M. Bahrul Ulum, M.Kom



Struktur Data

Queue

Program Studi Teknik Informatika

Fakultas Ilmu Komputer

Universitas Esa Unggul

2018

**QUEUE**

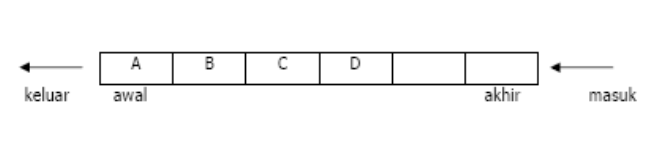
Teori

Queue (Antrian) adalah suatu bentuk khusus dari List Linier dengan operasi penyisipan (*insertion*) hanya diperbolehkan pada salah satu sisi, yang disebut sisi belakang (REAR), dan operasi penghapusan (*deletion*) hanya diperbolehkan pada sisi yang lainnya. Queue merupakan kumpulan data dengan penambahan data hanya melalui satu sisi, yaitu belakang (tail) dan penghapusan data hanya melalui sisi depan (head). Berbeda dengan stack yang bersifat LIFO maka queue bersifat FIFO (First In First Out), yaitu data yang pertama masuk akan keluar terlebih dahulu dan data yang terakhir masuk akan keluar terakhir.

**Hal yang perlu diingat :**

1. Queue disebut juga antrian dimana data masuk di satu sisi dan keluar di sisi yang lain.
2. Queue bersifat FIFO (First In First Out).





**Gambar 1. Ilustrasi Queue**

Elemen yang pertama kali masuk ke dalam queue disebut elemen depan (front/head of  
queue), sedangkan elemen yang terakhir kali masuk ke queue disebut elemen belakang  
(rear/tail of queue). Perbedaan antara stack dan queue terdapat pada aturan penambahan dan penghapusan elemen. Pada stack, operasi penambahan dan penghapusan elemen dilakukan di satu ujung. Elemen yang terakhir kali dimasukkan akan berada paling dekat dengan ujung atau dianggap paling atas sehingga pada operasi penghapusan, elemen teratas tersebut akan dihapus paling awal, sifat demikian dikenal dengan LIFO. Pada queue, operasi tersebut dilakukan di tempat yang berbeda. Penambahan elemen selalu dilakukan melalui salah satu ujung, menempati posisi di belakang elemen‐elemen yang sudah masuk sebelumnya atau menjadi elemen paling belakang. Sedangkan penghapusan elemen dilakukan di ujung yang berbeda, yaitu pada posisi elemen yang masuk paling awal atau elemen terdepan. Sifat yang demikian dikenal dengan FIFO.

Operasi Utama Queue

1. **Mendefinisikan queue dengan menggunakan struct dimana kita perlu menggunakan variable head dan tail sebagai penanda pada stack.**

typedef struct queue // Mendefinisikan queue dengan menggunakan struct

{

int head;

int tail;

char data [15][20]// menampung 15 data dengan jumlah

string max 20 huruf

};

1. **Mendefinisikan max untuk maksimum isi queue**

#define max 15

1. **Membuat variable array sebagai implementasi queue**

queue antri;

Operasi Operasi Queue

1. **Enqueue (menginputkan data pada queue)**

void enqueue(char d[20])

{

antri.head=0;

antri.tail++;

strcpy(antri.data[antri.tail],d);

printf("data berhasil dimasukkan");

}

1. **Dequeue (mengambil data dari queue)**

void dequeue()

{

printf ("data %s terambil",antri.data[antri.head]);

for (int i=antri.head;i<=antri.tail;i++)

strcpy (antri.data[i],antri.data[i+1]);

antri.tail--;

}

1. **isEmpty (mengecek apakah antrian kosong)**

int isEmpty()

{

if (antri.tail==-1)

{

antri.head=-1;

return 1;

}

else

return 0;

}

1. **isFull (mengecek apakah antrian penuh)**

int isFull()

{

if (antri.tail==max-1)

return 1;

else

return 0;

}

1. **Clear (membersihkan seluruh isi antrian)**

void clear()

{

antri.head=antri.tail=-1;

printf("semua data terhapus.");

}

1. **Print(mencetak seluruh isi antrian)**

void print()

{

for (int i=0;i<=antri.tail;i++)

printf ("%s\n",antri.data[i]);

}

**Implementasi Queue dengan Linear Array**

Disebut juga queue dengan model fisik, yaitu bagian depan queue selalu menempati posisi pertama array. Di bawah ini diperlihatkan suatu queue yang akan menempati N elemen array memori, serta cara pengurangan (delete) dan penambahan (added) elemen pada queue tersebut.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D |  |  |  | ..... |  |

Head   : 1

Tail      : 4

                        1          2          3          4        5         6          7        .....       N

REMOVE(Q)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | B | C | D |  |  |  | ..... |  |

Head   : 2

Tail      : 4

                        1          2          3          4          5          6       7       .....       N

INSERT(INSERT(E),F)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | B | C | D | E | F |  | ..... |  |

Head   : 2

Tail      : 6

                        1          2          3          4          5          6       7       .....        N

REMOVE(Q)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | C | D | E | F |  | ..... |  |

Head   : 3

Tail      : 6

                        1          2          3          4          5          6          7          .....       N

Dapat dilihat bahwa setiap terjadi penghapusan elemen pada queue nilai (index) dari Head bertambah satu (1) ; dapat ditulis HEAD = HEAD+1. Begitu pula bila terjadi penambahan elemen pada queue nilai (index) Tail bertambah satu (1); dapat ditulis TAIL = TAIL + 1.

