Rangkuman Struktur Data

Queue

Bersifat FIFO – First In First Out

* Elemen yang pertama masuk ke antrian akan keluar pertama kalinya.
* Inqueue adalah memasukan satu elemen kedalam suatu antrian.
* Dequeue adalah mengeluarkan satu elemen dari suatu antrian.
* Antrian dapat dibuat dengan menggunakan: Liniear Array dan Circular Array.

Kapan Empty

* 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.
* A picture containing diagram

  Description automatically generated

Kapan Full

* 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.
* Application, table

  Description automatically generated

Kapan Inqueue

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.

A picture containing graphical user interface

Description automatically generated

Kapan Dequeue

* Digunakan untuk menghapus elemen terdepan / pertama (Head) dari antrian.
* Dengan cara menggeser semua elemen antrian kedepan dan mengurangi Tail dengan 1 penggeseran dilakukan dengan menggunakan looping.
* Diagram

  Description automatically generated

Single Linked List

Menghapus Node Depan

Diagram

Description automatically generatedvoid hapusDepan(){

TNode \*hapus;

int d;

if (isEmpty()==0){

if(head!=tail){

hapus = head;

d = hapus->data;

head = head->next;

delete hapus;

}

else{

d = tail->data;

head=tail=NULL;

}

printf(“%d terhapus\n“,d);

}

else

printf("Masih kosong\n“);

}

}

Function di atas akan menghapus data terdepan (pertama) yang ditunjuk oleh head pada linked list.

* Penghapusan node tidak boleh dilakukan jika keadaan node sedang ditunjuk oleh pointer, maka harus dilakukan penunjukkan terlebih dahulu dengan pointer hapus pada head, kemudian dilakukan pergeseran head ke node berikutnya sehingga data setelah head menjadi head baru, kemudian menghapus pointer hapus dengan menggunakan perintah delete.
* Jika tail masih NULL maka berarti list masih kosong!

Diagram

Description automatically generated

\*Pointer first diarahkan pada data ke-2.

\*Pointer p diarahkan pada data ke-1.

\*Bebaskan pointer p (secara otomatis data ke-1 terhapus).

Menghapus Node Belakang

void hapusBelakang(){

TNode \*bantu,\*hapus;

int d;

if (isEmpty()==0){

bantu = head;

if(head!=tail){

while(bantu->next!=tail){

bantu = bantu->next;

Diagram

Description automatically generated}

hapus = tail;

tail=bantu;

d = hapus->data;

delete hapus;

tail->next = NULL;

}

else{

d = tail->data;

head=tail=NULL;}

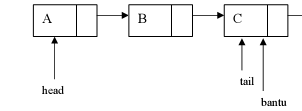
cout<<d<<" terhapus\n";

}

else

cout<<"Masih kosong\n";

}



Function di atas akan menghapus data terbelakang (terakhir) yang ditunjuk oleh tail pada linked list.

* Penghapusan node tidak boleh dilakukan jika keadaan node sedang ditunjuk oleh pointer, maka harus dilakukan penunjukkan terlebih dahulu dengan variabel hapus pada tail, kemudian dibutuhkan pointer bantu untuk membantu pergeseran dari head ke node berikutnya sampai sebelum tail, sehingga tail dapat ditunjukkan ke bantu tersebut, dan bantu tersebut akan menjadi tail yang baru.
* Setelah itu hapus pointer hapus dengan menggunakan perintah delete. Jika tail masih NULL maka berarti list masih kosong!

Diagram

Description automatically generated

\*Telusuri simpul s/d first->next = NULL.

\*Arahkan pointer p ke first.

\*Bebaskan pointer p->next (Simpul Terakhir).

\*Arahkan p->next ke NULL.

Menampilkan Data

void tampil(){

TNode \*bantu;

bantu = head;

if(isEmpty()==0){

while(bantu!=NULL){

printf(“%d\n”,bantu->data);

bantu=bantu->next;}printf(“\n”);

}

else

printf(“Masih kosong\n“);

}

A picture containing text, antenna

Description automatically generated