Федеральное государственное бюджетное образовательное учреждение высшего образования

**Национальный исследовательский университет «МЭИ»**

# Типовой расчет «Процедуры, функции, модули»

**Вариант 12**

*Задание выполнил: Нестеров А.С.*

*Студент группы A-06-23*

*Проверил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Оценка:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Замечания:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

Москва 2023 НИУ «МЭИ»

***Спецификация к Лабораторной работе №5 (Вариант 12)***

1. Постановка задачи (ПЗ).

**Задание**: Если **ни один из столбцов** матрицы, не содержит **два и более равных нулю** элемента, найти **сумму** элементов матрицы, лежащих на **главной диагонали** и **выше** нее.

1. Уточненная постановка задачи.

Дана **квадратная матрица** A **вещественных элементов** из **N строк** (N>1) и **N столбцов**. Если ни один из столбцов матрицы A, не содержит два и более равных нулю элемента Aij (|Aij| < 101), найти сумму элементов матрицы, лежащих на главной диагонали и выше нее.

1. Примеры

N = 4, все элементы по модулю < 100, ни один из столбцов матрицы не содержит более одного нулевого элемента => считаем сумму элементов, лежащих на главной диагонали и выше неё.

Sum = 1.31 – 2.12 + 3.999 – 4.49 + 2.12 – 3 + 4 + 3.001 – 4 + 4.49 = 5.31

1. Таблица данных

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Класс | **Имя** | Описание | Тип | Структура | Формат |
| Входные данные | A | Заданная матрица, |Aij|<=100 | вещ | Двухмерный массив (5х5) | +XX.XXX (:3:3) |
| N | число строк и столбцов в матрице, 1<N<=5 | цел | простая переменная | X (:1) |
| Выходные данные | summa | Сумма элементов главной диагонали и элементов над нею | вещ | простая | +XX.XXX (:3:3) |
| Промежуточные данные | q | Номер строки элемента матрицы, не подходящего под диапазон |Aij|<=100, 0<q<= 5 | цел | простая | XX (:2) |
| w | Номер столбца элемента матрицы, не подходящего под диапазон |Aij|<=100, 0<w<= 5 | цел | простая | XX (:2) |
| test | Файл с исходными данными с именем вида test#.txt | текст | файл | -- |
| answer | Файл с исходными данными с именем вида answer.txt | текст | файл | -- |
| matrixcheck | = True, если есть некорректные Aij, в противном случае False | лог | Простая переменная | -- |
| uslcheck | = True, если есть столбцы с количеством нулей больше одного | лог | Простая переменная | -- |

1. Входная форма (test.txt)

Обр 1 <n>

Обр 2 <A[1,1]> <A[1,2]> … < A[1,N]>

<A[2,1]> <A[2,2]> … < A[2,N]>

…………………………………..

<A[N,1]> <A[N,2]> … < A[N,N]>

1. Выходная форма

Обр 3 N = <N>

Некорректное число строк/столбцов

Обр 4 матрица:

<A[1,1]> <A[1,2]> … < A[1,N]>

<A[2,1]> <A[2,2]> … < A[2,N]>

…………………………………..

<A[N,1]> <A[N,2]> … < A[N,N]>

Обр 5 элемент матрицы <q> строки и <w> столбца, а именно: <A[q,w]>

является некорректным, так как не входит в диапазон [-100; 100]

Обр 6 В матрице содержится столбец, в котором содержится более чем один  
 нулевой элемент

Обр 7 Сумма элементов матрицы на верхней диагонали и над нею: <sum>

**Имена входного и выходного файлов передаются как параметры программы:**

Первый – имя файла с исходными данными

Второй – имя файла для вывода исходных данных и результатов

1. Аномалии

|  |  |  |  |
| --- | --- | --- | --- |
| № | Описание | Условие возникновения | Реакция |
| 1 | Некорректное число строк/столбцов | (n<2) | Обр 3 |
| 2 | Некорректное число строк/столбцов | (n>5) | Обр 3 |
| 3 | Есть некорректный элемент в матрице | (|A[q,w]|>100)  A[q,w]>100 | Обр 5 |
| 4 | Есть некорректный элемент в матрице | (|A[q,w]|>100)  A[q,w]<100 | Обр 5 |

