

**LAPORAN PRAKTIKUM  
ALGORITMA & STRUKTUR DATA  
MODUL 4**



**DOUBLE LINK LIST**

**Oleh:**

**Noor Khalisa      NIM. 2410817220012**

**PROGRAM STUDI TEKNOLOGI INFORMASI  
FAKULTAS TEKNIK  
UNIVERSITAS LAMBUNG MANGKURAT  
MEI 2025**

**LEMBAR PENGESAHAN**  
**LAPORAN PRAKTIKUM ALGORITMA & STRUKTUR DATA**  
**MODUL 4**

Laporan Praktikum Algoritma & Struktur Data Modul 4: Double Link List ini disusun sebagai syarat lulus mata kuliah Praktikum Algoritma & Struktur Data. Laporan Praktikum ini dikerjakan oleh:

Nama Praktikan : Noor Khalisa  
NIM : 24101817220012

Menyetujui,  
Asisten Praktikum

Mengetahui,  
Dosen Penanggung Jawab Praktikum

Muhammad Fauzan Ahsani  
NIM. 2310817310009

Muti'a Maulida, S.Kom., M.TI.  
NIP. 198810272019032013

## DAFTAR ISI

LEMBAR PENGESAHAN .....	2
DAFTAR ISI.....	3
DAFTAR GAMBAR .....	4
DAFTAR TABEL .....	5
SOAL 1 .....	6
A. Source Code .....	12
B. Output.....	21
C. Pembahasan.....	24
SOAL 2 .....	27
A. Pembahasan.....	27
SOAL 3 .....	28
A. Pembahasan.....	28
SOAL 4 .....	29
A. Pembahasan.....	29

## **DAFTAR GAMBAR**

Gambar 1 Tampilan Awal Program DLLNC.....	21
Gambar 2 Tampilan Awal Program DLLNC Head .....	22
Gambar 3 Tambah Depan .....	22
Gambar 4 Tambah Belakang.....	22
Gambar 5 Tampilkan Data .....	22
Gambar 6 Hapus Depan .....	22
Gambar 7 Hapus Belakang.....	22
Gambar 8 Reset .....	22
Gambar 9 Tampilkan Data Setelah Reset .....	22
Gambar 10 Hapus Depan Jika Data Kosong.....	23
Gambar 11 Hapus Belakang Jika Data Kosong .....	23
Gambar 12 Tampilan Awal Program DLLNC Head dan Tail .....	23
Gambar 13 Tambah Depan .....	23
Gambar 14 Tambah Belakang.....	23
Gambar 15 Tambah Belakang Lagi .....	23
Gambar 16 Hapus Depan .....	23
Gambar 17 Hapus Belakang.....	23
Gambar 18 Tampilkan Data Setelah Hapus Depan dan Belakang.....	24
Gambar 19 Reset .....	24
Gambar 20 Tampilkan Data Setelah Reset .....	24
Gambar 21 Quit.....	24

## DAFTAR TABEL

Tabel 1 Source Code Soal 1 .....	12
Tabel 2 Output Soal 4 .....	29

## SOAL 1

Lengkapi coding pada function tambahDepanH() agar bisa berjalan dengan lancar. running, simpan program, dan screenshoot hasil running!

```
1  #include <conio.h>
2  #include <iostream>
3  #include <stdlib.h>
4
5  using namespace std;
6
7  typedef struct TNode {
8      string data;
9      TNode *next;
10     TNode *prev;
11 };
12
13 TNode *head, *tail;
14
15 int pil, menu;
16 char pilihan[1];
17 string dataBaru;
18
19 void inith();
20 void inithT();
21 int isEmptyH();
22 int isEmptyHT();
23
24 void tambahDepanH();
25 void tambahDepanHT();
26 void tambahBelakangH();
27 void tambahBelakangHT();
28 void hapusDepanH();
29 void hapusDepanHT();
30 void hapusBelakangH();
31 void hapusBelakangHT();
32 void tampilkanH();
33 void tampilkanHT();
34 void clearH();
35 void clearHT();
36
37 int main()
38 {
39     menu:
40     cout<<"Double Linked List Non Circular (DLLNC)"<<endl;
41     cout<<"===== "<<endl;
42     cout<<"Silahkan pilih program DLLNC yang ingin dijalankan!"<<endl;
43     cout<<"1. DLLNC dengan Head"<<endl;
44     cout<<"2. DLLNC dengan Head dan Tail"<<endl;
45     cout<<"3. Quit"<<endl;
46     cout<<"Pilihan : ";
47     cin>>menu;
48     system("cls");
```

```
49     if(menu==1){
50         do {
51             cout<<"Double Linked List Non Circular (DLLNC) (Head)"<<endl;
52             cout<<"===== "<<endl;
53             cout<<"1. Tambah Depan"<<endl;
54             cout<<"2. Tambah Belakang"<<endl;
55             cout<<"3. Tampilkan Data"<<endl;
56             cout<<"4. Hapus Depan"<<endl;
57             cout<<"5. Hapus Belakang"<<endl;
58             cout<<"6. Reset"<<endl;
59             cout<<"7. Kembali ke Menu"<<endl;
60             cout<<"Pilihan : ";
61             cin>>pilihan;
62             pil=atoi(pilihan);
```

```

63
64     switch(pil) {
65     case 1:
66         tambahDepanH();
67         break;
68     case 2:
69         tambahBelakangH();
70         break;
71     case 3:
72         tampilkanH();
73         break;
74     case 4:
75         hapusDepanH();
76         break;
77     case 5:
78         hapusBelakangH();
79         break;
80     case 6:
81         clearH();
82         break;
83     default:
84         system("cls");
85         goto menu;
86     }
87
88     cout<<"\npress any key to continue"<<endl;
89     getch();
90     system("cls");
91
92     } while (pil<7);
93 } else if(menu==2){
94     do {

