**Донецкий национальный университет**

**Кафедра компьютерных технологий**

**Отчет**

по лабораторной работе №1

**«**Моделирование представления в памяти ЭВМ векторов и таблиц**»**

по курсу “ ***Структуры и организация данных в ЭВМ*** ”

Выполнил студент

Группы: ИСПР А1

**Присяжный М.Б.**

Преподаватель:

**Демин В.А.**

Донецк - 2011

**Цель работы**

Приобретение и закрепление навыков размещения в памяти векторов и таблиц. Закрепление знаний о модульности программы с точки зрения обрабатываемых данных.

**Темы для предварительной проработки**

Простые структуры данных, статические структуры данных.

1. **Описание алгоритма.**

Программа была реализована в среде Turbo Pascal 7.0

В качестве индивидуального задания была представлена следующая задача:

**Разработать способ экономного размещения в памяти заданной разреженной таблицы. Разработать процедуры/функции, обеспе- чивающие доступ к элементам таблицы по номерам строки и столбца. В матрице все элементы четных столбцов – нулевые. (Вариант 9).**

**Суть алгоритма:**

Используя одномерный массив, заполненный псевдослучайными числами, мы создаём матрицу, четные столбцы которой заполнены фоновыми элементами (равны 0). При этом матрица не записывается в двумерный массив, а лишь выводится для наглядности на экран. Ненулевые значения записываются в одномерный массив, формируя сжатое представление матрицы. В двойном цикле проходим каждую строку матрицы и если позиция текущего элемента четная выводим 0 и не записываем его в массив, если четная – выводим значение, записываем в массив и увеличиваем счетчик массива.

Затем по заданным двумерным координатам определяем номер текущего элемента в одномерном массиве (сжатом представлении матрицы). Если элемент фоновый или выходит за границы матрицы – выводим соответствующее сообщение. Сначала превращаем двумерные индексы в одномерные. Для этого был также использован двойной цикл. Во внешнем мы проходим до строки, содержащей заданный элемент, во внутреннем – до последнего столбца матрицы. Программа проходит до тех пор пока иксовая координата заданного элемента не совпадёт с текущей в матрице. Если совпадение происходит, осуществляем выход из цикла. Во время прохождения цикла программа считает количество всех пройденных элементов четных столбцов. Полученное число и есть нужный одномерный индекс. Далее просто осуществляем поиск элемента в одномерном массиве по полученному индексу, и выводим сам элемент и его позицию в массиве на экран.

1. **Описание программных элементов.**

**Было разработано 5 процедур:**

* **Procedure NewIndex;** - процедура выполняет переход от индексов в матрице - x,y - к одному индексу в одномерном массиве по формуле.

Procedure NewIndex;

begin

for i:=1 to x do

for j:=1 to m do

if (j mod 2<>0) then

if (i=x) and (j>y) then break

else

inc(k);

end;

Здесь x – заданная пользователем х-координата искомого элемента (номер строки), m – первоначальное число столбцов матрицы, к – счетчик (получаемый индекс).

* **Procedure PutTab** - процедура выполняет запись значения value в сжатое представление матрицы на место, определяемое координатами x,y и выводит на экран саму матрицу.

Procedure PutTab;

begin

for i:=1 to n do

begin

for j:=1 to m do

begin

if (j mod 2=0) then

Write('0':3,'|')

else

begin

inc(k);

a[k]:=random(maxval)+1;

Write(a[k]:3,'|');

end;

end;

WriteLn;

end;

end;

Здесь n и m – первоначальные размеры матрицы, a[k] – элемент одномерного массива, к – счетчик массива.

* **Procedure GetTab; -** процедура возвращает значение некоторой переменной, выбранное из матрицы по координатам **x,y**, в том числе и **0** для "**пустых**" мест матрицы.

Procedure GetTab;

begin

repeat

{/////////////////////////////////////////////////////////////}

{Ввод координат искомого элемента}

{/////////////////////////////////////////////////////////////}

DrawLine;

window(1,n+5,80,25);

TextBackground(White);

TextColor(Black);

clrscr;

k:=0;

{$I-}

WriteLn('Enter x and y you search:');

repeat

window(1,n+5,80,25);

clrscr;

Write('x=');

ReadLn(x);

until IOresult=0;

repeat

window(1,n+7,80,25);

clrscr;

Write('y=');

ReadLn(y);

until IOresult=0;

{$I+}

{/////////////////////////////////////////////////////////////////////////////}

{Проверка, фиксирующая фоновые элементы}

{/////////////////////////////////////////////////////////////////////////////}

if y mod 2=0 then empty:=true;

{/////////////////////////////////////////////////////////////////////////////////////////////////////////////////}

