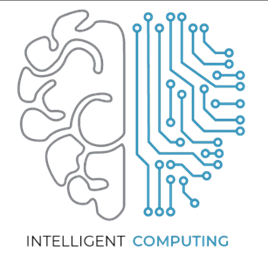
**LAPORAN PRAKTIKUM**

**STRUKTUR DATA**



NAMA : Lulu El Maknun

NIM : 202332124

KELAS : B

DOSEN : Nasril Sany , S.Kom., M.Kom

NO.PC :

ASISTEN : 1. Viana Salsabila Fairuz Syahla

2. M Raehand Fathahillah

3. Kashrina Masyid Azka

4. Ridho Chaerullah

**INSTITUT TEKNOLOGI PLN**

**SISTEM INFORMASI**

**2024**

**Pengertian Linked List**

Linked List adalah salah satu struktur data penting dalam pemrograman yang digunakan untuk menyimpan dan mengelola data secara dinamis. Struktur data ini memungkinkan kita untuk dengan mudah membuat tempat baru untuk menyimpan data kapan saja dibutuhkan.

Dalam Linked List, data disimpan dalam bentuk simpul atau node yang saling terhubung satu sama lain dengan menggunakan referensi atau alamat dari simpul selanjutnya dalam urutan. Setiap simpul berisi dua hal penting: data yang ingin kita simpan dan alamat referensi ke simpul berikutnya dalam urutan. Dengan begitu, setiap simpul dapat menyimpan data dan mengetahui alamat simpul selanjutnya.

Linked list dapat diilustrasikan seperti kereta api, dimana kereta api terdiri dari gerbong-gerbong yang saling terhubung yang dapat mengangkut penumpang. Gerbong disini setara dengan node dalam link list yang berfungsi untuk menyimpan data.

**Fungsi Linked List**

Linked List memiliki beberapa fungsi penting, antara lain:

1. Menyimpan dan mengelola data dalam urutan tertentu.
2. Memudahkan penambahan dan penghapusan data secara dinamis tanpa harus menggeser data lain.
3. Digunakan dalam implementasi berbagai algoritma dan struktur data lain seperti stack (timpukan) dan queue (antrean).

**Jenis-jenis Linked List**

Linked List adalah struktur data penting dalam pemrograman yang digunakan untuk menyimpan dan mengatur data secara dinamis.

1. Pada Single Linked List, setiap simpul memiliki dua bagian: data yang ingin disimpan dan referensi ke simpul berikutnya. Simpul terakhir menunjuk ke null sebagai penanda akhir Linked List. Keuntungan Single Linked List adalah struktur datanya yang ringan karena hanya memerlukan satu referensi untuk setiap simpul. Namun, untuk mencari elemen tertentu, kita harus melintasi seluruh Linked List dari awal hingga elemen yang diinginkan.
2. Double Linked List lebih kompleks daripada Single Linked List. Setiap simpul memiliki tiga bagian: data, referensi ke simpul berikutnya, dan referensi ke simpul sebelumnya. Keuntungan Double Linked List adalah kemampuan untuk melintasi Linked List maju (dari depan ke belakang) dan mundur (dari belakang ke depan) dengan mudah.

Namun, kelemahannya adalah memerlukan lebih banyak ruang memori karena setiap simpul harus menyimpan dua referensi.

1. Circular Linked List adalah tipe Linked List yang membentuk lingkaran, dengan simpul terakhir menunjuk kembali ke simpul pertama. Tidak ada simpul yang menunjuk ke null seperti pada Single Linked List dan Double Linked List. Keuntungan Circular Linked List adalah kemampuan untuk berputar terus menerus tanpa perlu mengubah referensi. Circular Linked List sering digunakan dalam implementasi antrian berputar dan penjadwalan.
2. Multiple Linked List terdiri dari beberapa kumpulan Linked List yang terpisah, tetapi dapat diakses bersamaan. Setiap kumpulan Linked List berisi simpul-simpul yang terkait dalam satu himpunan.  
   Jenis Linked List ini memungkinkan pengaturan data dalam kelompok-kelompok terpisah dan menjaga struktur yang rapi dan terorganisir. Dengan pemahaman tentang perbedaan jenis-jenis Linked List ini, kita dapat membuat pilihan yang tepat dalam menggunakan struktur data yang sesuai dengan kebutuhan aplikasi.