```

```

95         cout<<"Double Linked List Non Circular (DLLNC) (Head dan Tail)"<<endl;
96         cout<<"===== "<<endl;
97         cout<<"1. Tambah Depan"<<endl;
98         cout<<"2. Tambah Belakang"<<endl;
99         cout<<"3. Tampilkan Data"<<endl;
100        cout<<"4. Hapus Depan"<<endl;
101        cout<<"5. Hapus Belakang"<<endl;
102        cout<<"6. Reset"<<endl;
103        cout<<"7. Kembali ke Menu"<<endl;
104        cout<<"Pilihan : ";
105        cin>>pilihan;
106        pil=atoi(pilihan);
107
108        switch(pil) {
109        case 1:
110            tambahDepanHT();
111            break;
112        case 2:
113            tambahBelakangHT();
114            break;
115        case 3:
116            tampilkanHT();
117            break;
118        case 4:
119            hapusDepanHT();
120            break;
121        case 5:
122            hapusBelakangHT();
123            break;
124        case 6:
125            clearHT();
126            break;
127        default:
128            system("cls");
129            goto menu;
130        }
131
132        cout<<"\npress any key to continue"<<endl;
133        getch();
134        system("cls");
135
136    } while (pil<7);
137 } else {
138     cout<<"\nTERIMA KASIH"<<endl;
139     cout<<"Program was made by Nama (NIM)."<<endl;
140 }
141 }
142

```

```

143 void initH(){
144     head = NULL;
145 }
146
147 void initHT(){
148     head = NULL;
149     tail = NULL;
150 }
151
152 int isEmptyH(){
153     if(head == NULL) return 1;
154     else return 0;
155 }
156
157 int isEmptyHT(){
158     if(tail == NULL) return 1;
159     else return 0;
160 }
161
162 void tambahDepanH() {
163     cout<<"Masukkan data : ";
164
165
166
167
168
169
170
171
172
173
174
175
176
177     cout << "Data \"<<dataBaru<<\" berhasil dimasukkan di bagian depan.";
178 }
179
180 void tambahDepanHT() {
181     cout<<"Masukkan data : ";
182     cin>>dataBaru;
183     TNode *baru;
184     baru = new TNode;
185     baru->data = dataBaru;
186     baru->next = NULL;
187     baru->prev = NULL;
188     if(isEmptyHT() == 1) {
189         head = baru;
190         tail = baru;

```



```

191     } else {
192         baru->next = head;
193         head->prev = baru;
194         head = baru;
195     }
196     cout << "Data \""<<dataBaru<<"\" berhasil dimasukkan di bagian depan.";
197 }
198
199 void tambahBelakangH() {
200     cout<<"Masukkan data : ";
201     cin>>dataBaru;
202     TNode *baru, *bantu;
203     baru = new TNode;
204     baru->data = dataBaru;
205     baru->next = NULL;
206     baru->prev = NULL;
207     if(isEmptyH() == 1) {
208         head = baru;
209     } else {
210         bantu = head;
211         while(bantu->next != NULL){
212             bantu = bantu->next;
213         }
214         bantu->next = baru;
215         baru->prev = bantu;
216     }
217     cout << "Data \""<<dataBaru<<"\" berhasil dimasukkan di bagian belakang.";
218 }
219

```

```

220 void tambahBelakangHT() {
221     cout<<"Masukkan data : ";
222     cin>>dataBaru;
223     TNode *baru;
224     baru = new TNode;
225     baru->data = dataBaru;
226     baru->next = NULL;
227     baru->prev = NULL;
228     if(isEmptyHT() == 1) {
229         head = baru;
230         tail = baru;
231     } else {
232         tail->next = baru;
233         baru->prev = tail;
234         tail = baru;
235     }
236     cout << "Data \""<<dataBaru<<"\" berhasil dimasukkan di bagian belakang.";
237 }

