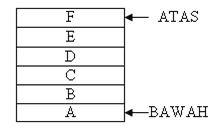
**BAB VII**

**STACK AND QUEUE**

* 1. Tujuan

1. Praktikan dapat memahami apa itu *stack* dan *queue.*
2. Praktikan dapat membedakan antara *stack* dan *queue*.
3. Praktikan dapat memahami dasar penggunaan antara *stack* dan *queue* di masing-masing bahasa pemrograman C#, Java dan PHP.
4. Praktikan mampu memahami setiap *source code* yang ada untuk menciptakan *stack* dan *queue* pada setiap bahasa pemrograman C#, Java dan PHP.
5. Praktikan mampu membuat program sederhana *stack* dan *queue* menggunakan bahasa pemrograman C#, Java dan PHP.
   1. Dasar Teori
      1. **Stack**

Pengertian [*Stack*](http://blog-arul.blogspot.com/2012/01/stack-queue-tree.html) adalah sebagai tumpukan dari benda, sekumpulan data yang seolah-olah diletakkan di atas data yang lain, koleksi dari objek-objek homogen, atau Suatu urutan elemen yang elemennya dapat diambil dan ditambah hanya pada posisi akhir (*top*) saja. [*Stack*](http://blog-arul.blogspot.com/2012/01/stack-queue-tree.html) pada [Struktur Data](http://blog-arul.blogspot.com/2012/01/pengertian-struktur-data.html) dapat diilustrasikan dengan dua buah kotak yang ditumpuk, kotak yang satu akan ditumpuk diatas kotak yang lainnya. Jika kemudian stack 2 kotak tadi, ditambah kotak ketiga, keempat, kelima, dan seterusnya, maka akan diperoleh sebuah stack kotak yang terdiri dari N kotak.

[](http://4.bp.blogspot.com/-T-SHVTTKhZ8/TyJFPHV8JAI/AAAAAAAAAJ0/OaArEc0-1wU/s1600/Picture1.png)

*Stack* bersifat LIFO (*Last In First Out*) artinya Benda yang terakhir masuk ke dalam *stack* akan menjadi yang pertama keluar dari *stack.*

Operasi-operasi yang biasanya terdapat pada *Stack* yaitu:

1. *Push* : digunakan untuk menambah item pada *stack* pada tumpukan paling atas.

2. *Pop* : digunakan untuk mengambil item pada *stack* pada tumpukan paling atas.

3. *Clear* : digunakan untuk mengosongkan *stack.*

4. *IsEmpty* : fungsi yang digunakan untuk mengecek apakah *stack* sudah kosong.

5. *IsFull* : fungsi yang digunakan untuk mengecek apakah *stack* sudah penuh.

Cara mendefenisikan *Stack* dengan *Array of Struct* yaitu:

1. Definisikan *Stack* dengan menggunakan *struct*

2. Definisikan konstanta MAX\_STACK untuk menyimpan maksimum isi *stack*

3. Buatlah variabel *array* data sebagai implementasi *stack*

4. Deklarasikan operasi-operasi/*function* di atas dan buat implemetasinya.

contoh :

//Deklarasi MAX\_STACK

                #define MAX\_STACK 10

//Deklarasi STACK dengan struct dan array data

                typedef struct STACK{

                                int top;

                                char data[10][10];

                };

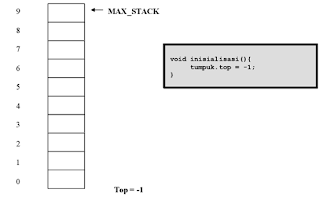
//Deklarasi/buat variabel dari struct

                STACK tumpuk;

Inisialisasi *Stack*

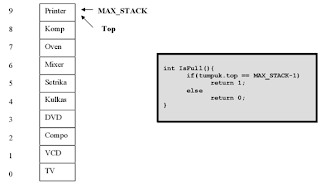
Pada mulanya isi top dengan -1, karena array dalam C dimulai dari 0, yang berarti *stack* adalah kosong.

*Top* adalah suatu variabel penanda dalam *STACK* yang menunjukkan elemen teratas *Stack* sekarang.  *Top Of Stack* akan selalu bergerak hingga mencapai *MAX of STACK* sehingga menyebabkan *stack* penuh.

[](http://2.bp.blogspot.com/-dorZWQ_e6-k/TyJIAh7JloI/AAAAAAAAAJ8/mfnpZv1jZZ8/s1600/Picture2.png)

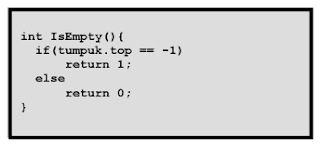
Ilustrasi Stack pada saat inisialisasi

*IsFull* berfungsi untuk memeriksa apakah stack sudah penuh atau tidak. Dengan cara, memeriksa top of stack, jika sudah sama dengan MAX\_STACK-1 maka full, jika belum (masih lebih kecil dari MAX\_STACK-1)  maka belum full.