**Perbedaan Antara Array dan Linked List**

Array dan Linked List adalah dua cara berbeda untuk menyimpan dan mengatur data dalam pemrograman.

**Pertama**, Array memerlukan alokasi memori kontiguitas, yang berarti elemen data harus disusun secara berurutan dalam memori. Ini membuat penambahan atau penghapusan elemen menjadi lebih sulit, karena kita perlu menggeser elemen lain untuk memberi ruang pada posisi yang diinginkan.

Selain itu, ukuran Array tetap dan ditentukan saat deklarasi, sehingga sulit untuk mengubah ukuran Array secara dinamis. Meskipun Array memberikan akses elemen yang cepat melalui indeks, namun ada keterbatasan dalam fleksibilitas dan efisiensi ketika datanya sering berubah. Elemen array hanya berisi data saja sedangkan linked list berisi data dan node untuk menghubungkan ke data berikutnya.

**Kedua,**Linked List menggunakan pendekatan yang berbeda. Linked List tidak memerlukan alokasi memori kontiguitas, sehingga setiap elemen data dapat ditempatkan di lokasi memori yang berbeda dan dihubungkan melalui referensi. Ini membuat penambahan dan penghapusan elemen menjadi lebih efisien, karena kita hanya perlu mengatur ulang referensi antar Node.

Ukuran Linked List dapat berubah selama runtime, sehingga memungkinkan penambahan atau penghapusan elemen dengan mudah. Namun, Linked List memerlukan pencarian melalui elemen sebelumnya untuk mengakses data, sehingga akses elemen membutuhkan waktu lebih lama. Meskipun Linked List memiliki keterbatasan, namun ini merupakan pilihan yang fleksibel untuk situasi di mana data sering berubah atau ketika ukuran data tidak diketahui sebelumnya.

**Penggunaan pointer dalam Linked List**

Berikut adalah beberapa informasi tentang penggunaan pointer dalam linked list:

1. Linked lists adalah sebuah data structure yang terdiri dari sekumpulan node, setiap node berisi referensi ke node berikutnya dalam list.
2. Pointers digunakan untuk mengimplementasikan linked lists, karena mereka memungkinkan efisien penambahan dan penghapusan node pada setiap posisi dalam list.
3. Dalam linked list, setiap node berisi pointer ke node berikutnya dalam list, serta pointer ke node sebelumnya (jika list adalah linked list ganda).
4. Node awal list adalah node pertama dalam list, sedangkan node akhir adalah node terakhir.
5. Dalam linked list single, node awal dan node akhir tidak eksplisit disebutkan, dan list diteruskan melalui pointer dari node satu ke node berikutnya.
6. Dalam linked list ganda, node awal dan node akhir eksplisit disebutkan, dan list diteruskan melalui pointer darinode satu ke node berikutnya, serta dari node sebelumnya ke node saat ini.
7. Penggunaan pointers dalam linked lists memungkinkan efisien penambahan dan penghapusan node pada setiap posisi dalam list, serta efisien traversal list.
8. Namun, penggunaan pointers juga memerlukan pengelolaan yang peduli untuk mencegah leak memory dan error lainnya.

Dalam hal yang spesifik tentang penggunaan pointers dalam linked lists, penting untuk diingat bahwa penggunaan pointers adalah detail implementasi yang tidak diberikan kepada pengguna. Pengguna interaksi dengan linked list melalui sekumpulan operasi yang disertakan dalam data yang disimpan dalam list, sehingga pengguna tidak perlu tahu detail spesifik tentang bagaimana list disimpan.Contohnya, linked list mungkin memberikan operasi seperti "insert at head," "insert at tail," "delete at head," dan "delete at tail," yang memungkinkan pengguna untuk menambahkan dan menghapus node dari list tanpa perlu tahu detail spesifik tentang bagaimana list disimpan. Implementasi operasi ini biasanya melibatkan manipulasi pointer ke node dalam list, tetapi pengguna tidak perlu tahu detail tersebut.Secara umum, penggunaan pointers dalam linked lists merupakan aspek yang penting dari pengimplementasian, karena mereka memungkinkan efisien penambahan dan penghapusan node pada setiap posisi dalam list, serta efisien traversal list. Namun, penggunaan pointers juga memerlukan pengelolaan yang peduli untuk mencegah leak memory dan error lainnya**.**

