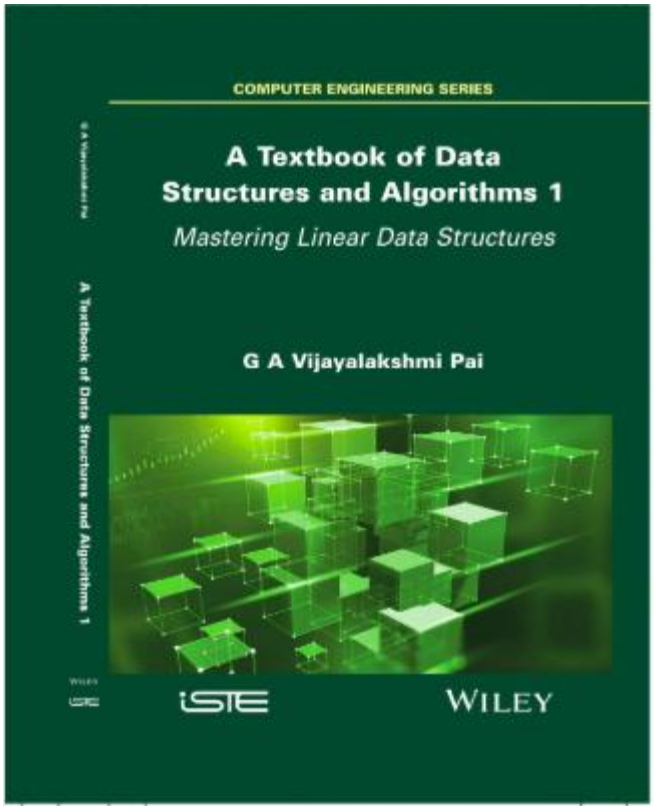
TIB29 – Struktur Data dan Algoritma

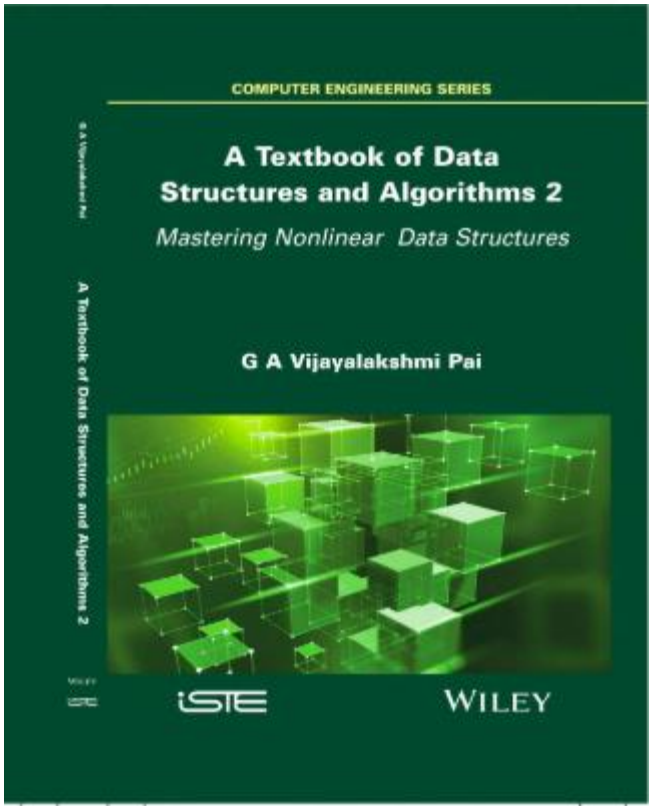
PERINGATAN HAK CIPTA

PERINGATAN HAK CIPTA

Stack dan Queue Berbasis Linked-List

Diadopsi Dari Sumber:





Sub-CPMK

- Mahasiswa mampu menggunakan linked list untuk membuat stack dan Queue beserta operasi-operasinya. (C3, A3)

Materi

1. Linked-List Base Stack
2. Linked-List Base Queue

1. Linked-List Base Stack

3.1. *Implementation Stacks* dengan *Linked List*

Data store yang dibutuhkan

- *Single Linked List* dengan *Head* Node
- *Head* berfungsi sebagai *top Stacks*
- (saran: dapat juga menggunakan *Tail* Node sebagai *top stack*, dengan demikian pengisian selalu dilakukan pada *Tail*)