[](http://1.bp.blogspot.com/-HgoWwQM46qA/TyJUQWzcUoI/AAAAAAAAAKE/qWhOyBAFW8w/s1600/Picture3.png)

Ilustrasi Stack pada kondisi Full

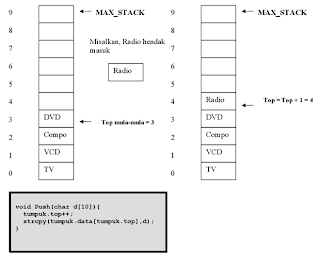
*IsEmpty* berfungsi untuk memeriksa apakah stack masih kosong atau tidak. Dengan cara memeriksa top of stack, jika masih -1 maka berarti stack masih kosong.

[](http://3.bp.blogspot.com/-ZS6LJG2C8Yc/TyJWL49TNhI/AAAAAAAAAKM/OdLZAi5CSEU/s1600/Picture4.png)

*Push* berfungsi untuk memasukkan elemen ke stack, selalu menjadi elemen teratas stack (yang ditunjuk oleh TOS).

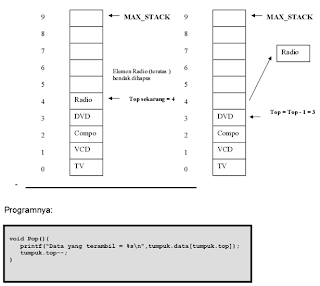
Tambah satu (increment)  nilai top of stack lebih dahulu setiap kali ada penambahan elemen stack.

Asalkan stack masih belum penuh, isikan data baru ke stack berdasarkan indeks top of stack setelah diincrement sebelumnya.

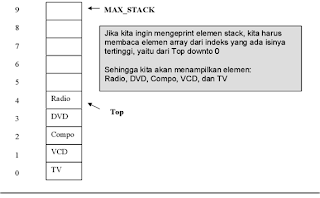
[](http://4.bp.blogspot.com/-6_Htwoc5MvI/TyJf-QUAHtI/AAAAAAAAAKU/K3EXJwm_Uko/s1600/Picture5.png)

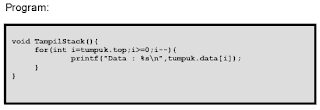
*Pop* berfungsi untuk mengambil elemen teratas (data yang ditunjuk oleh TOS) dari stack.

Ambil dahulu nilai elemen teratas stack dengan mengakses top of stack, tampilkan nilai yang akan dipop, baru dilakukan decrement nilai top of stack sehingga jumlah elemen stack berkurang.

[](http://3.bp.blogspot.com/-60G8G9aeDFg/TyJgXkEbqXI/AAAAAAAAAKc/fJRvTQp6DRQ/s1600/Picture6.png)

*Print* berfungsi untuk menampilkan semua elemen-elemen stack dengan cara looping semua nilai array secara terbalik, karena kita harus mengakses dari indeks array tertinggi terlebih dahulu baru ke indeks yang kecil.

[](http://4.bp.blogspot.com/-Rwk_6qYhzjw/TyJhCLiyGCI/AAAAAAAAAKk/qOlBZ2thVyk/s1600/Picture7.png)

[](http://3.bp.blogspot.com/--_0gZkB_Pbk/TyJhLiSvDBI/AAAAAAAAAKs/kRZXS9xtwiM/s1600/Picture8.png)

**Operasi Push**

void Push (NOD \*\*T, char item)

      {

        NOD \*n;

        n=NodBaru (item);

        n->next=\*T;

       \*T=n;

             }

**Operasi Pop**

char Pop (NOD \*\*T)

     {

         NOD \*n; char item;

         if (!StackKosong(\*T)) {

             P=\*T;

             \*T=(\*T)->next;

             item=P->data;

             free(P);

          }

          return item;

      }

*create* berfungsi untuk membuat sebuah stack baru yang masih kosong

*(sumber :* [*http://blog-arul.blogspot.com/2012/01/stack-pada-struktur-data.html*](http://blog-arul.blogspot.com/2012/01/stack-pada-struktur-data.html)*)*

* + 1. **Queue**

[*Queue*](http://blog-arul.blogspot.com/2012/01/stack-queue-tree.html) atau antrian adalah sekumpulan data yang mana penambahan elemen hanya bisa dilakukan pada suatu ujung disebut dengan sisi belakang (*rear*), dan penghapusan (pengambilan elemen) dilakukan lewat ujung lain (disebut dengan sisi depan atau *front*).