Akan terjadi ketidakefisienan bila penambahan elemen sudah pada posisi index N (Tail = N) maka tidak dapat lagi dilakukan penambahan elemen, sedangkan dilokasi memori yang lain (nilai di bawah N) masih terdapat memori array yang kosong. Untuk mengatasi hal tersebut maka kita bayangkan bahwa memori dari queue tersebut berbentuk melingkar dimana kedua ujungnya saling bertemu atau disebut juga dengan *Circular Queue.*

Queue dengan linear array secara umum dapat dideklarasikan sebagai berikut:  
Const  
 MaxQueue = {jumlah};  
Type  
 TypeQueue = byte;  
Var  
 Queue : Array[1..MaxQueue] of TypeQueue  
 Head, Tail : Byte;

Operasi‐operasi queue dengan linear array:

1. Create

Procedure create berguna untuk menciptakan queue yang baru dan kosong yaitu  
dengan cara memberikan nilai awal (head) dan nilai akhir (tail) dengan nol (0). Nol menunjukan bahwa queue masih kosong.  
Procedure Create;  
Begin  
 Head := 0; Tail := 0;  
End;

1. Empty

Function empty berguna untuk mengecek apakah queue masih kosong atau sudah berisi data. Hal ini dilakukan dengan mengecek apakah tail bernilai nol atau tidak, jika ya maka kosong.  
Function Empty : Boolean;  
Begin  
 If Tail = 0 then  
 Empty := true  
 Else  
 Empty := False;  
End;

1. FULL  
   Function Full : Boolean;  
   Begin  
    If Tail = MaxQueue then  
    Full := true  
    Else  
    Full := False  
   End;
2. EnQueue

Procedure EnQueue berguna untuk memasukkan 1 elemen ke dalam queue.  
Procedure Enqueue(Elemen : Byte);  
Begin  
 If Empty then  
 Begin  
 Head := 1;  
 Tail := 1;  
 Queue[head] := Elemen;  
End  
Else  
If Not Full then  
Begin  
 Inc(Tail);  
 Queue[tail] := Elemen;  
 End;  
End;

1. DeQueue

Procedure DeQueue berguna untuk mengambil 1 elemen dari Queue, operasi ini sering disebut juga SERVE. Hal ini dilakukan dengan cara memindahkan semua elemen satu langkah ke posisi di depannya, sehingga otomatis elemen yang paling depan akan tertimpa dengan elemen yang terletak dibelakangnya.

Procedure DeQueue;  
Var I : Byte;  
Begin  
If Not Empty then  
 Begin  
 For I := Head to Tail‐1 do  
 Queue[I] := Queue[I+1];  
 Dec(Tail);  
 End;  
End;

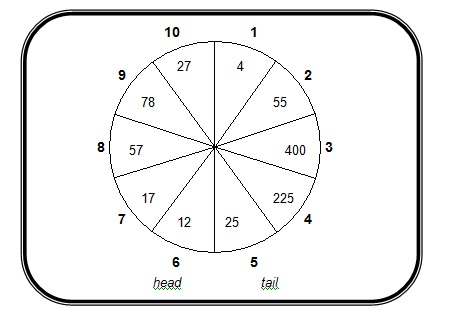
1. Clear

Procedure clear berguna untuk menghapus semua elemen dalam queue dengan jalan mengeluarkan semua elemen tersebut satu per satu sampai kosong dengan memanfaatkan procedure DeQueue.

Procedure Clear;  
Begin  
 While Not Empty then  
 DeQueue;  
End;

**Implementasi Queue dengan Circular Array**

Salah satu variasi array adalah array melingkar (circular array), artinya array dapat  
diakses mulai dari sembarang indeks (indeks awal) ke arah indeks terakhir (maksimum array), lalu memutar ke indeks pertama hingga kembali ke indeks awal. Circular array adalah array yang dibuat seakan‐akan merupakan sebuah lingkaran dengan titik awal dan titik akhir saling bersebelahan jika array tersebut masih kosong. Jumlah data yang dapat ditampung oleh array ini adalah besarnya ukuran array dikurangi 1. Misalnya besar array adalah 8, maka jumlah data yang dapat ditampung adalah 7. Ide yang paling sederhana adalah dengan mengisi tempat kosong (jika tersedia) yang berada pada awal array bila tail telah mencapai posisi pjg\_max sehingga penggunaan tempat menjadi lebih efisien. Jadi seolah-olah array di bawah ini dibulatkan manjadi sebuah lingkaran seperti digambarkan berikut ini (nama circular array berasal dari ide array yang dibulatkan).



**Gambar 2. Circular Array**

Aturan-aturan dalam queue yang menggunakan circular array adalah:

1. Proses penghapusan dilakukan dengan cara nilai depan (front) ditambah 1: front=front+1.
2. Proses penambahan elemen sama dengan linear array yaitu nilai belakang ditambah 1: tail=tail + 1.
3. Jika front = maks dan ada elemen yang akan dihapus, maka nilai front = 1.
4. Jika tail = maks dan front tidak 1 maka jika ada elemen yang akan ditambahkan, nilai tail = 1
5. Jika hanya tinggal 1 elemen di queue (front = tail), dan akan dihapus maka front diisi 0 dan tail diisi dengan 0 (queue kosong).

Front dan Tail akan bergerak maju, jika:

Untuk penambahan.

Tail sudah mencapai elemen terakhir array akan memakai elemen pertama array yang telah dihapus.

Untuk pngahapusan.

1. Front telah mencapai elemen terakhir array, maka akan menuju elemen pertama jika antrian masih berisi elemen.

Dengan circular array, meskipun posisi terakhir telah terpakai, elemen baru tetap dapat  
ditambahkan pada posisi pertama jika posisi pertama dalam keadaan kosong. Jika  
akhir=MAX dan awal=1, nilai head dan tail mencapai maksimum, maka akan dikembalikan ke posisi awal. Operasi‐operasi yang terdapat pada circular array tidak jauh berbeda dengan operasi pada linear array.