```

```

238
239 void tampilkanH() {
240     TNode *bantu;
241     bantu = head;
242     if(isEmptyH() == 0) {
243         while(bantu != NULL) {
244             cout<<bantu->data<<' ';
245             bantu = bantu->next;
246         }
247         cout<<endl;
248     } else cout<<"Tidak terdapat data pada Linked List";
249 }
250
251 void tampilkanHT() {
252     TNode *bantu;
253     bantu = head;
254     if(isEmptyHT() == 0) {
255         while(bantu != tail->next) {
256             cout<<bantu->data<<' ';
257             bantu = bantu->next;
258         }
259         cout<<endl;
260     } else cout<<"Tidak terdapat data pada Linked List";
261 }
262
263 void hapusDepanH() {
264     TNode *hapus;
265     string data;
266     if(isEmptyH() == 0) {
267         hapus = head;
268         data = hapus->data;
269         if(head->next != NULL) {
270             head = head->next;
271             head->prev = NULL;
272         } else {
273             initH();
274         }
275         delete hapus;
276         cout<<"Data \""<<data<<"\" yang berada di depan telah berhasil dihapus.";
277     } else cout<<"Tidak terdapat data pada Linked List";
278 }
279
280 void hapusDepanHT() {
281     TNode *hapus;
282     string data;
283     if(isEmptyHT() == 0) {
284         hapus = head;
285         data = hapus->data;

```

```

286         if(head->next != NULL) {
287             head = head->next;
288             head->prev = NULL;
289         } else {
290             initHT();
291         }
292         delete hapus;
293         cout<<"Data \\"<<data<<" yang berada di depan telah berhasil dihapus.";
294     } else cout<<"Tidak terdapat data pada Linked List";
295 }
296
297 void hapusBelakangH() {
298     TNode *hapus;
299     string data;
300     if(isEmptyH() == 0) {
301         hapus = head;
302         while(hapus->next != NULL){
303             hapus = hapus->next;
304         }
305         data = hapus->data;
306         if(head->next != NULL) {
307             hapus->prev->next = NULL;
308         } else {
309             initH();
310         }
311         delete hapus;
312         cout<<"Data \\"<<data<<" yang berada di belakang telah berhasil dihapus.";
313     } else cout<<"Tidak terdapat data pada Linked List";
314 }
315
316 void hapusBelakangHT() {
317     TNode *hapus;
318     string data;
319     if(isEmptyHT() == 0) {
320         hapus = tail;
321         data = hapus->data;
322         if(head->next != NULL) {
323             tail = tail->prev;
324             tail->next = NULL;
325         } else {
326             initHT();
327         }
328         delete hapus;
329         cout<<"Data \\"<<data<<" yang berada di belakang telah berhasil dihapus.";
330     } else cout<<"Tidak terdapat data pada Linked List";
331 }
332
333 void clearH() {

```

```

334     TNode *bantu, *hapus;
335     bantu = head;
336     while(bantu != NULL) {
337         hapus = bantu;
338         bantu = bantu->next;
339         delete hapus;
340     }
341     initH();
342     cout<<"Seluruh data pada Linked List telah dibersihkan.";
343 }
344
345 void clearHT() {
346     TNode *bantu, *hapus;
347     bantu = head;
348     while(bantu != NULL) {
349         hapus = bantu;
350         bantu = bantu->next;
351         delete hapus;
352     }
353     initHT();
354     cout<<"Seluruh data pada Linked List telah dibersihkan.";
355 }

```

## A. Source Code

Tabel 1 Source Code Soal 1

1	// write your code here
2	#include <conio.h>
3	#include <iostream>
4	#include <stdlib.h>
5	
6	using namespace std;
7	
8	typedef struct TNode {
9	string data;
10	TNode *next;
11	TNode *prev;
12	};
13	
14	TNode *head, *tail;
15	
16	int pil, menu;
17	char pilihan[1];
18	string dataBaru;
19	
20	void initH();
21	void initHT();
22	int isEmptyH();
23	int isEmptyHT();
24	
25	void tambahDepanH();
26	void tambahDepanHT();
27	void tambahBelakangH();

```

28 void tambahBelakangHT();
29 void hapusDepanH();
30 void hapusDepanHT();
31 void hapusBelakangH();
32 void hapusBelakangHT();
33 void tampilkanH();
34 void tampilkanHT();
35 void clearH();
36 void clearHT();
37
38 int main() {
39     menu:
40     cout << "Double Linked List Non Circular
(DLLNC)" << endl;
41     cout <<
"===== " << endl;
42     cout << "1. DLLNC dengan Head" << endl;
43     cout << "2. DLLNC dengan Head dan Tail" << endl;
44     cout << "3. Quit" << endl;
45     cout << "Pilihan: ";
46     cin >> menu;
47     system("cls");
48     if(menu==1) {
49         do {
50             cout << "Double Linked List Non Circular
(DLLNC) (Head)" << endl;
51             cout <<
"===== "
<< endl;
52             cout << "1. Tambah Depan" << endl;
53             cout << "2. Tambah Belakang" << endl;
54             cout << "3. Tampilkan Data" << endl;
55             cout << "4. Hapus Depan" << endl;
56             cout << "5. Hapus Belakang" << endl;
57             cout << "6. Reset" << endl;
58             cout << "7. Kembali ke Menu" << endl;
59             cout << "Pilihan: ";
60             cin >> pilihan;
61             pil=atoi(pilihan);
62
63             switch(pil) {
64                 case 1:
65                     tambahDepanH();
66                     break;
67                 case 2:

```

```

68         tambahBelakangH();
69         break;
70     case 3:
71         tampilkanH();
72         break;
73     case 4:
74         hapusDepanH();
75         break;
76     case 5:
77         hapusBelakangH();
78         break;
79     case 6:
80         clearH();
81         break;
82     default:
83         system("cls");
84         goto menu;
85     }
86
87     cout << "\npres any key to continue"
<< endl;
88     getch();
89     system("cls");
90
91     } while (pil<7);
92     } else if (menu == 2) {
93         do {
94             cout << "Double Linked List Non Circular
(DLLNC) (Head dan Tail)" << endl;
95             cout <<
"=====
===== " << endl;
96             cout << "1. Tambah Depan" << endl;
97             cout << "2. Tambah Belakang" << endl;
98             cout << "3. Tampilkan Data" << endl;
99             cout << "4. Hapus Depan" << endl;
100            cout << "5. Hapus Belakang" << endl;
101            cout << "6. Reset" << endl;
102            cout << "7. Kembali ke Menu" << endl;
103            cout << "Pilihan: ";
104            cin >> pilihan;
105            pil=atoi(pilihan);
106
107            switch(pil) {
108            case 1:

```

```

109         tambahDepanHT();
110         break;
111     case 2:
112         tambahBelakangHT();
113         break;
114     case 3:
115         tampilkanHT();
116         break;
117     case 4:
118         hapusDepanHT();
119         break;
120     case 5:
121         hapusBelakangHT();
122         break;
123     case 6:
124         clearHT();
125         break;
126     default:
127         system("cls");
128         goto menu;
129     }
130
131     cout << "\npress any key to continue"
132     << endl;
133     getch();
134     system("cls");
135     } while (pil<7);
136     } else {
137         cout << "\nTERIMA KASIH" << endl;
138         cout << "Program was made by Noor Khalisa
139         (2410817220012)." << endl;
140     }
141
142     void initH() {
143         head = NULL;
144     }
145
146     void initHT() {
147         head = NULL;
148         tail = NULL;
149     }
150
151     int isEmptyH() {

```

```

152     if (head == NULL) return 1;
153     else return 0;
154 }
155
156 int isEmptyHT() {
157     if (tail == NULL) return 1;
158     else return 0;
159 }
160
161 void tambahDepanH() {
162     cout << "Masukkan data: ";
163     cin >> dataBaru;
164     TNode *baru;
165     baru = new TNode;
166     baru -> data = dataBaru;
167     baru -> next = NULL;
168     baru -> prev = NULL;
169     if(isEmptyH() == 1) {
170         head = baru;
171     } else {
172         baru -> next = head;
173         head -> prev = baru;
174         head = baru;
175     }
176     cout << "Data \""<<dataBaru<<"\" berhasil
dimasukkan di bagian depan.";
177 }
178
179 void tambahDepanHT() {
180     cout << "Masukkan data: ";
181     cin >> dataBaru;
182     TNode *baru;
183     baru = new TNode;
184     baru -> data = dataBaru;
185     baru -> next = NULL;
186     baru -> prev = NULL;
187     if (isEmptyHT() == 1) {
188         head = baru;
189         tail = baru;
190     } else {
191         baru -> next = head;
192         head -> prev = baru;
193         head = baru;
194     }

```



```

195         cout << "Data \""<<dataBaru<<"\" berhasil
dimasukkan di bagian depan.";
196     }
197
198 void tambahBelakangH() {
199     cout << "Masukkan data: ";
200     cin >> dataBaru;
201     TNode *baru, *bantu;
202     baru = new TNode;
203     baru -> data = dataBaru;
204     baru -> next = NULL;
205     baru -> prev = NULL;
206     if(isEmptyH() == 1) {
207         head = baru;
208     } else {
209         bantu = head;
210         while(bantu -> next != NULL) {
211             bantu = bantu -> next;
212         }
213         bantu -> next = baru;
214         baru -> prev = bantu;
215     }
216     cout << "Data \""<<dataBaru<<"\" berhasil
dimasukkan di bagian belakang.";
217 }
218
219 void tambahBelakangHT() {
220     cout << "Masukkan data: ";
221     cin >> dataBaru;
222     TNode *baru;
223     baru = new TNode;
224     baru -> data = dataBaru;
225     baru -> next = NULL;
226     baru -> prev = NULL;
227     if(isEmptyHT() == 1) {
228         head = baru;
229         tail = baru;
230     } else {
231         tail -> next = baru;
232         baru -> prev = tail;
233         tail = baru;
234     }
235     cout << "Data \""<<dataBaru<<"\" berhasil
dimasukkan di bagian belakang.";
236 }

```

```

237
238 void tampilkanH() {
239     TNode *bantu;
240     bantu = head;
241     if(isEmptyH() == 0) {
242         while(bantu != NULL) {
243             cout << bantu -> data << ' ';
244             bantu = bantu -> next;
245         }
246         cout << endl;
247     } else cout << "Tidak terdapat data pada Linked
List";
248 }
249
250 void tampilkanHT() {
251     TNode *bantu;
252     bantu = head;
253     if(isEmptyHT() == 0) {
254         while(bantu != tail -> next) {
255             cout << bantu -> data << ' ';
256             bantu = bantu -> next;
257         }
258         cout << endl;
259     } else cout << "Tidak terdapat data pada Linked
List";
260 }
261
262 void hapusDepanH() {
263     TNode *hapus;
264     string data;
265     if(isEmptyH() == 0) {
266         hapus = head;
267         data = hapus -> data;
268         if(head -> next != NULL) {
269             head = head -> next;
270             head -> prev = NULL;
271         } else {
272             initH();
273         }
274         delete hapus;
275         cout << "Data \""<<data<<"\" yang berada di
depan telah berhasil dihapus." << endl;
276     } else cout << "Tidak terdapat data pada Linked
List" << endl;
277 }

