LAPORAN PRAKTIKUM

MODUL 13

MULTI LINKED LIST

Disusun Oleh:

Rizaldy Aulia Rachman (2311104051)

S1SE-07-02

Dosen :

Wahyu Andi Saputra, S.Pd., M.Eng

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. TUJUAN

1. Memahami penggunaan *Multi Linked list*.
2. Mengimplementasikan *Multi Linked list* dalam beberapa studi kasus.

II. LANDASAN TEORI

2.1 Multi Linked List

Linked list Multi List merupakan sekumpulan *list* yang berbeda yang memiliki suatu keterhubungan satu sama lain. Tiap elemen dalam *multi link list* dapat membentuk *list* sendiri. Biasanya ada yang bersifat sebagai *list* induk dan *list* anak .

2.2 Kelebihan Multi Linked List

1. Fleksibilitas Tinggi
    - Memungkinkan representasi hubungan kompleks antara data, seperti relasi hierarki, graf, atau tabel multidimensi.
2. Efisiensi dalam Menyimpan Data Terkait
    - Struktur ini memungkinkan setiap node untuk menyimpan beberapa referensi ke data yang relevan atau terkait.
3. Akses Relasi Lebih Cepat
    - Jika data memiliki banyak relasi atau koneksi, multi linked list memungkinkan navigasi lebih cepat melalui pointer tambahan.
4. Cocok untuk Struktur Data Kompleks
    - Digunakan dalam representasi graf atau multidimensional arrays, di mana konektivitas antar elemen sangat penting.

2.3 Kekurangan Multi Linked List

1. Pengelolaan Pointer Lebih Rumit
    - Karena setiap node memiliki banyak pointer, pengelolaan memori dan operasi seperti *insert* atau *delete* menjadi lebih kompleks.
2. Penggunaan Memori Lebih Besar
    - Adanya lebih dari satu pointer per node menyebabkan kebutuhan memori meningkat dibandingkan *single linked list*.
3. Sulit untuk Diimplementasikan
    - Dibutuhkan pemahaman lebih dalam tentang struktur data dan algoritma karena strukturnya lebih kompleks.

4. Proses Traversal Rumit
    - Traversal pada *multi linked list* memerlukan perhatian ekstra untuk menghindari pointer yang tidak valid atau perulangan tanpa akhir.

III. GUIDED

1. Guided1

Code:

```cpp
#include <iostream>
#include <string>

using namespace std;


struct Node {
    int data;
    Node* next;
    Node* child;

    Node(int val) : data(val), next(nullptr), child(nullptr) {}
};


class MultiLinkedList {
private:
    Node* head;

public:
    MultiLinkedList() : head(nullptr) {}


    void addParent(int data) {
        Node* newNode = new Node(data);
        newNode->next = head;
        head = newNode;
    }


    void addChild(int parentData, int childData) {
        Node* parent = head;
        while (parent != nullptr && parent->data != parentData) {
            parent = parent->next;
        }
        if (parent != nullptr) {
            Node* newChild = new Node(childData);
            newChild->next = parent->child;
            parent->child = newChild;
        } else {
            cout << "Parent not found!" << endl;
        }
    }


    void display() {
        Node* current = head;
        while (current != nullptr) {
            cout << "Parent: " << current->data << " -> ";
            Node* child = current->child;
            while (child != nullptr) {
                cout << child->data << " ";
                child = child->next;
            }
            cout << endl;
            current = current->next;
        }
    }

    ~MultiLinkedList() {

        while (head != nullptr) {
            Node* temp = head;
            head = head->next;


            while (temp->child != nullptr) {
                Node* childTemp = temp->child;
                temp->child = temp->child->next;
                delete childTemp;
            }
            delete temp;
        }
    }
};

int main() {
    MultiLinkedList mList;

    mList.addParent(1);
    mList.addParent(2);
    mList.addParent(3);

    mList.addChild(1, 10);
    mList.addChild(1, 11);
    mList.addChild(2, 20);
    mList.addChild(2, 20);
    mList.addChild(3, 30);
    mList.addChild(3, 30);
    mList.display();

    return 0;
}