Operasi‐operasi queue dengan circular array:

1. Create

Procedure Create;  
Begin  
 Head := 1;  
 Tail := MaxQueue;  
End;

1. Empty

Function Empty : Boolean;  
 Begin  
 If (Tail Mod Max\_Queue ) + 1 = Head then  
 Empty := true  
 Else  
 Empty := False;  
End;

1. FULL

Function Full : Boolean;  
 Var  
 X : 1 .. Max\_queue;  
 Begin  
 X := (Tail Mod Max\_queue)+1;  
 If (x Mod Max\_queue) + 1 = head then  
 Full := True;  
 Else

Full := False;  
End;

1. EnQueue

Procedure EnQueue (Elemen : TypeElemen);  
 Begin  
 If Not Full Then  
 Begin  
 Tail := (Tail Mod Max\_queue) + 1 ;  
 Queue[Tail] := Elemen;  
 End;  
 End;

1. DeQueue

Procedure Dequeue;  
 Begin  
 If Not Empty then  
 Head := (Head Mod Max\_queue) + 1;  
 End;

Penerapan Queue

Meski Queue sangat sederhana, namun Queue merupakan kakas dasar penyelesaian masalah-masalah besar. Penggunaan Queue yang utama adalah untuk simulasi fenomena antrian di dunia nyata, serta fenomena antrian di pengolahan data.

1. **Pada Pembelian Tiket**

Dalam kehidupan sehari-hari kita bisa dapati melalui penerapan pembelian tiket kereta api, tiket pesawat, tiket kapal laut, pembayaran tiket tol, pembayaran listrik, pembayaran air, dan lain sebagainya. Saat mengantri di loket untuk membeli tiket. Istilah yang cukup sering dipakai seseorang masuk dalam sebuah antrian adalah enqueue. Dalam suatu antrian, yang datang terlebih dahulu akan dilayani lebih dahulu. Istilah yang sering dipakai bila seseorang keluar dari antrian adalah dequeue. Walaupun berbeda implementasi, tapi pada prinsipnya sama saja. Aplikasi dalam pembelian tiket kereta api:

**a.**         **Enqueue       :** Seseorang membeli tiket melalui tempat pembayaran tiket yang disediakan.

**b.**        **Dequeue       :** Setelah membeli tiket, langsung menuju tempat penungguan kereta, dengan sebelumnya petugas memeriksa cek tiket tersebut.

**c.**         **Clear            :** Pembeli tiket tersebut telah terhapus dari antrian karena sudah melewati pembayaran administrasi tersebut.

**d.**        **IsEmpty       :** Petugas tiket Kereta melihat tidak ada lagi yang ingin membeli tiket kereta.

**e.**         **IsFull            :** Petugas Tiket Kereta melihat masih ada pembeli tiket kereta.

1. **Pada Aplikasi Download Manager IDM**

**Queue** juga dipakai sebagai salah satu fitur dari ***Internet Download Manager*** atau yang biasa disebut dengan ***IDM***. Fitur ini sangat membantu bagi para pecinta download. Fitur ini akan membantu pengguna untuk mendownload file yang anda pilih satu-persatu, jadi sebanyak apapun anda mendownload, tetapi akan tetap dibuat antrian atau istilahnya queueing. Sistem yang diterapkan dalam fitur ini adalah pada saat download file, maka ***IDM*** akan mendownload satu per satu, hingga download file selesai maka baru akan secara otomatis mendownload file berikutnya.

1. **Antrian Mobil**

Ketika sebuah mobil datang, dari belakang akan menuju kedepan dari antrian. Setelah mobil mendapatkan karcis tol, antrian yang berada didepan akan maju. Pada saat menempatkan data pada ujung (tail) dari queue disebut dengan enqueue. Pada saat memindahkan data dari kepala (head) sebuah queue disebut dengan dequeue.



**Gambar 3. Antrian Mobil**

1. **Pada Aplikasi Download Manager IDM**

**Queue** juga dipakai sebagai salah satu fitur dari ***Internet Download Manager*** atau yang biasa disebut dengan ***IDM***. Fitur ini sangat membantu bagi para pecinta download. Fitur ini akan membantu pengguna untuk mendownload file yang anda pilih satu-persatu, jadi sebanyak apapun anda mendownload, tetapi akan tetap dibuat antrian atau istilahnya queueing. Sistem yang diterapkan dalam fitur ini adalah pada saat download file, maka ***IDM*** akan mendownload satu per satu, hingga download file selesai maka baru akan secara otomatis mendownload file berikutnya.

1. **Printer Sharing**

Pada suatu jaringan terdapat beberapa aplikasi yang dapat membantu kita untuk mempermudah pekerjaan kita untuk melakukan cetak hanya dengan menggunakan 1 printer yang dapat dipakai oleh banyak orang yaitu dengan cara printer sharing. Aplikasi ini menggunakan cara kerja queue atau juga disebut dengan antrian, untuk mencetak suatu document yang hanya menggunakan 1 printer yang dapat diakses oleh orang banyak, penggunaan queue diperlukan untuk mengatur sistem jaringan agar tidak terjadi error, dengan cara siapa yang terlebih dahulu mencetak suatu dokumen, dia akan dilayani terlebih dahulu untuk mencetak dokumen dan jika ada seseorang yang ingin mencetak dokumen dia akan dimasukan kedalam antirian dan menunggu untuk dapat mencetak sebuah document.

1. **Sistem Produksi**

Dalam sistem produksi terdapat banyak sekali aplikasi antrian, misalnya pada mesin pengisi (filling) botol minuman otomatis di pabrik. Mesin ini digunakan agar mempermudah pekerjaan, meminimalisir waktu, dan masih banyak lagi manfaat lainnya. Cara kerja dari mesin filling otomatis ini yaitu :

1.      Beberapa botol disiapkan pada tempatnya.

2.      Kemudian mesin akan mengisi air kedalam botol sesuai dengan posisinya.

3.      Botol yang diletakkan paling depan akan diisi oleh mesin paling awal, setelah botol paling depan terisi baru akan berjalan ke botol yang dibelakangnya, dan seterusnya.

Prinsip queue atau antrian digunakan dalam mesin ini, karena FIFO (First In First Out). Jadi botol yang paling awal/depan sampai pada pengisi air, maka botol itu juga yang akan pertama keluar/diisi.