1. Функциональные тесты

|  |  |  |  |
| --- | --- | --- | --- |
| № теста | Входные данные | Ожидаемый результат | Смысл теста |
| 1 | N = 1 | Обр 3 | Аномальная ситуация 1: n<2 |
| 2 | N = 6 | Обр 3 | Аномальная ситуация 2: n>5 |
| 3 | N = 2 | Обр 5  A22 = 113 | Аномальна ситуация 3: A[q,w]>100 |
| 4 | N= 2 | Обр 5  A12 = -100.45 | Аномальна ситуация 4: A[q,w]<100 |
| 5 | N= 4 | Обр 6 | Более чем один нулевой элемент в одном из столбцов (в 4) |
| 6 | N = 5 | Обр 7  Summa = 15 + (0.01+0.15)\*15/2 = 15 + 1.2 = 16.2 | Проверка подсчёта суммы элементов диагонали и элементов над ней (элементы исключительно положительные) |
| 7 | N = 4 | Обр 7  Summa = 1.31 – 2.12 + 3.999 – 4.49 + 2.12 – 3 + 4 + 3.001 – 4 + 4.49 = 5.31 | Проверка подсчёта суммы элементов диагонали и элементов над ней (элементы как положительные, так и отрицательные) |
| 8 | N = 3 | Обр 7  Summa = -100 – 99.99 – 100 = -299.99 | Проверка подсчёта суммы элементов диагонали и элементов над ней (есть нулевые элементы, но ни один из столбцов не содержит более одного нулевого элемента) |

1. Алгоритм

1. Подзадача А0.1. Ввести из файла (обр.1) и проверить значение количества строк/столбцов (обр.3)

2. Подзадача А0.2. Ввести матрицу из файла (обр.2).

3. Подзадача А0.3. Вывести матрицу (обр.4)

4. Подзадача А0.4. Проверить значения элементов матрицы (обр.5)

5. Подзадача А0.5. Проверить на наличие столбцов с количеством нулевых элементов, большим 1 (обр 6)

6. Подзадача А0.6. Подсчитать сумму элементов на диагонали матрицы и над ней (обр 7)

1. Алгоритм

Начало

Ввод N по обр 1

N<2 or N>5 вывод по обр3

Вх. N

Ввод матрицы из файла

vvodmatrix(test, n, a);

Вх. N,A Вых A

Вывод матрицы в файл

vivodmatrix(answer, N, A);

Вх. N,A Вых –

Проверка элементов матрицы

matrixcheck := checkmatrix(n, a , q, w);

Вых matrixcheck, q,w

Not matrixcheck вывод по обр5

Вх. N,A

Проверка условия на наличие столбца

с более чем 1 нулем

uslcheck := checkusl(N,A);

вых. Uslcheck

Not uslcheck вывод по обр6

Вх N,A

Вычисление суммы элементов на диагонали и выше неё

summa := PoiskSumVerhDiag(N, A);

Вых. Summa

Вывод по обр7

Конец

1. Код

program laba12;

{$APPTYPE CONSOLE}

{$R \*.res}

uses

System.SysUtils,

MODUL12 in 'MODUL12.pas';

Var {раздел описания переменных -----------------------------------------------------------------}

N, q, w: byte;

summa : currrency;

A: matrix;

test, answer : TextFile;

matrixcheck, uslcheck : boolean;

begin

AssignFile(test, ParamSTR(1));

reset(test);

AssignFile(answer, ParamSTR(2));

append(answer);

read(test, N); {обр1} {ввод n и его проверка}

if(N<2) or( N>5) then // А0.1

begin

writeln(answer, 'N = ', N);

writeln(answer, 'Некорректное число строк/столбцов');

closefile(test);