{Проверка, фиксирующая элементы, выходящие за рамки матрицы}

{/////////////////////////////////////////////////////////////////////////////////////////////////////////////////}

if (x>n) or (y>m) or (x<1) or (y<1) then

out:=true;

NewIndex;

WriteLn;

{/////////////////////////////////////////////////////////////}

{Вывод результата}

{/////////////////////////////////////////////////////////////}

if not out then

if not(empty) then

WriteLn('Element=',a[k],' found in position ',k)

else WriteLn('Error!!! Empty place!!!')

else WriteLn('Error!!! Element is out of matrix!!!');

WriteLn;

WriteLn('Try Again?');

WriteLn('[Y/N]');

repeat

p:=readkey;

until (p='n') or (p='y') or (p=#27) ;

until p<>'y';

end;

Здесь x,y – задаваемые координаты искомого элемента, a[k] – текущий элемент массива, n и m – размеры матрицы.

* **Procedure InitArray; -** процедура содержит ввод входных данных и вызывает процедуру PutTab, тем самым создавая массив и выводя его на экран.

Procedure InitArray;

begin

repeat

clrscr;

WriteLn('Enter size of array:');

WriteLn;

{$I-}

repeat

window(1,3,80,25);

clrscr;

WriteLn('Enter x <=10:');

ReadLn(n);

until IOresult=0;

WriteLn;

repeat

window(1,5,80,25);

clrscr;

WriteLn('Enter y <=10:');

ReadLn(m);

until IOresult=0;

{$I+}

until (n>1) and (n<=maxnum) and (m>1) and (m<=maxnum);

window(1,1,80,25);

clrscr;

PutTab;

WriteLn;

maxk:=k;

{////////////////////////////////////////////////////////////////////}

{Вывод сжатого представления матрицы}

{////////////////////////////////////////////////////////////////////}

for k:=1 to maxk do

Write(a[k],' ');

end;

* **Procedure DrawLine; -** процедура рисует вспомогательную линию.

В коде используются процедуры из модуля crt, но они, как и процедура **DrawLine** и многочисленные проверки (с использованием IOresult) служат лишь для удобного диалога с пользователем и улучшенного интерфейса. В частности в коде можно заметить множество циклических конструкций, обработанных для организации повторного ввода данных.

**Листинг программы**

program laba\_1;

uses crt;

const max=100; maxline=80; maxval=98; maxnum=10;

var a:array[1..max] of byte;

v,x,y,i,j,k,n,m,maxk:byte;

empty,out:boolean;

p:char;

Procedure DrawLine;

begin

window(1,n+4,80,25);

for i:=1 to maxline do write('\_');

end;

Procedure NewIndex;

begin

for i:=1 to x do

for j:=1 to m do

if (j mod 2<>0) then

if (i=x) and (j>y) then break

else

inc(k);

end;

Procedure PutTab;

begin

for i:=1 to n do

begin

for j:=1 to m do

begin

if (j mod 2=0) then

Write('0':3,'|')

else

begin

inc(k);

a[k]:=random(maxval)+1;

Write(a[k]:3,'|');

end;

end;

WriteLn;

end;

end;

Procedure InitArray;

begin

repeat

clrscr;

WriteLn('Enter size of array:');

WriteLn;

{$I-}

repeat

window(1,3,80,25);

clrscr;

WriteLn('Enter x <=10:');

ReadLn(n);

until IOresult=0;

WriteLn;

repeat

window(1,5,80,25);

clrscr;

WriteLn('Enter y <=10:');

ReadLn(m);

until IOresult=0;

{$I+}

until (n>1) and (n<=maxnum) and (m>1) and (m<=maxnum);

window(1,1,80,25);

clrscr;

PutTab;

WriteLn;

maxk:=k;

for k:=1 to maxk do

Write(a[k],' ');

end;

Procedure GetTab;

begin

repeat

DrawLine;

window(1,n+5,80,25);

TextBackground(White);

TextColor(Black);

clrscr;

k:=0;

{$I-}

WriteLn('Enter x and y you search:');

repeat

window(1,n+5,80,25);

clrscr;

Write('x=');

ReadLn(x);

until IOresult=0;

repeat

window(1,n+7,80,25);

clrscr;

Write('y=');

ReadLn(y);

until IOresult=0;

{$I+}

if y mod 2=0 then empty:=true;

if (x>n) or (y>n) or (x<1) or (y<1) then

out:=true;

NewIndex;

WriteLn;

if not out then

if not(empty) then

WriteLn('Element=',a[k],' found in position ',k)

else WriteLn('Error!!! Empty place!!!')

else WriteLn('Error!!! Element is out of matrix!!!');

WriteLn;

WriteLn('Try Again?');

WriteLn('[Y/N]');

repeat

p:=readkey;

until (p='n') or (p='y') or (p=#27) ;

until p<>'y';

end;