Pada [*Stack*](http://blog-arul.blogspot.com/2012/01/stack-pada-struktur-data.html) atau tumpukan menggunakan prinsip “Masuk terakhir keluar pertama” atau LIFO (*Last In First Out*), Maka pada [*Queue*](http://blog-arul.blogspot.com/2012/01/stack-queue-tree.html) atau antrian prinsip yang digunakan adalah “Masuk Pertama Keluar Pertama” atau FIFO (*First In First Out*).

[*Queue*](http://blog-arul.blogspot.com/2012/01/stack-queue-tree.html) atau antrian banyak kita jumpai dalam kehidupan sehari-hari, ex: antrian Mobil diloket Tol, Antrian mahasiswa Mendaftar, dll.

Contoh lain dalam bidang komputer adalah pemakaian sistem komputer berbagi waktu(*time-sharing computer system*) dimana ada sejumlah pemakai yang akan menggunakan sistem tersebut secara serempak.

Pada [*Queue*](http://blog-arul.blogspot.com/2012/01/stack-queue-tree.html) atau antrian Terdapat satu buah pintu masuk di suatu ujung dan satu buah pintu keluar di ujung satunya dimana membutuhkan variabel *Head* dan *Tail* ( depan/*front*, belakang/*rear*).

Karakteristik [*Queue*](http://blog-arul.blogspot.com/2012/01/stack-queue-tree.html) atau antrian :

1. elemen antrian

2. *front* (elemen terdepan antrian)

3. *tail* (elemen terakhir)

4. jumlah elemen pada antrian

5. status antrian

Operasi pada [*Queue*](http://blog-arul.blogspot.com/2012/01/stack-queue-tree.html) atau antrian

1. tambah (menambah item pada belakang antrian)

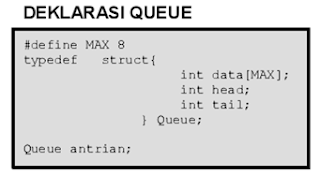
2. hapus (menghapus elemen depan dari antrian)

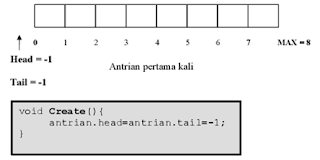
3. kosong (mendeteksi apakah pada antrian mengandung elemen atau tidak)

Operasi-operasi [*Queue*](http://blog-arul.blogspot.com/2012/01/stack-queue-tree.html) :

1. Create()

Untuk menciptakan dan menginisialisasi *Queue* Dengan cara membuat *Head* dan *Tail*  = -1

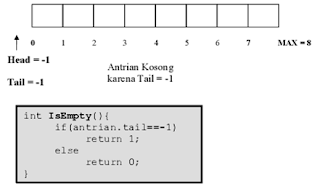
[](https://4.bp.blogspot.com/-hC7nluM_Ti8/TyOBKRKzhGI/AAAAAAAAAK0/vi2x0TBN0Gk/s1600/Picture1.png)

[](https://4.bp.blogspot.com/-Ea-PJoxWVnI/TyOBW6F_N5I/AAAAAAAAAK8/td_YKfaGol8/s1600/Picture2.png)

2. IsEmpty()

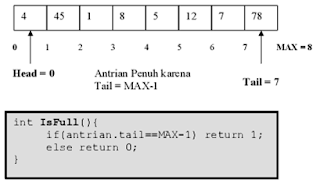
Untuk memeriksa apakah Antrian sudah penuh atau belum Dengan cara memeriksa nilai *Tail*, jika *Tail* = -1 maka *empty* Kita tidak memeriksa *Head*, karena *Head* adalah tanda untuk kepala antrian (elemen pertama dalam antrian) yang tidak akan berubah-ubah.

Pergerakan pada Antrian terjadi dengan penambahan elemen Antrian kebelakang, yaitu menggunakan nilai *Tail*.

[](https://1.bp.blogspot.com/-aWu4p-roOiI/TyOB5dCvL3I/AAAAAAAAALE/q-7erLuj0zU/s1600/Picture3.png)

3. IsFull

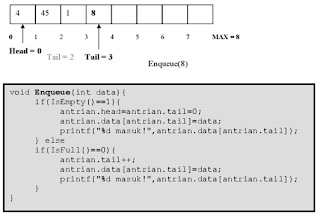
Untuk mengecek apakah Antrian sudah penuh atau belum Dengan cara mengecek nilai *Tail*, jika *Tail* >= MAX-1 (karena MAX-1 adalah batas elemen array pada C) berarti sudah penuh

[](https://4.bp.blogspot.com/-K5Hykpcgat4/TyOCNlRF7vI/AAAAAAAAALM/NIY1W1qBV0k/s1600/Picture4.png)

4. Enqueue

Untuk menambahkan elemen ke dalam Antrian, penambahan elemen selalu ditambahkan di elemen paling belakang.