**Struktur Node**

Struktur node dalam linked list adalah sebuah unit data yang mengandung data dan referensi ke node berikutnya. Setiap node dalam linked list berisi dua hal penting: data yang ingin disimpan dan alamat referensi ke simpul berikutnya dalam urutan. Dengan begitu, setiap simpul dapat menyimpan data dan mengetahui alamat simpul selanjutnya. Setiap linked list memiliki dua elemen khusus, yaitu "head" dan "tail".

### Cara kerja dari linked list adalah sebagai berikut:

Linked list adalah representasi data yang memanfaatkan pointer untuk menghubungkan antara node-node yang berisi data. Cara kerja linked list melibatkan manipulasi node-node yang terhubung oleh pointer. Untuk menambahkan data baru ke linked list, node baru dibuat dan dihubungkan dengan node yang sudah ada menggunakan pointer. Untuk menghapus data dari linked list, node yang akan dihapus dicari dan dihapus dari linked list. Pointer pada node sebelum dan sesudah node yang dihapus diperbarui untuk menghindari adanya pointer yang rusak. Untuk mencari data tertentu pada linked list, node-node diperulangi hingga data yang dicari ditemukan atau sampai akhir linked list. Untuk menampilkan data pada linked list, node-node diperulangi dan data pada setiap node ditampilkan. Dalam implementasi linked list, seringkali digunakan dua pointer pada setiap node, yaitu pointer ke node sebelumnya (prev) dan pointer ke node selanjutnya (next). Hal ini memungkinkan linked list dapat diakses dari kedua arah, yaitu dari node awal ke node akhir dan sebaliknya. Linked list yang memanfaatkan dua pointer ini disebut linked list doubly (dua arah).

**Keuntungan Linked List**

* **Struktur data dinamis**: Linked list adalah himpunan dinamis sehingga dapat bertambah dan menyusut saat runtime dengan mengalokasikan dan membatalkan alokasi memori. Jadi kita tidak perlu memberikan ukuran awal dari linked list.
* **Tidak boros memori**: Dalam linked list, pemanfaatan memori yang efisien dapat dicapai karena ukuran linked list bertambah atau berkurang pada runtime sehingga tidak ada pemborosan memori dan tidak perlu mengalokasikan memori sebelumnya.
* **Implementasi**: Struktur data linier seperti stack dan queue seringkali mudah diimplementasikan menggunakan linked list.
* **Operasi penyisipan dan penghapusan**: Operasi penyisipan dan penghapusan cukup mudah dalam linked list. Kita tidak perlu menggeser elemen setelah operasi penyisipan atau penghapusan elemen, hanya alamat yang ada di pointer berikutnya saja yang perlu diperbarui.

**Kelemahan Linked List**

* **Penggunaan memori**: Linked list memerlukan lebih banyak memori dibandingkan dengan array. Karena dalam linked list, pointer juga perlu menyimpan alamat elemen berikutnya dan membutuhkan memori tambahan untuk dirinya sendiri.
* **Traversal**: Dalam traversal, linked list lebih banyak memakan waktu dibandingkan dengan array. Akses langsung ke elemen tidak bisa dilakukan pada linked list seperti array yang dapat akses elemen berdasarkan indeks. Untuk mengakses sebuah simpul pada posisi n dari linked list, kita harus melintasi semua simpul sebelumnya.
* **Reverse Traversing**: Dalam single linked list, reverse traversing tidak dimungkinkan, tetapi dalam kasus double-linked list, ini dapat dimungkinkan karena berisi pointer ke node yang terhubung sebelumnya dengan setiap node. Untuk melakukannya, diperlukan memori tambahan untuk pointer sebelumnya sehingga ada pemborosan memori.
* **Akses Acak**: Akses acak tidak bisa dilakukan dalam linked list karena alokasi memorinya yang dinamis.

