

PRAKTIKUM 5

STACK & QUEUE

A. Tujuan

Setelah praktikum ini, praktikan diharapkan dapat:

1. Memahami tipe data dasar
2. Memahami tipe data bentukan
3. Struktur Program menggunakan bahasa C++

B. Peralatan

1. PC Desktop
2. Windows 7
3. Notepad++ dan MinGW atau Dev++

C. Stack

Stack adalah sebuah kumpulan data dimana data yang diletakkan diatas data yang lain. Proses penambahan dan penghapusan data selalu dilakukan pada bagian akhir data, yang disebut dengan **top of stack**.

Dengan demikian stack adalah struktur data yang menggunakan konsep LIFO (Last In First Out).

D. Queue

Queue atau antrian merupakan kumpulan data dimana penambahan dan pengambilannya melalui dua jalan yang berbeda. Penambahan elemen hanya bisa dilakukan pada suatu ujung yang disebut dengan sisi belakang (rear), dan penghapusan (pengambilan elemen) dilakukan lewat ujung lain (disebut dengan sisi depan atau front). Contoh *queue* dalam kehidupan sehari – hari misalnya antrian pembayaran di kasir, antrian mobil saat pengisian BBM di SPBU, dll.

Sebagai ilustrasi saat pengisian BBM di SPBU, maka mobil yang datang pertamalah yang akan diisi pertama, demikian seterusnya sampai yang mendapat giliran terakhir adalah yang datang terakhir. Oleh karena itu, struktur data ini mencerminkan konsep antrian yang sering kita alami di dunia nyata.

Hal yang sama juga berlaku pada data, yaitu data yang masuk pertama akan keluar pertama juga dan data yang terakhir masuk akan keluar terakhir. Sifat ini sering disebut dengan istilah FIFO (*First In First Out*).

D. Praktikum

Percobaan 1

Berikut adalah contoh program yang menerapkan konsep stack menggunakan array.

```
1  #include <iostream>
2  using namespace std;
3
4  int maksimal = 5;
5  string arrayBuku[5];
6  int top = 0;
7
8  bool isFull()
9  {
10     if( top == maksimal ){
11         return true;
12     }else{
13         return false;
14     }
15 }
16
17 bool isEmpty()
18 {
19     if( top == 0 ){
20         return true;
21     }else{
22         return false;
23     }
24 }
25
26 void pushArray(string data){
27     if( isFull() ){
28         cout << "Data penuh" << endl;
29     }else{
30         arrayBuku[top] = data;
31         top++;
32     }
33 }
34
35 void popArray()
36 {
37     if( isEmpty() ){
38         cout << "Data kosong!!" << endl;
39     }else{
40         arrayBuku[top-1] = "";
41         top--;
42     }
43 }
```

```

45 void displayArray(){
46     if( isEmpty() ){
47         cout << "Data kosong!!" << endl;
48     }else{
49         cout << "Data stack array : " << endl;
50         cout << "-----" << endl;
51         for( int i = maksimal - 1; i >= 0; i-- ){
52             if( arrayBuku[i] != "" ){
53                 cout << "Stack " << i << ": " << arrayBuku[i] << endl;
54             }
55         }
56         cout << "\n" << endl;
57     }
58 }
59
60 void destroyArray(){
61     for( int i = 0; i < top; i++ ){
62         arrayBuku[i] = "";
63     }
64     top = 0;
65 }
66
67 int main(){
68
69     pushArray("Delapan");
70     displayArray();
71
72     pushArray("Sembilan");
73     pushArray("Tiga");
74     displayArray();
75
76     popArray();
77     displayArray();
78
79     pushArray("Lima");
80     pushArray("Enam");
81     pushArray("Empat");
82     pushArray("Tujuh");
83     displayArray();
84
85     popArray();
86     displayArray();
87
88     popArray();
89     displayArray();
90
91     cout << "Apakah data full ? : " << isFull() << endl;
92     cout << "Apakah data kosong ? : " << isEmpty() << endl<< endl;
93
94     destroyArray();
95
96     cout << "Setelah di clear " << endl;
97     cout << "Apakah data full ? : " << isFull() << endl;
98     cout << "Apakah data kosong ? : " << isEmpty() << endl;
99
100 }