Penambahan elemen selalu menggerakan variabel *Tail* dengan cara *increment counter Tail* terlebih dahulu

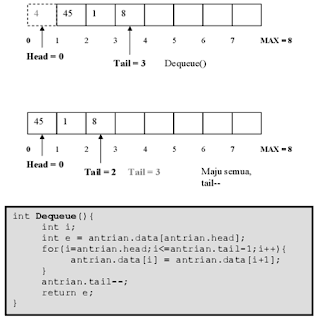
[](https://3.bp.blogspot.com/-PMEARXIXkNM/TyOClkIIF3I/AAAAAAAAALU/EwpVm82Gg8U/s1600/Picture5.png)

5. Dequeue()

Digunakan untuk menghapus elemen terdepan/pertama (*head*) dari Antrian

Dengan cara menggeser semua elemen antrian kedepan dan mengurangi *Tail* dgn 1

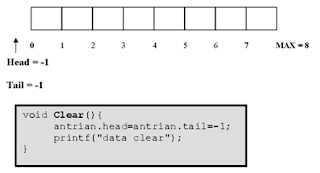
Penggeseran dilakukan dengan menggunakan *looping.*

[](https://1.bp.blogspot.com/-YCne1JWAjyI/TyODEm7T4jI/AAAAAAAAALc/j3-pSsjDEHw/s1600/Picture6.png)

6. Clear()

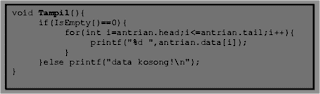
Untuk menghapus elemen-elemen Antrian dengan cara membuat *Tail* dan *Head* = -1

Penghapusan elemen-elemen Antrian sebenarnya tidak menghapus arraynya, namun hanya mengeset indeks pengaksesan-nya ke nilai -1 sehingga elemen-elemen Antrian tidak lagi terbaca

[](https://2.bp.blogspot.com/-xBVkqFuwKo8/TyODdbwpfwI/AAAAAAAAALk/2peRJ-aGoSU/s1600/Picture8.png)

7. Tampil()

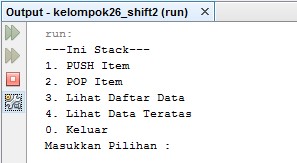
Untuk menampilkan nilai-nilai elemen Antrian Menggunakan *looping* dari *head* s/d *tail*

[](https://4.bp.blogspot.com/-iupXLXpNb1o/TyOEDpnwolI/AAAAAAAAALs/uRYXmc_c3OM/s1600/Picture9.png)

*(sumber :*[*https://blog-arul.blogspot.com/2012/01/queue-pada-struktur-data.html*](https://blog-arul.blogspot.com/2012/01/queue-pada-struktur-data.html)*)*

* 1. Hasil Percobaan dan Analisis
     1. **Stack pada Java**

|  |
| --- |
| package kelompok26\_shift2;  import java.util.Scanner;  import java.util.Stack;  public class Kelompok26\_shift2 {  public static void main(String[] args) {  int pilihan;  int data;  Stack result = new Stack();  //looping until false condition  do{  //Displaying Menu  System.out.println("---Ini Stack---");  System.out.println("1. PUSH Item");  System.out.println("2. POP Item");  System.out.println("3. Lihat Daftar Data");  System.out.println("4. Lihat Data Teratas");  System.out.println("0. Keluar");  Scanner input = new Scanner(System.in);  System.out.print("Masukkan Pilihan : ");  pilihan = input.nextInt();  //condition for choice  if(pilihan==1){  System.out.print("Data yang ditambahkan : ");  data = input.nextInt();  result.push(data);  System.out.println("");  }  else if(pilihan==2){  result.pop();  System.out.println("");  }  else if(pilihan==3){  System.out.print(result + " ");  System.out.println("");  }  else if(pilihan==4){  System.out.println("Data teratas : "+result.peek());  System.out.println("");  }  else if(pilihan==0){  System.exit(0);  }  else{  System.out.println("Pilihan Tidak Ada!!");  }//end of condition  }while(pilihan!=0);//end looping  }  } |