**Perbandingan double linked list dan circular linked list dengan single linked list.**

Double linked lists dan circular linked lists adalah jenis linked lists yang lebih kompleks dari single linked list. Double linked lists memiliki pointer tambahan yang disebut "previous pointer", yang menunjuk ke node sebelumnya. Hal ini memungkinkan untuk melintas dalam arah yang berlawanan dari node ke node. Double linked lists juga dapat digunakan untuk mengimplementasikan data structures seperti stack dan queue, yang memerlukan akses ke node sebelumnya dan node berikutnya.

Circular linked lists adalah jenis linked list yang memiliki pointer ke node kepala dan node ekor, sehingga kita harus berhati-hati saat melintas. Circular linked lists dapat digunakan untuk mengimplementasikan data structures seperti queue dan stack, yang memerlukan akses ke node terakhir dan node pertama.

Perbandingan antara double linked list dan circular linked list dengan single linked list adalah:

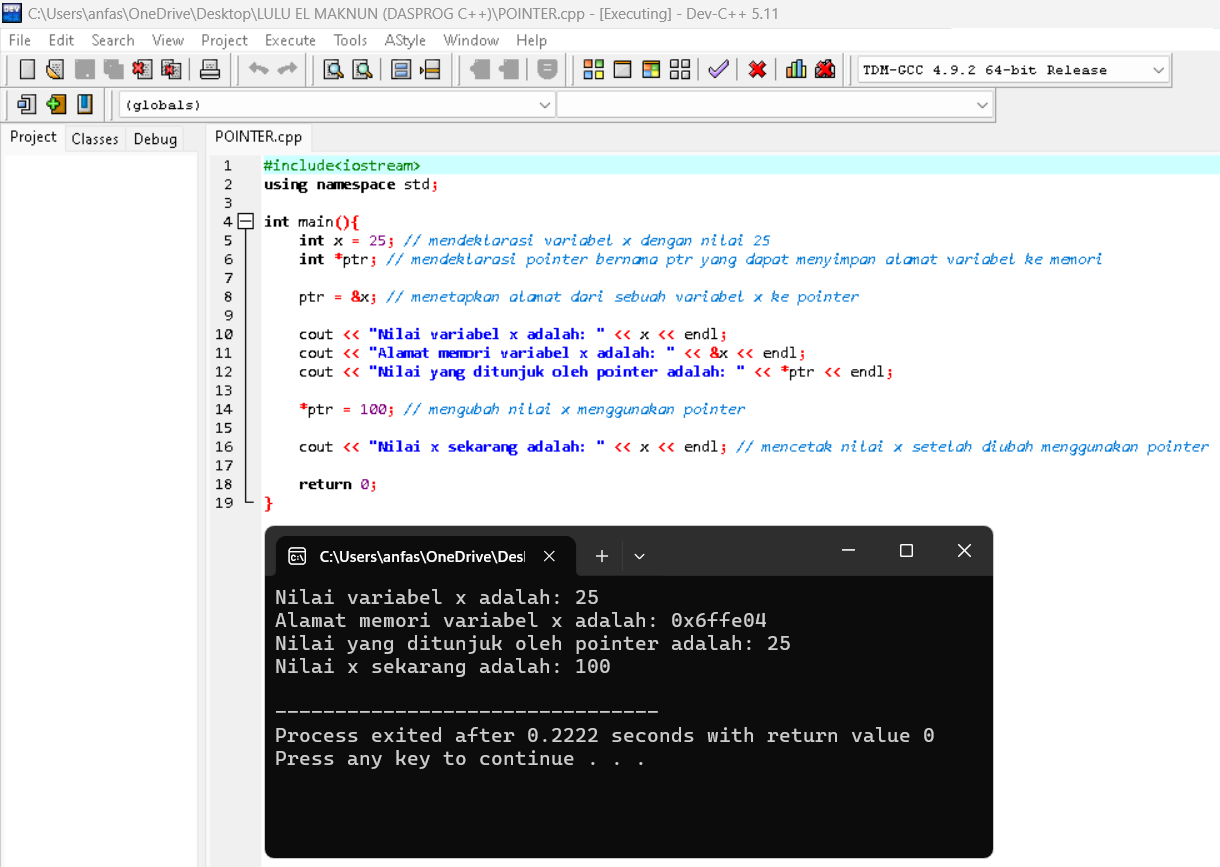
* Double linked list dapat melintas dalam arah yang berlawanan, yang memungkinkan untuk melintas ke node sebelumnya dan node berikutnya.
* Circular linked list memiliki pointer ke node kepala dan node ekor, yang memungkinkan untuk melintas dalam sebuah loop.
* Single linked list hanya dapat melintas dalam arah yang satu, yaitu dari node ke node ekor.

