TIF1202 – Pemrograman Berorientasi Objek

HO 09 - Set, Stack, dan Queue

Opim Salim Sitompul

Department of Information Technology
Universitas Sumatera Utara

Outline

Tujuan

- Setelah menyelesaikan modul ini mahasiswa diharapkan:
  - Memahami konsep struktur data set
  - Memahami konsep struktur data sederhana Stack
  - Memahami konsep struktur data sederhana Queue
  - Dapat mengimplementasikan set, stack dan queue pada array
  - Dapat mengimplementasikan stack dan queue pada linked-list

Konsep Dasar Set

- Set adalah struktur data untuk menampung elemen-elemen tunggal yang memenuhi urutan tertentu.
- Nilai sebuah elemen dalam set juga bertindak sebagai kunci untuk mengenali elemen tersebut.
- Nilai elemen dalam sebuah set tidak dapat diubah.

Konsep Dasar Set

- Operasi-operasi dasar pada set antara lain adalah:
  - Menambah elemen
  - Membuang elemen
  - Memeriksa apakah sebuah elemen adalah anggota set
  - Menggabungkan dua buah set
  - Melakukan irisan dua buah set.
- Contoh Program 9.1:

Implementasi Set

```
1  //Contoh9_1.cpp
2  #include <iostream>
3  #include <cstdlib>
4  #include <ctime>
5
6  #define MAX 100
7  enum Bool {False, True};
8
9  using namespace std;
10
11 class Set
12 {
13   private:
14     int elemen[MAX]; // elemen himpunan
15     int kardinalitas; // kardinalitas himpunan
```

Implementasi Set

```
16    public:
17       Set() {srand(time(NULL)); kardinalitas=0;}
18       Bool Anggota (const int);
19       Bool tambahElemen (const int);
20       void bubble_sort();
21       void buangElemen (const int);
22       int getElemen(const int);
23       void Copy (Set&);
24       Bool Equal (Set&);
25       void Irisan (Set&, Set&);
26       void Gabungan (Set&, Set&);
27       void Print (string);
28    };
```

Implementasi Set

```
29  Bool Set::Anggota (const int elem)
30  {
31    for (int i = 0; i < kardinalitas; ++i)
32      if (elemen[i] == elem)
33        return True;
34      return False;
35  }
36
37  int Set::getElemen(const int i)
38  {
39    if (i < kardinalitas)
40      return elemen[i];
41    else
42      return -99;
43  }
```

Implementasi Set

```
44  Bool Set::tambahElemen (const int elem)
45  {
46    if (Anggota(elem))
47      return False;
48    if (kardinalitas < MAX)
49    {
50      elemen[kardinalitas++] = elem;
51      bubble_sort();
52      return True;
53    }
54    else
55      cout << "Set overflow\n";
56    return False;
57  }
```

Implementasi Set

```
58  void Set::bubble_sort()
59  {
60    int i, j, temp;
61    for(i = 0; i < kardinalitas-1; i++)
62      for(j = kardinalitas-1; j > i; j--)
63        if(elemen[j-1] > elemen[j])
64        {
65          temp = elemen[j - 1];
66          elemen[j - 1] = elemen[j];
67          elemen[j] = temp;
68        }
69  }
```

Implementasi Set

```
70  void Set::buangElemen (const int elem)
71  {
72    for (int i = 0; i < kardinalitas; ++i)
73      if (elemen[i] == elem)
74      {
75        for (; i < kardinalitas-1; ++i)
              elemen[i] = elemen[i+1];
76        --kardinalitas;
77      }
78  }
```

Implementasi Set

```
79  void Set::Copy (Set &set)
80  {
81    for (int i = 0; i < kardinalitas; ++i)
82      set.elemen[i] = elemen[i];
83    set.kardinalitas = kardinalitas;
84  }
85
86  Bool Set::Equal (Set &set)
87  {
88    if (kardinalitas != set.kardinalitas)
89      return False;
90    for (int i = 0; i < kardinalitas; ++i)
91      if (!set.Anggota(elemen[i]))
92        return False;
93      return True;
94  }
```