Kesimpulan Queue

*Queue* (antrian) adalah salah satu *list linier* dari struktur data yang beroperasi dengan cara *First In First Out* (FIFO) yaitu elemen pertama yang masuk merupakan elemen yang pertama keluar. Data-data di dalam antrian dapat bertipe *integer*, *real, record* dalam bentuk sederhana atau terstruktur. Queue dilakukan dengan cara penyisipan di satu ujung, sedang penghapusan di ujung lain. Ujung penyisipan biasa disebut *rear/tail*, sedang ujung penghapusa disebut *front/head*.

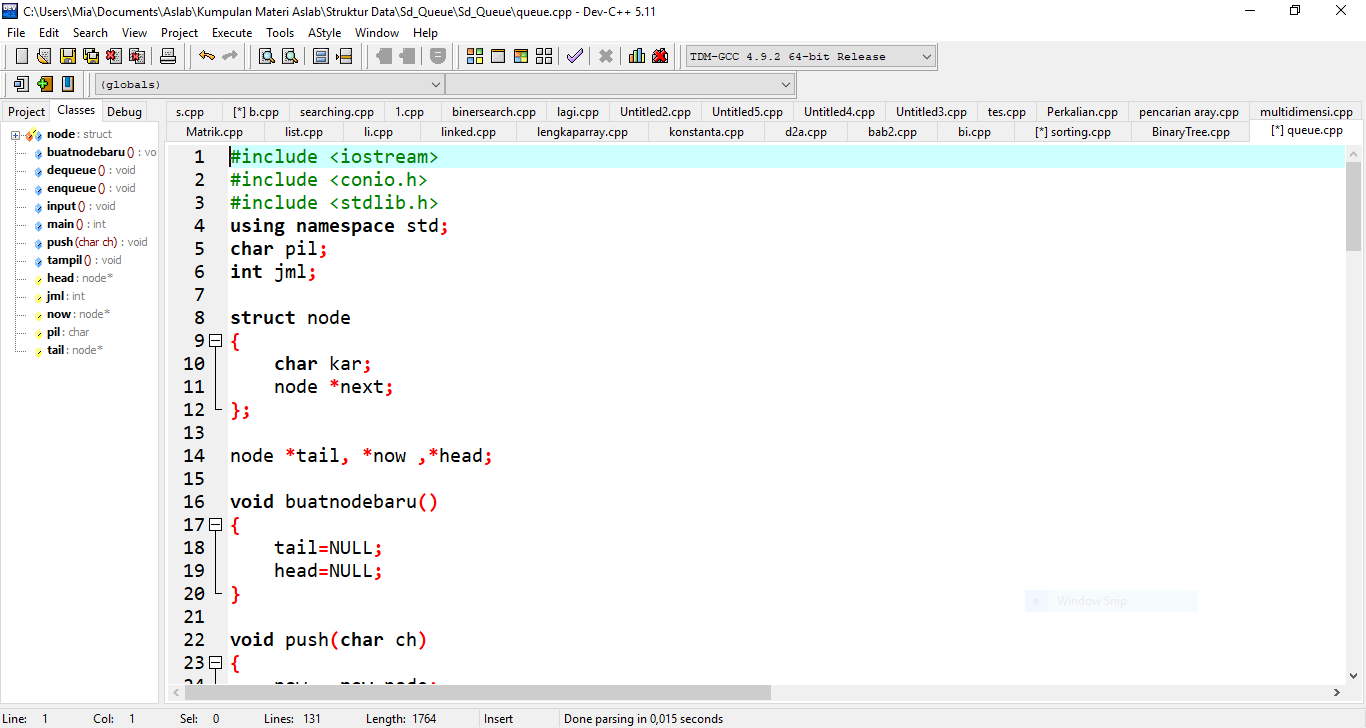
Sebuah *queue* dalam program setidaknya harus mengandung tiga variabel, yakni: *head* untuk penanda bagian depan antrian, *tail* unttuk penanda bagian belakang antrian, dan *array data* untuk menyimpan data-data yang dimasukkan ke dalam queue tersebut.