```

```

278
279 void hapusDepanHT() {
280     TNode *hapus;
281     string data;
282     if(isEmptyHT() == 0) {
283         hapus = head;
284         data = hapus -> data;
285         if(head -> next != NULL) {
286             head = head -> next;
287             head -> prev = NULL;
288         } else {
289             initHT();
290         }
291         delete hapus;
292         cout << "Data \""<<data<<"\" yang berada di
depan telah berhasil dihapus." << endl;
293     } else cout << "Tidak terdapat data pada Linked
List" << endl;
294 }
295
296 void hapusBelakangH() {
297     TNode *hapus;
298     string data;
299     if(isEmptyH() == 0) {
300         hapus = head;
301         while(hapus -> next != NULL) {
302             hapus = hapus -> next;
303         }
304         data = hapus -> data;
305         if(head -> next != NULL) {
306             hapus -> prev -> next = NULL;
307         } else {
308             initH();
309         }
310         delete hapus;
311         cout << "Data \""<<data<<"\" yang berada di
belakang telah berhasil dihapus." << endl;
312     } else cout << "Tidak terdapat data pada Linked
List" << endl;
313 }
314
315 void hapusBelakangHT() {
316     TNode *hapus;
317     string data;
318     if(isEmptyHT() == 0) {

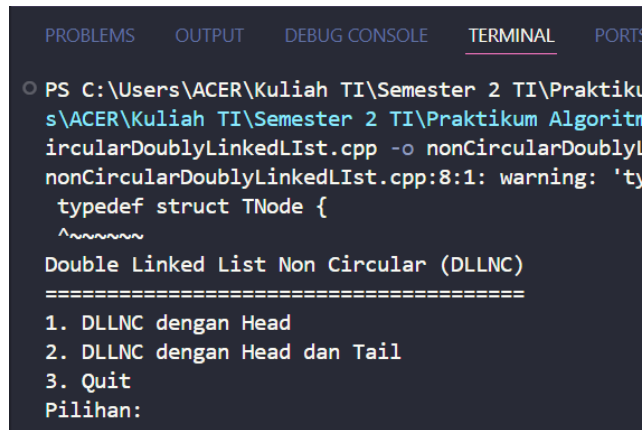
```

```

319         hapus = tail;
320         data = hapus -> data;
321         if(head -> next != NULL) {
322             tail = tail -> prev;
323             tail -> next = NULL;
324         } else {
325             initHT();
326         }
327         delete hapus;
328         cout << "Data \""<<data<<"\" yang berada di
belakang telah berhasil dihapus." << endl;
329     } else cout << "Tidak terdapat data pada Linked
List" << endl;
330 }
331
332 void clearH() {
333     TNode *bantu, *hapus;
334     bantu = head;
335     while(bantu != NULL) {
336         hapus = bantu;
337         bantu = bantu -> next;
338         delete hapus;
339     }
340     initH();
341     cout << "Seluruh data pada Linked List telah
dibersihkan." << endl;
342 }
343
344 void clearHT() {
345     TNode *bantu, *hapus;
346     bantu = head;
347     while(bantu != NULL) {
348         hapus = bantu;
349         bantu = bantu -> next;
350         delete hapus;
351     }
352     initHT();
353     cout << "Seluruh data pada Linked List telah
dibersihkan." << endl;
354 }
355
356 // need tutorial? watch these video from MIT
357 //
https://youtu.be/xvFZjo5PgG0?si=5ZdaPDqYTYsfCKbF

```

## B. Output



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\ACER\Kuliah TI\Semester 2 TI\Praktikum Algoritma\ACER\Kuliah TI\Semester 2 TI\Praktikum Algoritma\circularDoublyLinkedList.cpp -o nonCircularDoublyLinkedList.cpp:8:1: warning: 'typedef struct TNode {'
~~~~~
Double Linked List Non Circular (DLLNC)
=====
1. DLLNC dengan Head
2. DLLNC dengan Head dan Tail
3. Quit
Pilihan:
```

Gambar 1 Tampilan Awal Program DLLNC

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
○ Double Linked List Non Circular (DLLNC) (Head)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali ke Menu
Pilihan: █
```

Gambar 2 Tampilan Awal Program DLLNC Head

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
○ Double Linked List Non Circular (DLLNC) (Head)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali ke Menu
Pilihan: 1
Masukkan data: 1
Data "1" berhasil dimasukkan di bagian depan.
press any key to continue
█
```

Gambar 3 Tambah Depan

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
○ Double Linked List Non Circular (DLLNC) (Head)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali ke Menu
Pilihan: 2
Masukkan data: 9
Data "9" berhasil dimasukkan di bagian belakang.
press any key to continue
█
```

Gambar 4 Tambah Belakang

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
○ Double Linked List Non Circular (DLLNC) (Head)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali ke Menu
Pilihan: 3
1 9
press any key to continue
█
```