```

Output:

```
Parent: 3 -> 30 30
Parent: 2 -> 20 20
Parent: 1 -> 11 10
PS C:\Praktikum Struktur data\pertemuan10\output>
```

2. Guided2

Code:

```cpp
#include <iostream>
#include <string>

using namespace std;


struct EmployeeNode {
    string name;
    EmployeeNode* next;
    EmployeeNode* subordinate;

    EmployeeNode(string empName) : name(empName), next(nullptr), subordinate(nullptr) {}
};


class EmployeeList {
private:
    EmployeeNode* head;

public:
    EmployeeList() : head(nullptr) {}


    void addEmployee(string name) {
        EmployeeNode* newEmployee = new EmployeeNode(name);
        newEmployee->next = head;
        head = newEmployee;
    }


    void addSubordinate(string managerName, string subordinateName) {
        EmployeeNode* manager = head;
        while (manager != nullptr && manager->name != managerName) {
            manager = manager->next;
        }
        if (manager != nullptr) {
            EmployeeNode* newSubordinate = new EmployeeNode(subordinateName);
            newSubordinate->next = manager->subordinate;
            manager->subordinate = newSubordinate;
        } else {
            cout << "Manager not found!" << endl;
        }
    }


    void display() {
        EmployeeNode* current = head;
        while (current != nullptr) {
            cout << "Manager: " << current->name << " -> ";
            EmployeeNode* sub = current->subordinate;
            while (sub != nullptr) {
                cout << sub->name << " ";
                sub = sub->next;
            }
            cout << endl;
            current = current->next;
        }
    }

    ~EmployeeList() {

        while (head != nullptr) {
            EmployeeNode* temp = head;
            head = head->next;


            while (temp->subordinate != nullptr) {
                EmployeeNode* subTemp = temp->subordinate;
                temp->subordinate = temp->subordinate->next;
                delete subTemp;
            }
            delete temp;
        }
    }
};

int main() {
    EmployeeList empList;

    empList.addEmployee("Alice");
    empList.addEmployee("Bob");
    empList.addEmployee("Charlie");

    empList.addSubordinate("Alice", "David");
    empList.addSubordinate("Alice", "Eve");
    empList.addSubordinate("Bob", "Frank");

    empList.addSubordinate("Charlie", "Frans");
    empList.addSubordinate("Charlie", "Brian");

    empList.display();

    return 0;
}

```

Output:

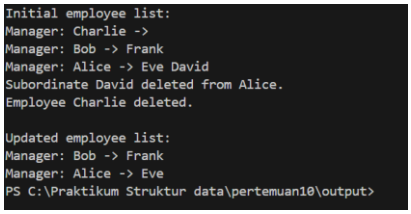
```
Manager: Charlie -> Brian Frans
Manager: Bob -> Frank
Manager: Alice -> Eve David
PS C:\Praktikum Struktur data\pertemuan10\output>
```