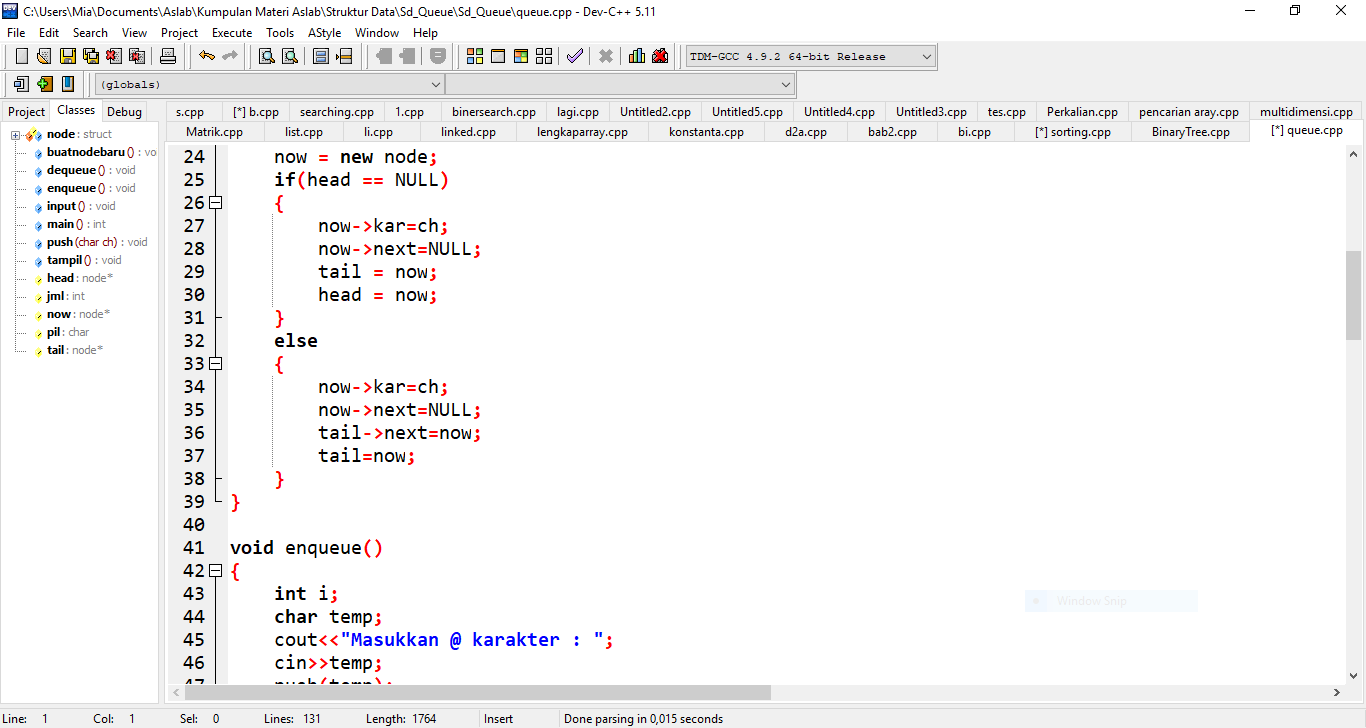
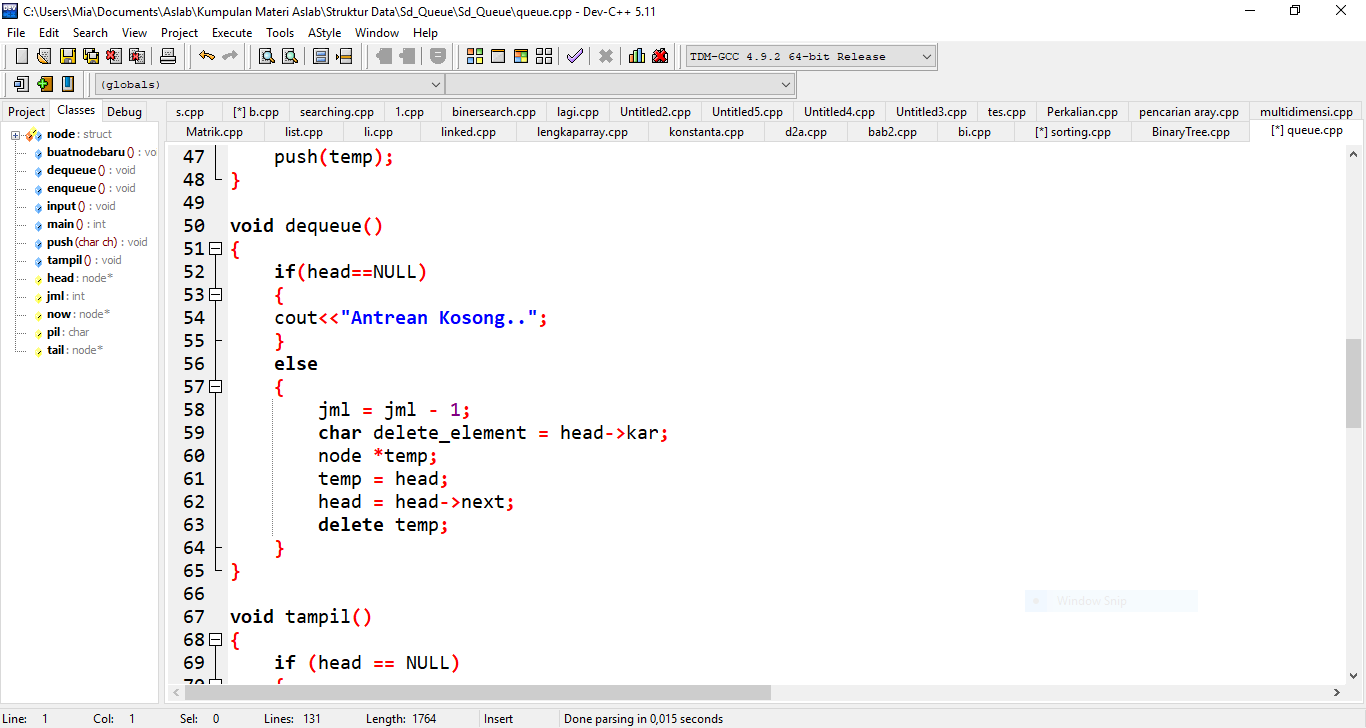
Pada queue ada operasi – operasi dasar, yaitu: prosedur *create* untuk membuat queue baru yang kosog, fungsi *IsEmpty* untuk mengecek queue tersebut kosong atau tidak, fungsi *IsFull* untuk mengecek queue tersebut penuh atau tidak, prosedur *EnQueue* untuk memasukkan data kedalam queue, prosedur *DeQueue* untuk mengeluarkan sebuah elemen pada posisi *head* dari queue, fungsi *clear* untuk menghapus elemen queue, dan prosedur *tampil* untuk menampilakn elemen yang ada pada queue.

