

TUGAS PENDAHULUAN 4
STRUKTUR DATA

Nama : Bagas Tri Wibowo
NIM : 1301194051
Kelas : IF-43-04

Header :

```
main.cpp  X  doublelinkedlist.cpp  X  doublelinkedlist.h  X
1      #ifndef DOUBLELINKEDLIST_H_INCLUDED
2      #define DOUBLELINKEDLIST_H_INCLUDED
3      #define next(P) P->next
4      #define info(P) P->info
5      #define prev(P) P->prev
6      #define first(L) L.first
7      #define last(L) L.last
8
9      typedef int infotype;
10     typedef struct elmList *address;
11     struct elmList {
12         infotype info;
13         address next;
14         address prev;
15     };
16     struct List {
17         address first;
18         address last;
19     };
20     bool isEmpty (List L);
21     void createList(List &L);
22     void createNewElm(address &P, infotype x);
23     void insertFirst(List &L, address P);
24     void insertAfter(List &L, address Prec, address P);
25     void insertLast(List &L, address P);
26     void deleteFirst(List &L, address &P);
27     void deleteAfter(List &L, address Prec, address &P);
28
29     void deleteLast(List &L, address &P);
30     void concat(List L1, List L2, List &L3);
31     float median(List L);
32     void printInfo(List L);
33     void InserAscending(List &L, infotype x);
34     void deleteElm(List &L, infotype x);
35     int CountElm(List L);
36
37     #endif // DOUBLELINKEDLIST_H_INCLUDED
```

Implementation :

```
main.cpp × doublelinkedlist.cpp × doublelinkedlist.h ×
1      #include <iostream>
2      #include "doublelinkedlist.h"
3
4      using namespace std;
5      bool isEmpty (List L) {
6          if (first(L) != NULL) {
7              return false;
8          } else {
9              return true;
10         }
11     }
12     void createList(List &L) {
13         first(L) = NULL;
14         last(L) = NULL;
15     }
16     void createNewElm(address &P, infotype x) {
17         P = new elmList;
18         info(P) = x;
19         next(P) = NULL;
20         prev(P) = NULL;
21     }
22     void insertFirst(List &L, address P) {
23         if (isEmpty(L)) {
24             first(L) = P;
25             last(L) = P;
26         } else {
27             next(P) = first(L);
28             prev(first(L)) = P;
29             first(L) = P;
30         }
31     }
```

```
main.cpp x doublelinkedlist.cpp x doublelinkedlist.h x
32 void insertAfter(List &L, address Prec, address P) {
33     next(P) = next(Prec);
34     prev(P) = Prec;
35     next(Prec) = P;
36 }
37 void insertLast(List &L, address P) {
38     if (isEmpty(L)) {
39         first(L) = P;
40         last(L) = P;
41     } else {
42         prev(P) = last(L);
43         next(last(L)) = P;
44         last(L) = P;
45     }
46 }
47 void deleteFirst(List &L, address &P) {
48     P = first(L);
49     first(L) = next(P);
50     prev(next(P)) = NULL;
51     next(P) = NULL;
52 }
53 void deleteAfter(List &L, address Prec, address &P) {
54     P = next(Prec);
55     next(Prec) = next(P);
56     prev(next(P)) = prev(P);
57     next(P) = NULL;
58     prev(P) = NULL;
59 }
```

```
main.cpp x doublelinkedlist.cpp x doublelinkedlist.h x
60 void deleteLast(List &L, address &P) {
61     P = last(L);
62     last(L) = prev(P);
63     next(prev(P)) = NULL;
64     prev(P) = NULL;
65 }
66 void concat(List L1, List L2, List &L3) {
67     first(L3) = first(L1);
68     last(L3) = last(L2);
69     next(last(L1)) = first(L2);
70     prev(first(L2)) = last(L1);
71 }
72 float median(List L) {
73     int jumElm = CountElm(L);
74     int j = (jumElm/2)+1;
75     address P = first(L);
76     if (jumElm%2 != 0) {
77         for(int i=1;i<j;i++) {
78             P = next(P);
79         }
80         return info(P);
81     } else {
82         for(int i=1;i<j-1;i++) {
83             P = next(P);
84         }
85         return (info(P)+info(next(P)))/2;
86     }
87 }
```

```
main.cpp x doublelinkedlist.cpp x doublelinkedlist.h x
88 void printInfo(List L) {
89     address P = first(L);
90     cout<<" ";
91     while (P != NULL) {
92         cout<<info(P);
93         if (next(P) != NULL) {
94             cout<<" ";
95         }
96         P = next(P);
97     }
98     cout<<" ";
99     cout<<endl;
100 }
101 void InsertAscending(List &L, infotype x) {
102     address P;
103     createNewElm(P,x);
104     if (isEmpty(L)) {
105         insertFirst(L,P);
106     } else {
107         if (x <= info(first(L))) {
108             insertFirst(L,P);
109         } else if (x >= info(last(L))) {
110             insertLast(L,P);
111         } else {
112             address Prec = first(L);
113             while (x > info(next(Prec))) {
114                 Prec = next(Prec);
115             }
116             insertAfter(L,Prec,P);
117         }
118     }
```

```
120 void deleteElm(List &L, infotype x) {
121     address P;
122     createNewElm(P,x);
123     if (x == info(first(L))) {
124         deleteFirst(L,P);
125     } else if (x == info(last(L))) {
126         deleteLast(L,P);
127     } else {
128         address Prec = first(L);
129         while (x != info(next(Prec))) {
130             Prec = next(Prec);
131         }
132         deleteAfter(L,Prec,P);
133     }
134 }
135 int CountElm(List L) {
136     address P = first(L);
137     int i = 0;
138     while (P != NULL) {
139         i++;
140         P = next(P);
141     }
142     return i;
143 }
144
```