3. Guided3

Code:

```cpp
#include <iostream>
#include <string>

using namespace std;

// Struktur untuk node karyawan
struct EmployeeNode {
    string name; // Nama karyawan
    EmployeeNode* next; // Pointer ke karyawan berikutnya
    EmployeeNode* subordinate; // Pointer ke subordinate pertama

    EmployeeNode(string empName) : name(empName), next(nullptr), subordinate(nullptr) {}
};

// Kelas untuk Multi Linked List Karyawan
class EmployeeList {
private:
    EmployeeNode* head; // Pointer ke kepala list

public:
    EmployeeList() : head(nullptr) {}

    // Menambahkan karyawan (induk)
    void addEmployee(string name) {
        EmployeeNode* newEmployee = new EmployeeNode(name);
        newEmployee->next = head; // Menyambungkan ke karyawan sebelumnya
        head = newEmployee; // Memperbarui head
    }

    // Menambahkan subordinate ke karyawan tertentu
    void addSubordinate(string managerName, string subordinateName) {
        EmployeeNode* manager = head;
        while (manager != nullptr && manager->name != managerName) {
            manager = manager->next;
        }
        if (manager != nullptr) { // Jika manajer ditemukan
            EmployeeNode* newSubordinate = new EmployeeNode(subordinateName);
            newSubordinate->next = manager->subordinate; // Menyambungkan ke subordinate sebelumnya
            manager->subordinate = newSubordinate; // Memperbarui subordinate
        } else {
            cout << "Manager not found!" << endl;
        }
    }

    // Menghapus karyawan (induk)
    void deleteEmployee(string name) {
        EmployeeNode** current = &head;
        while (*current != nullptr && (*current)->name != name) {
            current = &((*current)->next);
        }

        if (*current != nullptr) { // Jika karyawan ditemukan
            EmployeeNode* toDelete = *current;
            *current = (*current)->next;

            // Hapus semua subordinate dari node ini
            while (toDelete->subordinate != nullptr) {
                EmployeeNode* subTemp = toDelete->subordinate;
                toDelete->subordinate = toDelete->subordinate->next;
                delete subTemp;
            }
            delete toDelete;
            cout << "Employee " << name << " deleted." << endl;
        } else {
            cout << "Employee not found!" << endl;
        }
    }

    // Menghapus subordinate dari karyawan tertentu
    void deleteSubordinate(string managerName, string subordinateName) {
        EmployeeNode* manager = head;
        while (manager != nullptr && manager->name != managerName) {
            manager = manager->next;
        }

        if (manager != nullptr) { // Jika manajer ditemukan
            EmployeeNode** currentSub = &(manager->subordinate);
            while (*currentSub != nullptr && (*currentSub)->name != subordinateName) {
                currentSub = &((*currentSub)->next);
            }

            if (*currentSub != nullptr) { // Jika subordinate ditemukan
                EmployeeNode* toDelete = *currentSub;
                *currentSub = (*currentSub)->next; // Menghapus dari list

                delete toDelete; // Menghapus node subordinate
                cout << "Subordinate " << subordinateName << " deleted from " << managerName << "." << endl;
            } else {
                cout << "Subordinate not found!" << endl;
            }
        } else {
            cout << "Manager not found!" << endl;
        }
    }

    // Menampilkan daftar karyawan dan subordinate mereka
    void display() {
        EmployeeNode* current = head;
        while (current != nullptr) {
            cout << "Manager: " << current->name << " -> ";
            EmployeeNode* sub = current->subordinate;
            while (sub != nullptr) {
                cout << sub->name << " ";
                sub = sub->next;
            }
            cout << endl;
            current = current->next;
        }
    }

    ~EmployeeList() {
        // Destructor untuk membersihkan memori
        while (head != nullptr) {
            EmployeeNode* temp = head;
            head = head->next;

            // Hapus semua subordinate dari node ini
            while (temp->subordinate != nullptr) {
                EmployeeNode* subTemp = temp->subordinate;
                temp->subordinate = temp->subordinate->next;
                delete subTemp;
            }
            delete temp;
        }
    }
};

int main() {
    EmployeeList empList;

    empList.addEmployee("Alice");
    empList.addEmployee("Bob");
    empList.addEmployee("Charlie");

    empList.addSubordinate("Alice", "David");
    empList.addSubordinate("Alice", "Eve");
    empList.addSubordinate("Bob", "Frank");

    cout << "Initial employee list:" << endl;
    empList.display(); // Menampilkan isi daftar karyawan

    empList.deleteSubordinate("Alice", "David"); // Menghapus David dari Alice
    empList.deleteEmployee("Charlie"); // Menghapus Charlie

    cout << "\nUpdated employee list:" << endl;
    empList.display(); // Menampilkan isi daftar setelah penghapusan

    return 0;
}
```

Output:



IV. UNGUIDED

1. Manajemen Data Pegawai dan Proyek
Buatlah program menggunakan Multi Linked List untuk menyimpan data pegawai dan proyek yang dikelola setiap pegawai.
- Setiap pegawai memiliki data: Nama Pegawai dan ID Pegawai.
- Setiap proyek memiliki data: Nama Proyek** dan **Durasi (bulan).
Instruksi:
1. Masukkan data pegawai berikut:
- Pegawai 1: Nama = "Andi", ID = "P001".
- Pegawai 2: Nama = "Budi", ID = "P002".
- Pegawai 3: Nama = "Citra", ID = "P003".
2. Tambahkan proyek ke pegawai:
- Proyek 1: Nama = "Aplikasi Mobile", Durasi = 12 bulan (Untuk Andi).
- Proyek 2: Nama = "Sistem Akuntansi", Durasi = 8 bulan (Untuk Budi).
- Proyek 3: Nama = "E-commerce", Durasi = 10 bulan (Untuk Citra).
3. Tambahkan proyek baru:
- Proyek 4: Nama = "Analisis Data", Durasi = 6 bulan (Untuk Andi).
4. Hapus proyek "Aplikasi Mobile" dari Andi.
5. Tampilkan data pegawai dan proyek mereka.

Jawaban:

Code:

```cpp
#include <iostream>
#include <string>
using namespace std;

struct Proyek {
    string namaProyek;
    int durasi;
    Proyek* nextProyek;
};

struct Pegawai {
    string namaPegawai;
    string idPegawai;
    Proyek* headProyek;
    Pegawai* nextPegawai;
};

Pegawai* headPegawai = nullptr;

void tambahPegawai(string nama, string id) {
    Pegawai* pegawaiBaru = new Pegawai;
    pegawaiBaru->namaPegawai = nama;
    pegawaiBaru->idPegawai = id;
    pegawaiBaru->headProyek = nullptr;
    pegawaiBaru->nextPegawai = headPegawai;
    headPegawai = pegawaiBaru;
}

void tambahProyek(string idPegawai, string namaProyek, int durasi) {
    Pegawai* currPegawai = headPegawai;
    while (currPegawai != nullptr) {
        if (currPegawai->idPegawai == idPegawai) {
            Proyek* proyekBaru = new Proyek;
            proyekBaru->namaProyek = namaProyek;
            proyekBaru->durasi = durasi;
            proyekBaru->nextProyek = currPegawai->headProyek;
            currPegawai->headProyek = proyekBaru;
            return;
        }
        currPegawai = currPegawai->nextPegawai;
    }
    cout << "Pegawai dengan ID " << idPegawai << " tidak ditemukan!\n";
}

void hapusProyek(string idPegawai, string namaProyek) {
    Pegawai* currPegawai = headPegawai;
    while (currPegawai != nullptr) {
        if (currPegawai->idPegawai == idPegawai) {
            Proyek* currProyek = currPegawai->headProyek;
            Proyek* prevProyek = nullptr;
            while (currProyek != nullptr) {
                if (currProyek->namaProyek == namaProyek) {
                    if (prevProyek == nullptr) {
                        currPegawai->headProyek = currProyek->nextProyek;
                    } else {
                        prevProyek->nextProyek = currProyek->nextProyek;
                    }
                    delete currProyek;
                    return;
                }
                prevProyek = currProyek;
                currProyek = currProyek->nextProyek;
            }
        }
        currPegawai = currPegawai->nextPegawai;
    }
    cout << "Proyek " << namaProyek << " tidak ditemukan untuk pegawai dengan ID " << idPegawai << "!\n";
}

void tampilkanData() {
    Pegawai* currPegawai = headPegawai;
    while (currPegawai != nullptr) {
        cout << "Pegawai: " << currPegawai->namaPegawai << " (ID: " << currPegawai->idPegawai << ")\n";
        Proyek* currProyek = currPegawai->headProyek;
        while (currProyek != nullptr) {
            cout << "  Proyek: " << currProyek->namaProyek << " (Durasi: " << currProyek->durasi << " bulan)\n";
            currProyek = currProyek->nextProyek;
        }
        currPegawai = currPegawai->nextPegawai;
    }
}

int main() {
    int pilihan;
    do {
        cout << "\n=Menu:\n";
        cout << "1. Tambah Pegawai\n";
        cout << "2. Tambah Proyek\n";
        cout << "3. Hapus Proyek\n";
        cout << "4. Tampilkan Data\n";
        cout << "5. Keluar\n";
        cout << "Pilihan: ";
        cin >> pilihan;
        cin.ignore();

        if (pilihan == 1) {
            string nama, id;
            cout << "Masukkan Nama Pegawai: ";
            getline(cin, nama);
            cout << "Masukkan ID Pegawai: ";
            getline(cin, id);
            tambahPegawai(nama, id);
        } else if (pilihan == 2) {
            string id, namaProyek;
            int durasi;
            cout << "Masukkan ID Pegawai: ";
            getline(cin, id);
            cout << "Masukkan Nama Proyek: ";
            getline(cin, namaProyek);
            cout << "Masukkan Durasi Proyek (bulan): ";
            cin >> durasi;
            cin.ignore();
            tambahProyek(id, namaProyek, durasi);
        } else if (pilihan == 3) {
            string id, namaProyek;
            cout << "Masukkan ID Pegawai: ";
            getline(cin, id);
            cout << "Masukkan Nama Proyek yang akan dihapus: ";
            getline(cin, namaProyek);
            hapusProyek(id, namaProyek);
        } else if (pilihan == 4) {
            tampilkanData();
        } else if (pilihan != 5) {
            cout << "Pilihan tidak valid.\n";
        }
    } while (pilihan != 5);

    return 0;
}