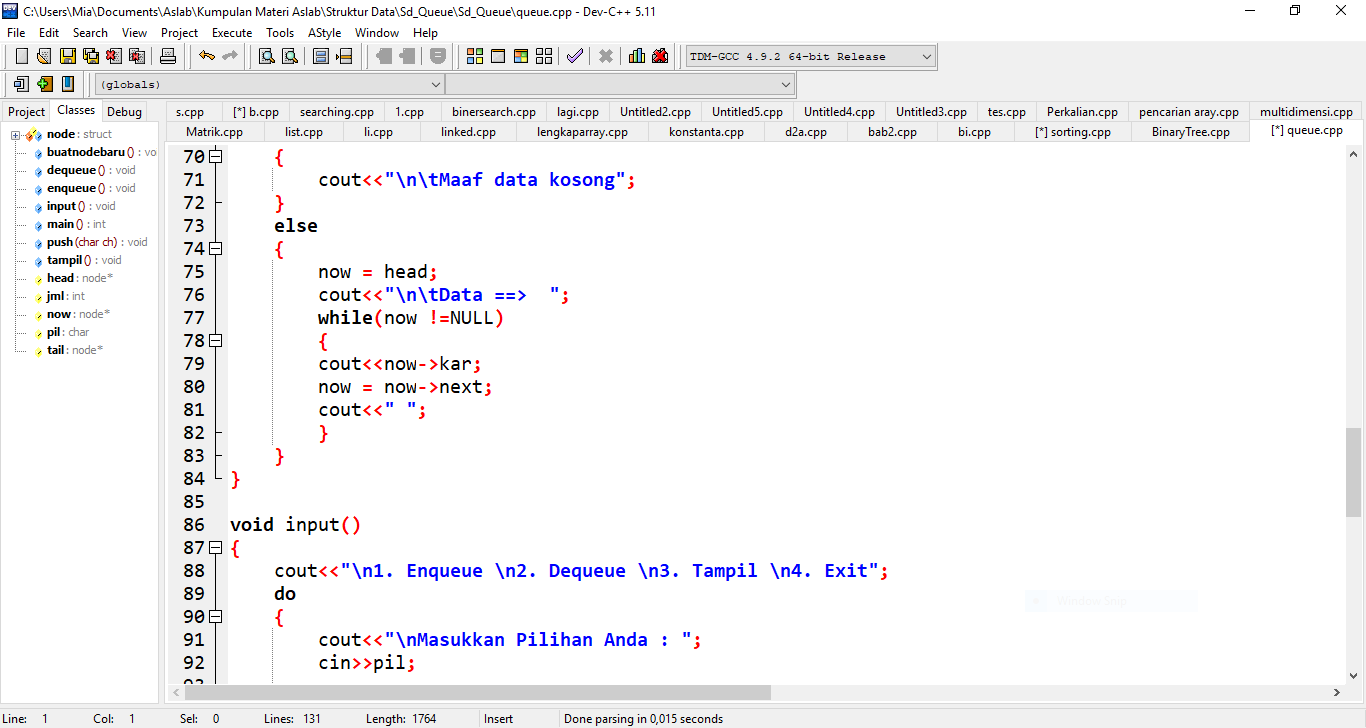
Ada beberapa queue, yaitu: *queue dengan linear array* yaitu queue yang dibuat seakan-akan merupakan suatu garis lurus dengan satu pintu masuk dan satu pintu keluar, *queue dengan circular array* yaitu queue yang dibuat seakan-akan merupakan sebuah lingkaran dengan titik awal (head) dan titik akhir (tail) saling bersebelahan jika array tersebut masih kosong, dan *queue dengan linked-list*.

