STRUKTUR DATA

Tugas Kelompok Multilist

Anggota:

- Muhammad Calvin (140810220061)
- Muhammad Is’ad (140810220065)
- Robert William (140810220087)

Tanggal mengumpulkan: 4 Juni 2023

UNIVERSITAS PADJAJARAN

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

Program Studi S-1 Teknik Informatika

2023

1. Multilist Divisi, Pegawai, Anak
    - Source Code

```
/*
Nama program: matriks.cpp
Anggota:
- Muhammad Calvin (140810220061)
- Muhammad Is'ad (140810220065)
- Robert William (140810220087)

Tanggal mengerjakan:04-06-2023
Deskripsi: Multilist Divisi, Pegawai, Anak
*/

#include <iostream>
using namespace std;

struct Anak {
   string namaAnak;
   Anak* nextAnak;
};

struct Pegawai {
   string namaPegawai;
   string jabatan;
   Pegawai* nextPegawai;
   Anak* listAnak;
};

struct Divisi {
   string namaDivisi;
   Pegawai* listPegawai;
   Divisi* nextDivisi;
};

using pointerPeg = Pegawai*;
using pointerDivisi = Divisi*;
using pointerAnak = Anak*;
```

```

// Membuat list divisi baru
void createListDiv(pointerDivisi& First) {
    First = NULL;
}

void createElementPeg(pointerPeg& pBaru) {
    pBaru = new Pegawai;
    cout << "Masukkan nama pegawai: ";
    cin >> pBaru->namaPegawai;
    pBaru->nextPegawai = NULL;
    pBaru->listAnak = NULL;
}

void createElementAnak(pointerAnak& pBaru) {
    pBaru = new Anak;
    cout << "Masukkan nama anak: ";
    cin >> pBaru->namaAnak;
    pBaru->nextAnak = NULL;
}

void traversalPeg(pointerPeg First) {
    if (First == NULL) {
        cout << "Tidak ada pegawai dalam divisi ini." << endl;
    } else {
        pointerPeg p = First;
        cout << "Daftar pegawai dalam divisi:" << endl;
        while (p != NULL) {
            cout << "- " << p->namaPegawai << endl;
            p = p->nextPegawai;
        }
    }
}

void linearSearch(pointerPeg First, string key, int& status, pointerPeg& pCari) {
    status = 0;  // status = 0 berarti tidak ditemukan
```

```
    pCari = First;
    while (pCari != NULL) {
      if (pCari->namaPegawai == key) {
        status = 1;  // status = 1 berarti ditemukan
        break;
      }
      pCari = pCari->nextPegawai;
    }
  }

  void insertFirstPeg(pointerPeg& First, pointerPeg pBaru) {
    if (First == NULL) {
      First = pBaru;
    } else {
      pBaru->nextPegawai = First;
      First = pBaru;
    }
  }

  void deleteFirstPeg(pointerPeg& First, pointerPeg& pHapus) {
    if (First == NULL) {
      pHapus = NULL;
      cout << "Tidak ada pegawai dalam divisi ini." << endl;
    } else if (First->nextPegawai == NULL) {
      pHapus = First;
      First = NULL;
    } else {
      pHapus = First;
      First = First->nextPegawai;
      pHapus->nextPegawai = NULL;
    }
  }

  void insertFirstAnak(pointerPeg& First, string key, pointerAnak pBaru) {
    pointerPeg pCari;
    int status;
```

```
    linearSearch(First, key, status, pCari);
    if (status == 1) {
        if (pCari->listAnak == NULL) {
            pCari->listAnak = pBaru;
        } else {
            pBaru->nextAnak = pCari->listAnak;
            pCari->listAnak = pBaru;
        }
        cout << "Anak berhasil ditambahkan." << endl;
    } else {
        cout << "Pegawai tidak ditemukan." << endl;
    }
}

void deleteFirstAnak(pointerPeg& First, string key, pointerAnak& pHapus) {
    pointerPeg pCari;
    int status;
    linearSearch(First, key, status, pCari);
    if (status == 1) {
        if (pCari->listAnak == NULL) {
            pHapus = NULL;
            cout << "Pegawai tidak memiliki anak." << endl;
        } else if (pCari->listAnak->nextAnak == NULL) {
            pHapus = pCari->listAnak;
            pCari->listAnak = NULL;
        } else {
            pHapus = pCari->listAnak;
            pCari->listAnak = pCari->listAnak->nextAnak;
            pHapus->nextAnak = NULL;
        }
        cout << "Anak berhasil dihapus." << endl;
    } else {
        cout << "Pegawai tidak ditemukan." << endl;
    }
}