3.2. *Stacks-Linked List Operation*

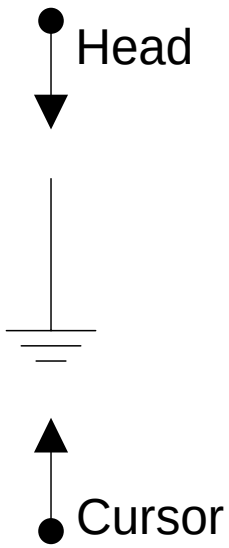
- Clear()
  - *Set Variabel Head* dengan Null
- isEmpty()

  Check isi simpul *Head*
  - Head Value = NULL → empty
  - Head Value Not NULL → not empty

3.2. *Stacks-Linked List Operation* (Lanj.)

- push(el) :
  - Buat Node baru
  - *Set Next Link* dari Node baru ke node *Head*
  - *Set* Node baru sebagai *Head*
  - Ambil data el dan isi ke node baru
- pop() :
  - Ambil *Head*->data dan masukkan ke *variable* yang akan digunakan
  - Isi variabel *Head* dengan *Head->Next*
- topEl()
  - Ambil *Head*->data dan masukkan ke *variable* yang akan digunakan

3.3. *Stacks-Linked List Operation*: isEmpty()

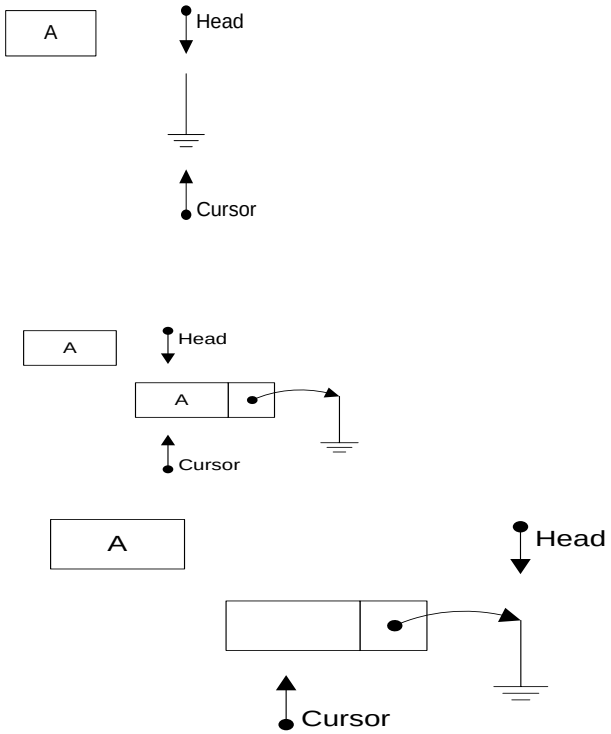


- Jika Cursor atau pointer berada pada Head, atau top, dan sama-sama berisi NULL, berarti stack dalam kondisi kosong.

3.4. *Stacks-linked List Operation: Push Stack* Pada *Empty Stacks*

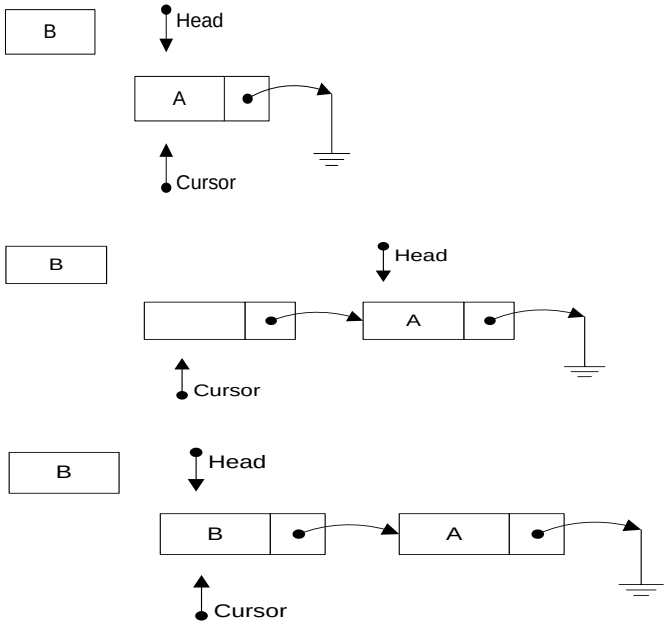
push(A)