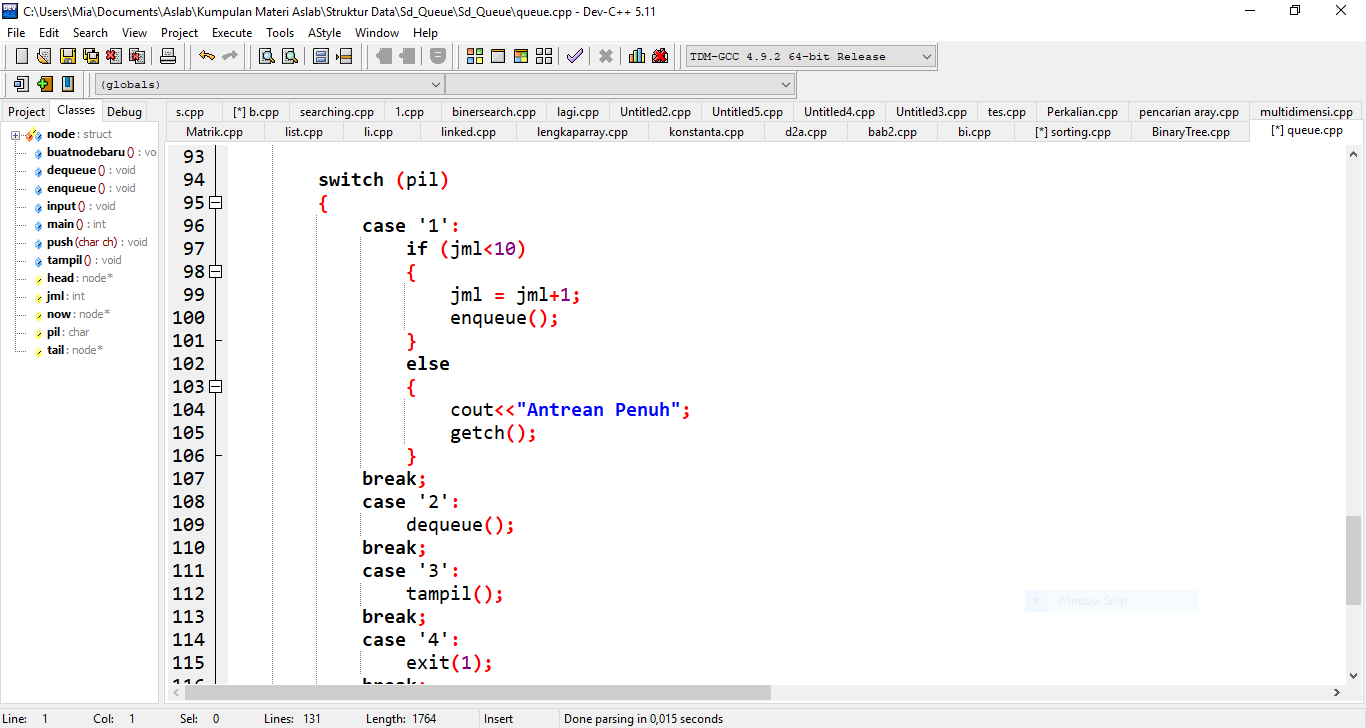
Pelaksanaan Praktikum

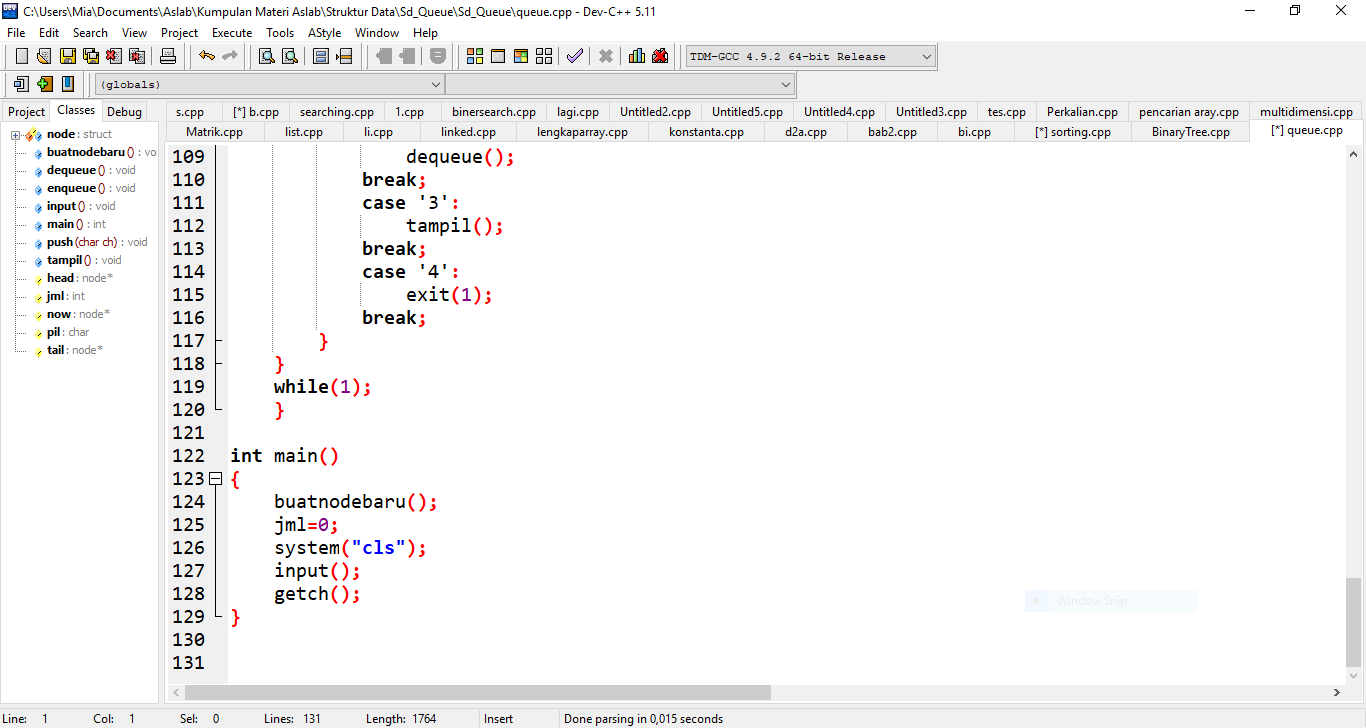
**Program**









Latihan

1. Deskripsikan program pelaksanaan praktikum tersebut !
2. Buatlah sebuah program yang menunjukkan program tersebut sebuah program queue. Gunakan tampilan menarik dan penjelasan apa saja yang digunakan dalam program tersebut !

SOAL LATIHAN

## Guided STACK:

1. Dari flowchart dibawah ini, buatlah fungsi utama untuk menjalankan stack diatas dengan menggunakan menu.

# Kunci jawaban soal GUIDED STACK:

start

int a;

char input[20]

tumpuk.top 🡨 -1

Cetak “menu: 1.push 2. Pop 3.clear 4.print

Pilihan:”

Input a

switch (a)

a==1

Isfull()==1

Cetak “Stack penuh”

Cetak “masukkan data”

gets input

push(input)

a==2

IsEmpty()==1

Cetak “Stack kosong”

pop()

a==3

clear()

Cetak “Stack kosong”

a==4

IsEmpty()==1

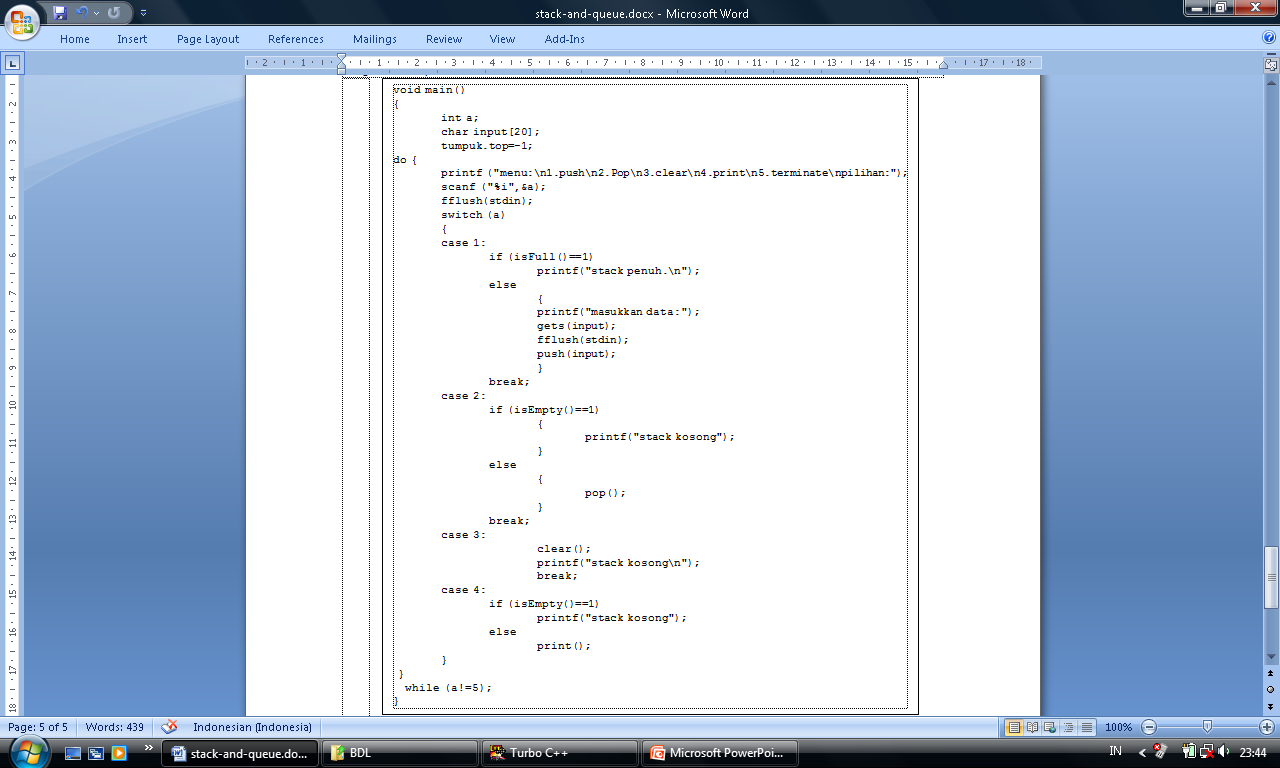
Cetak “Stack kosong”