```

Percobaan 2

Berikut adalah contoh program yang menerapkan konsep stack menggunakan struct.

```
1 #include <iostream>
2
3 using namespace std;
4
5 const int MAX_STACK = 5;
6
7 struct Stack
8 {
9     int top;
10    int data[MAX_STACK];
11 };
12
13 // prototype fungsi stack
14 void inisialisasi();
15 void push(int data); // menambahkan item pada stack
16 void pop(); // menghapus item pada stack
17 void clear(); // mengosongkan stack
18 bool isEmpty(); // untuk mengecek apakah stack kosong
19 bool isFull(); // untuk mengecek apakah stack penuh
20 void print(); // mencetak item stack
21
22 Stack stack; // deklarasi var stack dg tipe struct Stack
23
24 int main()
25 {
26     // inisialisasi
27     inisialisasi();
28
29     int pilihanMenu;
30     int data;
31
32     do
33     {
34         cout << ">>> PILIHAN MENU STACK <<<" << endl << endl;
35         cout << "1. Menambah item stack" << endl;
36         cout << "2. Menghapus item stack" << endl;
37         cout << "3. Menampilkan item stack" << endl;
38         cout << "4. Mengosongkan stack" << endl;
39         cout << "5. Selesai" << endl << endl;
40
41         cout << "Masukkan pilihan Anda : "; cin >> pilihanMenu;
42         cout << endl;
43     }
```

```

44     switch (pilihanMenu)
45     {
46         case 1: // menambah item stack
47             cout << "Masukkan data : "; cin >> data;
48             push(data);
49             break;
50
51         case 2: // menghapus item stack
52             pop();
53             break;
54
55         case 3: // menampilkan stack
56             print();
57             break;
58
59         case 4: // mengosongkan stack
60             clear();
61             break;
62     }
63
64     } while (pilihanMenu != 5);
65
66     cout << endl;
67
68     system("pause");
69     return 0;
70 }
71
72 // definisi fungsi stack
73 void inisialisasi()
74 {
75     stack.top = -1;
76 }
77
78 void push(int data)
79 {
80     stack.top++;
81     stack.data[stack.top] = data;
82     cout << "Data berhasil ditambahkan" << endl << endl;
83 }
84
85 void pop()
86 {
87     cout << "Data " << stack.data[stack.top] << " sudah dihapus" << endl << endl;
88     stack.top--;
89 }
90
91 void clear()
92 {
93     stack.top = -1;
94     cout << "Stack sudah dikosongkan" << endl << endl;
95 }
96
97 bool isEmpty()
98 {
99     return (stack.top == -1);
100 }

```

```

101
102 bool isFull()
103 {
104     return (stack.top >= (MAX_STACK - 1));
105 }
106
107 void print()
108 {
109     cout << "Isi stack :" << endl << endl;
110     for (int i = stack.top; i >= 0; i--)
111     {
112         cout << stack.data[i] << endl;
113     }
114     cout << endl << endl;
115 }

```

Simpan dengan nama **stack-1-xxxx** dan untuk melihat hasilnya klik menu **Execute - > Compile & Run** atau cukup dengan menekan tombol **F9**. Contoh output :

```

E:\Kuliah\Struktur Data\Teori\Src\Stack\stack-1.exe
>>> PILIHAN MENU STACK <<<
1. Menambah item stack
2. Menghapus item stack
3. Menampilkan item stack
4. Mengosongkan stack
5. Selesai
Masukkan pilihan Anda : 1
Masukkan data : 10
Data berhasil ditambahkan
>>> PILIHAN MENU STACK <<<
1. Menambah item stack
2. Menghapus item stack
3. Menampilkan item stack
4. Mengosongkan stack
5. Selesai
Masukkan pilihan Anda : 1
Masukkan data : 20
Data berhasil ditambahkan
>>> PILIHAN MENU STACK <<<
1. Menambah item stack
2. Menghapus item stack
3. Menampilkan item stack
4. Mengosongkan stack
5. Selesai
Masukkan pilihan Anda : 3_