- Simpul baru dibentuk,
- elemen A diisi pada simpul baru
- Head dan cursor / pointer menunjuk ke simpul baru

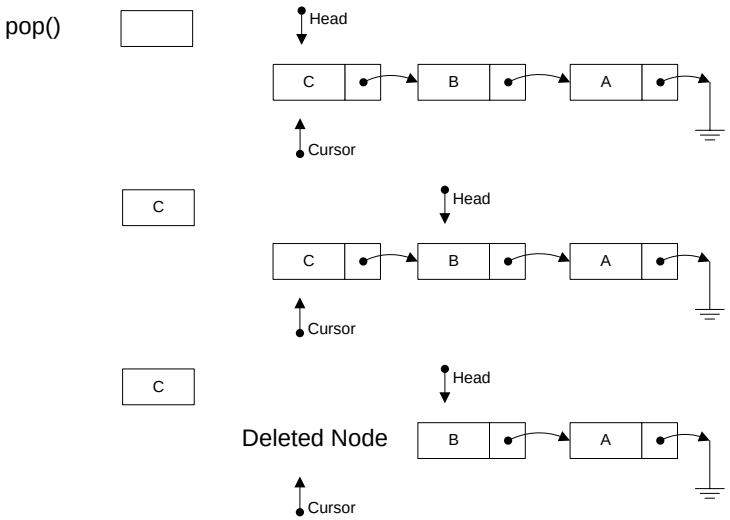


3.5. *Stacks-linked List Operation*: *Push Stack* Pada *Stacks* Yang Berisi

- push(B)
- Simpul baru yang akan diisi elemen B dibuat dan dilink nya diarahkan ke Head
- Selanjutnya Head dipindahkan ke simpul baru

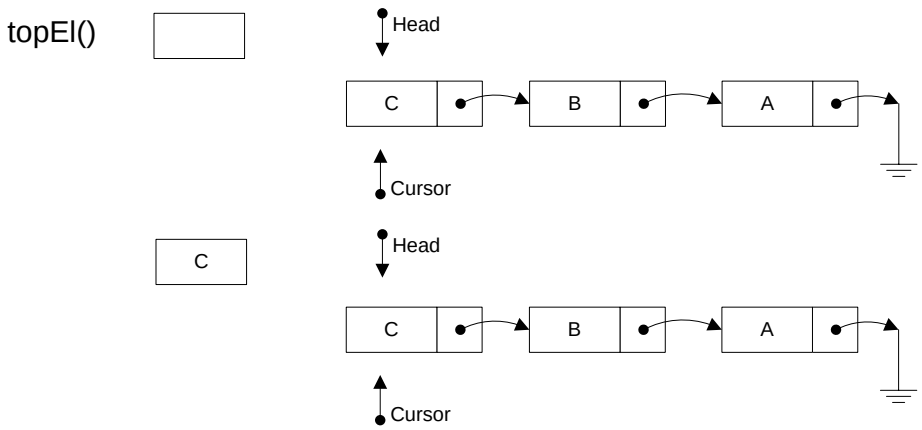


3.6. *Stacks-Linked List Operation:* pop()



- Ptr diarahkan ke Head
- Data / elemen pada top stack ditampung pada variabel
- Pindahkan Head
- Hapus node yang ditunjuk oleh cursor/ptr

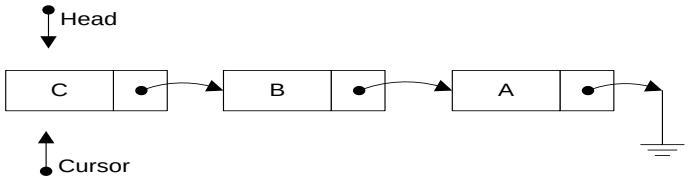
3.7. *Stacks-Linked List Operation:* topEl()