Gambar 5 Tampilkan Data

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
○ Double Linked List Non Circular (DLLNC) (Head)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali ke Menu
Pilihan: 4
Data "1" yang berada di depan telah berhasil dihapus.
press any key to continue
█
```

Gambar 6 Hapus Depan

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
○ Double Linked List Non Circular (DLLNC) (Head)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali ke Menu
Pilihan: 5
Data "9" yang berada di belakang telah berhasil dihapus.
press any key to continue
█
```

Gambar 7 Hapus Belakang

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
○ Double Linked List Non Circular (DLLNC) (Head)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali ke Menu
Pilihan: 6
Seluruh data pada Linked List telah dibersihkan.
press any key to continue
█
```

Gambar 8 Reset

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
○ Double Linked List Non Circular (DLLNC) (Head)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali ke Menu
Pilihan: 3
Tidak terdapat data pada Linked List
press any key to continue
█
```

Gambar 9 Tampilkan Data Setelah Reset

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
○ Double Linked List Non Circular (DLLNC) (Head)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali ke Menu
Pilihan: 4
Tidak terdapat data pada Linked List

press any key to continue
```

Gambar 10 Hapus Depan Jika Data Kosong

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
○ Double Linked List Non Circular (DLLNC) (Head)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali ke Menu
Pilihan: 5
Tidak terdapat data pada Linked List

press any key to continue
```

Gambar 11 Hapus Belakang Jika Data Kosong

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GIT
○ Double Linked List Non Circular (DLLNC) (Head dan Tail)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali ke Menu
Pilihan:
```

Gambar 12 Tampilan Awal Program DLLNC Head dan Tail

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GIT
○ Double Linked List Non Circular (DLLNC) (Head dan Tail)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali ke Menu
Pilihan: 1
Masukkan data: 2
Data "2" berhasil dimasukkan di bagian depan.
press any key to continue
```

Gambar 13 Tambah Depan

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GIT
○ Double Linked List Non Circular (DLLNC) (Head dan Tail)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali ke Menu
Pilihan: 2
Masukkan data: 5
Data "5" berhasil dimasukkan di bagian belakang.
press any key to continue
```

Gambar 14 Tambah Belakang

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GIT
○ Double Linked List Non Circular (DLLNC) (Head dan Tail)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali ke Menu
Pilihan: 2
Masukkan data: 8
Data "8" berhasil dimasukkan di bagian belakang.
press any key to continue
```

Gambar 15 Tambah Belakang Lagi

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GIT
○ Double Linked List Non Circular (DLLNC) (Head dan Tail)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali ke Menu
Pilihan: 4
Data "2" yang berada di depan telah berhasil dihapus.

press any key to continue
```

Gambar 16 Hapus Depan

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GIT
○ Double Linked List Non Circular (DLLNC) (Head dan Tail)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali ke Menu
Pilihan: 5
Data "8" yang berada di belakang telah berhasil dihapus.

press any key to continue
```

Gambar 17 Hapus Belakang

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLE
○ Double Linked List Non Circular (DLLNC) (Head dan Tail)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali ke Menu
Pilihan: 3
5
press any key to continue

```

Gambar 18 Tampilkan Data Setelah Hapus Depan dan Belakang

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLE
○ Double Linked List Non Circular (DLLNC) (Head dan Tail)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali ke Menu
Pilihan: 6
Seluruh data pada Linked List telah dibersihkan.
press any key to continue

```

Gambar 19 Reset

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLE
○ Double Linked List Non Circular (DLLNC) (Head dan Tail)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali ke Menu
Pilihan: 3
Tidak terdapat data pada Linked List
press any key to continue

```

Gambar 20 Tampilkan Data Setelah Reset

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLE
TERIMA KASIH
Program was made by Noor Khalisa (2410817220012).
PS C:\Users\ACER\Kuliah TI\Semester 2 TI\Praktikum Algoritma da

```

Gambar 21 Quit

### C. Pembahasan

1. `int main()` digunakan untuk mengelola antarmuka menu utama dan sub-menu untuk masing-masing mode DLLNC (Head atau Head-Tail). Pengguna diberi pilihan untuk menjalankan mode dengan Head saja atau dengan Head dan Tail. Setelah memilih salah satu, pengguna dapat melakukan berbagai operasi seperti menambah, menghapus, menampilkan, dan mereset data pada list. Fungsi ini memastikan alur program berjalan interaktif melalui penggunaan struktur perulangan dan switch-case, serta pengendalian tampilan dengan `system("cls")` dan `getch()` agar pengalaman pengguna lebih terstruktur dan nyaman.
2. `initH()` digunakan untuk menginisialisasi linked list versi head-only. Fungsi ini hanya menetapkan pointer head ke `NULL`, menandakan bahwa list belum memiliki data atau sudah dikosongkan.



