TIF1202 – Pemrograman Berorientasi Objek HO 08 - Alokasi Memori Dinamik

Opim Salim Sitompul

Department of Information Technology Universitas Sumatera Utara







Outline

- Tujuan
- 2 Konsep Dasar
- Pengalokasian Memori Secara Dinamik
- 4 Linked-List menggunakan struct
- Doubly Linked-List







Tujuan

- Setelah menyelesaikan modul ini mahasiswa diharapkan:
 - Memahami konsep pengalokasian memori secara dinamik
 - Dapat mengalokasikan array secara dinamik
 - Memahami konsep struktur data struct
 - Dapat mengimplementasikan konsep linked-list
 - Dapat mengimplementasikan konsep doubly linked-list







Konsep Dasar

- Mengalokasikan memori sebanyak yang diperlukan.
- Mengurangi jumlah perubahan yang harus dibuat terhadap program apabila kebutuhan program berubah.







- Sebelum memperkenalkan konsep alokasi memori dinamik, diberikan terlebih dahulu contoh alokasi memory statik.
- Program berikut mengalokasikan memori sebanyak 20 bytes ke array integer kemudian mengisi elemen-elemen array dengan bilangan acak.
- Bilangan acak yang dihasilkan adalah berdasarkan benih (seed) yang diberikan berdasarkan fungsi waktu time().
- Fungsi bubble_sort() kemudian akan mensortir elemen-elemen array tersebut secara menaik (asscending).
- Contoh Program 8.1:





```
//Contoh8_1.cpp
   #include <iostream>
3
   #include <string>
   #include <cstdlib>
5
   #include <ctime>
6
   #define N 100
8
   using namespace std;
9
10
   class Array
11
12
     private:
13
       int i, array[N];
```







Konsep Dasar

0000000

```
14
      public:
15
        Array();
16
        void bubble sort();
17
        void printArray(string);
18
        ~Array(){}
19
    };
20
21
   Array::Array()
22
23
      time t seed:
24
25
      srand(time(&seed));
26
      for (int i=0; i < N; i++)</pre>
27
        array[i] = rand() % N;
28
```





Doubly L



```
29
   void Array::bubble_sort()
30
    {
31
      for (int i=0; i < N; i++)
32
        for (int j=0; j < N; j++)
33
          if(array[i] < array[j])</pre>
34
35
             int temp = array[i];
36
             array[i] = array[j];
37
             array[j] = temp;
38
39
```







```
40  void Array::printArray(string str)
41  {
42   cout << str << endl;
43   for(int i=0; i < N; i++)
44    cout << array[i] << "_";
45   cout << endl;
46 }</pre>
```







```
47
   int main()
48
49
     Array myArr;
50
51
     myArr.printArray("Sebelum_disortir:_");
52
     myArr.bubble sort();
53
      myArr.printArray("Setelah, disortir: ");
54
55
      return 0:
56
```







- Pengalokasian memori secara dinamik dilakukan melalui konsep pointer.
- Operasi yang digunakan untuk mengalokasikan memori adalah new.
- Memori yang sudah tidak diperlukan lagi dapat dikembalikan ke pool memori dengan operasi delete.
- Contoh Program 8.2:
- Pada contoh ini, operasi delete dilakukan di dalam destructor.







```
//Contoh8_2.cpp
   #include <iostream>
3
   #include <string>
   #include <cstdlib>
5
   #include <ctime>
6
   using namespace std;
8
9
   class Array
10
11
     private:
12
     int *array;
13
     int size;
```







```
14
      public:
15
        Array(int);
16
        void generate();
17
        void bubble sort();
18
        void printArray(string);
19
        ~Array() { delete [] array; }
20
21
22
   Array::Array(int size)
23
24
      time t seed;
25
      this->size = size;
26
      srand(time(&seed));
27
```