Implementasi Set

```
95  void Set::Irisan (Set &set, Set &res)
96  {
97    res.kardinalitas = 0;
98
99    for (int i = 0; i < kardinalitas; ++i)
100     if (set.Anggota(elemen[i]))
101       res.elemen[res.kardinalitas++] =
102                         elemen[i];
103 }
104
105 void Set::Gabungan (Set &set, Set &res)
106 {
107   set.Copy(res);
108   for (int i = 0; i < kardinalitas; ++i)
109     res.tambahElemen(elemen[i]);
110 }
```

Implementasi Set

```
111 void Set::Print (string ss)
112 {
113   cout << ss << "␣=␣{";
114   for (int i = 0; i < kardinalitas-1; ++i)
115     cout << elemen[i] << ",";
116   if (kardinalitas > 0)
117     cout << elemen[kardinalitas-1];
118   cout << "}\n";
119 }
120
121 int main ()
122 {
123   Set s1, s2, s3;
124   int i, pil=0;
125   string himpunan;
```

Implementasi Set

```
126     i=0;
127     while(i<10)
128       if (s1.tambahElemen(rand()%MAX))
129         i++;
130     s1.Print("s1");
131
132     i=0;
133     while(i<10)
134       if (s2.tambahElemen(rand()%MAX))
135         i++;
136     s2.Print("s2");
```

Implementasi Set

```
137     cout <<"Buang elemen ke 1 dari s1:" <<endl;
138     s1.buangElemen(1);
139     s1.Print("s1");
140
141     if (s1.Anggota(20))
142       cout << "20 anggota s1\n";
143     else
144       cout << "20 bukan anggota s1\n";
145
146     s1.Irisan(s2,s3);
```

Implementasi Set

```
147     cout << "s1 Irisan s2: ";
148     s3.Print("s3");
149     s1.Gabungan(s2,s3);
150     cout << "s1 Gabungan s2: ";
151     s3.Print("s3");
152     if (!s1.Equal(s2)) cout << "s1 != s2\n";
153
154     return 0;
155 }
```

Konsep Dasar Stack

- Stack adalah daftar berurut yang menyimpan jenis data yang sama.
- Data disimpan mengikuti struktur Last In First Out (LIFO)
- Operasi pada stack hanya dilakukan pada puncak stack.
- Ada dua operasi dasar stack, yaitu *push()* dan *pop()*.
- *Push()* menambahkan elemen di puncak stack.
- *pop()* mengambil satu elemen dari puncak stack.
- Contoh program 9.2:

Implementasi Stack pada Array

```
1  //Contoh9_2.cpp
2  #include <iostream>
3  #include <cstdlib>
4  #include <ctime>
5  #define N 10
6
7  using namespace std;
8
9  class stack
10 {
11   private:
12     int *storage;
13     int elements;
14     int isempty;
15     int isfull;
16     int stack_size;
```

Implementasi Stack pada Array

```
17    public:
18       stack(int size);
19       void push (int value);
20       int pop(void);
21       void show_stack();
22       int is_empty(void) { return (isempty); }
23       int is_full(void) { return (isfull);}
24       ~stack(){delete [] storage;}
25 };
```

Implementasi Stack pada Array

```
26  stack::stack(int size)
27  {
28    storage = new int[size];
29    elements = 0;
30    isempty = 1;
31    isfull = 0;
32    stack_size = size;
33  }
```

Implementasi Stack pada Array

```
34  int stack::pop(void)
35  {
36    if (!is_empty())
37    {
38        if (--elements == 0)
39                isempty = 1;
40      return (storage[elements]);
41    }
42  }
```

Implementasi Stack pada Array

```
43  void stack::push(int value)
44  {
45    if (!is_full())
46    {
47      storage[elements++] = value;
48      if (elements == stack_size)
49        isfull = 1;
50      isempty = 0;
51    }
52  }
```

Implementasi Stack pada Array

```
53  void stack::show_stack()
54  {
55    int i=0;
56
57    while(i<stack_size)
58    {
59      cout << storage[i] << "␣";
60      i++;
61    }
62    cout  << endl;
63  }
```

Implementasi Stack pada Array

```
64  int main()
65  {
66    stack values(N);
67
68    cout << "Push:" << endl;
69    for(int i = 0;  !values.is_full(); i++)
70      values.push(rand()%N);
71    values.show_stack();
72
73    cout << "Pop:" << endl;
74    while(!values.is_empty())
75      cout << values.pop() << "␣";
76    cout << endl;
77
78    return 0;
79  }
```

Konsep Dasar Stack

- Implementasi Stack pada Linked-List
- Contoh 9.3.