```

```
void traversalOrtuAnak(pointerDivisi First) {
    pointerDivisi pDivisi = First;

    while (pDivisi != NULL) {
        cout << "Divisi: " << pDivisi->namaDivisi << endl;

        pointerPeg pPegawai = pDivisi->listPegawai;

        while (pPegawai != NULL) {
            cout << "  Pegawai: " << pPegawai->namaPegawai << endl;

            pointerAnak pAnak = pPegawai->listAnak;

            while (pAnak != NULL) {
                cout << "    Anak: " << pAnak->namaAnak << endl;
                pAnak = pAnak->nextAnak;
            }

            pPegawai = pPegawai->nextPegawai;
        }

        pDivisi = pDivisi->nextDivisi;
    }
}

void insertDivisi(pointerDivisi& First, const string& namaDivisi) {
    pointerDivisi pBaru = new Divisi;
    pBaru->namaDivisi = namaDivisi;
    pBaru->listPegawai = NULL;
    pBaru->nextDivisi = NULL;

    if (First == NULL) {
        First = pBaru;
    } else {
        pointerDivisi lastDivisi = First;
        while (lastDivisi->nextDivisi != NULL) {
```

```
            lastDivisi = lastDivisi->nextDivisi;
        }

        lastDivisi->nextDivisi = pBaru;
    }
}

void insertPegawai(pointerDivisi& First, const string& namaDivisi, pointerPeg pegawai) {
    pointerDivisi pDivisi = First;

    while (pDivisi != NULL) {
        if (pDivisi->namaDivisi == namaDivisi) {
            pegawai->nextPegawai = pDivisi->listPegawai;
            pDivisi->listPegawai = pegawai;
            break;
        }

        pDivisi = pDivisi->nextDivisi;
    }
}

void insertAnak(pointerDivisi& First, const string& namaPegawai, pointerAnak anak) {
    pointerDivisi pDivisi = First;

    while (pDivisi != NULL) {
        pointerPeg pPegawai = pDivisi->listPegawai;

        while (pPegawai != NULL) {
            if (pPegawai->namaPegawai == namaPegawai) {
                if (pPegawai->listAnak == NULL) {
                    pPegawai->listAnak = anak;
                } else {
                    anak->nextAnak = pPegawai->listAnak;
                    pPegawai->listAnak = anak;
                }
                break;
```

```
            }

            pPegawai = pPegawai->nextPegawai;
        }

        pDivisi = pDivisi->nextDivisi;
    }
}

void showMenu() {
    cout << "======== MENU ========" << endl;
    cout << "1. Tambah Divisi" << endl;
    cout << "2. Tambah Pegawai ke Divisi" << endl;
    cout << "3. Tambah Anak ke Pegawai" << endl;
    cout << "4. Tampilkan Orang Tua dan Anak" << endl;
    cout << "5. Keluar" << endl;
    cout << "======================" << endl;
    cout << "Pilihan: ";
}

int main() {
    pointerDivisi divisi = NULL;
    int pilihan = 0;
    string namaDivisi;
    string namaPegawai;
    string namaAnak;

    do {
        showMenu();
        cin >> pilihan;

        switch (pilihan) {
            case 1: {
                cout << "Nama Divisi: ";
                cin >> namaDivisi;
                insertDivisi(divisi, namaDivisi);
```

```
            cout << "Divisi " << namaDivisi << " ditambahkan." << endl;
            break;
        }
        case 2: {
            cout << "Nama Divisi: ";
            cin >> namaDivisi;
            cout << "Nama Pegawai: ";
            cin >> namaPegawai;
            pointerPeg pegawai = new Pegawai;
            pegawai->namaPegawai = namaPegawai;
            pegawai->listAnak = NULL;
            pegawai->nextPegawai = NULL;
            insertPegawai(divisi, namaDivisi, pegawai);
            cout << "Pegawai " << namaPegawai << " ditambahkan ke Divisi " << namaDivisi
<< "." << endl;
            break;
        }
        case 3: {
            cout << "Nama Pegawai: ";
            cin >> namaPegawai;
            cout << "Nama Anak: ";
            cin >> namaAnak;
            pointerAnak anak = new Anak;
            anak->namaAnak = namaAnak;
            anak->nextAnak = NULL;
            insertAnak(divisi, namaPegawai, anak);
            cout << "Anak " << namaAnak << " ditambahkan pada Pegawai " << namaPegawai
<< "." << endl;
            break;
        }
        case 4: {
            traversalOrtuAnak(divisi);
            break;
        }
        case 5: {
            cout << "Terima kasih. Program selesai." << endl;
            break;
```