```

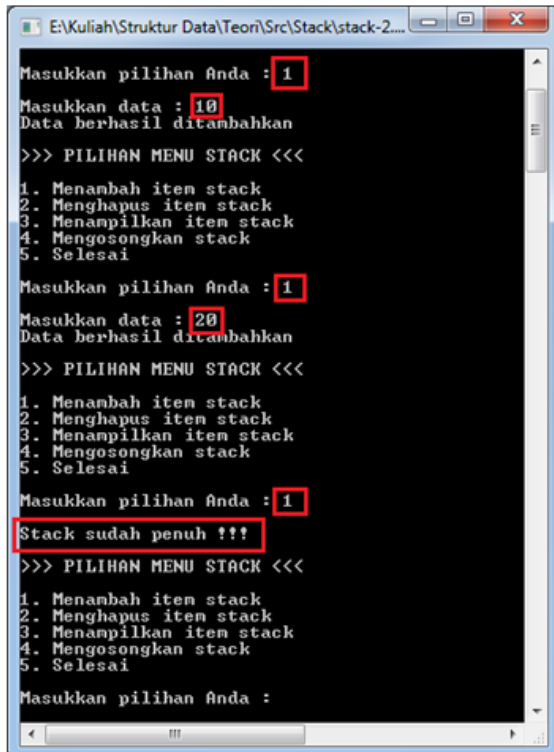
E. Latihan 1

Penambahan validasi pada operasi stack

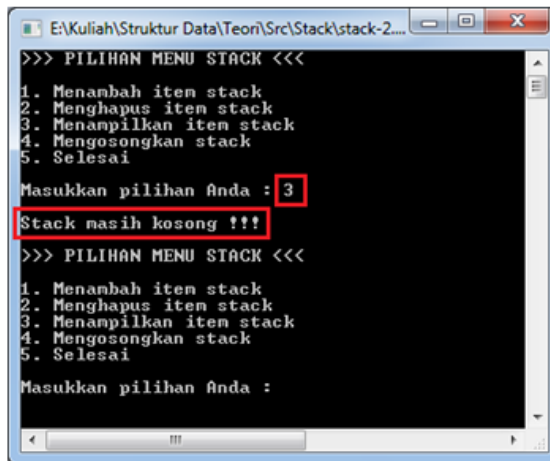
Lakukan revisi pada source code #1, dengan menambahkan validasi berikut :

1. Pada saat menambahkan item stack (pilihan menu 1), jika kondisi stack sudah penuh tampilkan pesan **"Stack sudah penuh !!!"**
2. Pada saat menghapus item stack (pilihan menu 2), menampilkan stack (pilihan menu 3) dan mengosongkan stack (menu 4), jika kondisi stack masih kosong tampilkan pesan **"Stack masih kosong !!!"**

Simpan dengan nama **stack-2-xxxx** dan untuk melihat hasilnya klik menu **Execute - > Compile & Run** atau cukup dengan menekan tombol **F9**. Contoh output :



```
E:\Kuliah\Struktur Data\Teori\Src\Stack\stack-2-xxxx
Masukkan pilihan Anda : 1
Masukkan data : 10
Data berhasil ditambahkan
>>> PILIHAN MENU STACK <<<
1. Menambah item stack
2. Menghapus item stack
3. Menampilkan item stack
4. Mengosongkan stack
5. Selesai
Masukkan pilihan Anda : 1
Masukkan data : 20
Data berhasil ditambahkan
>>> PILIHAN MENU STACK <<<
1. Menambah item stack
2. Menghapus item stack
3. Menampilkan item stack
4. Mengosongkan stack
5. Selesai
Masukkan pilihan Anda : 1
Stack sudah penuh !!!
>>> PILIHAN MENU STACK <<<
1. Menambah item stack
2. Menghapus item stack
3. Menampilkan item stack
4. Mengosongkan stack
5. Selesai
Masukkan pilihan Anda :
```



```
E:\Kuliah\Struktur Data\Teori\Src\Stack\stack-2-xxxx
>>> PILIHAN MENU STACK <<<
1. Menambah item stack
2. Menghapus item stack
3. Menampilkan item stack
4. Mengosongkan stack
5. Selesai
Masukkan pilihan Anda : 3
Stack masih kosong !!!
>>> PILIHAN MENU STACK <<<
1. Menambah item stack
2. Menghapus item stack
3. Menampilkan item stack
4. Mengosongkan stack
5. Selesai
Masukkan pilihan Anda :
```