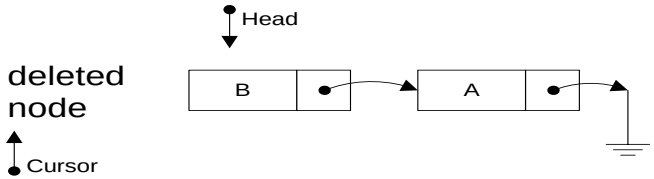
- TopEl hanya membaca data pada top Stack
- Cursor / Ptr arahkan ke Head
- Copykan isi / elemen yang ditunjuk oleh Head ke variabel penampungnya

3.8. *Stacks-Linked List Operation*: clear()

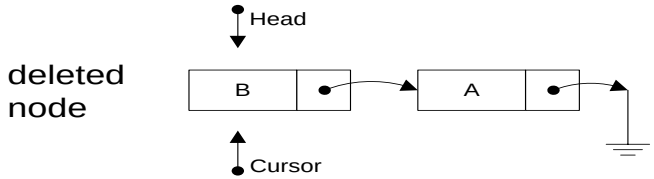
1.



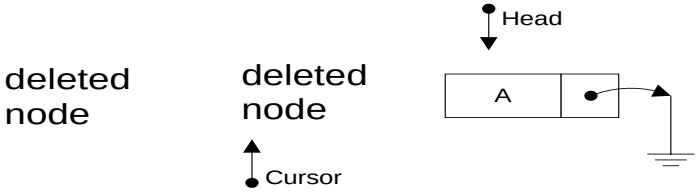
2.



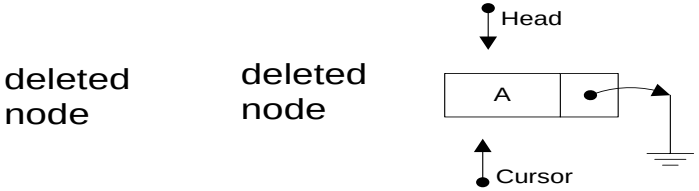
3.



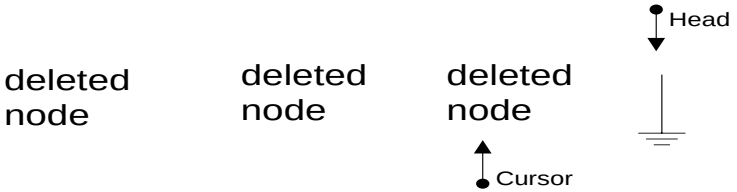
4.



5.



6.



3.8. *Stacks-Linked List Operation*: clear() (Lanj.)

Sesuai dengan yang diperagakan pada gambar:

- Arahkan cursor ke Head, kemudian pindahkan Head ke simpul berikutnya
- Hapus simpul yang ditunjuk oleh cursor
- Ulangi langkah dari langkah pertama sampai Cursor dan Head menunjuk NULL

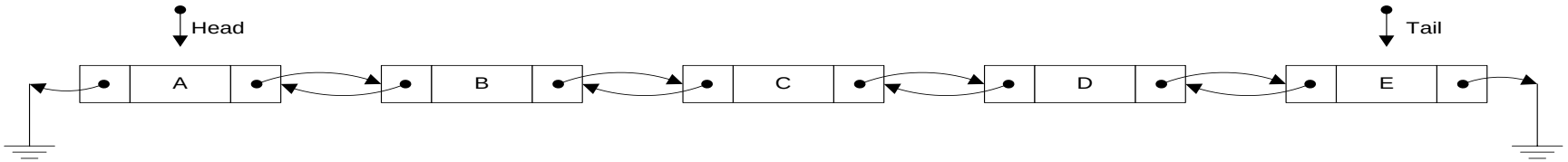
2. *Linked-List Base Queue*

3.1. Penerapan *Queue* dengan *Linked List*

- Hanya diperlukan *Double Linked List* dan cara yang mudah untuk mendapatkan *queue* pertama dan terakhir.
- (jika menggunakan *single linked list*, akan sedikit lebih repot pada saat melakukan dequeue, pada first step harus menyimpan *current head* ke *temporary variable*, untuk membebaskan *current head* dengan mudah).
- *First queue* dapat diterapkan dengan *head.*
- *Last queue* dapat diterapkan dengan *tail*

  (atau kebalikannya: *head* sebagai *Last queue* dan *tail* sebagai *First queue*, tapi harus punya *pointer* untuk menunjuk ke *previous tail* – dapat diterapkan dengan *double linked list* or *single list* dengan informasi *Previous Tail*).

