YASSIN MARMOUD
NWYUK6
ymarmoud@gmail.com
Group1

Assignment/1. Task 3

first date: 20 March 2020
update: 30 june 2020

TASK

Implement the N matrix type which contains integers.
These are square matrices that can contain nonzero entries only in their first and last column, and in their main diagonal. Don't store the zero entries. Store only the entries that can be nonzero in a sequence. Implement as methods:
getting the entry located at index (i, j), adding and multiplying two N-matrices, and printing the N-matrix (in a
square shape).

Modules:

There are four main modules in my program one is for printing a N matrix, getting the value from a N matrix by indexes, add two N-matrix and multiply two N-matrix.

❖ *Print:*

The print is divided into two main functions; the first takes the input from the user including the size of N-matrix and the values. The second only takes size of matrix as input and then make a random N-matrix according to the given size.

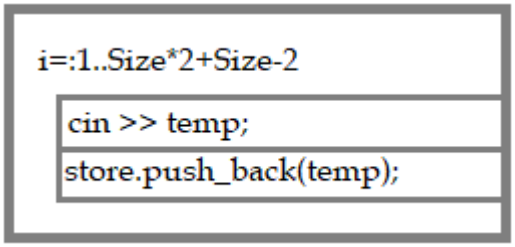
- **Printing the N-matrix with user input:**

  - the first part of print:

Specification:

A = (Size:ℤ, temp:ℤ , store:Vec)
Pre = (Size=Size' temp=temp')
Post = { Pre ^ store.push_back(temp) }

Structogram:



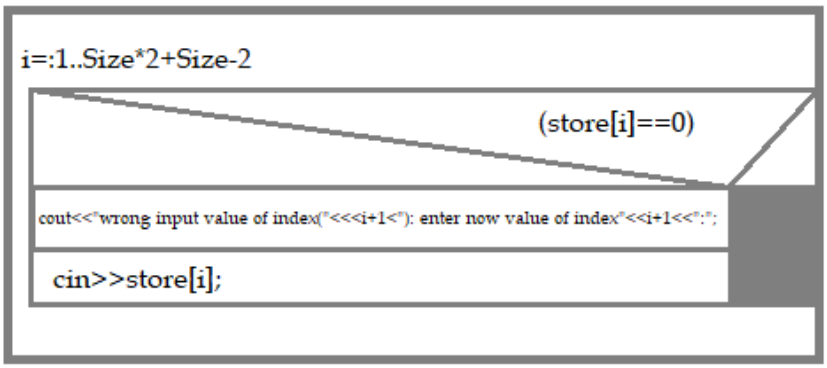
  - The second part of print:

This part check the wrong input value and give change to correct it:

Specification:

A = (Size:ℤ , store:Vec)
Pre = (Size=Size')
Post = { Pre ^ store[i] }

Structogram:



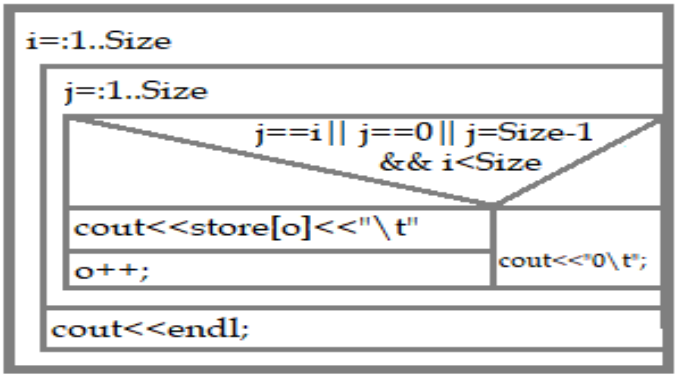
- The third part of print:

This part check the wrong input value and give change to correct it:

Specification:

A = (Size:ℤ , store:Vec)
Pre = (Size=Size')
Post = { Pre ^ store[o] }
j=I or j=0 or
j=size-1 and i<size

Structogram:

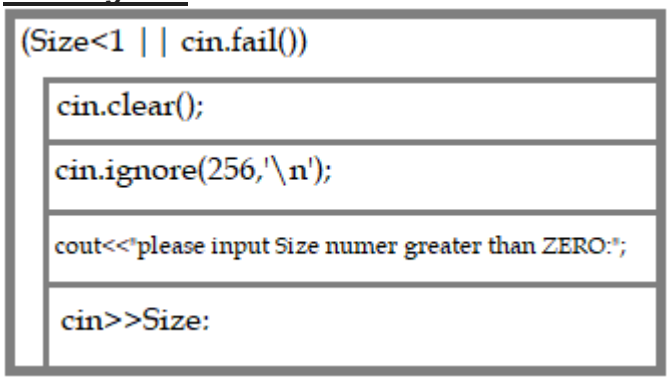


- **Printing random N-Matrix:**
  - the first part of print:

Specification:

A = (Size:ℤ)
Pre = (Size=Size')
Post = if the Size input is not a number cin>>Size again;

Structogram:

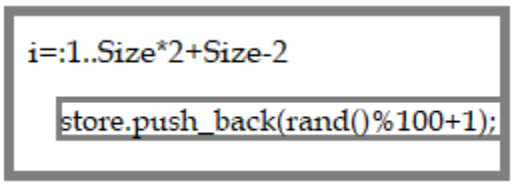


  - the Second part of print:

Specification:

We push_back the store vectore with random numbers between 1 and 100 of course base on the Size of the vector.

Structogram:



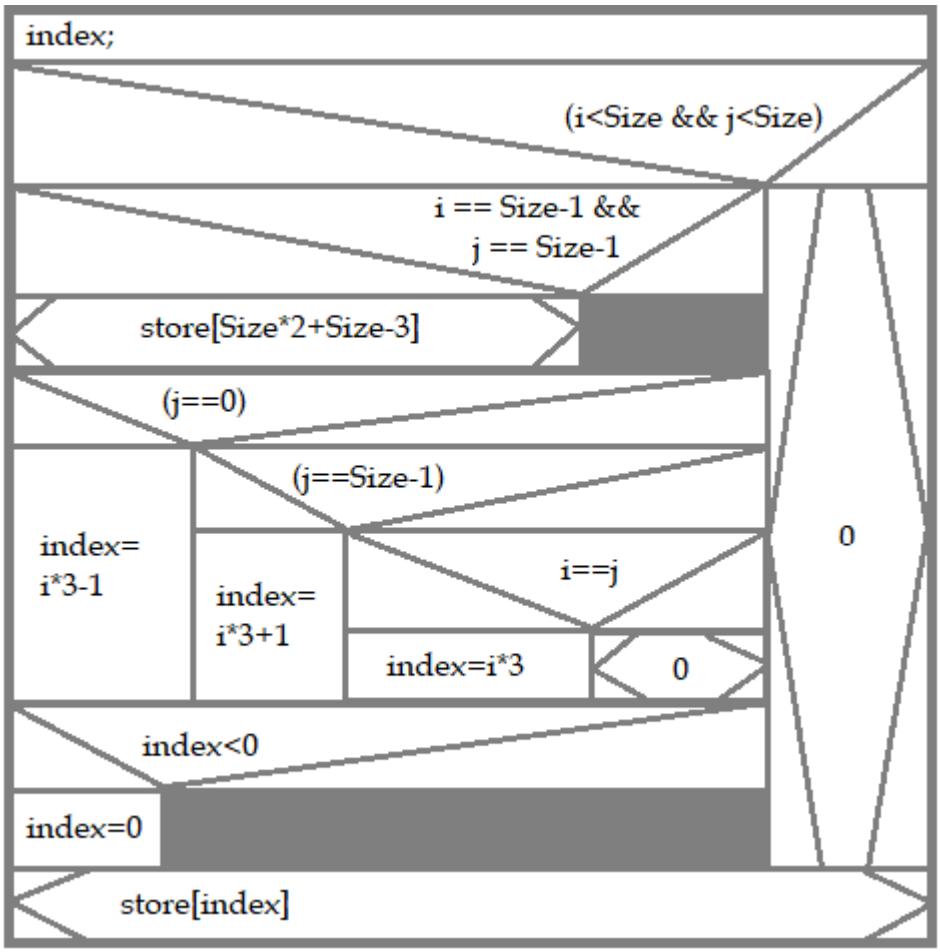
  - the third part of print:

this part of the code is same with the third part of print with inputting values.

❖ ***Get Entry:***

The user is asked to give the ith and jth index for the matrix if the matrix already exists in the program otherwise the user is first asked to generate the matrix and then give the ith and jth index.
getEntry:
The getEntry of N-matrice A (represented by vectors store) goes to check the i and j are small then size of N-matrix A after the connection with the index of the vector store.