Doubly L



```
28
   void Array::generate()
29
30
      if ((array = new int[size]) == NULL)
31
        cerr << "Error alokasi memori..." << endl;</pre>
32
      else
33
34
        for(int i=0; i < size; i++)</pre>
35
          array[i] = rand() % size;
36
37
```







```
38
    void Array::bubble_sort()
39
40
      for(int i=0; i < size; i++)</pre>
41
        for(int j=0; j < size; j++)</pre>
42
           if(array[i] < array[j])</pre>
43
44
             int temp = array[i];
45
             array[i] = array[j];
46
             array[j] = temp;
47
48
```







```
49  void Array::printArray(string str)
50  {
51    cout << str << endl;
52    for(int i=0; i < size; i++)
53       cout << array[i] << "_";
54    cout << endl;
55 }</pre>
```







Doubly L

```
56
   int main()
57
58
     Array myArr(100);
59
60
     //Generate random array
61
     myArr.generate();
62
63
     //Print unsorted array
64
      myArr.printArray("Sebelum disortir:");
65
      //Sort array
66
     myArr.bubble_sort();
67
      //Print sorted array
68
      myArr.printArray("Sesudah disortir:");
69
70
      return 0;
                                      イロン イ押 とくき とくき とうきょう
```

- Linked-list menggunakan struktur data struct dalam membentuk simpul-simpul (nodes) list.







 Masing-masing simpul dalam linked list terdiri dari dua jenis field yaitu field data dan field pointer ke simpul berikutnya:



Gambar 1: Struktur data linked-list

 Linked-list di atas disebut singly linked-list atau biasa disebut linked-list saja.







- Linked list dapat dibuat pada stack.
- Ketika objek itu meninggalkan scope fungsi main(), secara otomatis akan di keluarkan dari stack (popped off), dan destructor akan dipanggil.
- Contoh:
 - LinkList myList;
 myList.clear();
- Contoh Program 8.3:







```
//Contoh8_3.cpp
2 #include <iostream>
3
   #include <iomanip>
   #include <cstdlib>
5
   #include <ctime>
6
   #define N 100
7
8
   using namespace std;
9
10
   struct Node
11
12
  int value;
13
     Node *next;
14
```







```
15
   class LinkList
16
17
      private:
18
        Node *first;
19
      public:
20
        LinkList();
21
        void show list();
22
        void append value(int);
23
        void clear();
24
       ~LinkList(){}
25
```







```
26
   LinkList::LinkList()
27
28
      srand(time(NULL));
29
      first = NULL;
30
31
32
   void LinkList::show_list()
33
34
      Node *node=first;
35
      while (node)
36
37
        cout << node->value << ".";
38
        node = node->next;
39
40
      cout << endl;
41
```





Doubly L

```
42
   void LinkList::append value(int value)
43
44
     Node *tmp, *tail;
45
     tmp = new Node;
46
     tmp->value = value;
47
     tmp->next = NULL;
48
     if (first == NULL)
49
        first = tmp;
50
     else
51
52
        tail = first;
53
        while(tail->next)
54
          tail = tail->next;
55
        tail->next = tmp;
56
57
```





Doubly L

```
58
   void LinkList::clear() {
59
      Node* temp;
60
61
      cout << "deleting...";</pre>
62
63
      while (first != NULL) {
64
        cout << first->value << " ";
65
        temp = first->next;
66
        delete first:
67
        first = temp;
68
69
      cout << endl;
70
      first = NULL;
71
```







```
72
   int main()
73
74
      LinkList myList;
75
76
      cout << "Linked list:" << endl;</pre>
77
      for (int i=0; i < N; i++)
78
        myList.append_value(rand() % N);
79
      myList.show_list();
80
81
      myList.clear();
82
83
      return 0;
84
```







- Linked-list juga dapat dialokasikan pada heap.
- Contoh:

```
LinkList *myList = new LinkList();
```

- Linked-list kemudian dapat dihapus dengan: delete myList;
- Jika list ingin dikosongkan atau memori yang digunakan dikembalikan sebelum dihapus, maka dapat dilakukan dengan memanggil fungsi clear().

```
myList->clear();
delete myList;
```