3.1. Penerapan *Queue* dengan *Linked List* (Lanj.)



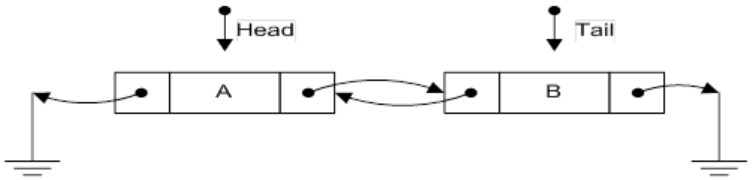
3.2. *Queue Operation* dengan *Linked List*

- clear() →
  - Hancurkan setiap node
  - set *Head* dengan NULL
- isEmpty() →
  - Periksa jika *Head* == NULL, maka queue is empty
- enqueue(el) →
  - Buat node baru pada *tail*,
  - Isi *element* el pada node baru.
- Dequeue →
  - Ambil *element* dari node yang ditunjuk oleh *Head*,
  - *set next* Node sebagai *Head*
  - *destroy* node yang semula *Head*
- firstEl() →
  - Ambil *element* dari node *Head*.

3.2. *Queue Operation* dengan *Linked List* (Lanj.)

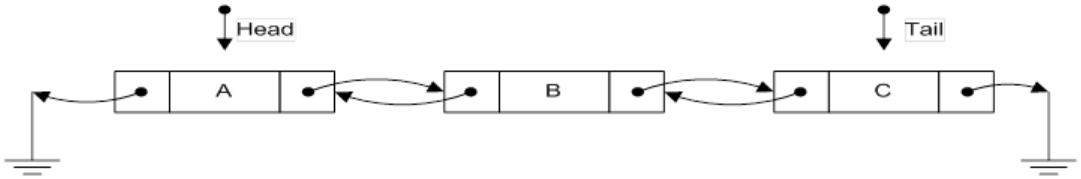
enqueue()

- Buat node baru pada *tail*,
- Isi *element* el pada node baru.



After enqueue(A)

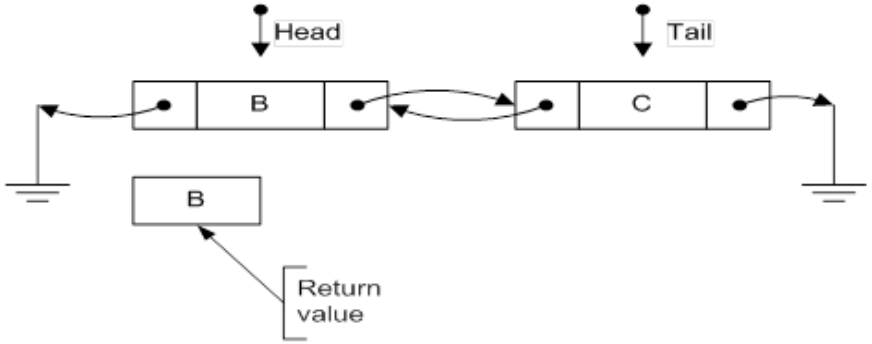
After enqueue(B)



After enqueue(C)

3.2. *Queue Operation* dengan *Linked List* (Lanj.)