Main Program :

```
main.cpp x doublelinkedlist.cpp x doublelinkedlist.h x
1  #include <iostream>
2  #include "doublelinkedlist.cpp"
3
4  using namespace std;
5
6  int main()
7  {
8      /**
9          Nama      : Bagas Tri Wibowo
10         Kelas     : IF-43-04
11         NIM        : 1301194051
12     */
13     List L1,L2,L3;
14     createList(L1);
15     createList(L2);
16     createList(L3);
17
18     cout<<"====> List 1 <===="<<endl;
19     cout<<"Insert Asc (10)"<<endl;
20     InserAscending(L1,10);
21     printInfo(L1);
22     cout<<"Insert Asc (5)"<<endl;
23     InserAscending(L1,5);
24     printInfo(L1);
25     cout<<"Insert Asc (15)"<<endl;
26     InserAscending(L1,15);
27     printInfo(L1);
28     cout<<"Insert Asc (6)"<<endl;
29     InserAscending(L1,6);
30
31     printInfo(L1);
32
33     cout<<"Median List 1 : ";
34     cout<<median(L1)<<endl<<endl;
35
36     cout<<"====> List 2 <===="<<endl;
37     InserAscending(L2,30);
38     InserAscending(L2,70);
39     InserAscending(L2,24);
40     InserAscending(L2,20);
41     InserAscending(L2,42);
42     InserAscending(L2,10);
43     InserAscending(L2,14);
44     printInfo(L2);
45     cout<<"delete Elm (10)"<<endl;
46     deleteElm(L2,10);
47     printInfo(L2);
48     cout<<"delete Elm (70)"<<endl;
49     deleteElm(L2,70);
50     printInfo(L2);
51     cout<<"delete Elm (24)"<<endl;
52     deleteElm(L2,24);
53     printInfo(L2);
54     cout<<"Median List 2 : ";
55     cout<<median(L2)<<endl<<endl;
56
57     cout<<"====> List 3 <===="<<endl;
58     cout<<"List 3 = List 1 + List 2"<<endl;
59     concat(L1,L2,L3);
60     printInfo(L3);
61     return 0;
62 }
```

Output :

```
====> List 1 <====
Insert Asc (10)
{10}
Insert Asc (5)
{5 10}
Insert Asc (15)
{5 10 15}
Insert Asc (6)
{5 6 10 15}
Median List 1 : 8

====> List 2 <====
{10 14 20 24 30 42 70}
delete Elm (10)
{14 20 24 30 42 70}
delete Elm (70)
{14 20 24 30}
delete Elm (24)
{14 20 30}
Median List 2 : 20

====> List 3 <====
List 3 = List 1 + List 2
{5 6 10 15 14 20 30}

Process returned 0 (0x0)   execution time : 0.060 s
Press any key to continue.
```