Implementasi Stack pada Linked-List

```
1  //Contoh9_3.cpp
2  #include <iostream>
3  #include <cstdlib>
4  #include <ctime>
5  #define N 10
6
7  using namespace std;
8
9  struct Node
10 {
11   int value;
12   Node *next;
13 };
```

Implementasi Stack pada Linked-List

```
14  class Stack
15  {
16    private:
17      Node *top;
18      Node *bottom;
19      int isempty;
20    public:
21      Stack() {top = NULL; bottom = NULL;
            isempty = 1;}
22      void push (int value);
23      int pop(void);
24      void show_stack();
25      int is_empty(void) { return (isempty); }
26  };
```

Implementasi Stack pada Linked-List

```
27  void Stack::show_stack()
28  {
29    Node *temp;
30
31    temp = top;
32    while(temp)
33    {
34      cout << temp->value << " ";
35      temp = temp->next;
36    }
37    cout  << endl;
38  }
39
40  void Stack::push(int value)
41  {
42    Node *temp;
```

Implementasi Stack pada Linked-List

```
43      temp = new Node;
44      temp->value = value;
45      temp->next = NULL;
46
47      if (is_empty())
48      {
49        top = temp;
50        bottom = top;
51      }
52      else
53      {
54        temp->next = top;
55        top = temp;
56      }
57      isempty = 0;
58  }
```

Implementasi Stack pada Linked-List

```
59  int Stack::pop(void)
60  {
61    Node *temp;
62    int val=-99;
63
64    if (!is_empty())
65    {
66      temp = top;
67      val = temp->value;
68      if(temp->next)
69      {
70        temp = temp->next;
71        top->next = NULL;
72        delete top;
73        top = temp;
74      }
```

Implementasi Stack pada Linked-List

```
75        else
76        {
77          top->next = NULL;
78          delete top;
79          top = NULL;
80          bottom = NULL;
81          isempty = 1;
82        }
83      }
84      return val;
85    }
```

Implementasi Stack pada Linked-List

```
86  int main()
87  {
88    Stack myStack;
89
90    cout << "Push:" << endl;
91    for(int i = 0;  i<N; i++)
92      myStack.push(rand()%N);
93    myStack.show_stack();
94
95    cout << "Pop:" << endl;
96    while(!myStack.is_empty())
97      cout << myStack.pop() << "␣";
98    cout << endl;
99
100   return 0;
101 }
```

Konsep Dasar Queue

- Queue adalah sebuah struktur data abstrak yang menirukan prinsip antrian.
- Queue menggunakan struktur First In First Out (FIFO).
- Operasi pada elemen-elemen queue dilakukan dari dua ujung pertama dan terakhir.
- Queue memiliki dua operasi dasar, yaitu enqueue dan dequeue.
- Enqueue() menambahkan elemen ke dalam antrian, sedangkan dequeue() mengeluarkan elemen dari antrian.
- Elemen yang pertama ditambahkan merupakan elemen pertama yang keluar dari antrian.
- Contoh Program 9.4:

Implementasi Queue pada Array

```
1  //Contoh9_4.cpp
2  #include <iostream>
3  #include <cstdlib>
4  #include <ctime>
5  #define N 10
6
7  using namespace std;
8
9  class Queue
10 {
11   private:
12     int *storage;
13     int first;
14     int last;
15     int isempty;
```