Contoh Program 8.4:







```
//Contoh8_4.cpp
2 #include <iostream>
3
   #include <iomanip>
   #include <cstdlib>
5
   #include <ctime>
6
   #define N 100
7
8
   using namespace std;
9
10
   struct Node
11
12
  int value;
13
     Node *next;
14
```







```
15
   class LinkList
16
17
      private:
18
        Node *first;
19
      public:
20
        LinkList();
21
        void show list();
22
        void append value(int);
23
        void clear();
24
       ~LinkList(){}
25
```







```
29
   void Array::bubble_sort()
30
    {
31
      for (int i=0; i < N; i++)
32
        for (int j=0; j < N; j++)
33
          if(array[i] < array[j])</pre>
34
35
             int temp = array[i];
36
             array[i] = array[j];
37
             array[j] = temp;
38
39
```







```
42
   void LinkList::append value(int value)
43
44
     Node *tmp, *tail;
45
     tmp = new Node;
46
     tmp->value = value;
47
     tmp->next = NULL;
48
     if (first == NULL)
49
     first = tmp;
50
     else
51
52
        tail = first;
53
        while(tail->next)
54
       tail = tail->next;
55
        tail->next = tmp;
56
57
```





Doubly L

```
58
   void LinkList::clear() {
59
      Node* temp;
60
61
      cout << "deleting...";</pre>
62
63
      while (first != NULL) {
64
        cout << first->value << " ";
65
        temp = first->next;
66
        delete first:
67
        first = temp;
68
69
      cout << endl;
70
      first = NULL;
71
```







```
72
   int main()
73
74
      LinkList *myList = new LinkList();
75
76
      cout << "Linked list:" << endl;</pre>
77
      for (int i=0; i < N; i++)
78
        myList->append value(rand() % N);
79
      myList->show list();
80
81
      myList->clear();
82
      delete myList;
83
84
      return 0;
85
```







Contoh Program 8.5: reverse list

```
//Contoh8_5.cpp
   #include <iostream>
3
   #include <iomanip>
   #include <cstdlib>
5
   #include <ctime>
6
   #define N 100
8
   using namespace std;
9
10
   struct Node
11
12
   int value;
13 Node *next;
14
```







```
15
   class LinkList
16
17
     private:
18
        Node *first;
19
      public:
20
        LinkList();
21
        void show list();
22
        void append value(int);
23
        void reverseList();
24
       ~LinkList(){}
25
```







```
26
   LinkList::LinkList()
27
28
      srand(time(NULL));
29
      first = NULL;
30
31
32
   void LinkList::show_list()
33
34
      Node *node = first;
35
      while (node)
36
37
        cout << node->value << ".";
38
        node = node->next;
39
40
      cout << endl;
41
```





Doubly L

```
42
   void LinkList::append value(int value)
43
44
      Node *tmp, *tail;
45
46
     tmp = new Node;
47
      tmp->value = value;
48
     tmp->next = NULL;
49
      if (first == NULL)
50
        first = tmp;
51
      else
52
53
        tail = first;
54
        while (tail->next)
55
          tail = tail->next;
56
        tail->next = tmp;
57
```



Kampus Merdeka

```
59
   void LinkList::reverseList()
60
61
      Node \star temp = first;
62
      Node* nextnode = NULL;
63
      Node* revnode = NULL;
64
65
      while (temp != NULL)
66
67
            first = temp;
68
            nextnode = temp->next;
69
            temp->next = revnode;
70
            revnode = temp;
71
            temp = nextnode;
72
73
```







```
74
    int main()
75
76
      LinkList myList;
77
78
      cout << "Linked_list_awal:" << endl;</pre>
79
      for (int i=0; i < N; i++)
80
        myList.append_value(rand() % N);
81
      myList.show_list();
82
83
      myList.reverseList();
84
      cout << "Linked_list_setelah_dibalik:"</pre>
85
        << endl;
86
      myList.show list();
87
88
      return 0;
89
```