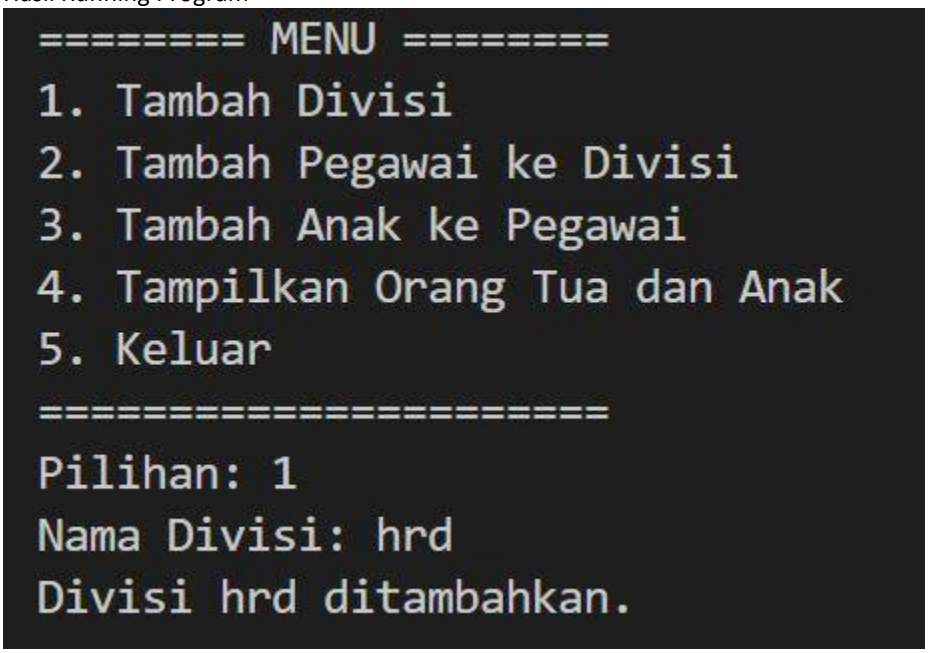
```
            }
            default: {
                cout << "Pilihan tidak valid." << endl;
                break;
            }
        }

        cout << endl;

    } while (pilihan != 5);

    return 0;
}
```