Implementasi Queue pada Array

```
16      int isfull;
17      int q_size;
18    public:
19      Queue(int size);
20      void enqueue(int value);
21      int dequeue(void);
22      void show_queue();
23      int getLast(void) {return last;}
24      int getFirst(void) {return first;}
25      int is_empty(void) { return isempty; }
26      int is_full(void) { return isfull;}
27      ~Queue(){delete [] storage;}
28  };
```

Implementasi Queue pada Array

```
29  Queue::Queue(int size)
30  {
31    srand(time(NULL));
32    storage = new int[size];
33    first = 0;
34    last = 0;
35    isempty = 1;
36    isfull = 0;
37    q_size = size;
38  }
```

Implementasi Queue pada Array

```
39  void Queue::show_queue()
40  {
41    int i=first;
42
43    while(i<q_size)
44      cout << storage[i++] << " ";
45    cout  << endl;
46  }
```

Implementasi Queue pada Array

```
47  int Queue::dequeue(void)
48  {
49    if (!is_empty())
50    {
51      if (first == q_size)
52      {
53        isempty = 1;
54        isfull = 0;
55        first = 0;
56        last = 0;
57        return isempty;
58      }
59      else return storage[first++];
60    }
61    return 0;
62  }
```

Implementasi Queue pada Array

```
63  void Queue::enqueue(int value)
64  {
65    if (is_full())
66      return 0;
67    else
68    {
69      storage[last++] = value;
70      if (last == q_size-1)
71        isfull = 1;
72      isempty = 0;
73    }
74  }
```

Implementasi Queue pada Array

```
75  int main()
76  {
77    Queue values(N);
78
79    cout << "Enqueue:" << endl;
80    while(!values.is_full())
81      values.enqueue(rand()%N);
82    values.show_queue();
83    cout << "Queue is full... " << endl;
84    cout << "\nDequeue:" << endl;
85    for(int i=0; i < N; i++)
86        cout << values.dequeue() << " ";
87    cout << "\nQueue is empty..." << endl;
88    return 0;
89  }
```

Implementasi Queue pada Linked-List

```
1  //Contoh9_5.cpp
2  #include <iostream>
3  #include <cstdlib>
4  #include <ctime>
5  #define N 15
6
7  using namespace std;
8
9  struct Node
10 {
11   int val;
12   Node *next;
13 };
```

Implementasi Queue pada Linked-List

```
14  class Queue
15  {
16    private:
17      Node *first;
18      Node *last;
19      int empty;
20    public:
21      Queue() {first=NULL; empty=1;}
22      void enqueue(int);
23      int dequeue();
24      void print();
25      int is_empty(){return empty;}
26      ~Queue(){}
27  };
```

Implementasi Queue pada Linked-List

```
28  void Queue::enqueue(int x)
29  {
30    Node *temp;
31    temp = new Node;
32    temp->val = x;
33    temp->next = NULL;
34    if(first==NULL)
35      first = temp;
36    else
37    {
38      last = first;
39      while(last->next!=NULL)
40        last = last->next;
41      last->next = temp;
42    }
43    empty=0;
44  }
```

Implementasi Queue pada Linked-List

```
45  int Queue::dequeue()
46  {
47    if(first == NULL)
48    {
49      cout << "Queue Kosong" << endl;
50      return -1;
51    }
52    else
53    {
54      int r = first->val;
55      first = first->next;
56      if(first==NULL)
57        empty = 1;
58      return r;
59    }
60  }
```

Implementasi Queue pada Linked-List

```
61  void Queue::print()
62  {
63    last = first;
64    while(last != NULL)
65    {
66      cout << last->val << "␣";
67      last = last->next;
68    }
69  }
```

Implementasi Queue pada Linked-List

```
70  int main()
71  {
72    Queue q;
73
74    srand(time(NULL));
75    cout << "Entering␣Q..." << endl;
76    for(int i=0; i<N; i++)
77      q.enqueue(rand()%N);
78    q.print();
79    cout << endl;
80    cout << "Exiting␣Q..." << endl;
81    while(!q.is_empty())
82       cout << q.dequeue() << "␣";
83    cout << endl;
84    return 0;
85  }
```