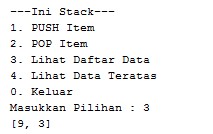
Gambar 7.1 Tampilan *stack* pada Java

C:\Users\Wahyu\Documents\Lightshot\Modul 6\Screenshot_2.jpg

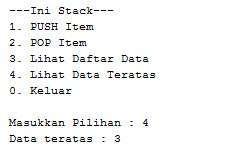
Gambar 7.2 *push* item

C:\Users\Wahyu\Documents\Lightshot\Modul 6\Screenshot_3.jpg

Gambar 7.3 *push* item



Gambar 7.4 Lihat daftar data

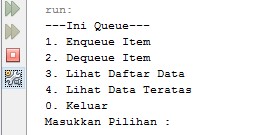


Gambar 7.5 Lihat data teratas

Untuk menggunakan *stack* dibutuhkan *library* dari java dengan cara import java.util.Stack, lalu deklarasikan variabel yang digunakan. Lalu membuat objek baru dari *class stack* yang di*import* Stack result = new Stack(). Setelah itu digunakan perulanagan untuk membuat tampilan menu yang hasilnya seperti gambar 7.1, gunakan *scanner* untuk mengambil nilai *input* dan masukkan pada variabel pilihan. Setelah itu gunakan perkondisian *if* untuk menjalankan kode sesuai pilihan pada menu. Jika pilihannya 1 maka akan diminta untuk memasukkan nilai yang nanti disimpan di variabel data, lalu variabel tersebut akan di-*push* pada *stack* seperti gambar 7.2*.* Jika pilihannya 2 maka data akan di­-*pop* dari *stack.* Jika pilihannya 3 maka akan menampilkan semua data yang ada pada *stack* seperti gambar 7.4*.* Jika pilihannya 4 maka akan menampilkan data teratas(*peek*) dari *stack* seperti gambar 7.5*.* Lalu jika pilihannya 0 program akan berhenti. Kemudian jika semua kondisi tidak terpenuhi akan ditampilkan pesan “Pilihan Tidak Ada!!” program akan terus diulang selama pilihan tidak sama dengan 0.

* + 1. **Queue pada Java**

|  |
| --- |
| package kelompok26\_queue;  import java.util.Arrays;  import java.util.LinkedList;  import java.util.Queue;  import java.util.Scanner;  public class Kelompok26\_queue {  public static void main(String[] args) {  int pilihan;  int data;  Queue<Integer> antrian = new LinkedList<Integer>();  //looping until false condition  do{  //Displaying Menu  System.out.println ("---Ini Queue---");  System.out.println("1. Enqueue Item");  System.out.println("2. Dequeue Item");  System.out.println("3. Lihat Daftar Data");  System.out.println("4. Lihat Data Teratas");  System.out.println("0. Keluar");  Scanner input = new Scanner(System.in);  System.out.print("Masukkan Pilihan : ");  pilihan = input.nextInt();  //condition for choice  if(pilihan==1){  System.out.print("Data yang ditambahkan : ");  data = input.nextInt();  antrian.add(data);  System.out.println("");  }  else if(pilihan==2){  antrian.poll();  System.out.println("");  }  else if(pilihan==3){  System.out.println(antrian+ " ");  System.out.println("");  }  else if(pilihan==4){  System.out.println("Data teratas : "+ antrian.peek());  System.out.println("");  }  else if(pilihan==0){  System.exit(0);  }  else{  System.out.println("Pilihan Tidak Ada!!");  System.out.println("");  }//end of condition  } while(pilihan!=0);//end looping  }  } |

****

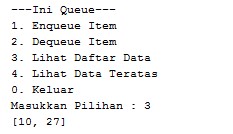
Gambar 7.6 Tampilan *queue* pada Java

**C:\Users\Wahyu\Documents\Lightshot\Modul 6\Screenshot_7.jpg**

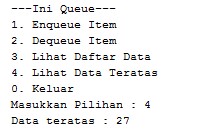
Gambar 7.7 *Enqueue* item

**C:\Users\Wahyu\Documents\Lightshot\Modul 6\Screenshot_8.jpg**

Gambar 7.8 *Enqueue* item

****

Gambar 7.9 Lihat daftar data

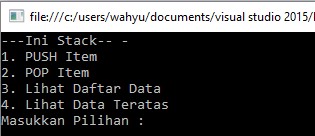
****

Gambar 7.10 Lihat data teratas

Untuk menggunakan *queue* dibutuhkan *library* dari java dengan cara import java.util.Queue, import java.util.LinkedList, lalu deklarasikan variabel yang digunakan. Lalu membuat objek baru dari *class queue* yang di*import* Queue<Integer> antrian = new LinkedList<Integer>(). Setelah itu digunakan perulanagan untuk membuat tampilan menu yang hasilnya seperti gambar 7.6, gunakan *scanner* untuk mengambil nilai *input* dan masukkan pada variabel pilihan. Setelah itu gunakan perkondisian *if* untuk menjalankan kode sesuai pilihan pada menu. Jika pilihannya 1 maka akan diminta untuk memasukkan nilai yang nanti disimpan di variabel data, lalu variabel tersebut akan di-*enqueue* pada *queue* seperti gambar 7.7*.* Jika pilihannya 2 maka data akan di­-*dequeue* dari *queue.* Jika pilihannya 3 maka akan menampilkan semua data yang ada pada *queue* seperti gambar 7.9*.* Jika pilihannya 4 maka akan menampilkan data teratas(*peek*) dari *queue* seperti gambar 7.10*.* Lalu jika pilihannya 0 program akan berhenti. Kemudian jika semua kondisi tidak terpenuhi akan ditampilkan pesan “Pilihan Tidak Ada!!” program akan terus diulang selama pilihan tidak sama dengan 0.