{///////////MAIN PROGRAM/////////////}

begin

Randomize;

TextBackground(White);

TextColor(Black);

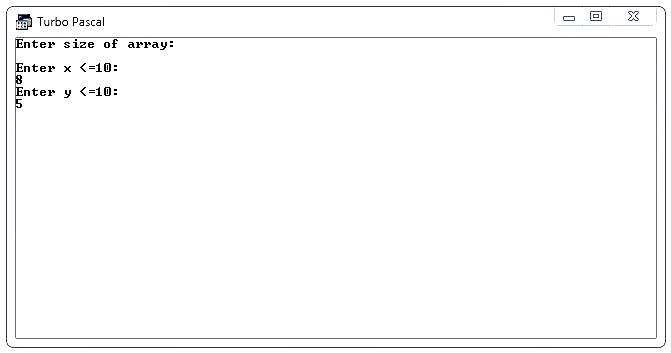
InitArray;

GetTab;

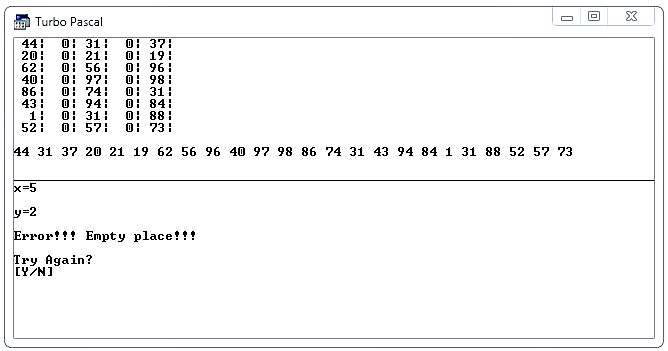
end.

1. **Описание результатов контрольного тестирования задачи.**

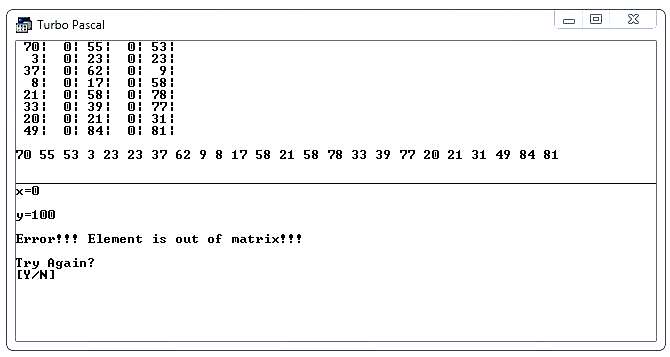
Ввод размерности матрицы:



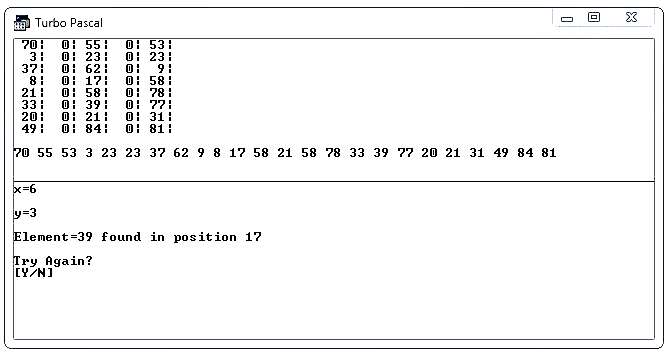
Вывод логического и внутреннего представления матрицы, ввод координат фонового элемента:



Ввод координат элемента, выходящего за границы матрицы:



Ввод координат существующего, значащего элемента и выдача значения пользователю:



1. **Выводы.**

В результатах тестирования программы были представлены следующие пункты:

* внутреннее представление разреженной матрицы
* значение элемента разреженной матрицы для случая, когда запрашиваемые координаты соответствуют местоположению значащего элемента
* значение элемента разреженной матрицы для случая, когда запрашиваемые координаты соответствуют местоположению фонового элемента
* значение элемента разреженной матрицы для случая, когда запрашиваемые координаты не вписываются в размеры матрицы
* выдача предупреждающих сообщений пользователю в случае некорректного задания вводимых параметров

Обобщив результаты, полученные после контрольного тестирования программы индивидуального задания (Вариант №9), можно свидетельствовать, что:

* матрица формируется верно (все элементы четных столбцов – фоновые)
* выдача сообщений об ошибках организована корректно
* значение искомого элемента определяется верно

За время написания программы:

* приобрел и закрепил навыки размещения в памяти векторов и таблиц
* закрепил знания о модульности программы с точки зрения обрабатываемых данных
* повторил основы организации циклов, условий и других конструкций языка Pascal
* повторил работу с подпрограммами модуля crt.