3. `initHT()` digunakan untuk menginisialisasi linked list versi head-tail. Head dan tail di-set ke `NULL`, sehingga list berada dalam keadaan kosong sepenuhnya dan siap untuk diisi data baru dari kedua ujung.
4. `isEmptyH()` digunakan untuk memeriksa apakah list dalam keadaan kosong. Fungsi akan mengembalikan nilai 1 jika head bernilai `NULL`, dan 0 jika tidak, menandakan list tidak kosong.
5. `isEmptyHT()` digunakan untuk memeriksa kondisi kosong pada list versi head-tail, untuk mengembalikan 1 jika tail adalah `NULL`, yang berarti tidak ada data di dalam list.
6. `tambahDepanH()` digunakan untuk menambahkan node baru di bagian depan list pada versi head-only. Jika list kosong, node baru menjadi head. Jika tidak kosong, node baru dihubungkan sebagai head baru dan prev dari head lama dihubungkan ke node baru.
7. `tambahDepanHT()` serupa dengan `tambahDepanH()`, tetapi berlaku untuk versi head-tail. Selain mengatur node baru sebagai head, fungsi ini juga mengatur tail jika list sebelumnya kosong, sehingga node tunggal menjadi elemen pertama dan terakhir.
8. `tambahBelakangH()` digunakan untuk menambahkan data di akhir list untuk versi head-only. Jika list kosong, node baru langsung menjadi head. Jika tidak, fungsi melakukan traversal hingga akhir list dan menambahkan node baru sebagai next dari node terakhir.
9. `tambahBelakangHT()` digunakan untuk menambahkan data di bagian belakang list versi head-tail. Jika kosong, node baru menjadi head dan tail. Jika tidak, node baru langsung ditambahkan setelah tail, lalu tail diperbarui untuk menunjuk ke node tersebut.
10. `tampilkanH()` digunakan untuk menampilkan semua data dari list versi head-only, dimulai dari head hingga akhir (ketika next adalah `NULL`). Jika list kosong, pesan bahwa data tidak tersedia akan ditampilkan.

11. `tampilkanHT()` digunakan untuk menampilkan semua elemen list versi head-tail dengan cara traversal dari head hingga tail, termasuk tail. Jika list kosong, akan ditampilkan pesan bahwa tidak ada data.
12. `hapusDepanH()` digunakan untuk menghapus node dari bagian depan list versi head-only. Jika hanya ada satu node, list diinisialisasi ulang (kosong). Jika lebih dari satu node, head dipindah ke node berikutnya dan node lama dihapus dari memori.
13. `hapusDepanHT()` digunakan untuk menghapus node paling depan pada list versi head-tail. Jika hanya ada satu node, maka head dan tail dikosongkan. Jika tidak, head dipindah ke node selanjutnya dan node sebelumnya dihapus dari memori.
14. `hapusBelakangH()` digunakan untuk menghapus node terakhir dari list versi head-only. Jika hanya satu node, list direset. Jika lebih, pointer prev dari node terakhir digunakan untuk menghapus node tersebut, namun perlu traversal hingga ke node terakhir.
15. `hapusBelakangHT()` digunakan untuk menghapus node dari bagian belakang pada versi head-tail. Dengan adanya tail, proses ini lebih efisien karena langsung menunjuk node terakhir. Jika hanya satu node, list direset. Jika lebih, tail dipindah ke node sebelumnya.
16. `clearH()` digunakan untuk menghapus seluruh node dari list versi head-only. Fungsi melakukan traversal dari head, menghapus setiap node satu per satu, hingga semua node dihapus dan list dikembalikan ke kondisi kosong.
17. `clearHT()` digunakan untuk menghapus seluruh data pada list versi head-tail. Fungsi ini bekerja sama seperti `clearH()`, namun setelah semua node dihapus, ia juga mengatur tail ke NULL selain head, sehingga list benar-benar bersih.

## SOAL 2

Apa fungsi next pada coding?

### A. Pembahasan

Next adalah pointer yang menunjuk ke node berikutnya dalam linked list. Dengan kata lain, next berfungsi untuk menyambungkan suatu node dengan node yang datang setelahnya.

- Pada fungsi tambahDepanH() dan tambahDepanHT(), next digunakan untuk menyambungkan node baru ke node yang sebelumnya menjadi head, sehingga node baru berada di posisi paling depan dalam list.
- Pada fungsi tambahBelakangH() dan tambahBelakangHT(), next berfungsi untuk menghubungkan node yang sebelumnya menjadi node terakhir dengan node baru, menjadikannya node terakhir yang baru.
- Di fungsi hapusDepanH() dan hapusDepanHT(), pointer next membantu memindahkan head ke node berikutnya dan melewati node yang akan dihapus.
- Pada fungsi hapusBelakangH(), next digunakan untuk menelusuri hingga ke node terakhir dan kemudian memutuskan koneksi dengan mengatur next dari node sebelumnya menjadi NULL.
- Dalam fungsi hapusBelakangHT(), meskipun lebih banyak menggunakan prev, next pada node baru tetap diatur menjadi NULL agar tidak menunjuk ke node yang sudah dihapus.
- Di fungsi tampilkanH() dan tampilkanHT(), next digunakan untuk berjalan dari head ke node terakhir sambil menampilkan data tiap node.
- Pada fungsi clearH() dan clearHT(), next digunakan untuk menyimpan referensi ke node berikutnya sebelum node saat ini dihapus, sehingga proses penghapusan semua node bisa dilakukan dengan aman.