Penjelasan algoritma reverse pada program 8.5:

```
59
   void LinkList::reverseList()
60
61
      Node \star temp = first;
62
      Node* nextnode = NULL;
63
      Node* revnode = NULL:
64
65
      while (temp != NULL)
66
67
            first = temp;
68
            nextnode = temp->next;
69
            temp->next = revnode;
70
            revnode = temp;
71
            temp = nextnode;
72
```

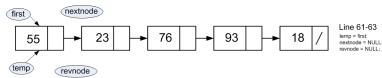


Doubly L





Keadaan awal sebelum loop:

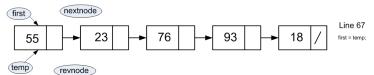








Iterasi 1:

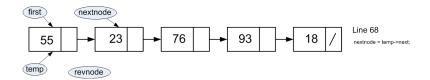








• Iterasi 1:

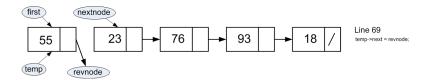








Iterasi 1:

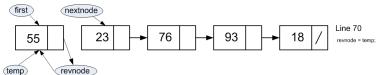








Iterasi 1:

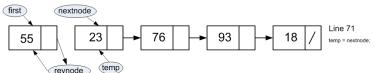








• Iterasi 1:









Contoh Program 8.6:

```
1 // Contoh8_6.cpp
2 // Program untuk menyimpan titik koordinat
      dalam linked list
  #include <iostream>
4
5
   using namespace std;
6
   struct Simpul
8
9
       int absis;
10
       int ordinat;
11
       Simpul *berikut;
```







```
13
   class titikKoord
14
15
      private:
16
        Simpul *first;
17
      public:
18
        titikKoord();
19
        void tambahTitik();
20
        void buanqTitik();
21
        void cetakDaftar();
22
   };
23
24
   titikKoord::titikKoord()
25
26
      first = NULL;
27
```







```
28
   void titikKoord::cetakDaftar()
29
30
      Simpul *node;
31
32
      cout << "Daftar titik koordinat:" << endl;</pre>
33
      node = first;
34
      while (node)
35
36
        cout << "(" << node->absis << ",.."
                 << node->ordinat << ")";
37
38
        node = node->berikut;
39
40
      cout << endl;
41
```







```
42
   void titikKoord::tambahTitik()
43
44
      Simpul *baru, *ekor;
45
46
     baru = new Simpul;
47
48
      cout << "Berikan absis :: ";
49
      cin >> baru->absis;
50
      cout << "Berikan ordinat: ";</pre>
51
      cin >> baru->ordinat;
52
      baru->berikut = NULL;
53
54
      if(first == NULL)
55
        first = baru;
```







```
56
      else
57
58
        ekor = first;
59
        while (ekor->berikut)
60
          ekor = ekor->berikut;
61
        ekor->berikut = baru;
62
63
64
65
   void titikKoord::buangTitik()
66
67
      int x, y, ada = 0;
68
      Simpul *cari, *hapus;
69
      cout << "Titik mana?" << end rensform
70
```



```
71
      cout << "Berikan absis dan ordinat:..";</pre>
72
      cin >> x >> y;
73
74
      cari = first;
75
      while (cari && !ada)
76
77
        if((cari->absis == x) \&\&
78
                 (cari->ordinat == y))
79
          ada = 1;
80
        else
81
          cari = cari->berikut;
82
83
      if (ada)
84
85
        hapus = first;
```





Doubly L

イロン イ押 とくき とくき とうきょう

```
86
        if (hapus == cari)
87
88
           first = first->berikut;
89
           cari->berikut = NULL;
90
           delete cari:
91
92
        else
93
94
           while ((hapus->berikut) &&
95
                         (hapus->berikut != cari))
96
             hapus = hapus->berikut;
97
           hapus->berikut = cari->berikut;
98
           cari->berikut = NULL;
99
           delete cari:
100
101
```

