1. ЛАБОРАТОРНАЯ РАБОТА ПО ПРОГРАММИРОВАНИЮ УЧЕНИКА 10д КЛАССА ШКОЛЫ N57

АХМАНОВА СЕРГЕЯ ПО ТЕМЕ "СОРТИРОВКИ".

2. ПОСТАНОВКА ЗАДАЧИ.

Дан файл, содержащий числа типа longint, расположенные в произвольном порядке. Требуется расположить эти числа по возрастанию, используя не более 40 килобайт оперативной памяти и дискового пространства не более чем в два раза больше исходного файла.

3. АЛГОРИТМ (метод решения).

Сначала исходный файл разбивается на куски по 10000 чисел, каждый кусок сортируется в памяти и записывается в один из двух временных файлов, причем так, что количество кусков в этих файлах отличается не более чем на 1(далее - первоначальная сортировка).

Затем, несколько раз выполняется операция "склеивание"(одно выполнение операции "склеивание" мы будем незывать "шаг"), т.е два исходных файла, в которых находились отсортированные куски копируются в два других файла, при этом из двух кусков, находящихся в разных файлах и имеющих одинаковые номера создается один отсортированный кусок. Этот кусок записывается в первый выходной файл если исходные куски имели нечетные номера и во второй, если исходные куски имели четные номера.

4. ВНУТРЕННЯЯ СПЕЦИФИКАЦИЯ ПРОГРАММЫ.

При написании программы использовалась среда Borland Pascal 7.0 и встроенный компилятор.

Для ускоренного обмена с диском применялся блоковый ввод-вывод, т.е информация читается и записывается целыми кластерами. Для осуществления этого способа ввода-вывода был написан модуль(Files), с помощью которого ввод-вывод внешне не отличается от обычного.

Схема программы предельно проста: сначала выполняется первоначльная сортировка(процедура firstsort), затем вызываем склеивание(процедура ftrans(in1, in2, out1, out2: workfile);), где пары файлов все время меняются и после каждого запуска процедуры проверяется условие выхода.

Процедура ftrans открывает все файлы, затем выполняет несколько раз процедуру слива одного куска(onestep) и закрывает файлы.

5. КОММЕНТИРОВАННЫЙ ТЕКСТ ПРОГРАММЫ.

Модуль Files.

Сдесь переписаны все процедуры и функции необходимые для работы с файлами, работающие с блоками. Работа с ними осуществляется также как и с обычными процедурами модуля System.

unit Files;

interface

const typesize=4;

const bufsize = 2048;

type using=longint;

type buffer = array[1..bufsize] of using;

type pbuffer = ^buffer;

type filemode = (fread, fwrite, closed);

type tfile = record

buf: pbuffer;

mode: filemode;

f: file;

count, leng: integer;

end;

procedure fAssign(var w: tfile; name: string);

procedure fReWrite(var w: tfile);

procedure fReset(var w: tfile);

procedure fPut(var w: tfile; d: using);

procedure fGet(var w: tfile; var d: using);

procedure fClose(var w: tfile);

function fEof(var w: tfile): boolean;

implementation

procedure fAssign(var w: tfile; name: string);

begin

Assign(w.f, name); w.mode:=closed;

end;

procedure fReWrite(var w: tfile); begin

if w.mode=closed then

begin

ReWrite(w.f, typesize); new(w.buf); w.count:=0; w.leng:=0; w.mode:=fwrite;

end;

end;

procedure fReset(var w: tfile); begin

if w.mode=closed then

begin

Reset(w.f, typesize); new(w.buf);

BlockRead(w.f, w.buf^, bufsize, w.leng); w.count:=1;

w.mode:=fread;

end;

end;

procedure fPut(var w: tfile; d: using);

begin

if w.mode=fwrite then

begin

w.count:=w.count+1;

w.buf^[w.count]:=d;

if w.count=bufsize then

begin

BlockWrite(w.f, w.buf^, w.count); w.count:=0;

end;

end;

end;

procedure fGet(var w: tfile; var d: using); begin

if (w.mode=fread) then

begin

d:=w.buf^[w.count];

if w.leng=w.count then

begin

BlockRead(w.f, w.buf^, bufsize, w.leng); w.count:=1;

end else w.count:=w.count+1;

end;

end;

procedure fClose(var w: tfile);

begin

if w.mode=fwrite then BlockWrite(w.f, w.buf^, w.count); dispose(w.buf);

w.mode:=closed;

Close(w.f); end;

function fEof(var w: tfile): boolean;

begin

if (w.mode=fread) and (w.leng=0) then fEof:=true

else fEof:=false;

end;

begin

end.