//closefile(test); closefile(answer);

end

else

begin

vvodmatrix(test, n, a); // А0.2 {обр1}

closefile(test);

vivodmatrix(answer, N, A); // А0.3{обр4} {ввод и вывод матрицы A}

matrixcheck := checkmatrix(n, a , q, w);

if not matrixcheck then// А0.4 {проверка элементов матрицы}

begin

writeln(answer, 'элемент матрицы ', q , ' строки и ', w, ' столбца, а именно: ',a[q][w] );

writeln(answer, 'является некорректным, так как не входит в диапазон [-100; 100]'); {обр5}

end

else

begin

uslcheck := checkusl(N,A);

if not (uslcheck) then// А0.5 {проверка условия с нулевыми элементами в столбцах}

begin

writeln(answer, 'В матрице содержится столбец, в котором содержится более чем один нулевой элемент'); {обр6}

end

else

begin

summa := PoiskSumVerhDiag(N, A);// А0.6 {поиск суммы элементов на диагонали и выше нее}

writeln(answer, 'Сумма элементов матрицы на верхней диагонали и над нею: ', summa:3:3); {обр7}

end;

end;

end;

closeFile(answer);

end.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Модуль MODUL12.pas со всеми процедурами и функциями\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unit MODUL12;

interface

const Nmax = 5;

Type matrix = array[1..Nmax,1..Nmax] of currency; //для отрицательных вещественных чисел

procedure vvodmatrix(var testt: TextFile;const nn : byte; out aa:matrix);

|  |  |  |
| --- | --- | --- |
| Формальные параметры | Фактические параметры | Тип |
| testt | test | TextFile |
| nn | n | byte |
| aa | a | matrix |

procedure vivodmatrix(var answerr: TextFile; const Nn : integer; var Aa : matrix);

|  |  |  |
| --- | --- | --- |
| Формальные параметры | Фактические параметры | Тип |
| answerr | answer | TextFile |
| Nn | n | byte |
| Aa | a | matrix |

function checkmatrix(const nn: byte; const aa : matrix; out ii, jj : byte) : boolean;

|  |  |  |
| --- | --- | --- |
| Формальные параметры | Фактические параметры | Тип |
| Nn | n | byte |
| Aa | a | matrix |
| ii | q | byte |
| jj | w | byte |

function checkusl(const Nn: byte; const Aa : matrix) : boolean;

|  |  |  |
| --- | --- | --- |
| Формальные параметры | Фактические параметры | Тип |
| Nn | n | byte |
| Aa | a | matrix |

function PoiskSumVerhDiag(const Nn : byte; const Aa: matrix) : currency;

|  |  |  |
| --- | --- | --- |
| Формальные параметры | Фактические параметры | Тип |
| Nn | n | byte |
| Aa | a | matrix |

implementation

{Ввод матрицы из файла, файл уже открыт }

procedure vvodmatrix(var testt: TextFile;const nn : byte; out aa:matrix);

var i, j : byte;

begin

for i := 1 to nn do

begin

for j := 1 to nn do read(testt,aa[i,j]);

readln(testt);

end;

end;

{Вывод матрицы в файл, файл уже открыт }

procedure vivodmatrix(var answerr: TextFile; const Nn : integer; var Aa : matrix);

var i,j : integer;

begin

writeln(answerr, 'матрица: '+ #13#10);

for i := 1 to Nn do

begin

for j := 1 to Nn do write(answerr,Aa[i][j]:3:3, ' ');

writeln(answerr);

end;

writeln(answerr);

end;

{проверка элементов матрицы }

function checkmatrix(const nn: byte; const aa : matrix; out ii, jj : byte) : boolean;

var flag : boolean; i, j : byte;

begin

{Заглушка} writeln(’ Заглушка - проверка элементов матрицы’);

{тест 3}//ii := 2; jj := 2; flag := True; checkmatrix := flag;

{тест 4}//ii := 1; jj := 2; flag := True; checkmatrix := flag;

{остальные тесты}

end;

{проверка условия наличия хотя бы одного столбца, в котором минимум 2 нуля}

function checkusl(const Nn: byte; const Aa : matrix) : boolean;

var flag : boolean; i, j : byte; counter : integer;

begin

{Заглушка} writeln(’ Заглушка - проверка условия наличия хотя бы одного столбца, в котором минимум 2 нуля’);

{тест 5} //flag := true; checkusl := flag;

{остальные тесты}

end;

{поиск суммы элементов на диагонали матрицы и выше неё}

function PoiskSumVerhDiag(const Nn : byte; const Aa: matrix) : currency;

var sum : currency; i, j : byte;

begin

{Заглушка} writeln(’ Заглушка - поиск суммы элементов на диагонали матрицы и выше неё);

{тест 6}//sum:= 16.2 PoiskSumVerhDiag := sum;

{тест 7}//sum:= 5.31 PoiskSumVerhDiag := sum;

{тест 8}//sum:= -299.99 PoiskSumVerhDiag := sum;

{остальные тесты}

end;

end.