* + 1. **Stack pada C#**

|  |
| --- |
| using System;  using System.Collections;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace kelompok26\_stack  {  class Program  {  static void Main(string[] args)  {  int pilihan;  int data;  Stack tumpukan = new Stack();  //looping until false condition  do  {  //Displaying Menu  Console.WriteLine("---Ini Stack-- -");  Console.WriteLine("1. PUSH Item");  Console.WriteLine("2. POP Item");  Console.WriteLine("3. Lihat Daftar Data");  Console.WriteLine("4. Lihat Data Teratas");  Console.Write("Masukkan Pilihan : ");  pilihan = int.Parse(Console.ReadLine());  //condition for choice  if (pilihan == 1)  {  Console.Write("Data yang ditambahkan : ");  data = int.Parse(Console.ReadLine());  tumpukan.Push(data);  Console.WriteLine("");  }  else if (pilihan == 2)  {  tumpukan.Pop();  Console.WriteLine("");  }  else if (pilihan == 3)  {  foreach (int stack in tumpukan)  {  Console.WriteLine(stack);  }  Console.WriteLine("\n");  }  else if (pilihan == 4)  {  Console.WriteLine("Data teratas : " + tumpukan.Peek());  Console.WriteLine("");  }  else  {  Console.WriteLine("Pilihan Tidak Ada!!");  }//end of condition  } while (pilihan != 0);//end looping  }  }  } |

****

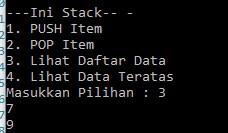
Gambar 7.11 Tampilan *stack* pada C#

**C:\Users\Wahyu\Documents\Lightshot\Modul 6\Screenshot_12.jpg**

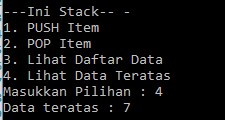
Gambar 7.12 *push* item

**C:\Users\Wahyu\Documents\Lightshot\Modul 6\Screenshot_13.jpg**

Gambar 7.13 *push* item

****

Gambar 7.14 Lihat daftar data

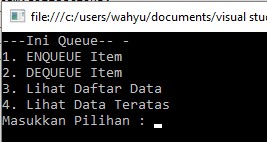
****

Gambar 7.15 Lihat data teratas

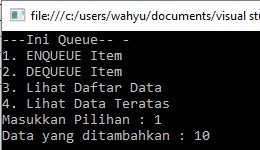
Untuk menggunakan *stack* dibutuhkan *library* dari C# dengan cara using System.Collections, lalu deklarasikan variabel yang digunakan. Lalu membuat objek baru dari *class stack* yang di*import* Stack tumpukan = new Stack(). Setelah itu digunakan perulanagan untuk membuat tampilan menu yang hasilnya seperti gambar 7.11, gunakan int.Parse(Console.ReadLine())untuk mengambil nilai *input* dan masukkan pada variabel pilihan. Setelah itu gunakan perkondisian *if* untuk menjalankan kode sesuai pilihan pada menu. Jika pilihannya 1 maka akan diminta untuk memasukkan nilai yang nanti disimpan di variabel data, lalu variabel tersebut akan di-*push* pada *stack* seperti gambar 7.12*.* Jika pilihannya 2 maka data akan di­-*pop* dari *stack.* Jika pilihannya 3 maka akan menampilkan semua data yang ada pada *stack* dengan perulangan *foreach* seperti gambar 7.14*.* Jika pilihannya 4 maka akan menampilkan data teratas(*peek*) dari *stack* seperti gambar 7.15*.* Kemudian jika semua kondisi tidak terpenuhi akan ditampilkan pesan “Pilihan Tidak Ada!!” program akan terus diulang selama pilihan tidak sama dengan 0.