```

1. Output intruksi 1:

```
Menu:
1. Tambah Pegawai
2. Tambah Proyek
3. Hapus Proyek
4. Tampilkan Data
5. Keluar
Pilihan: 1
Masukkan Nama Pegawai: Andi
Masukkan ID Pegawai: P001

Menu:
1. Tambah Pegawai
2. Tambah Proyek
3. Hapus Proyek
4. Tampilkan Data
5. Keluar
Pilihan: 1
Masukkan Nama Pegawai: Budi
Masukkan ID Pegawai: P002

Menu:
1. Tambah Pegawai
2. Tambah Proyek
3. Hapus Proyek
4. Tampilkan Data
5. Keluar
Pilihan: 1
Masukkan Nama Pegawai: Citra
Masukkan ID Pegawai: P003
```

2. Output intruksi 2:

```
Menu:
1. Tambah Pegawai
2. Tambah Proyek
3. Hapus Proyek
4. Tampilkan Data
5. Keluar
Pilihan: 2
Masukkan ID Pegawai: P001
Masukkan Nama Proyek: Aplikasi Mobile
Masukkan Durasi Proyek (bulan): 12

Menu:
1. Tambah Pegawai
2. Tambah Proyek
3. Hapus Proyek
4. Tampilkan Data
5. Keluar
Pilihan: 2
Masukkan ID Pegawai: P002
Masukkan Nama Proyek: Sistem Akuntansi
Masukkan Durasi Proyek (bulan): 8

Menu:
1. Tambah Pegawai
2. Tambah Proyek
3. Hapus Proyek
4. Tampilkan Data
5. Keluar
Pilihan: 2
Masukkan ID Pegawai: P003
Masukkan Nama Proyek: E-commerce
Masukkan Durasi Proyek (bulan): 10
```

3. Output intruksi 3:

```
Menu:
1. Tambah Pegawai
2. Tambah Proyek
3. Hapus Proyek
4. Tampilkan Data
5. Keluar
Pilihan: 2
Masukkan ID Pegawai: P001
Masukkan Nama Proyek: Analisis Data
Masukkan Durasi Proyek (bulan): 6
```

4. Output intruksi 4:

```
Menu:
1. Tambah Pegawai
2. Tambah Proyek
3. Hapus Proyek
4. Tampilkan Data
5. Keluar
Pilihan: 3
Masukkan ID Pegawai: P001
Masukkan Nama Proyek yang akan dihapus: Aplikasi Mobile
```

5. Output intruksi 5:

```
Menu:
1. Tambah Pegawai
2. Tambah Proyek
3. Hapus Proyek
4. Tampilkan Data
5. Keluar
Pilihan: 4
Pegawai: Andi (ID: P001)
  Proyek: Analisis Data (Durasi: 6 bulan)
Pegawai: Citra (ID: P003)
  Proyek: E-commerce (Durasi: 10 bulan)
Pegawai: Budi (ID: P002)
  Proyek: Sistem Akuntansi (Durasi: 8 bulan)
```

2. Sistem Manajemen Buku Perpustakaan

Gunakan Multi Linked List untuk menyimpan data anggota perpustakaan dan daftar buku yang dipinjam.