Double linked list dan circular linked list lebih kompleks dari single linked list, tetapi mereka memungkinkan untuk melintas dalam arah yang berlawanan dan mengimplementasikan data structures seperti stack dan queue.

**1. Pointer**

Pointer adalah sebuah penunjuk yang akan menunjukkan alamat sebuah variabel di memori.

Pada laporan ini saya akan menjelaskan codingan saya mengenai pointer.



Jadi dalam program ini library yang digunakan yaitu iostream yang berfungsi untuk menampilkan input dan outpun pada layar. Using namespace std adalah sebuah perintah yang digunakan untuk memudahkan penulisan tanpa harus menulis std secara keseluruhan. Int main adalah fungsi utama pemograman karena merupakan titik awal dan akhir program yang akan dijalankan.

**Deklarasi Variabel**

Int x = 25

Pada baris ke lima saya menggunakan tipe data integer dengan nama variabel nya yaitu x, dan x diberi nilai 25.

Kemudian, int \*ptr yaitu untuk mendeklarasikan pointer yang diberi nama ptr, terdapat operator reference(\*) yang digunakan untuk menyimpan alamat dari sebuah variabel ke memori. Operator tersebut diletakan di depan variabel.

**Menetapkan alamat ke pointer**

ptr = &x

Pada baris depalan saya menetapkan alamat variabel x ke pointer ptr menggunakan operator dereference(&) untuk mendapatkan alamat memori dari sebuah variabel.

**Mencetak nilai melalui pointer**

Pada baris sepuluh sampai dua belas ‘cout’ di gunakan untuk mencetak ke layar.

cout << "Nilai variabel x adalah: " << x << endl; (mencetak variabel)

pada baris ini menunjukan bahwa nilai variabel x saat ini adalah 25.

cout << "Alamat memori variabel x adalah: " << &x << endl;

**Alamat memori variabel x adalah: 0x6ffe04**. Baris ini menunjukkan alamat memori darivariabel x disimpan.

cout << "Nilai yang ditunjuk oleh pointer adalah: " << \*ptr << endl;

Baris ini menunjukkan bahwa nilai yang ditunjuk oleh pointer ptr adalah 25. Karena pointer ptr diinisialisasi dengan alamat variabel x, nilai yang ditunjuk oleh pointer sama dengan nilai variabel x.

\*ptr = 100; (mengubah nilai melalui ptr)

Pada baris empat belas, mengubah nilai variabel x menjadi 100 melalui pointer ptr. Dengan menggunakan operator \* pada ptr, kita dapat mengakses dan mengubah nilai variabel yang ditunjuk oleh pointer tersebut. Jadi, perintah tersebut akan mengubah nilai variabel x menjadi 100.

cout << "Nilai x sekarang adalah: (mencetak nilai setelah diubah)

Baris ini menunjukkan bahwa nilai variabel x telah diubah menjadi 100. Hal ini dilakukan dengan menggunakan pointer ptr untuk mengubah nilai yang disimpan di alamat memori yang ditunjuknya.

Return 0;

Mengembalikan nilai 0 sebagai penanda bahwa program berakhir.

**A screenshot of a computer program

Description automatically generated2. Linked List**

A screenshot of a computer

Description automatically generated**A screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generated**

Pada baris pertama saya menggunakan library iosteam yang berfungsi untuk menmapilkan input dan output pada layar, pada baris kedua saya menggunakan using namespace std yang berfungsi untuk mempermudah penulisan agar tidak perlu menulis std:: secara keseluruhan.

Kemudian pada baris lima, merupakan definisi struktur data node yang digunakan untuk meyimpan elemen dalam linked list. Setiap node memiliki dua anggota yaitu ada data untuk menyimpan nilai dari node dan next merupakan pointer ke node berikutnya dalam list.

