

LAPORAN PRAKTIKUM
PERTEMUAN 4



Nama :

Razhendriya Vania Ramadhan Suganjarsarwat (2311104048)

Dosen :

Wahyu Andi Saputra

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2024

I. TUJUAN

- Memahami penggunaan **linked list** dengan pointer operator dalam program.
- Memahami operasi-operasi dasar dalam **linked list** seperti penciptaan, penyisipan, penghapusan, dan penelusuran elemen.
- Membuat program menggunakan **linked list** sesuai dengan prototype yang ada

II. LANDASAN TEORI

Linked List adalah struktur data yang fleksibel, terdiri dari serangkaian elemen yang saling terhubung menggunakan pointer. Elemen ini dinamakan Node, yang terdiri dari dua bagian: data dan pointer menuju elemen berikutnya.

Keunggulan linked list dibandingkan array adalah sifatnya yang dinamis, sehingga elemen dapat bertambah atau berkurang sesuai kebutuhan. Ada berbagai jenis linked list, seperti Single Linked List, Double Linked List, dan Circular Linked List. Modul ini fokus pada Single Linked List, di mana setiap node hanya memiliki satu pointer yang mengarah ke elemen berikutnya.

Operasi dasar pada linked list meliputi:

- **Penciptaan dan inisialisasi list.**
- **Penyisipan elemen di awal, akhir, atau setelah node tertentu.**
- **Penghapusan elemen dari awal, akhir, atau elemen tertentu.**
- **Penelusuran dan menampilkan elemen-elemen pada list**

III. GUIDED

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

// Struktur data untuk simpul
struct Node {
    int data;
    Node* next;
};

// Variabel global
Node* head = NULL;
Node* tail = NULL;
int jumlah = 0;

// Fungsi untuk menambah node di awal
void tambahAwal(int data) {
    Node* newNode = new Node;
    newNode->data = data;
    newNode->next = head;
    head = newNode;
    jumlah++;
}

// Fungsi untuk menambah node di akhir
void tambahAkhir(int data) {
    Node* newNode = new Node;
    newNode->data = data;
    newNode->next = NULL;
    if (tail == NULL) {
        head = tail = newNode;
    } else {
        tail->next = newNode;
        tail = newNode;
    }
    jumlah++;
}

// Fungsi untuk menghitung jumlah node
int hitungJumlah() {
    return jumlah;
}

// Fungsi untuk menghapus node dari depan
void hapusDepan() {
    if (head == NULL) {
        cout << "List kosong" << endl;
        return;
    }
    Node* temp = head;
    head = head->next;
    delete temp;
    jumlah--;
}

// Fungsi untuk menghapus node dari belakang
void hapusBelakang() {
    if (head == NULL) {
        cout << "List kosong" << endl;
        return;
    }
    if (head == tail) {
        head = tail = NULL;
        delete head;
        jumlah--;
    } else {
        Node* temp = head;
        while (temp->next != tail) {
            temp = temp->next;
        }
        temp->next = NULL;
        delete tail;
        tail = temp;
        jumlah--;
    }
}

// Fungsi untuk membersihkan seluruh list
void bersihkanList() {
    while (head != NULL) {
        hapusDepan();
    }
    cout << "List telah dibersihkan" << endl;
}

// Fungsi untuk menampilkan list
void tampilkanList() {
    Node* temp = head;
    while (temp != NULL) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}

// Fungsi utama
int main() {
    // Menambah node di awal
    tambahAwal(10);
    tambahAwal(20);

    // Menambah node di akhir
    tambahAkhir(30);
    tambahAkhir(40);

    // Menampilkan list
    tampilkanList();

    // Menghitung jumlah node
    cout << "Jumlah node: " << hitungJumlah() << endl;

    // Menghapus node dari depan
    hapusDepan();
    tampilkanList();

    // Menghapus node dari belakang
    hapusBelakang();
    tampilkanList();