конец files.pas

----------------------------------------------------------------------------

Файл sort.pas - сортировка в памяти.

var k: integer;

function SwapTops(no: integer): integer;

var t: longint;

begin

if (memo^[2\*no+1]>memo^[2\*no]) then

begin

t:=memo^[no];

memo^[no]:=memo^[2\*no+1];

memo^[2\*no+1]:=t;

SwapTops:=2\*no+1; end else begin

t:=memo^[no];

memo^[no]:=memo^[2\*no];

memo^[2\*no]:=t;

SwapTops:=2\*no; end;

end;

procedure SwapHalf(no: integer);

var t: longint;

begin

if memo^[no]<memo^[2\*no] then

begin

t:=memo^[no];

memo^[no]:=memo^[2\*no];

memo^[2\*no]:=t;

end;

end;

function Reg(no: integer): boolean;

begin

if (2\*no)>k then Reg:=true else

if (2\*no+1)>k then

begin

SwapHalf(no);

Reg:=true; end else

if (memo^[2\*no]<=memo^[no]) and (memo^[2\*no+1]<=memo^[no]) then Reg:=true

else Reg:=false;

end;

procedure HalfReg(no: integer);

var next: integer;

begin

next:=no;

while (not Reg(next)) do next:=SwapTops(next);

end;

procedure RegTree;

var i: integer;

begin

for i:=k downto 1 do HalfReg(i);

end;

procedure SwapLeaves(l1, l2: integer);

var t: longint;

begin

t:=memo^[l1];

memo^[l1]:=memo^[l2];

memo^[l2]:=t;

end;

procedure SortMemo(len: integer);

begin

k:=len;

RegTree;

for k:=len-1 downto 1 do

begin

SwapLeaves(1, k+1);

HalfReg(1); end;

end;

конец sort.pas

----------------------------------------------------------------------------

Основная пограмма

uses Dos, FilesПодключение модуля, осуществляющего ввод-вывод.;

const memlen=10000;Размер памяти, разрешенной для использования

type tmemo = array[0 .. memlen] of longint;

type pmemo = ^ tmemo;Тип-указатель на основной массив, используемый

программой

var memo : pmemo;

$I sort.pas Подключение файла, содержащего процедуру сортировки

массива за время n\*(log n), не используя дополнительной памяти(сортировка

деревом).

type workfile = record

mainосновной файл,

infфайл, содержащий длины отсортированных кусков: tfile;

end;tfile - тип, определенный в unit Files, который заменяет файловые типы

var

t1, t2, t3, t4, dest, seur: workfile;

временные файлы входной и выходной файл

Инициализация

procedure Init;

var tmp: string;

begin

tmp:=getenv('TEMP');

fAssign(t1.main, tmp+'\~fsort-1.tmp');

fAssign(t2.main, tmp+'\~fsort-2.tmp');

fAssign(t3.main, tmp+'\~fsort-3.tmp');

fAssign(t4.main, tmp+'\~fsort-4.tmp');

fAssign(t1.inf, tmp+'\~finf-1.tmp');

fAssign(t2.inf, tmp+'\~finf-2.tmp');

fAssign(t3.inf, tmp+'\~finf-3.tmp');

fAssign(t4.inf, tmp+'\~finf-4.tmp');

fAssign(seur.main,ParamStr(1));

fAssign(dest.main,ParamStr(2));

end;

Первоначальная сортировка

procedure firstsort(var inp, out1, out2: workfile);

var i, k: longint;

begin

fReset(inp.main);

fRewrite(out1.main);

fRewrite(out2.main);

fRewrite(out1.inf);

fRewrite(out2.inf);

new(memo);

repeat

for i:=1 to memlen do

if fEof(inp.main) then

begin

i:=i-1;

break

end else fGet(inp.main, memo^[i]);

k:=i;

sortmemo(k);

for i:=1 to k do fPut(out1.main, memo^[i]);

fPut(out1.inf, k);

if k=memlen then

begin

for i:=1 to memlen do

if fEof(inp.main) then

begin

i:=i-1;

break;

end

else fGet(inp.main, memo^[i]);

k:=i;

sortmemo(k);

for i:=1 to k do fPut(out2.main, memo^[i]);

fPut(out2.inf, k);

end;

until fEof(inp.main);

dispose(memo);

fClose(inp.main);

fClose(out1.main);

fClose(out2.main);

fClose(out1.inf);

fClose(out2.inf);

end;

Процедура, копирующая заданное количество эл-тов из одного файла в другой.

Используется при сливании для копирования оставшейся части куска(если другой кусок иссяк).

procedure Copy(var inp, out: workfile; c0: longint);

var

c, n: longint;

Done: boolean; begin

for c:=c0 downto 1 do begin

fGet(inp.main, n); fPut(out.main, n);

end;

end;

Сливает два очередных куска из файлов in1 и in2 и записывает в out. procedure onestep(var in1, in2, out: workfile; c01, c02: longint); var n1, n2, c1, c2, c: longint;

Done: boolean; begin

Done:=false; c1:=c01-1; c2:=c02-1; c:=0; fGet(in1.main, n1); fGet(in2.main, n2); repeat

if n1<n2 then begin

fPut(out.main, n1); c:=c+1; if c1=0 then begin

fPut(out.main, n2); c:=c+1; Copy(in2, out, c2); c:=c+c2; Done:=true;

end else

begin

fGet(in1.main, n1); c1:=c1-1;

end;

end else

begin

fPut(out.main, n2);

c:=c+1;

if c2=0 then

begin

fPut(out.main, n1);

c:=c+1;