Pada baris sebelas, untuk menambahkan node baru di awal linked list yaitu dengan menggunakan addFirst. Pertama, buat node baru dengan operator new dan kemudian menyimpan data di dalamnya. Selanjutnya mengatur pointer next dari node baru untuk menunjuk ke node yang saat ini menjadi head linked list. Terakhir, dengan mengatur pointer head untuk menunjuk ke node baru, sehingga node baru menjadi head linked list.

Di baris sembilan belas, fungsi addLast untuk menambahkan node baru di akhir linked list. Kita membuat node baru dengan menggunakan operator new dan menyimpan data di dalamnya. Selanjutnya, mengatur pointer next dari node baru menuju ke NULL. Selanjutnya, kita periksa apakah linked list kosong atau tidak dengan memeriksa apakah head menunjuk ke NULL. Jika linked list kosong, maka head akan langsung menunjuk ke node baru yang telah dibuat. Jika linked list tidak kosong, kita membuat pointer last yang awalnya menunjuk ke node pertama. Kemudian, kita traverse linked list hingga menemukan node terakhir (node yang next-nya NULL). Setelah itu, kita menghubungkan node terakhir dengan node baru sehingga node baru menjadi node terakhir dalam linked list.

Di baris berikutnya yaitu mengimplementasi fungsi deleteFirst dan deleteLast.

Fungsi deleteFirst:

Pertama, kita periksa apakah linked list kosong atau tidak dengan memeriksa apakah head menunjuk ke NULL.

Jika linked list kosong, maka kita mencetak pesan "List kosong" dan langsung keluar dari fungsi. Jika linked list tidak kosong, kita membuat pointer temp yang menunjuk ke node pertama. Selanjutnya, kita mengatur head untuk menunjuk ke node kedua dalam linked list. Terakhir, kita hapus node yang semula menjadi node pertama menggunakan operator delete.

Fungsi deleteLast:

Pertama, kita periksa apakah linked list kosong atau tidak dengan memeriksa apakah head menunjuk ke NULL.

Jika linked list kosong, maka kita mencetak pesan "List kosong" dan langsung keluar dari fungsi.

Jika linked list tidak kosong, kita periksa apakah linked list hanya memiliki satu node. Jika ya, kita hapus node tersebut dan mengatur head untuk menunjuk ke NULL. Jika linked list memiliki lebih dari satu node, kita traverse linked list hingga menemukan node kedua terakhir. Kemudian, kita hapus node terakhir dan mengatur pointer next dari node kedua terakhir untuk menunjuk ke NULL, sehingga node terakhir dihapus dari linked list.

Fungsi untuk mencetak printList

printList digunakan untuk mencetak isi dari linked list ke layar. Pertama, fungsi ini membuat pointer temp yang menunjuk ke node pertama dari linked list. Selanjutnya, fungsi melakukan iterasi melalui linked list dengan menggunakan loop while yang berjalan selama temp tidak NULL. Pada setiap iterasi, fungsi mencetak data dari node yang ditunjuk oleh temp dan menggerakkan pointer temp ke node selanjutnya dalam linked list. Setelah selesai iterasi, fungsi mencetak endl untuk mengakhiri baris pencetakan.

Dengan demikian, setelah fungsi printList dijalankan, isi dari linked list akan dicetak ke layar dengan dipisahkan oleh spasi.

Fungsi untuk mencari node

Fungsi search menerima dua parameter: head yang merupakan pointer ke node pertama dari linked list, dan data yang merupakan nilai yang ingin dicari dalam linked list.

Fungsi ini menggunakan sebuah pointer current yang awalnya menunjuk ke node pertama dari linked list. Selama current tidak NULL, fungsi akan memeriksa apakah nilai data dari node yang ditunjuk oleh current sama dengan nilai data yang dicari. Jika nilai data cocok, fungsi akan mengembalikan pointer ke node tersebut. Jika tidak ditemukan node dengan nilai data yang sesuai setelah iterasi selesai, fungsi akan mengembalikan NULL sebagai indikasi bahwa data tidak ditemukan dalam linked list.