* + 1. **Queue pada C#**

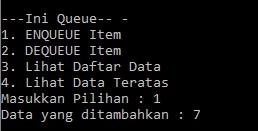
|  |
| --- |
| using System;  using System.Collections;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace kelompok26\_queue  {  class Program  {  static void Main(string[] args)  {  int pilihan;  int data;  Queue antrian = new Queue();  //looping until false condition  do  {  //Displaying Menu  Console.WriteLine("---Ini Queue-- -");  Console.WriteLine("1. ENQUEUE Item");  Console.WriteLine("2. DEQUEUE Item");  Console.WriteLine("3. Lihat Daftar Data");  Console.WriteLine("4. Lihat Data Teratas");  Console.Write("Masukkan Pilihan : ");  pilihan = int.Parse(Console.ReadLine());  //condition for choice  if (pilihan == 1)  {  Console.Write("Data yang ditambahkan : ");  data = int.Parse(Console.ReadLine());  antrian.Enqueue(data);  Console.WriteLine("");  }  else if (pilihan == 2)  {  antrian.Dequeue();  Console.WriteLine("");  }  else if (pilihan == 3)  {  foreach (int value in antrian)  {  Console.Write(value + " ");  }  Console.WriteLine("\n");  }  else if (pilihan == 4)  {  Console.WriteLine("Data teratas : " + antrian.Peek());  Console.WriteLine("");  }  else  {  Console.WriteLine("Pilihan Tidak Ada!!");  }//end of condition  } while (pilihan != 0);//end looping  }  }  } |

****

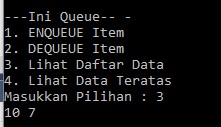
Gambar 7.16 Tampilan *queue* pada C#

****

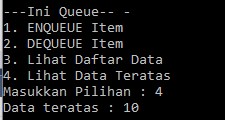
Gambar 7.17 *enqueue* item

****

Gambar 7.18 *enqueue* item

****

Gambar 7.19 Lihat daftar data

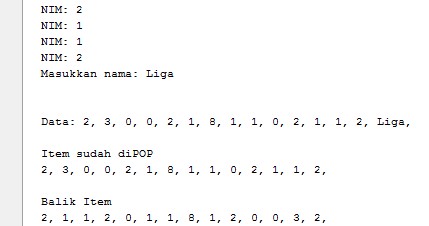
****

Gambar 7.20 Lihat data teratas

Untuk menggunakan *queue* dibutuhkan *library* dari C# dengan cara using System.Collections, lalu deklarasikan variabel yang digunakan. Lalu membuat objek baru dari *class queue* yang di*import* Queue antrian = new Queue(). Setelah itu digunakan perulanagan untuk membuat tampilan menu yang hasilnya seperti gambar 7.16, gunakan int.Parse(Console.ReadLine())untuk mengambil nilai *input* dan masukkan pada variabel pilihan. Setelah itu menggunakan perkondisian *if* untuk menjalankan kode sesuai pilihan pada menu. Jika pilihannya 1 maka akan diminta untuk memasukkan nilai yang nanti disimpan di variabel data, lalu variabel tersebut akan di-*enqueue* pada *queue* seperti gambar 7.17*.* Jika pilihannya 2 maka data akan di­-*dequeue* dari *queue.* Jika pilihannya 3 maka akan menampilkan semua data yang ada pada *queue* dengan perulangan *foreach* seperti gambar 7.19*.* Jika pilihannya 4 maka akan menampilkan data teratas(*peek*) dari *queue* seperti gambar 7.20*.* Kemudian jika semua kondisi tidak terpenuhi akan ditampilkan pesan “Pilihan Tidak Ada!!” program akan terus diulang selama pilihan tidak sama dengan 0.

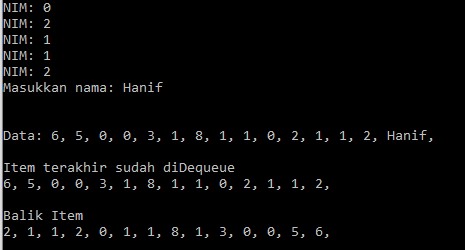
* 1. Tugas

|  |
| --- |
| package tugas6;  import java.util.Scanner;  import java.util.Stack;  public class Tugas6 {    public static void main(String[] args) {  int data;  String nama;  Stack result = new Stack();  Scanner input = new Scanner(System.in);  System.out.println("Masukkan NIM satu persatu");  for (int i = 0; i < 14; i++) {  System.out.print("NIM: ");  data = input.nextInt();  result.push(data);    }  Scanner name = new Scanner(System.in);  System.out.print("Masukkan nama: ");  nama = name.nextLine();  result.push(nama);  System.out.println();  System.out.println();  System.out.print("Data: ");  result.forEach(a->{  System.out.print(a+", ");  });  System.out.println();  System.out.println();  System.out.println("Item sudah diPOP");  result.pop();  result.forEach(a->{  System.out.print(a+", ");  });  System.out.println();  System.out.println();  System.out.println("Balik Item");  for(int i = 0; i < 14; i++)  {  System.out.print(result.pop()+", ");  }  System.out.println();  }  } |