Copy(in1, out, c1); c:=c+c1; Done:=true;

end else

begin

fGet(in2.main, n2); c2:=c2-1;

end;

end;

until Done;

end;

Процедура осуществляет один шаг(т.е. копирует файлы in1 и in2 в out1 и out2, при этом склеивая куски)

procedure ftrans(var in1,in2,out1,out2: workfile);

var c1, c2, c: longint;

begin

fReset(in1.main);

fReset(in2.main);

fReset(in1.inf);

fReset(in2.inf);

fRewrite(out1.main);

fRewrite(out2.main);

fRewrite(out1.inf);

fRewrite(out2.inf);

while (not fEof(in1.inf)) and (not fEof(in2.inf)) do

begin

fGet(in1.inf, c1);

fGet(in2.inf, c2);

onestep(in1, in2, out1, c1, c2);

c:=c1+c2;

fPut(out1.inf, c);

if (not fEof(in1.inf)) and (not fEof(in2.inf)) then

begin

fGet(in1.inf, c1);

fGet(in2.inf, c2);

onestep(in1, in2, out2, c1, c2);

c:=c1+c2;

fPut(out2.inf, c);

end;

end;

if fEof(in1.inf) xor fEof(in2.inf) then

if fEof(in1.inf) then

begin

fGet(in2.inf, c2);

Copy(in2, out2, c2); fPut(out2.inf, c2);

end else

if fEof(in2.inf) then begin

fGet(in1.inf, c1); Copy(in1, out2, c1); fPut(out2.inf, c1);

end; fClose(in1.main); fClose(in2.main); fClose(in1.inf); fClose(in2.inf); fClose(out1.main); fClose(out2.main); fClose(out1.inf); fClose(out2.inf);

end;

Копирование файла f1 в f2.(Используется при завершении работы для

копирования конечного файла из временной директории в указанную).

procedure FCopy(f1, f2: tfile);

var t: longint;

begin

write('копирование');

fRewrite(f2);

fReset(f1);

while (not fEof(f1)) do

begin

fGet(f1, t);

fPut(f2, t);

end;

fClose(f1);

fClose(f2);

end;

Принимает значение True, если файл отсортирован и больше не надо склеивать.

(Условие выхода)

function Fin: boolean;

begin

fReset(t2.main);

fReset(t4.main);

if fEof(t2.main) then

begin

Fin:=true;

FCopy(t1.main, dest.main); end else

if fEof(t4.main) then

begin

Fin:=true;

FCopy(t3.main, dest.main); end else Fin:=false; fClose(t2.main); fClose(t4.main);

end;

begin

writeln;

if ParamCount<2 then

begin

writeln('Слишком мало параметров.');

Exit; end; write('Инициализация...'); Init; writeln('готово'); write('Первоначальная сортировка...'); firstsort(seur, t1, t2); writeln('готово'); ReWrite(dest.main.f); Close(dest.main.f); writeln('Склеивание:'); repeat

ftrans(t1, t2, t3, t4);

writeln('шаг');

if (not Fin) then

begin

ftrans(t3, t4, t1, t2);

writeln('шаг');

end;

until Fin;

writeln('готово');

end.

----------------------------------------------------------------------------

6. ВНЕШНЯЯ СПЕЦИФИКАЦИЯ.

Для корректной работы программы необходим компьютер AT286, 40K свободной conventional памяти, операционная система MS-DOS 3.0 или более поздняя версия. Возможны версии программы, использующие меньше памяти, процессоры слабее 286 и т.д. Программа использует место на диске вдвое большее исходного файла(не считаяя сам файл).

7. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ.

При запуске программы обязательно должна быть определена переменная среды TEMP!

Формат запуска программы:

f\_sort[.exe] <входной файл> <выходной файл>

Программа не задает ни каких вопросов, что сильно упрощает работу с ней.

Результат работы можно поверить с помощью прилагаемой утилиты f\_check, создать случайный исходный файл - с промощью f\_make.

Причинами ошибок могут служить не соответствие системы требованиям, изложенным в п. 6, недостаточное место на диске, размер(в байтах) исходного файла не кратен 4.

В данном отчете описывается самая эфективная версия этой программы, но существуют версии, не использующие ввод-вывод блоками, требующие меньше ресурсов системы.

8. ОПИСАНИЕ ТЕСТОВ.

Программма тестировалась неодноктатно, на входе использовались, в основном, файлы из случайных чисел различной длины. На выходе были получены файлы тойже длины, не содержащие ошибок, т.е. числа в этох файлах оказались в порядке не строгого возрастания. Содержимое этих файлов полностью совпало с разультатами работы других программ сортировки на тех же входных файлах, что сильно снижает вероятность дифектов программы.

При тестировании использовались операционные системы MS-DOS 6.22, Windows`95, компьютеры PC AT 486DX4-100, 486SX-25, работающие с локальным винчестером, робочие станции 486DX-40, 386SX, работающие в сети Novell.

Результаты тестирования(на файле размером 4M) занесены в таблицу: компьютер работа в сети время работы

486DX4-100 нет 3 мин.

486SX-25 нет 7 мин.

486DX-40 да

386SX да