print()

a!=5

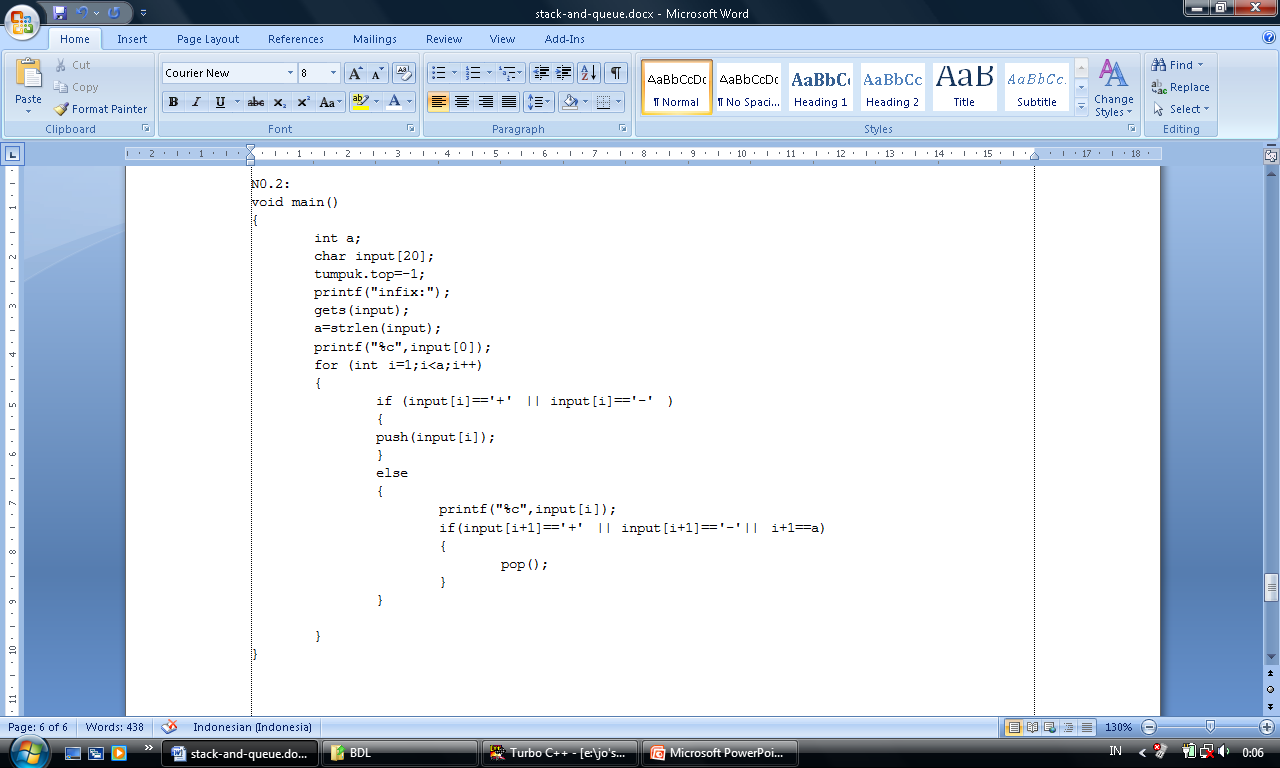
selesai

**No.1: Fungsi utamanya:**



No.2:

**Jangan lupa sesuaikan fungsi pop, dan push yang sudah ada dengan kontex penggunaannya dan juga tipe datanya!**



**Latihan Queue**

1. Buatlah fungsi utama untuk menjalankan queue pada modul dengan menggunakan menu. (untuk alurnya, hampir sama dengan flowchart untuk stack diatas. Hanya saja fungsi diganti dengan fungsi-fungsi queue.)
2. Carilah total, rata-rata, nilai terkecil, nilai terbesar dari nilai yang telah di inputkan pada queue

**Latihan Mandiri Kelas Queue**

1. Tambahkanlah function untuk mencari suatu elemen dalam queue & stack
2. Tambahkan function untuk mengedit suatu elemen dalam queue & stack
3. Buatlah suatu fungsi untuk mencetak elemen stack ganjil
4. Buatlah suatu fungsi untuk mencetak elemen queue genap

**Latihan Mandiri Rumah /Tes Queue**

1. Buatlah program untuk membuat kalkulator scientific!
2. Buatlah program untuk membalikkan kata yang berguna untuk membalik susunan elemen dalam queue Q dengan menggunakan bantuan sebuah stack
3. Buatlah program untuk menentukan apakah kata yang dimasukan merupakan palindrome atau bukan