### SOAL 3

Apa fungsi prev pada coding?

#### A. Pembahasan

Prev adalah pointer yang menunjuk ke node sebelumnya dalam linked list. Ini memungkinkan traversal atau operasi balik (backward traversal), serta mempermudah penghapusan dan penyisipan node di belakang.

- Pada fungsi tambahDepanH() dan tambahDepanHT(), prev pada node yang sebelumnya menjadi head diatur menunjuk ke node baru, sehingga node baru menjadi node pertama dan node lama menjadi node kedua yang menunjuk kembali ke node baru.
- Di fungsi tambahBelakangH() dan tambahBelakangHT(), prev pada node baru diatur ke node yang sebelumnya adalah node terakhir, sehingga node baru bisa "menunjuk balik" ke node sebelumnya.
- Pada fungsi hapusDepanH() dan hapusDepanHT(), prev pada node baru yang menjadi head diatur menjadi NULL, karena tidak ada node di depannya lagi.
- Dalam fungsi hapusBelakangH() dan hapusBelakangHT(), prev digunakan untuk memindahkan pointer tail (pada HT) atau untuk mengakses node sebelum node terakhir, lalu menghapus node terakhir dan memastikan node sebelumnya tidak lagi menunjuk ke node yang dihapus.
- Fungsi tampilkanH() dan tampilkanHT() sebenarnya tidak langsung menggunakan prev karena penelusuran dilakukan dari head ke next, tapi prev tetap penting untuk menjaga hubungan dua arah antar node.
- Pada fungsi clearH() dan clearHT(), prev secara implisit ikut terhapus bersama node karena seluruh node dihapus satu per satu, sehingga hubungan mundur juga hilang.

## SOAL 4

Gantilah baris 243 dan 255 dari `cout<<"bantu->data<<"`; menjadi `cout<<"head->data<<"`; lalu jawab pertanyaan berikut:

- Apa yang terjadi jika anda menambahkan beberapa data pada program lalu tampilkan datanya, dan screenshoot hasilnya.
- Jelaskan mengapa hal tersebut bisa terjadi dan data apa yang ditampilkan oleh program?

### A. Pembahasan

- Setelah beberapa data dimasukkan ke dalam program dan perintah untuk menampilkan data dijalankan, program akan mencetak data yang sama secara berulang sebanyak jumlah node yang ada di dalam linked list. Meskipun data yang dimasukkan berbeda, hasil tampilan hanya akan menunjukkan data dari node pertama (head) secara berulang.

Tabel 2 Output Soal 4

 <pre> PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL ... o Double Linked List Non Circular (DLLNC) (Head) ===== 1. Tambah Depan 2. Tambah Belakang 3. Tampilkan Data 4. Hapus Depan 5. Hapus Belakang 6. Reset 7. Kembali ke Menu Pilihan: 1 Masukkan data: 0 Data "0" berhasil dimasukkan di bagian depan. press any key to continue </pre>	 <pre> PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL ... o Double Linked List Non Circular (DLLNC) (Head) ===== 1. Tambah Depan 2. Tambah Belakang 3. Tampilkan Data 4. Hapus Depan 5. Hapus Belakang 6. Reset 7. Kembali ke Menu Pilihan: 2 Masukkan data: 2 Data "2" berhasil dimasukkan di bagian belakang. press any key to continue </pre>
 <pre> PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL ... o Double Linked List Non Circular (DLLNC) (Head) ===== 1. Tambah Depan 2. Tambah Belakang 3. Tampilkan Data 4. Hapus Depan 5. Hapus Belakang 6. Reset 7. Kembali ke Menu Pilihan: 3 0 0 press any key to continue </pre>	

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL ... Code +
○ Double Linked List Non Circular (DLLNC) (Head dan Tail)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali ke Menu
Pilihan: 1
Masukkan data: 0
Data "0" berhasil dimasukkan di bagian depan.
press any key to continue

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL ... Code +
○ Double Linked List Non Circular (DLLNC) (Head dan Tail)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali ke Menu
Pilihan: 2
Masukkan data: 9
Data "9" berhasil dimasukkan di bagian belakang.
press any key to continue

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL ... Code +
○ Double Linked List Non Circular (DLLNC) (Head dan Tail)
=====
1. Tambah Depan
2. Tambah Belakang
3. Tampilkan Data
4. Hapus Depan
5. Hapus Belakang
6. Reset
7. Kembali ke Menu
Pilihan: 3
0 0
press any key to continue
```

B. Hal tersebut terjadi karena dalam fungsi `tampilkanH()` dan `tampilkanHT()`, perintah yang seharusnya mencetak data dari node yang sedang ditelusuri menggunakan pointer bantu, telah diubah menjadi mencetak `head -> data`. Perubahan ini menyebabkan program tidak menampilkan data dari setiap node dalam list, melainkan hanya mencetak data dari node head sebanyak jumlah node yang ada. Oleh karena itu, meskipun terdapat beberapa node dengan data berbeda, program hanya akan menampilkan isi dari node pertama (head) secara berulang.