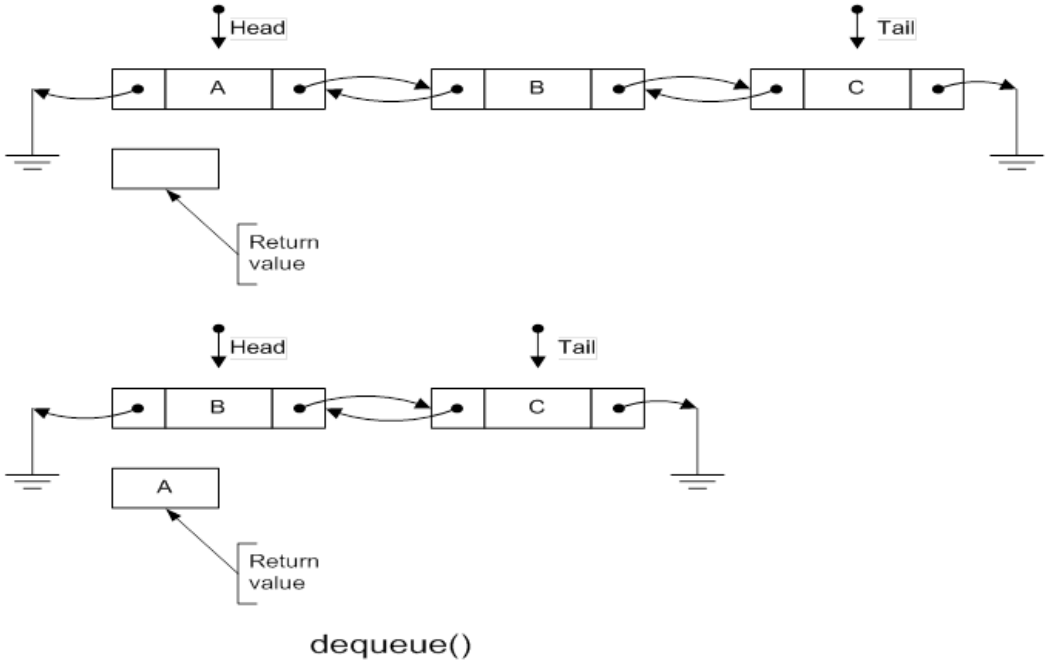
- firstEl() →
  - Ambil *element* dari node *Head*.



firstEl()

3.2. *Queue Operation* dengan *Linked List* (Lanj.)

- Dequeue →
  - Ambil *element* dari node yang ditunjuk oleh *Head*,
  - *set next* Node sebagai *Head*
  - *destroy* node yang semula *Head*



dequeue()

3.3. Tambahan

- Untuk penggunaan *Single List*, ketika melakukan dequeue,
  - anda harus menyimpan dahulu alamat *head* saat ini pada variabel temp.
  - Setelah data diambil dari *queue*, arahkan head ke node berikutnya.
  - Dan *free* kan node yang sebelumnya merupakan *head* yang sebelumnya yang sekarang ditunjuk oleh *variable* temp di atas
- Untuk penggunaan *double linked list*, anda cukup mengarahkan *head* ke node berikutnya, kemudian *free* kan node yang sebelumnya merupakan *head* yang sekarang ditunjuk oleh *Head*->Prev

Contoh Program Linked-List Base Stack

```
class Node:
    def __init__(self, dat):
        self.dat = dat
        self.next = None

class StackLinkedList:
    def __init__(self):
        self.top = None

    def push(self, simpulBaru):
        if not self.top:
            self.top = simpulBaru
        else:
            simpulBaru.next = self.top
            self.top = simpulBaru

    def pop(self):
        if not self.top:
            return None
        else:
            popped_node = self.top
            self.top = self.top.next
            popped_node.next = None  # Memutuskan koneksi dari tumpukan
            return popped_node
```

Contoh Program Linked-List Base Stack (Lanj.)

```
    def cetak(self):
        if not self.top:
            print("Stack Linked List Kosong")
        else:
            current = self.top
            while current:
                print("node: [", current.dat, "|", current.next, "]")
                current = current.next

# Implementasi Stack Linked List
a = [34, 77, 91, 23, 10, 32, 90, 60, 50, 11]
print("mulai")
stack_ll = StackLinkedList()

for i in range(0, 10):
    temp = Node(a[i])
    stack_ll.push(temp)

stack_ll.cetak()
print("ok")
print()
```