Абстракция А0.4.

1.Условие. Проверить, есть ли в матрице элементы, абсолютная величина которых больше 100

2.Уточненная постановка задачи. Дана вещественная квадратная матрица A из n строк и столбцов. Проверить (checkmatrix), есть ли в матрицы элементы, абсолютная величина которых больше 100. Если есть, то присвоить checkmatrix значение True и найти номер строки (ii) и столбца (jj) первого такого элемента, иначе присвоить checkmatrix значение False.

3. Примеры.

(тест 3 – есть некорректный элемент) Aa22 = 113

checkmatrix = True

(тест 4 – есть некорректный элемент) Aa12 = -100.45

checkmatrix = True

(тесты 5-8 – некорректных элементов нет)

Например, 7 тест:

checkmatrix = False

4. Таблица данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Класс | Имя | Описание (смысл), диапазон, точность | Тип | Структура |
| Входные данные | nn | число строк и столбцов в матрице Aa, 1<nn<=5 | цел | Простая переменная |
| aa | Заданная матрица, |Aaij|<=100 | вещ | Двухмерный массив (5х5) |
| Выходные данные | ii | Строка с некорр. Aa[ii,jj], 0 | цел | Простая переменная |
| jj | Столбец с некорр. Aa[ii,jj], 0 | цел | Простая переменная |
| flag | = True, если нет некорректных Aaij, в противном случае False | лог | Простая переменная |
| Промежуточные данные | i | Номер текущей строки, 0 < i < 6 | цел | Простая переменная |
| j | Номер текущего столбца, 0 < j <6 | цел | Простая переменная |

9.Метод

Пусть flag=

Предположим сначала, что такого элемента в матрице нет (Flag:=True). Затем будем просматривать строки, начиная с первой (i:=1) Пока не просмотрены все (i≤=nn) и не найден такой элемент (Flag=True)

В каждой строке будем просматривать элементы, начиная с первого (j :=1). Пока не посмотрены все (j<=nn) и не найден такой элемент (Flag=True)

Если рассматриваемый элемент Aaij по модулю больше 100

ТО

искомый элемент найден! (Flag:=False)

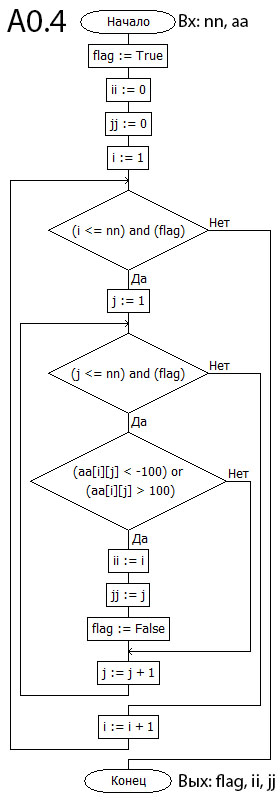
Запоминаем его местонахождение (ii:=i; jj:=j)

Переходим к следующему элементу в строке (j:=j+1)

Переходим к следующей строке (i:=i+1)

Значения flag, ii, jj искомы

1. Алгоритм



11.Программа на паскале. Функция.

function checkmatrix(const nn: byte; const aa : matrix; out ii, jj : byte) : boolean;

var flag : boolean; i, j : byte;

begin

flag := True;

ii := 0; jj:=0;

i := 1;

while (i <= nn) and (flag) do

begin

j:= 1;

while (j <= nn) and (flag) do

begin

if (aa[i][j] < -100) or (aa[i][j] > 100) then

begin

ii := i; jj := j;

flag := False;

end;

j := j + 1;

end;

i:= i + 1;

end;

checkmatrix := flag;

end;

Абстракция А0.5.

1.Условие. Проверить, есть ли в матрице

2.Уточненная постановка задачи. Дана вещественная квадратная матрица A из n строк и столбцов. Проверить (checkusl), есть ли в матрице столбцы, в которых содержится более чем один нулевой элемент. Если есть, то присвоить checkusl значение True, иначе присвоить checkusl значение False.

3. Примеры.