- Setiap anggota memiliki data: Nama Anggota dan ID Anggota.
- Setiap buku memiliki data: Judul Buku dan Tanggal Pengembalian.

Instruksi:

1. Masukkan data anggota berikut:
- Anggota 1: Nama = "Rani", ID = "A001".
- Anggota 2: Nama = "Dito", ID = "A002".
- Anggota 3: Nama = "Vina", ID = "A003".

2. Tambahkan buku yang dipinjam:

- Buku 1: Judul = "Pemrograman C++", Pengembalian = "01/12/2024" (Untuk Rani).
- Buku 2: Judul = "Algoritma Pemrograman", Pengembalian = "15/12/2024" (Untuk Dito).
3. Tambahkan buku baru:
- Buku 3: Judul = "Struktur Data", Pengembalian = "10/12/2024" (Untuk Rani).
4. Hapus anggota Dito beserta buku yang dipinjam.
5. Tampilkan seluruh data anggota dan buku yang dipinjam.

Jawaban:

Code:

```cpp
#include <iostream>
#include <string>
using namespace std;

struct Buku {
    string judulBuku;
    string tanggalPengembalian;
    Buku* nextBuku;
};

struct Anggota {
    string namaAnggota;
    string idAnggota;
    Buku* headBuku;
    Anggota* nextAnggota;
};

Anggota* headAnggota = nullptr;

void tambahAnggota(string nama, string id) {
    Anggota* anggotaBaru = new Anggota;
    anggotaBaru->namaAnggota = nama;
    anggotaBaru->idAnggota = id;
    anggotaBaru->headBuku = nullptr;
    anggotaBaru->nextAnggota = headAnggota;
    headAnggota = anggotaBaru;
}

void tambahBuku(string idAnggota, string judulBuku, string tanggalPengembalian) {
    Anggota* currAnggota = headAnggota;
    while (currAnggota != nullptr) {
        if (currAnggota->idAnggota == idAnggota) {
            Buku* bukuBaru = new Buku;
            bukuBaru->judulBuku = judulBuku;
            bukuBaru->tanggalPengembalian = tanggalPengembalian;
            bukuBaru->nextBuku = currAnggota->headBuku;
            currAnggota->headBuku = bukuBaru;
            return;
        }
        currAnggota = currAnggota->nextAnggota;
    }
    cout << "Anggota dengan ID " << idAnggota << " tidak ditemukan!\n";
}

void hapusAnggota(string idAnggota) {
    Anggota* currAnggota = headAnggota;
    Anggota* prevAnggota = nullptr;
    while (currAnggota != nullptr) {
        if (currAnggota->idAnggota == idAnggota) {
            if (prevAnggota == nullptr) {
                headAnggota = currAnggota->nextAnggota;
            } else {
                prevAnggota->nextAnggota = currAnggota->nextAnggota;
            }
            // Hapus semua buku yang dipinjam anggota
            Buku* currBuku = currAnggota->headBuku;
            while (currBuku != nullptr) {
                Buku* temp = currBuku;
                currBuku = currBuku->nextBuku;
                delete temp;
            }
            delete currAnggota;
            return;
        }
        prevAnggota = currAnggota;
        currAnggota = currAnggota->nextAnggota;
    }
    cout << "Anggota dengan ID " << idAnggota << " tidak ditemukan!\n";
}

void tampilkanData() {
    Anggota* currAnggota = headAnggota;
    while (currAnggota != nullptr) {
        cout << "Anggota: " << currAnggota->namaAnggota << " (ID: " << currAnggota->idAnggota << ")\n";
        Buku* currBuku = currAnggota->headBuku;
        while (currBuku != nullptr) {
            cout << "  Buku: " << currBuku->judulBuku << " (Pengembalian: " << currBuku->tanggalPengembalian << ")\n";
            currBuku = currBuku->nextBuku;
        }
        currAnggota = currAnggota->nextAnggota;
    }
}

int main() {
    int pilihan;
    do {
        cout << "\nMenu:\n";
        cout << "1. Tambah Anggota\n";
        cout << "2. Tambah Buku\n";
        cout << "3. Hapus Anggota\n";
        cout << "4. Tampilkan Data\n";
        cout << "5. Keluar\n";
        cout << "Pilihan: ";
        cin >> pilihan;
        cin.ignore();

        if (pilihan == 1) {
            string nama, id;
            cout << "Masukkan Nama Anggota: ";
            getline(cin, nama);
            cout << "Masukkan ID Anggota: ";
            getline(cin, id);
            tambahAnggota(nama, id);
        } else if (pilihan == 2) {
            string id, judulBuku, tanggalPengembalian;
            cout << "Masukkan ID Anggota: ";
            getline(cin, id);
            cout << "Masukkan Judul Buku: ";
            getline(cin, judulBuku);
            cout << "Masukkan Tanggal Pengembalian: ";
            getline(cin, tanggalPengembalian);
            tambahBuku(id, judulBuku, tanggalPengembalian);
        } else if (pilihan == 3) {
            string id;
            cout << "Masukkan ID Anggota yang akan dihapus: ";
            getline(cin, id);
            hapusAnggota(id);
        } else if (pilihan == 4) {
            tampilkanData();
        } else if (pilihan != 5) {
            cout << "Pilihan tidak valid.\n";
        }
    } while (pilihan != 5);

    return 0;
}
```