    // Bersihkan seluruh list
    bersihkanList();
    tampilkanList();

    return 0;
}
```

Program ini mendemonstrasikan cara membuat linked list, menambah node di awal dan akhir, menghitung jumlah node, menghapus node dari depan dan belakang, serta membersihkan seluruh list.

[illegible]

Program ini menunjukkan cara kerja dasar linked list, termasuk menambahkan elemen di awal/akhir, menampilkan elemen, menghitung jumlah elemen, dan menghapus semua elemen dari list.

IV. UNGUIDED

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

// Definisi Node untuk linked list
struct Node {
    int data;
    Node* next;
};

// Fungsi untuk mengalokasikan node baru
Node* createNode(int value) {
    Node* newNode = new Node;
    newNode->data = value;
    newNode->next = nullptr;
    return newNode;
}

// Menambahkan node di depan linked list
void insertDepan(Node* &head, int value) {
    Node* newNode = createNode(value);
    newNode->next = head;
    head = newNode;
}

// Menambahkan node di belakang linked list
void insertBelakang(Node* &head, int value) {
    Node* newNode = createNode(value);
    if (head == nullptr) {
        head = newNode;
    } else {
        Node* temp = head;
        while (temp->next != nullptr) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
}

// Fungsi untuk mencetak isi linked list
void printList(Node* head) {
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data;
        if (temp->next != nullptr) cout << " -> ";
        temp = temp->next;
    }
    cout << endl;
}

int main() {
    Node* head = nullptr;

    // Tambah node sesuai instruksi
    insertDepan(head, 10);
    insertBelakang(head, 20);
    insertDepan(head, 5);

    // Cetak isi linked list
    cout << "Isi linked list: ";
    printList(head);

    return 0;
}
```

createNode: Mengalokasikan memori untuk node baru dan mengisinya dengan data yang diberikan.

insertDepan: Menyisipkan node di awal linked list dengan menghubungkan node baru ke head dan memperbarui head.

insertBelakang: Menambahkan node di akhir linked list dengan cara menelusuri node terakhir.

