STRUKTUR DATA
Tugas Kelompok Matriks Sparse

MUHAMMAD LUTHFI AZIZ SUNARYA 140810230081
BAGAS DIATAMA WARDOYO 140810230061
MUHAMAD HISYAM AZ-ZAHRAN 140810230065

Dikumpulkan tanggal :
02 Juni 2024

Universitas Padjadjaran
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
Program Studi S-1 Teknik Informatika
2024

Source Code:

```
/* Nama program : penjumlahan_MatriksSparse
Nama            : Bagas Diatama Wardoyo 140810230061
Nama            : MUHAMMAD LUTHFI AZIZ SUNARYA 140810230081
Nama            : MUHAMAD HISYAM AZ-ZAHRAN 140810230065
Tanggal buat    : 02/06/2024
Deskripsi       : Tugas Kelompok Matriks Sparse
*****************************************************************
*******/
#include <iostream>
using namespace std;

struct Node {
    int baris, kolom, nilai;
    Node* kanan;
    Node* bawah;
};

typedef Node* NodePtr;
typedef NodePtr* NodePtrArray;

void createNode(NodePtr& newNode, int baris, int kolom, int
nilai) {
    newNode = new Node;
    newNode->baris = baris;
    newNode->kolom = kolom;
    newNode->nilai = nilai;
    newNode->kanan = newNode; // Circular
    newNode->bawah = newNode; // Circular
}

void insertNode(NodePtrArray rowHead, NodePtrArray colHead, int
baris, int kolom, int nilai) {
    NodePtr newNode;
    createNode(newNode, baris, kolom, nilai);

    // Insert in row
    if (!rowHead[baris]) {
        rowHead[baris] = newNode;
    } else {
```

```
        NodePtr temp = rowHead[baris];
        while (temp->kanan != rowHead[baris] &&
temp->kanan->kolom < kolom)
            temp = temp->kanan;
        if (temp->kolom == kolom) {
            temp->nilai += nilai;
            if (temp->nilai == 0) { // Remove zero value node
                if (temp->kanan == temp) {
                    rowHead[baris] = nullptr;
                } else {
                    NodePtr prev = rowHead[baris];
                    while (prev->kanan != temp) {
                        prev = prev->kanan;
                    }
                    prev->kanan = temp->kanan;
                    if (temp == rowHead[baris]) {
                        rowHead[baris] = temp->kanan;
                    }
                    delete temp;
                }
            }
            delete newNode; // Avoid memory leak
        } else {
            newNode->kanan = temp->kanan;
            temp->kanan = newNode;
            if (kolom < rowHead[baris]->kolom) {
                while (temp->kanan != rowHead[baris]) {
                    temp = temp->kanan;
                }
                temp->kanan = newNode;
                rowHead[baris] = newNode;
            }
        }
    }

    // Insert in column
    if (!colHead[kolom]) {
        colHead[kolom] = newNode;
    } else {
        NodePtr temp = colHead[kolom];
```

```
        while (temp->bawah != colHead[kolom] &&
temp->bawah->baris < baris)
            temp = temp->bawah;
        if (temp->baris == baris) {
            temp->nilai += nilai;
            if (temp->nilai == 0) { // Remove zero value node
                if (temp->bawah == temp) {
                    colHead[kolom] = nullptr;
                } else {
                    NodePtr prev = colHead[kolom];
                    while (prev->bawah != temp) {
                        prev = prev->bawah;
                    }
                    prev->bawah = temp->bawah;
                    if (temp == colHead[kolom]) {
                        colHead[kolom] = temp->bawah;
                    }
                    delete temp;
                }
            }
        } else {
            newNode->bawah = temp->bawah;
            temp->bawah = newNode;
            if (baris < colHead[kolom]->baris) {
                while (temp->bawah != colHead[kolom]) {
                    temp = temp->bawah;
                }
                temp->bawah = newNode;
                colHead[kolom] = newNode;
            }
        }
    }
}

void sumSparseMatrices(NodePtrArray rowHeadA, NodePtrArray
colHeadA, NodePtrArray rowHeadB, NodePtrArray colHeadB,
NodePtrArray& rowHeadC, NodePtrArray& colHeadC, int numRows,
int numCols) {
    for (int i = 0; i < numRows; ++i) {
        rowHeadC[i] = nullptr;
```

```
    }

    for (int j = 0; j < numCols; ++j) {
        colHeadC[j] = nullptr;
    }

    for (int i = 0; i < numRows; ++i) {
        if (rowHeadA[i] != nullptr) {
            Node* temp = rowHeadA[i];
            do {
                insertNode(rowHeadC, colHeadC, temp->baris,
temp->kolom, temp->nilai);
                temp = temp->kanan;
            } while (temp != rowHeadA[i]);
        }
        if (rowHeadB[i] != nullptr) {
            Node* temp = rowHeadB[i];
            do {
                insertNode(rowHeadC, colHeadC, temp->baris,
temp->kolom, temp->nilai);
                temp = temp->kanan;
            } while (temp != rowHeadB[i]);
        }
    }
}

void printSparseMatrix(NodePtrArray rowHead, int numRows, int
numCols) {
    for (int i = 0; i < numRows; ++i) {
        Node* temp = nullptr;
        if (rowHead[i] != nullptr) {
            temp = rowHead[i];
        }
        for (int j = 0; j < numCols; ++j) {
            if (temp != nullptr && temp->kolom == j) {
                cout << temp->nilai << " ";
                temp = temp->kanan;
            } else {
                cout << "0 ";
            }
```

```
        }
        cout << endl;
    }
}

void inputMatrix(NodePtrArray rowHead, NodePtrArray colHead,
int numRows, int numCols) {
    int n, row, col, value;
    cout << "Masukkan jumlah elemen non-nol: ";
    cin >> n;

    for (int i = 0; i < n; ++i) {
        cout << "Masukkan baris, kolom, dan nilai: ";
        cin >> row >> col >> value;
        insertNode(rowHead, colHead, row, col, value);
    }
}

int main() {
    const int numRows = 5;
    const int numCols = 6;

    NodePtrArray rowHeadA = new NodePtr[numRows]{ nullptr };
    NodePtrArray colHeadA = new NodePtr[numCols]{ nullptr };
    NodePtrArray rowHeadB = new NodePtr[numRows]{ nullptr };
    NodePtrArray colHeadB = new NodePtr[numCols]{ nullptr };
    NodePtrArray rowHeadC = new NodePtr[numRows]{ nullptr };
    NodePtrArray colHeadC = new NodePtr[numCols]{ nullptr };

    int choice;

    do {
        cout << "\nMenu:\n";
        cout << "1. Input Matriks 1\n";
        cout << "2. Input Matriks 2\n";
        cout << "3. Jumlahkan Matriks dan Tampilkan Hasil\n";
        cout << "4. Tampilkan Matriks 1\n";
        cout << "5. Tampilkan Matriks 2\n";
        cout << "6. Keluar\n";
        cout << "Masukkan pilihan Anda: ";
```

```
        cin >> choice;

        switch (choice) {
            case 1:
                inputMatrix(rowHeadA, colHeadA, numRows,
numCols);
                break;
            case 2:
                inputMatrix(rowHeadB, colHeadB, numRows,
numCols);
                break;
            case 3: {
                sumSparseMatrices(rowHeadA, colHeadA, rowHeadB,
colHeadB, rowHeadC, colHeadC, numRows, numCols);
                cout << "Matriks Hasil (C = A + B):\n";
                printSparseMatrix(rowHeadC, numRows, numCols);
                break;
            }
            case 4:
                cout << "Matriks 1:\n";
                printSparseMatrix(rowHeadA, numRows, numCols);
                break;
            case 5:
                cout << "Matriks 2:\n";
                printSparseMatrix(rowHeadB, numRows, numCols);
                break;
            case 6:
                cout << "Keluar...\n";
                break;
            default:
                cout << "Pilihan tidak valid. Silakan coba
lagi.\n";
        }
    } while (choice != 6);

    delete[] rowHeadA;
    delete[] colHeadA;
    delete[] rowHeadB;
    delete[] colHeadB;
    delete[] rowHeadC;
```

```
    delete[] colHeadC;

    return 0;
}
```