- Hasil Running Program



```
======== MENU ========
1. Tambah Divisi
2. Tambah Pegawai ke Divisi
3. Tambah Anak ke Pegawai
4. Tampilkan Orang Tua dan Anak
5. Keluar
======================
Pilihan: 2
Nama Divisi: hrd
Nama Pegawai: budi
Pegawai budi ditambahkan ke Divisi hrd.
```

```
======== MENU ========
1. Tambah Divisi
2. Tambah Pegawai ke Divisi
3. Tambah Anak ke Pegawai
4. Tampilkan Orang Tua dan Anak
5. Keluar
======================
Pilihan: 3
Nama Pegawai: budi
Nama Anak: jamal
Anak jamal ditambahkan pada Pegawai budi.
```

```
======== MENU ========
1. Tambah Divisi
2. Tambah Pegawai ke Divisi
3. Tambah Anak ke Pegawai
4. Tampilkan Orang Tua dan Anak
5. Keluar
======================
Pilihan: 4
Divisi: hrd
  Pegawai: cica
    Anak: didinding
  Pegawai: budi
    Anak: bangkit
    Anak: jamal
Divisi: produksi
  Pegawai: santi
Divisi: pemasaran
  Pegawai: lala
    Anak: suman

======== MENU ========
1. Tambah Divisi
2. Tambah Pegawai ke Divisi
3. Tambah Anak ke Pegawai
4. Tampilkan Orang Tua dan Anak
5. Keluar
======================
Pilihan: 5
Terima kasih. Program selesai.
```