1. Output intruksi 1:

```
Menu:
1. Tambah Anggota
2. Tambah Buku
3. Hapus Anggota
4. Tampilkan Data
5. Keluar
Pilihan: 1
Masukkan Nama Anggota: Rani
Masukkan ID Anggota: A001

Menu:
1. Tambah Anggota
2. Tambah Buku
3. Hapus Anggota
4. Tampilkan Data
5. Keluar
Pilihan: 1
Masukkan Nama Anggota: Dito
Masukkan ID Anggota: A002

Menu:
1. Tambah Anggota
2. Tambah Buku
3. Hapus Anggota
4. Tampilkan Data
5. Keluar
Pilihan: 1
Masukkan Nama Anggota: Vina
Masukkan ID Anggota: A003
```

2. Output intruksi 2:

```
Menu:
1. Tambah Anggota
2. Tambah Buku
3. Hapus Anggota
4. Tampilkan Data
5. Keluar
Pilihan: 2
Masukkan ID Anggota: A001
Masukkan Judul Buku: Pemrograman C++
Masukkan Tanggal Pengembalian: 01/12/2024

Menu:
1. Tambah Anggota
2. Tambah Buku
3. Hapus Anggota
4. Tampilkan Data
5. Keluar
Pilihan: 2
Masukkan ID Anggota: A002
Masukkan Judul Buku: Algoritma Pemrograman
Masukkan Tanggal Pengembalian: 15/12/2024
```

3. Output intruksi 3:

```
Menu:
1. Tambah Anggota
2. Tambah Buku
3. Hapus Anggota
4. Tampilkan Data
5. Keluar
Pilihan: 2
Masukkan ID Anggota: A001
Masukkan Judul Buku: Struktur Data
Masukkan Tanggal Pengembalian: 10/12/2024
```

4. Output intruksi 4:

```
Menu:
1. Tambah Anggota
2. Tambah Buku
3. Hapus Anggota
4. Tampilkan Data
5. Keluar
Pilihan: 3
Masukkan ID Anggota yang akan dihapus: A002
```

5. Output intruksi 5:

```
Menu:
1. Tambah Anggota
2. Tambah Buku
3. Hapus Anggota
4. Tampilkan Data
5. Keluar
Pilihan: 4
Anggota: Vina (ID: A003)
Anggota: Rani (ID: A001)
  Buku: Struktur Data (Pengembalian: 10/12/2024)
  Buku: Pemrograman C++ (Pengembalian: 01/12/2024)
```