Kampus Merdeka

Linked-List menggunakan struct

else

102

```
103
         cout << "Titik ("<< x << ", " << y
104
               << ") tidak ada dalam daftar." <<
                   endl;
105
106
107
    int main()
108
109
      titikKoord titikA;
110
      char pilih = '0';
111
      do
112
113
         system("cls");
114
         cout << "Menu:" << endl;
115
         cout << "1. Menambah_titik." << endl;</pre>
         cout << "2. Membuang titik
116
```

cout << "3. Mencetak daftar." << endl;</pre>

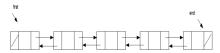
Linked-List menggunakan struct

117

100

```
118
         cout << "4. Selesai." << endl;</pre>
119
         cout << "Pilihan:.";</pre>
120
         cin >> pilih;
121
         switch (pilih)
122
123
              case '1': titikA.tambahTitik();
124
                  break:
125
              case '2': titikA.buangTitik();
126
                  break:
127
              case '3': titikA.cetakDaftar();
128
                system("pause");
129
                   break;
130
131
       } while (pilih < '4');</pre>
132
       return 0:
                                           4日 → 4周 → 4 目 → 4 目 → 9 Q P
```

 Sebuah struktur list lain disebut doubly linked-list memiliki dua field pointer; satu menunjuk ke simpul berikut dan satu lagi menunjuk ke simpul sebelumnya.



Gambar 2: Struktur data doubly linked-list







- Program berikut adalah implementasi dari struktur data doubly linked-list yang digunakan untuk menyimpan bilangan-bilangan bulat sebanyak N.
- Program ini menggunakan sebuah struktur untuk menyajikan simpul-simpul yang memuat data, satu pointer yang menunjuk ke simpul berikut dan satu pointer menunjuk ke simpul sebelumnya.
- Contoh Program 8.7:







```
//Contoh8_7.cpp
   #include <iostream>
3
   #include <iomanip>
   #include <cstdlib>
5
   #include <ctime>
6
   #define N 15
7
8
   using namespace std;
9
10
   struct Node
11
12
     int value;
13
     Node *next;
14
     Node *prev;
15
```







```
16
   class DoubleLinkList
17
18
     private:
19
        Node *first;
20
        Node *end;
21
      public:
22
        DoubleLinkList();
23
        void show list();
24
        void show_prevlist();
25
        Node *append value(int);
26
```





Doubly L



Doubly L

4 D > 4 P > 4 P > 4 P >

```
27
   DoubleLinkList::DoubleLinkList()
28
29
      srand(time(NULL));
30
      first = NULL:
31
     end = NULL:
32
33
34
   void DoubleLinkList::show list()
35
36
     Node *node;
37
38
      node = first;
39
      while (node)
40
41
        cout << node->value <<
42
        node = node->next;
```

```
46
   void DoubleLinkList::show prevlist()
47
48
      Node *node;
49
50
      node = end;
51
      while (node)
52
53
        cout << node->value << "..";
54
        node = node->prev;
55
56
      cout << endl;
57
```







イロン イ押 とくき とくき とうきょう

```
58
   Node *DoubleLinkList::append value(int value)
59
60
      Node *ptr = end;
61
      end = new Node;
62
      if (first == NULL)
63
        first = end;
64
      else
65
        ptr->next = end;
66
      if (end)
67
68
        end->next = NULL;
69
        end->prev = ptr;
70
        end->value = value;
71
72
      return end;
73
```

```
74
   int main()
75
76
      DoubleLinkList myList;
77
78
      cout << "Double Linked list:" << endl;</pre>
79
      for (int i=0; i < N; i++)
80
        myList.append value(rand() % N);
81
      cout << "Print maju: " << endl;
82
      myList.show list();
83
      cout << "Print mundur: " << endl;</pre>
84
      myList.show_prevlist();
85
86
      return 0;
87
```





Doubly L