printList: Menampilkan elemen-elemen yang ada di dalam linked list.

```

001  )
002  )
003  )
004  )
005  )
006  )
007  )
008  )
009  )
010  )
011  )
012  )
013  )
014  )
015  )
016  )
017  )
018  )
019  )
020  )
021  )
022  )
023  )
024  )
025  )
026  )
027  )
028  )
029  )
030  )
031  )
032  )
033  )
034  )
035  )
036  )
037  )
038  )
039  )
040  )
041  )
042  )
043  )
044  )
045  )
046  )
047  )
048  )
049  )
050  )
051  )
052  )
053  )
054  )
055  )
056  )
057  )
058  )
059  )
060  )
061  )
062  )
063  )
064  )
065  )
066  )
067  )
068  )
069  )
070  )
071  )
072  )
073  )
074  )
075  )
076  )
077  )
078  )
079  )
080  )
081  )
082  )
083  )
084  )
085  )
086  )
087  )
088  )
089  )
090  )
091  )
092  )
093  )
094  )
095  )
096  )
097  )
098  )
099  )
100  )
101  )
102  )
103  )
104  )
105  )
106  )
107  )
108  )
109  )
110  )
111  )
112  )
113  )
114  )
115  )
116  )
117  )
118  )
119  )
120  )
121  )
122  )
123  )
124  )
125  )
126  )
127  )
128  )
129  )
130  )
131  )
132  )
133  )
134  )
135  )
136  )
137  )
138  )
139  )
140  )
141  )
142  )
143  )
144  )
145  )
146  )
147  )
148  )
149  )
150  )
151  )
152  )
153  )
154  )
155  )
156  )
157  )
158  )
159  )
160  )
161  )
162  )
163  )
164  )
165  )
166  )
167  )
168  )
169  )
170  )
171  )
172  )
173  )
174  )
175  )
176  )
177  )
178  )
179  )
180  )
181  )
182  )
183  )
184  )
185  )
186  )
187  )
188  )
189  )
190  )
191  )
192  )
193  )
194  )
195  )
196  )
197  )
198  )
199  )
200  )
201  )
202  )
203  )
204  )
205  )
206  )
207  )
208  )
209  )
210  )
211  )
212  )
213  )
214  )
215  )
216  )
217  )
218  )
219  )
220  )
221  )
222  )
223  )
224  )
225  )
226  )
227  )
228  )
229  )
230  )
231  )
232  )
233  )
234  )
235  )
236  )
237  )
238  )
239  )
240  )
241  )
242  )
243  )
244  )
245  )
246  )
247  )
248  )
249  )
250  )
251  )
252  )
253  )
254  )
255  )
256  )
257  )
258  )
259  )
260  )
261  )
262  )
263  )
264  )
265  )
266  )
267  )
268  )
269  )
270  )
271  )
272  )
273  )
274  )
275  )
276  )
277  )
278  )
279  )
280  )
281  )
282  )
283  )
284  )
285  )
286  )
287  )
288  )
289  )
290  )
291  )
292  )
293  )
294  )
295  )
296  )
297  )
298  )
299  )
300  )
301  )
302  )
303  )
304  )
305  )
306  )
307  )
308  )
309  )
310  )
311  )
312  )
313  )
314  )
315  )
316  )
317  )
318  )
319  )
320  )
321  )
322  )
323  )
324  )
325  )
326  )
327  )
328  )
329  )
330  )
331  )
332  )
333  )
334  )
335  )
336  )
337  )
338  )
339  )
340  )
341  )
342  )
343  )
344  )
345  )
346  )
347  )
348  )
349  )
350  )
351  )
352  )
353  )
354  )
355  )
356  )
357  )
358  )
359  )
360  )
361  )
362  )
363  )
364  )
365  )
366  )
367  )
368  )
369  )
370  )
371  )
372  )
373  )
374  )
375  )
376  )
377  )
378  )
379  )
380  )
381  )
382  )
383  )
384  )
385  )
386  )
387  )
388  )
389  )
390  )
391  )
392  )
393  )
394  )
395  )
396  )
397  )
398  )
399  )
400  )
401  )
402  )
403  )
404  )
405  )
406  )
407  )
408  )
409  )
410  )
411  )
412  )
413  )
414  )
415  )
416  )
417  )
418  )
419  )
420  )
421  )
422  )
423  )
424  )
425  )
426  )
427  )
428  )
429  )
430  )
431  )
432  )
433  )
434  )
435  )
436  )
437  )
438  )
439  )
440  )
441  )
442  )
443  )
444  )
445  )
446  )
447  )
448  )
449  )
450  )
451  )
452  )
453  )
454  )
455  )
456  )
457  )
458  )
459  )
460  )
461  )
462  )
463  )
464  )
465  )
466  )
467  )
468  )
469  )
470  )
471  )
472  )
473  )
474  )
475  )
476  )
477  )
478  )
479  )
480  )
481  )
482  )
483  )
484  )
485  )
486  )
487  )
488  )
489  )
490  )
491  )
492  )
493  )
494  )
495  )
496  )
497  )
498  )
499  )
500  )
501  )
502  )
503  )
504  )
505  )
506  )
507  )
508  )
509  )
510  )
511  )
512  )
513  )
514  )
515  )
516  )
517  )
518  )
519  )
520  )
521  )
522  )
523  )
524  )
525  )
526  )
527  )
528  )
529  )
530  )
531  )
532  )
533  )
534  )
535  )
536  )
537  )
538  )
539  )
540  )
541  )
542  )
543  )
544  )
545  )
546  )
547  )
548  )
549  )
550  )
551  )
552  )
553  )
554  )
555  )
556  )
557  )
558  )
559  )
560  )
561  )
562  )
563  )
564  )
565  )
566  )
567  )
568  )
569  )
570  )
571  )
572  )
573  )
574  )
575  )
576  )
577  )
578  )
579  )
580  )
581  )
582  )
583  )
584  )
585  )
586  )
587  )
588  )
589  )
590  )
591  )
592  )
593  )
594  )
595  )
596  )
597  )
598  )
599  )
600  )
601  )
602  )
603  )
604  )
605  )
606  )
607  )
608  )
609  )
610  )
611  )
612  )
613  )
614  )
615  )
616  )
617  )
618  )
619  )
620  )
621  )
622  )
623  )
624  )
625  )
626  )
627  )
628  )
629  )
630  )
631  )
632  )
633  )
634  )
635  )
636  )
637  )
638  )
639  )
640  )
641  )
642  )
643  )
644  )
645  )
646  )
647  )
648  )
649  )
650  )
651  )
652  )
653  )
654  )
655  )
656  )
657  )
658  )
659  )
660  )
661  )
662  )
663  )
664  )
665  )
666  )
667  )
668  )
669  )
670  )
671  )
672  )
673  )
674  )
675  )
676  )
677  )
678  )
679  )
680  )
681  )
682  )
6
```