2. Program Matriks Sparse
   - Source Code

```
/*
Nama program: matriks.cpp
Anggota:
- Muhammad Calvin (140810220061)
- Muhammad Is'ad (140810220065)
- Robert William (140810220087)

Tanggal mengerjakan:04-06-2023
Deskripsi: penjumlahan matriks sparse
*/

#include <iostream>
using namespace std;

struct elmtNode {
   int value;
   int column;
   int row;
   elmtNode* right;
   elmtNode* down;
};

struct headerNode {
   int index;
   headerNode* down;
   headerNode* right;
   elmtNode* next;
};

headerNode* create_header(int index) {
   headerNode* newNode = new headerNode;
   newNode->index = index;
   newNode->down = NULL;
   newNode->right = NULL;
   newNode->next = NULL;
```

```
    return newNode;
}

elmtNode* create_elmt(int row, int column, int value) {
    elmtNode* newNode = new elmtNode;
    newNode->row = row;
    newNode->column = column;
    newNode->value = value;
    newNode->right = NULL;
    newNode->down = NULL;
    return newNode;
}

void insertElement(headerNode* headerrow[], headerNode* headercolumn[], int row, int column, int value) {
    elmtNode* newNode = create_elmt(row, column, value);

    if (headerrow[row]->next == NULL || headercolumn[column]->next == NULL) {
        headerrow[row]->next = newNode;
        headercolumn[column]->next = newNode;
    } else {
        elmtNode* pBantu = headerrow[row]->next;
        while (pBantu->right != NULL) {
            pBantu = pBantu->right;
        }
        pBantu->right = newNode;
    }

    elmtNode* pBantu = headercolumn[column]->next;
    while (pBantu->down != NULL) {
        pBantu = pBantu->down;
    }
    pBantu->down = newNode;
}

void batasHorizontal(int jumlahColumn) {
    for (int i = 0; i < jumlahColumn * 5; i++) {
```

```
        cout << "-";
    }
    cout << endl;
}

void tampilkan(headerNode* headerrow[], headerNode* headercolumn[], int jumlahRow, int jumlahColumn) {
    batasHorizontal(jumlahColumn);

    for (int i = 0; i < jumlahRow; i++) {
        elmtNode* pBantu = headerrow[i]->next;
        cout << "| ";
        for (int j = 0; j < jumlahColumn; j++) {
            if (pBantu != NULL && pBantu->column == j) {
                cout << pBantu->value << " | ";
                pBantu = pBantu->right;
            } else {
                cout << "0 | ";
            }
        }
        cout << endl;
        batasHorizontal(jumlahColumn);
    }
}



void addMatriks(headerNode* headerrow1[], headerNode* headercolumn1[], headerNode* headerrow2[], headerNode* headercolumn2[], int jumlahRow, int jumlahColumn) {
    headerNode* headercolumnHasil[jumlahColumn];
    headerNode* headerrowHasil[jumlahRow];
    for (int i = 0; i < jumlahRow; i++) {
        headerrowHasil[i] = create_header(i);
    }

    for (int i = 0; i < jumlahColumn; i++) {
        headercolumnHasil[i] = create_header(i);
```

```
    }

    for (int i = 0; i < jumlahRow; i++) {
      elmtNode* pBantu1 = headerrow1[i]->next;
      elmtNode* pBantu2 = headerrow2[i]->next;
      while (pBantu1 != NULL || pBantu2 != NULL) {
        int row, column, value;

        if (pBantu1 == NULL) {
          row = pBantu2->row;
          column = pBantu2->column;
          value = pBantu2->value;
          pBantu2 = pBantu2->right;
        } else if (pBantu2 == NULL) {
          row = pBantu1->row;
          column = pBantu1->column;
          value = pBantu1->value;
          pBantu1 = pBantu1->right;
        } else {
          if (pBantu1->column < pBantu2->column) {
            row = pBantu1->row;
            column = pBantu1->column;
            value = pBantu1->value;
            pBantu1 = pBantu1->right;
          } else if (pBantu2->column < pBantu1->column) {
            row = pBantu2->row;
            column = pBantu2->column;
            value = pBantu2->value;
            pBantu2 = pBantu2->right;
          } else {
            row = pBantu1->row;
            column = pBantu1->column;
            value = pBantu1->value + pBantu2->value;
            pBantu1 = pBantu1->right;
            pBantu2 = pBantu2->right;
          }
```

```
            }

            insertElement(headerrowHasil, headercolumnHasil, row, column, value);
        }
    }
    cout << "Hasil penjumlahan:" << endl;
    tampilkan(headerrowHasil, headercolumnHasil, jumlahRow, jumlahColumn);
}

int main() {
    int jumlahValue;
    int jumlahColumn;
    int jumlahRow;

    cout << "Masukkan jumlah row dan column: ";
    cin >> jumlahRow >> jumlahColumn;
    cout << endl;
    cout << "Masukkan banyak value matriks 1: ";
    cin >> jumlahValue;

    headerNode* headerrow1[jumlahRow];
    headerNode* headercolumn1[jumlahColumn];
    for (int i = 0; i < jumlahRow; i++) {
        headerrow1[i] = create_header(i);
    }
    for (int i = 0; i < jumlahColumn; i++) {
        headercolumn1[i] = create_header(i);
    }

    cout << "Masukkan elemen matriks 1:" << endl;
    for (int i = 0; i < jumlahValue; i++) {
        int row, column, value;
        cout << "Masukkan posisi row, column, dan value elemen ke-" << i+1 << ": ";
        cin >> row >> column >> value;
        insertElement(headerrow1, headercolumn1, row, column, value);
    }
```

```
    cout << "Masukkan banyak value matriks 2: ";
    cin >> jumlahValue;

    headerNode* headerrow2[jumlahRow];
    headerNode* headercolumn2[jumlahColumn];
    for (int i = 0; i < jumlahRow; i++) {
        headerrow2[i] = create_header(i);
    }
    for (int i = 0; i < jumlahColumn; i++) {
        headercolumn2[i] = create_header(i);
    }

    cout << "Masukkan elemen matriks 2:" << endl;
    for (int i = 0; i < jumlahValue; i++) {
        int row, column, value;
        cout << "Masukkan posisi row, column, dan value elemen ke-" << i+1 << ": ";
        cin >> row >> column >> value;
        insertElement(headerrow2, headercolumn2, row, column, value);
    }

    cout << "\nMatriks 1:" << endl;
    tampilkan(headerrow1, headercolumn1, jumlahRow, jumlahColumn);
    cout << endl;

    cout << "Matriks 2:" << endl;
    tampilkan(headerrow2, headercolumn2, jumlahRow, jumlahColumn);
    cout << endl;

    addMatriks(headerrow1, headercolumn1, headerrow2, headercolumn2, jumlahRow,
jumlahColumn);

    return 0;
}
```

- Hasil Running Program