Fungsi memperbarui data node, digunakan untuk mengganti nilai data pada node tertentu dalam linked list. Fungsi ini mencari node dengan nilai data tertentu menggunakan fungsi search, kemudian mengubah nilai data pada node tersebut jika ditemukan, atau mencetak pesan "Data tidak ditemukan" jika tidak ditemukan.

Fungsi untuk menggabungkan dua linked list, jika linked list pertama kosong (tidak memiliki node), maka linked list pertama akan diisi dengan seluruh linked list kedua. Jika linked list pertama tidak kosong, linked list kedua akan disambungkan ke ujung linked list pertama, sehingga semua node dari linked list kedua akan menjadi bagian dari linked list pertama.

Fungsi utama (int main)

Fungsi utama di mana program dimulai. Di dalam fungsi ini, beberapa operasi dasar seperti menambahkan node, mencetak list, menggabungkan list, dan memperbarui data node dilakukan.

1. Membuat linked list head1 kosong dan menambahkan node dengan nilai 1 di bagian depan linked list tersebut.

2. Menambahkan node dengan nilai 2 di bagian belakang linked list head1.

3. Menambahkan node dengan nilai 3 di bagian belakang linked list head1.

4. Mencetak isi linked list head1.

5. Membuat linked list head2 kosong dan menambahkan node dengan nilai 4 di bagian depan linked list tersebut.

6. Menambahkan node dengan nilai 5 di bagian belakang linked list head2.

7. Menambahkan node dengan nilai 6 di bagian belakang linked list head2.

8. Mencetak isi linked list head2.

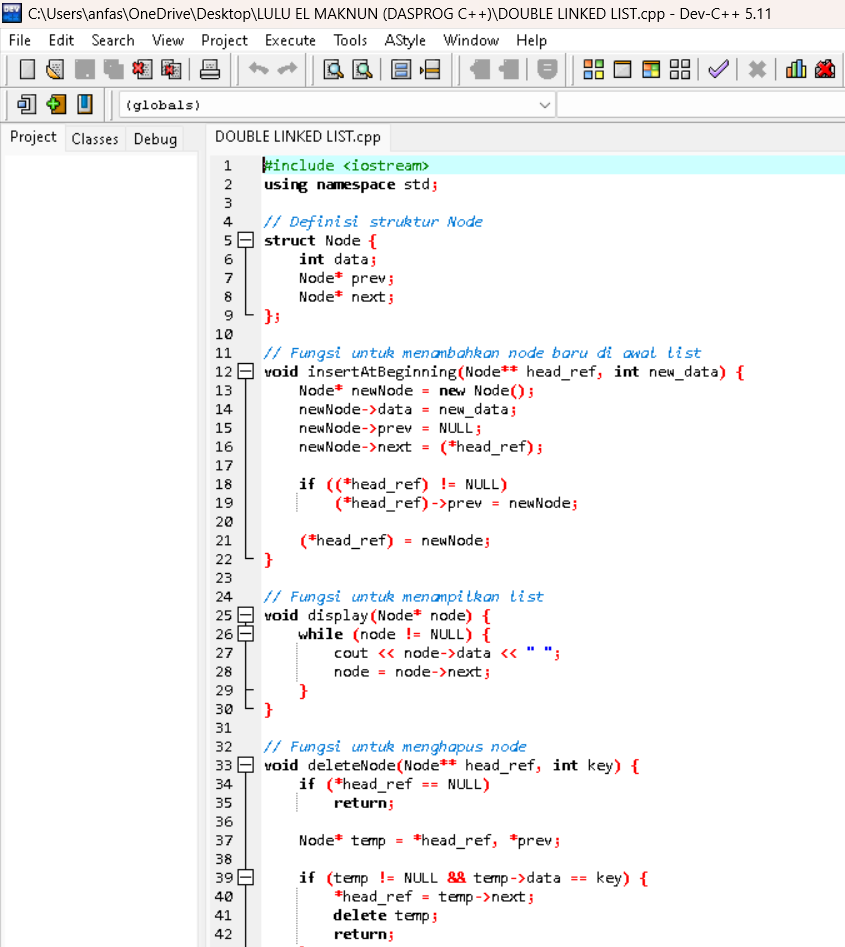
9. Menggabungkan linked list head2 ke linked list head1.