F. Studi Kasus

Membuat antrian dengan kapasitas 10 elemen.

```
#include <iostream>
#include <string>
#include <vector>

using namespace std;

class Antrian
{
private:
    vector<string> data;
    int depan, belakang;
    int maksElemen;
public:
    // Konstruktor
    Antrian(int ukuran)
    {
        depan = 0;
        belakang = 0;
        maksElemen = ukuran;
        data.resize(ukuran); // Ukuran vector
    }

    // Memasukkan data ke antrian
    // Nilai balik tidak ada
    void insert(string x)
    {
        int posisiBelakang;

        // Geser belakang ke posisi berikutnya
        if (belakang == maksElemen)
            posisiBelakang = 1;
        else
            posisiBelakang = belakang + 1;

        // Cek belakang apa sama dengan Depan
        if (posisiBelakang == depan)
            cout << "Antrian penuh" << endl;
        else
        {
            belakang = posisiBelakang;

            // Masukkan data
            data[belakang] = x;
        }
    }
}
```



```

    }

    string remove(void)
    {
        if (empty())
        {
            cout << "Antrian kosong" << endl;
            return "";
        }

        if (depan == maksElemen)
            depan = 1;
        else
            depan = depan + 1;

        return data[depan];
    }

    bool empty(void)
    {
        if (depan == belakang)
            return true;
        else
            return false;
    }
};

int main()
{
    int ukuran = 10;
    Antrian daftar(ukuran); // Buat objek

    // Masukkan 5 buah nama
    daftar.insert("Aman");
    daftar.insert("Budi");
    daftar.insert("Caca");
    daftar.insert("Didi");
    daftar.insert("Edi");

    // Kosongkan isi antrian dan tampilkan
    while (!daftar.empty())
    {
        string nama = daftar.remove();
        cout << nama << endl;
    }
}

```

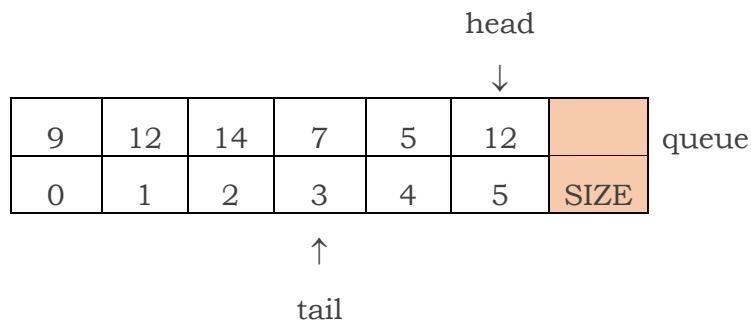
```

    return 0;
}

```

G. Latihan 2

1. Ilustrasikan antrian untuk daftar pemanggilan operasi berikut:
 store('Ana')
 store('Bina')
 store('Cika')
 retrieve
 store('Dana')
 retrieve
 store('Eka')
 store('Fina')
 retrieve
2. Saat kita mengalokasikan antrian menggunakan array dengan jumlah n elemen maka data yang dapat kita isikan hanya n-1 saja, mengapa?
3. Terdapat ilustrasi antrian sebagai berikut :



Sebutkan jumlah dari data antrian diatas dan bagaimana urutan antriannya?
 Apabila ditambahkan operasi remove diakhir proses maka gambarlah ilustrasi hasil akhirnya.

4. Apakah memungkinkan dilakukan penambahan elemen didepan? Mengapa?
5. Modifikasi source code antrian diatas supaya dapat melakukan penambahan dan penghapusan elemen saat program berjalan.