Hasil Run:
Menu:
1. Input Matriks 1
2. Input Matriks 2
3. Jumlahkan Matriks dan Tampilkan Hasil
4. Tampilkan Matriks 1
5. Tampilkan Matriks 2
6. Keluar
Masukkan pilihan Anda: 1
Masukkan jumlah elemen non-nol: 4
Masukkan baris, kolom, dan nilai: 0 1 5
Masukkan baris, kolom, dan nilai: 0 4 6
Masukkan baris, kolom, dan nilai: 2 2 2
Masukkan baris, kolom, dan nilai: 1 3 4

Menu:
1. Input Matriks 1
2. Input Matriks 2
3. Jumlahkan Matriks dan Tampilkan Hasil
4. Tampilkan Matriks 1
5. Tampilkan Matriks 2
6. Keluar
Masukkan pilihan Anda: 2
Masukkan jumlah elemen non-nol: 2
Masukkan baris, kolom, dan nilai: 1 1 1
Masukkan baris, kolom, dan nilai: 1 3 4

Menu:
1. Input Matriks 1
2. Input Matriks 2
3. Jumlahkan Matriks dan Tampilkan Hasil
4. Tampilkan Matriks 1
5. Tampilkan Matriks 2
6. Keluar
Masukkan pilihan Anda: 3
Matriks Hasil (C = A + B):
0 5 0 0 6 0
0 1 0 4 0 0

0 0 2 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0