл размер массива и времени работы

fprintf(out,"%d\t\t",size);

fprintf(out,"%f\n",time);

fclose(out);

return 0;

}