(тест 5 – в хотя бы одном из столбцов более чем один нулевой элемент)

Checkusl = True

(тесты 6-8 – каждый столбец не содержит более одного нулевого элемента)

Например, 8 тест:

Checkusl = False

4. Таблица данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Класс | Имя | Описание (смысл), диапазон, точность | Тип | Структура |
| Входные данные | nn | число строк и столбцов в матрице Aa, 1<nn<=5 | цел | Простая переменная |
| aa | Заданная матрица, |Aaij|<=100 | вещ | Двухмерный массив (5х5) |
| Выходные данные | flag | = True, если матрица не содержит столбцов с более чем одним нулем, в противном случае False | лог | Простая переменная |
| Промежуточные данные | i | Номер текущей строки, 0 < i < 6 | цел | Простая переменная |
| j | Номер текущего столбца, 0 < j <6 | цел | Простая переменная |
| counter | Счётчик нулей в столбце 0<=counter<=5 | цел | Простая переменная |

9.Алгоритм

Пусть flag=

Предположим сначала, что таких стобцов в матрице нет (Flag:=True). Затем будем просматривать столбцы, начиная с первого (j:=1) Пока не просмотрены все (j≤=nn) и не найден такой столбец (Flag=True)

В каждом столбце будем просматривать элементы, начиная с первого (i :=1). Пока не посмотрены все (i<=nn) и не найден такой элемент (Flag=True).

Заводим счётчик нулей(counter := 0);

Если рассматриваемый элемент Aaij равен нулю

ТО

Увеличиваем счётчик (counter := counter +1)

Если счётчик нулей больше 1 (counter > 1)

ТО

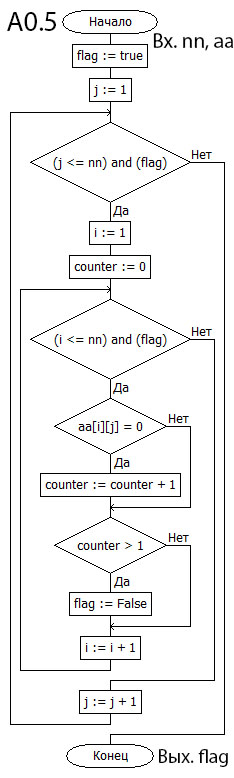
(Flag:=False).

Переходим к следующему элементу в столбце (i:=i+1)

Переходим к следующему столбцу (j:=j+1)

Значение flag искомо

10.Алгоритм



11.Программа на паскале. Функция.

function checkusl(const Nn: byte; const Aa : matrix) : boolean;

var flag : boolean; i, j : byte; counter : integer;

begin

flag:= true;

j := 1;

while (j <= nn) and (flag) do

begin

i:= 1;

counter := 0;

while (i <= nn) and (flag) do

begin

if aa[i][j] = 0 then

begin

counter := counter + 1;

end;

if counter > 1 then flag := False;

i := i + 1;

end;

j:= j + 1;

end;

checkusl := flag;

end;

Абстракция А0.6.

1.Условие. Посчитать сумму всех элементов, лежащих на диагонали и выше неё

2.Уточненная постановка задачи. Дана вещественная квадратная матрица A из n строк и столбцов. Посчитать сумму (sum) всех элементов, лежащих на диагонали и выше неё.

3. Примеры.

Тест 6 (исключительно положительные элементы)

Тест 7 (и положительные, и отрицательные элементы)

Тест 8 (присутствуют нули, матрица не противоречит условию)

4. Таблица данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Класс | Имя | Описание (смысл), диапазон, точность | Тип | Структура |
| Входные данные | nn | число строк и столбцов в матрице Aa, 1<nn<=5 | цел | Простая переменная |
| aa | Заданная матрица, |Aaij|<=100 | вещ | Двухмерный массив (5х5) |
| Выходные данные | sum | Сумма всех элементов на диагонали и выше неё | вещ | Простая переменная |
| Промежуточные данные | i | Номер текущей строки, 0 < i < 6 | цел | Простая переменная |
| j | Номер текущего столбца, 0 < j <6 | цел | Простая переменная |

9.Метод

Пусть сумма всех элементов на диагонали и выше неё будет равна 0 (для накопления суммы) (sum := 0)

Будем проходить все строки, начиная с первой (i:=1), пока не пройдем все (i≤=nn).