10. Mencetak isi linked list head1 setelah penggabungan.

11. Mengubah nilai node dengan data 2 dalam linked list head1 menjadi 7.

12. Mencetak isi linked list head1 setelah perubahan nilai.

A screenshot of a computer program

Description automatically generated**3. Double Linked List**

Pada baris lima, merupakan definisi dari sebuah node pada double linked list. Node ini memiliki tiga bagian yaitu, int data berguna untuk menyimpan data kemudian, node prev untuk pointer ke node sebelumnya dan terakhir node next untuk pointer node ke selanjutny.

Pada baris dua belas, insertAtBeginning berfungsi untuk menambhakan node baru di awal sebuah double linked list. Kemudian mengalokasikan memori untuk node baru dengan menggunakan operator new dan menyimpan alamat node baru salam variabel newNode. Lalu menetapkan data ke node baru, data yang diberikan sebagai argumen fungsi (new\_data)ditetapkan ke field data dari node baru. Kemudian mengatur pointer prev dari node baru ke NULL, node ini akan menjadi node pertama dalam list.

Pada baris dua puluh lima, display berfungsi untuk menampilkan semua elemen di double linked list. Baris dua puluh enam, inisialisasi iterasi fungsinya untuk menerima sebuah pointer ke node pertama. Kemudian menampilkan data node, fungsinya mencetak data yang di simpan dalam node saat ini. Data ini di tampilkan dengan menggunakan cout. Setelah menampilkan data, fungsi ini memperbarui pointer node untuk menunjuk ke node berikutnya dalam list. Ini dilakukan dengan mengatur node ke node->next, yang merupakan pointer ke node berikutnya.

Fungsi deleteNode untuk menghapus node dengan data tertentu.

Pemeriksaan List Kosong: Fungsi ini pertama-tama memeriksa apakah list kosong (yaitu, \*head\_ref adalah NULL). Jika ya, fungsi ini langsung kembali tanpa melakukan apa-apa. Kemudian menginisialisasi dua pointer, temp dan prev. temp diatur ke node pertama dalam list, dan prev diatur ke NULL.

Pemeriksaan Node Pertama: Fungsi ini memeriksa apakah node pertama dalam list memiliki data yang sama dengan key yang dicari. Jika ya, fungsi ini menghapus node pertama dengan mengatur \*head\_ref ke node berikutnya dan menghapus node pertama. Jika node pertama tidak memiliki data yang sama dengan key, fungsi ini melanjutkan pencarian node dengan data yang sama. Ini dilakukan dengan iterasi melalui list sampai menemukan node dengan data yang sama atau mencapai akhir list. Jika pencarian selesai dan temp masih NULL, berarti node dengan data yang dicari tidak ditemukan dalam list. Fungsi ini kembali tanpa melakukan apa-apa. Jika node ditemukan, fungsi ini menghapus node tersebut dengan memperbarui pointer next dari node sebelumnya dan prev dari node berikutnya. Kemudian, node tersebut dihapus.

Output dari program ini akan menampilkan elemen-elemen dalam list doubly linked yang telah diinisialisasi dan dimasukkan elemen ke dalamnya. Yaitu 8 4 1 7 6.

A screenshot of a computer program

Description automatically generated**4. Circular Linked List**

A screenshot of a computer

Description automatically generated

Fungsi addNode digunakan untuk menambahkan node baru di awal linked list. Fungsi ini menerima dua parameter, yaitu pointer ke pointer ke node awal linked list (head\_ref) dan data baru yang akan dimasukkan ke node (data). Jika linked list kosong, maka node baru yang ditambahkan akan menjadi node pertama dan node terakhir pada linked list. Untuk melakukan ini, pointer new\_node->next diperbarui menjadi node baru (new\_node). Setelah node baru ditambahkan, pointer ke node awal linked list (head\_ref) diperbarui menjadi node baru (new\_node). Hal ini akan membuat node baru menjadi node awal linked list.

Kemudian Fungsi printList digunakan untuk menampilkan seluruh data yang ada dalam linked list. Fungsi ini menerima satu parameter, yaitu pointer ke node awal linked list (head). Seluruh data yang ada dalam linked list kemudian ditampilkan menggunakan fungsi cout. Hasil yang diharapkan adalah 10 8 7 5 2.