deleteNode: Fungsi ini menghapus node dengan nilai tertentu.

Jika node yang akan dihapus adalah node pertama, ia memperbarui head. Jika node ditemukan di bagian lain list, ia memutus hubungan node dan menghapus node tersebut dari memori.

```

1 #include <iostream>
2 using namespace std;
3
4 // Definisi Node untuk Linked List
5 struct Node {
6     int data;
7     Node* next;
8 };
9
10 // Fungsi untuk mengalokasikan node baru
11 Node* createNode(int value) {
12     Node* newNode = new Node;
13     newNode->data = value;
14     newNode->next = nullptr;
15     return newNode;
16 }
17
18 // Menambahkan node di depan linked list
19 void insertDepan(Node* &head, int value) {
20     Node* newNode = createNode(value);
21     newNode->next = head;
22     head = newNode;
23 }
24
25 // Menambahkan node di belakang linked list
26 void insertBelakang(Node* &head, int value) {
27     Node* newNode = createNode(value);
28     if (head == nullptr) {
29         head = newNode;
30     } else {
31         Node* temp = head;
32         while (temp->next != nullptr) {
33             temp = temp->next;
34         }
35         temp->next = newNode;
36     }
37 }
38
39 // Mencari node dengan nilai tertentu
40 bool cariNode(Node* head, int value) {
41     Node* temp = head;
42     while (temp != nullptr) {
43         if (temp->data == value) return true;
44         temp = temp->next;
45     }
46     return false;
47 }
48
49 // Menghitung panjang linked list
50 int panjangList(Node* head) {
51     int count = 0;
52     Node* temp = head;
53     while (temp != nullptr) {
54         count++;
55         temp = temp->next;
56     }
57     return count;
58 }
59
60 // Fungsi untuk mencetak isi linked list
61 void printList(Node* head) {
62     Node* temp = head;
63     while (temp != nullptr) {
64         cout << temp->data;
65         if (temp->next != nullptr) cout << " -> ";
66         temp = temp->next;
67     }
68     cout << endl;
69 }
70
71 int main() {
72     Node* head = nullptr;
73
74     // Tambah node sesuai instruksi
75     insertDepan(head, 10);
76     insertBelakang(head, 20);
77     insertDepan(head, 5);
78
79     // Mencari node
80     int cari = 20;
81     if (cariNode(head, cari)) {
82         cout << "Node dengan nilai " << cari << " ditemukan." << endl;
83     } else {
84         cout << "Node dengan nilai " << cari << " tidak ditemukan." << endl;
85     }
86
87     // Cetak panjang linked list
88     cout << "Panjang linked list: " << panjangList(head) << endl;
89
90     return 0;
91 }

```

cariNode: Fungsi untuk mencari apakah suatu node dengan nilai tertentu ada dalam linked list.

panjangList: Fungsi untuk menghitung jumlah node dalam linked list dengan cara menelusuri seluruh node.

V. KESIMPULAN

Single Linked List memungkinkan penyimpanan dan pengelolaan data dengan lebih efisien dan fleksibel. Dengan memahami operasi-operasi dasar seperti penyisipan, penghapusan, dan penelusuran, kita dapat memanfaatkan linked list untuk menyimpan data yang terus berkembang atau berubah ukurannya. Primitif-primitif yang diimplementasikan seperti insertFirst, insertLast, deleteFirst, deleteLast, dan printList adalah bagian penting dalam manipulasi linked list, dan penggunaannya sangat sesuai dengan kebutuhan program berbasis pointer