Untuk menggunakan *stack* dibutuhkan *library* dari java dengan cara import java.util.Stack, lalu deklarasikan variabel data untuk menyimpan masukan angka dari *user* dan variabel *string* nama untuk menyimpan masukan bertipe *string.* Lalu membuat objek baru dari *class stack* yang di*import* Stack result = new Stack(). Gunakan *scanner* untuk mengambil nilai *input* dan masukkan pada variabel data. Kemudian menggunakan perulangan *for* untuk meng-*inputkan* NIM sebanyak 14 digit yang nanti disimpan di variabel data, lalu variabel tersebut akan di-*push* pada *stack.* Setelah itu membuat *input* untuk nama yang akan disimpan di variabel nama dan di-*push* pada *stack*. Lalu untuk menampilkan semua data yang sudah disimpan di *stack* digunakan perulangan *foreach.* Kemudian untuk mengeluarkan nama dari *stack* diperlukan operasi *pop* yang dipanggil dengan sintaks result.pop(). Maka nama akan keluar dan tersisa NIM untuk ditampilkan. Untuk menampilkan NIM secara terbalik hanya perlu mencetak tiap data yang di ­*pop* dari *stack* karena prinsipnya yang menggunakan LIFO (*Last In First Out*). Data yang dimasukkan terakhir akan pertama keluar atau urutan keluaran kebalikan dari urutan masukannya.

|  |
| --- |
| using System;  using System.Collections;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace tugas6  {  class Program  {  static void Main(string[] args)  {  int data;  String nama;  Queue result = new Queue();  Console.WriteLine("Masukkan NIM satu persatu");  for (int i = 0; i < 14; i++)  {  Console.Write("NIM: ");  int.TryParse(Console.ReadLine(), out data);  result.Enqueue(data);  }  Console.Write("Masukkan nama: ");  nama = Console.ReadLine();  result.Enqueue(nama);  Console.WriteLine();  Console.WriteLine();  Console.Write("Data: ");  foreach (Object obj in result)  {  Console.Write(obj+", ");  }  Console.WriteLine();  Console.WriteLine();  Console.WriteLine("Item terakhir sudah diDequeue");  object first = result.Peek();  object current = null;  while (true)  {  current = result.Dequeue();  if (result.Peek() == first)  {  break;  }  result.Enqueue(current);  }  foreach (Object obj in result)  {  Console.Write(obj + ", ");  }  Console.WriteLine();  Console.WriteLine();  Console.WriteLine("Balik Item");  Stack stack = new Stack();  while (result.Count > 0)  {  stack.Push(result.Peek());  result.Dequeue();  }  while (stack.Count > 0)  {  result.Enqueue(stack.Peek());  stack.Pop();  }  foreach (Object obj in result)  {  Console.Write(obj + ", ");  }  Console.WriteLine();  Console.ReadKey();  }  }  } |



Untuk menggunakan *queue* dibutuhkan *library* dari C# dengan cara using System.Collections, lalu deklarasikan variabel data untuk menyimpan masukan angka dari *user* dan variabel *string* nama untuk menyimpan masukan bertipe *string.* Lalu membuat objek baru dari *class queue* yang di-*import* Queue result = new Queue(). Kemudian menggunakan perulangan *for* untuk meng-*inputkan* NIM sebanyak 14 digit yang nanti disimpan di variabel data, lalu variabel tersebut akan di-*enqueue* pada *queue.* Setelah itu membuat *input* untuk nama yang akan disimpan di variabel nama dan di-*enqueue* pada *queue*. Untuk membaca *input* digunakan perintah Console.ReadLine(). Lalu untuk menampilkan semua data yang sudah disimpan di *queue* digunakan perulangan *foreach.* Kemudian untuk mengeluarkan nama dari *queue* diperlukan operasi *dequeuer,* karena prinsip *queue* menggunakan FIFO(*First In First Out*) artinya data yang pertama masuk akan pertama keluar. Sedangkan nama dimasukkan terakhir atau menjadi data terakhir yang keluar maka untuk mengeluarkan data terakhir digunakan cara *dequeue* semua data yang ada di *queue* dan disimpan dalam suatu variabel hingga tersisa data terakhir. Lalu *enqueuer* lagi data yang tadi dikeluarkan sehingga meninggalkan data terakhir. Maka nama akan keluar dan tersisa NIM untuk ditampilkan. Untuk menampilkan NIM secara terbalik digunakan cara memasukkan semua data *queue* kedalam *stack* maka urutan data akan menjadi terbalik dan memasukkan lagi data yang sudah terbalik kedalam *queue* untuk selanjutnya ditampilkan dengan perintah *foreach.*

* 1. Kesimpulan (minimal 7)