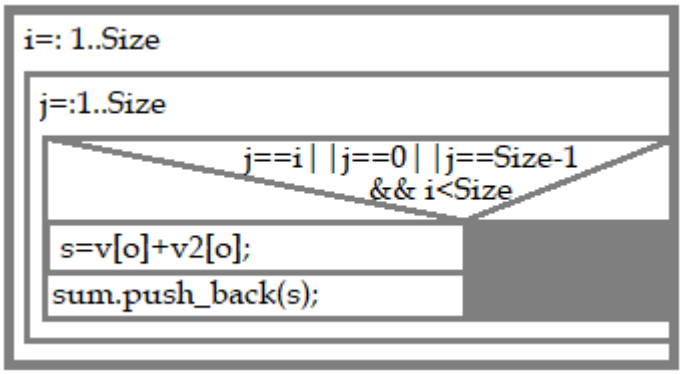
Specification:
A = (Size:ℤ, i:ℤ , j:ℤ , index: ℤ , store:Vec)
Pre = (Size=Size' i=i' j=j' index=index')
Post = { Pre ^ getEntry(i,j) }
Structogram:



❖ ***Addition two N-matrix:***

The function first asks the user to give the size of the matrix you want to add. After getting size the function adds them together and shows the results.
Sum:
The sum of N-matrices A and B (represented by vectors v1 and v2) goes to N-matrix c (represented by vector sum), where all of the vectors have to have the same size.
∀i∈[0..n-1]: sum[i]:= v[i] + v2[i]

Specification:
A = (Size:ℤ, s:ℤ , v:Vec , v2:Vec , sum:Vec)
Pre = (Size=Size' s=s')
Post = { Pre ^ sum =$\sum_{i=0}^{Size} v(i) + v2(i)$}
Structogram:

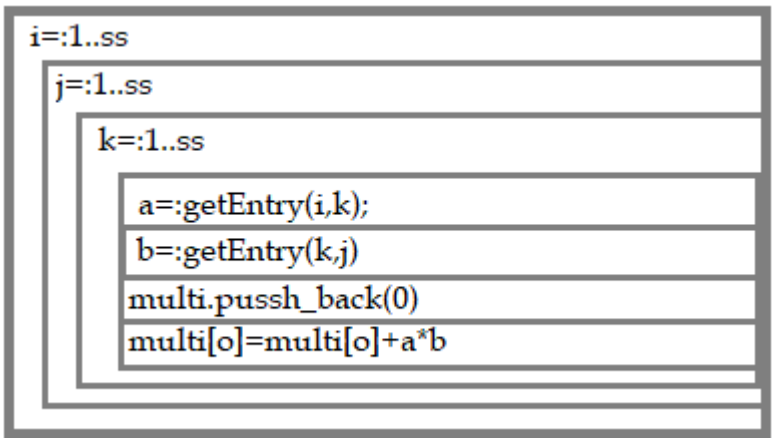


❖ ***Multiplication two N-matrix***

The function first asks the user to give the size of the matrix you want to multiply. After getting size the function multiplies them together and shows the results.
Multiplication:
The multi of N-matrices A and B (represented by vectors v and v2) goes to N-matrix C (represented by vector Multi), where all of the vectors have to have the same size.
∀i∈[0..n-1]: sum[i]:= v[i] + v2[i]

Specification:
A = (ss:ℤ, a:ℤ , b:ℤ , multi:Vec)
Pre = (ss=ss' a=a' b=b')
Post = { Pre ^ multi =$\sum_{i=0}^{ss} multi + a * b$}
a=getEntry(i,k)
b=getEntry(k,j)

Structogram:



❖ ***Class***

I have two classes in my program.

They are:

menuN:

the menuN is showing the main menu of the program and the submenu for the options in main menu

Menu
- v: vector<int> - temp, size: int - mm : matrix
- run() : void - getIndex(): void - add() : void - mult() : void - print() : void

Matrix:

The matrix class is managing all the functions of the matrix. We are using setters and getters in this class to manage the actual matrix. And getting the value by indexes and add two N-matrix and multiplication.

Matrix
- Size: int - Store: vectore<int>
- getMatrix():vector<int> - setMatrix():void - getSize():int - setSize():void - getEntry():int - printN():void - printNR():void - store2get():void - storeR2get():void - sum2():void - multi2():void - operator+: friend matrix - operator*: friend matrix

❖ ***Testing***

1. **Create matrix**
    - Check matrix a
    - Check matrix b
    - Check matrix c
2. **Copy constructor**
    - Check matrix a
    - Check matrix b
    - Check matrix c
3. **Assignment operator**
    - Check matrix a
    - Check matrix b
    - Check matrix c
4. **Addition**
    - Check sum of matrix a + b
    - Check sum of matrix b + a
5. **Multiplication**
    - Check multiplication of matrix a * b
    - Check multiplication of matrix b * a