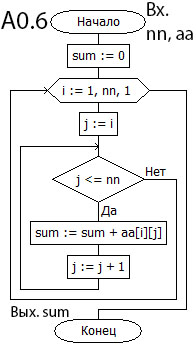
В каждой строке будем накапливать сумму(sum), начиная с элемента, расположенного на диагонали (j :=i). Пока не пройдены все (j<=nn).

Накапливаем сумму элемента на диагонали и элементов правее его (если они есть) (sum := sum + aa[i][j])

Переходим к следующему элементу в строке (j := j +1)

Переходим к следующей строке (i := i +1)

10.Алгоритм



11.Программа на паскале. Функция.

function PoiskSumVerhDiag(const Nn : byte; const Aa: matrix) : currency;

var sum : currency; i, j : byte;

begin

sum := 0;

for i := 1 to nn do

begin

j := i;

while j <= nn do

begin

sum := sum + aa[i][j];

j := j + 1;

end;

end;

PoiskSumVerhDiag := sum;

end;

Структурные тесты. Выявление случаев, не покрываемых функциональными тестами (1-8):

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Тест№  Структура | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| if(N<2) or( N>5) |  | | | | | | | |
| N<2 истина  N>5 истина | невозможно | | | | | | | |
| N<2 истина  N>5 ложь | N=1 | - | - | - | - | - | - | - |
| N<2 ложь  N>5 истина | - | N=6 | - | - | - | - | - | - |
| N<2 ложь (N=2)  N>5 ложь |  |  | N=2 | N=2 |  |  |  |  |
| N<2 ложь  N>5 ложь (N=5) |  |  |  |  |  | N=5 |  |  |
| N<2 ложь  N>5 ложь |  |  |  |  | N=4 |  | N=4 | N=3 |
| N<2 ложь (N=2)  N>5 ложь (N=5) | невозможно | | | | | | | |
| Ввод/вывод матрицы For i:=1 to n do |  | | | | | | | |
| 0 раз (i>n) | Н Е Т  В В О Д А  И  П  Р  О  В  Е  Р  К  И | | Мы проверили, что N>1 | | | | | |
| 1 раз (i=n) |
| более 1 раза (i<n) | 2 | 2 | 4 | 5 | 4 | 3 |
| 0 раз (j>n) | Мы проверили, что N>1 | | | | | |
| 1 раз (j=n) |
| более 1 раза (j<n) | 2 | 2 | 4 | 5 | 4 | 3 |
| Проверка элементов матрицы (function checkmatrix) |  | | | | | |
| if (aa[i][j] < -100) or (aa[i][j] > 100) |  | | | | | |
| (aa[i][j] < -100) истина  (aa[i][j] > 100) истина | невозможно | | | | | |
| (aa[i][j] < -100) истина  (aa[i][j] > 100) ложь | - | + | - | - | - | - |
| (aa[i][j] < -100) ложь  (aa[i][j] > 100) истина | + | - | - | - | - | - |
| (aa[i][j] < -100) ложь  (aa[i][j] > 100) ложь | - | - | + | + | + | + |
| if not matrixcheck then |  | | | | | |
| (not matrixcheck) = истина | + | + | - | - | - | - |
| (not matrixcheck) = ложь | - | - | + | + | + | + |
| Проверка условия на нули function checkusl | НЕКОРРЕКТНЫЕ ЭЛЕМЕНТЫ МАТРИЦЫ | |  | | | |
| Counter >1 |  | | | |
| Counter < 0 | невозможно | | | |
| Counter = 0 | - | + | + | - |
| Counter = 1 | - | - | - | + |
| Counter > 1 | + | - | - | - |
| if not uslcheck then |  | | | |
| (not uslcheck) истина | + | - | - | - |
| (not uslcheck) ложь | - | + | + | + |
| Функция поиска суммы (function PoiskSumVerhDiag) | В матрице содержится столбец, в котором содержится более чем один нулевой элемент |  |  |  |
| 0 раз (i>n) | невозможно | | |
| 1 раз (i=n) |
| более 1 раза (i<n) | + | + | + |
| 0 раз (j>n) | невозможно | | |
| 1 раз (j=n) |
| более 1 раза (j<n) | + | + | + |
|  |  |  |  |