Contoh Program C++

```
#include <stdio.h>
#include <conio.h>
#include <iostream.h>
#include <string.h>
#include <iomanip.h>

//Record Definition
struct TheCell
{
  int dat;
  struct TheCell *sebelum;
  struct TheCell *berikut;
};
struct TheCell *ptrCell=NULL;
struct TheCell *first=NULL;
struct TheCell *last=NULL;
```

Contoh Program C++ (Lanj.)

```
void enqueue(isi)
{
  struct TheCell *baru;
  baru=(struct TheCell *) malloc(sizeof(struct TheCell));
  if (first!=NULL) //untuk mengecek stack kosong atau tidak.
                   //atau bisa juga pakai isEmpty()
  {  //kalau tidak kosong, baru-> diarahkan ke first
    baru->berikut = first;
  }
  else
  {
    baru->berikut = NULL;
  }
  baru->sebelum = NULL;
  first = baru;
  baru->dat = isi;
}
```

Contoh Program C++ (Lanj.)

```
int dequeue()
{
  if (first!=NULL)
  {
    int getData;
    getData = last->dat;
    ptrCell = last;
    last = last->sebelum;
    free(last->berikut);
    last->berikut=NULL
    return(getData);
  }
  else
  {
    return(NULL);
  }
}
```

Contoh Program C++ (Lanj.)

```
//program utama
void main()
{
  //deklarasi variable
  int i; int bilRandom;
  //pengisian bilangan random ke dalam stack
  for (i=1;i<=10;i++)
  {
    bilRandom = rand();
    enqueue(bilRandom);
  }
}
```

Contoh Program (Python)

```
class TheCell:
    def __init__(self, dat):
        self.dat = dat
        self.sebelum = None
        self.berikut = None

ptrCell = None
first = None
last = None

def main():
    # Your main program logic goes here

if __name__ == "__main__":
    main()
```

Contoh Program Python (Lanj.)

```
class TheCell:
    def __init__(self, dat):
        self.dat = dat
        self.sebelum = None
        self.berikut = None

def enqueue(isi):
    global first  # Assuming 'first' is a global variable

    baru = TheCell(isi)

    if first is not None:
        baru.berikut = first
    else:
        baru.berikut = None

    baru.sebelum = None
    first = baru

# Example usage:
# Assuming 'first' is initialized somewhere in your program
# and you can call enqueue like this:
# enqueue(your_data)
```

Contoh Program Python (Lanj.)

```
def dequeue():
    global first, last, ptrCell  # Assuming 'first', 'last', and 'ptrCell' are global
variables

    if first is not None:
        getData = last.dat
        ptrCell = last
        last = last.sebelum

        if last is not None:
            free_ptr = ptrCell.berikut
            del free_ptr
            ptrCell.berikut = None

        return getData
    else:
        return None

# Example usage:
# Assuming 'first', 'last', and 'ptrCell' are initialized somewhere in your program
# and you can call dequeue like this:
# result = dequeue()
```

Contoh Program Python (Lanj.)

```
import random

# Assume that enqueue is already defined as per the previous code

def main():
    # Deklarasi variable
    i = 0
    bilRandom = 0

    # Pengisian bilangan random ke dalam stack
    for i in range(1, 11):
        bilRandom = random.randint(0, 100)  # Adjust the range as needed
        enqueue(bilRandom)

# Example usage:
# If 'enqueue' function and 'TheCell' class are defined elsewhere, you can
call the 'main' function like this:
# main()
```

Contoh Program Python Queue

```
class Queue:
    def __init__(self):
        self.items = []

    def is_empty(self):
        return len(self.items) == 0

    def enqueue(self, item):
        self.items.append(item)

    def dequeue(self):
        if not self.is_empty():
            return self.items.pop(0)
        else:
            print("Queue is empty")
            return None

    def size(self):
        return len(self.items)

# Contoh penggunaan Queue
if __name__ == "__main__":
```

Ringkasan

- *Stack berbasis linked-list sama mudahnya digunakan seperti pada stack berbasis array*
- *Queue berbasis linked-list lebih mudah* diterapkan daripada Queue berbasis Array

PERINGATAN HAK CIPTA

PERINGATAN HAK CIPTA

Terimakasih

TUHAN Memberkati Anda

Teady Matius Surya Mulyana (tmulyana@bundamulia.ac.id)