LAPORAN TUGAS STRUKTUR DATA LINKED LIST



Disusun Oleh :

Kevin Indra Mahendra A11.2017.10325

Adis Wing Kasenda A11.2017.10316

**UNIVERSITAS DIAN NUSWANTORO**

**SEMARANG**

**2018**

# PENDAHULUAN

**Array** adalah sekumpulan variabel yang memiliki tipe data yang sama dan dinyatakan dengan nama yang sama. Array merupakan konsep yang penting dalam pemrograman, karena array memungkinkan untuk menyimpan data maupun referensi objek dalam jumlah banyak dan terindeks.

**Linked List** atau dikenal juga dengan sebutan senarai berantai adalah struktur data yang terdiri dari urutan record data dimana setiap record memiliki field yang menyimpan alamat/referensi dari record selanjutnya (dalam urutan). Elemen data yang dihubungkan dengan link pada Linked List disebut Node. Biasanya didalam suatu linked list, terdapat istilah head dan tail. **Single Linked List** merupakan suatu linked list yang hanya memiliki satu variabel pointer saja. Dimana pointer tersebut menunjuk ke node selanjutnya. Biasanya field pada tail menunjuk ke NULL.

**Perbedaan Array dan Single Linked List**

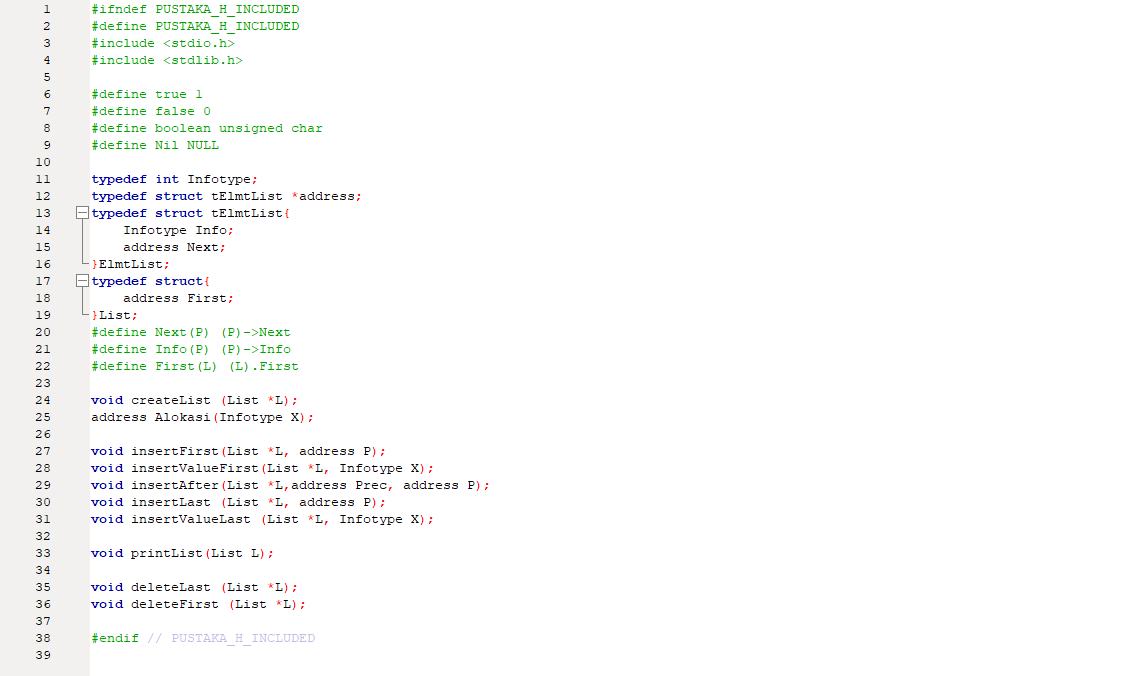
**ARRAY**

* Elemen data bisa menggunakan RECORD.
* Bersifat Statis
* volumenya selalu tetap tidak tergantung pada jumlah data.
* alokasi memori dilakukan pada saat array didefinisikan.
* pembebasan memori dilakukan pada saat program berhenti.
* Cara akses bersifat random dengan menggunakan nomor index.

**SINGLE LINKED LIST**

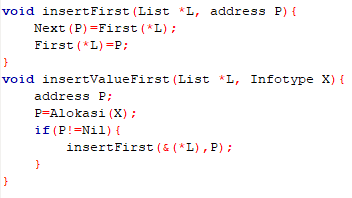
* Elemen data selalu menggunakan RECORD.
* Bersifat Dinamis
* ukurannya berubah-ubah disesuaikan dengan kebutuhan.
* alokasi memori ditentukan pada saat data baru dibuat.
* pembebasan memori dilakukan setiap ada penghapusan data.
* Cara akses ke masing-masing class data dilakukan secara linier (selalu dimulai dari elemen pertama).

# HEADER



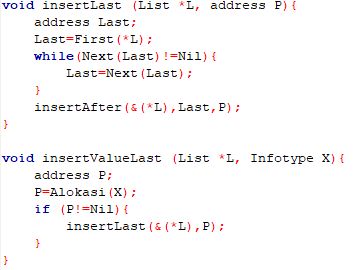
1. Mendefinisikan Tipe Data **Boolean**
2. Variabel **Nil** didefinisikan dengan NULL
3. Mendefinisikan tipe data **Infotype** bertipe integer
4. Mendeklarasikan tipe data bentukan **tElmList \*address** untuk menampung element array didalam list
5. Tipe data bentukan **ElmList** berisi variabel **Info** bertipe Infotype (integer) dan **Next** bertipe address (pointer struct **tElmtList)**
6. Mendeklarasikan tipe data bentukan **List** yang berisi variabel **First** bertipe address
7. Mendefinisikan macro **Next(P)** , **Info(P)** , **First(L)** menjadi **(P)->Next** , **(P)->Info** , **(L).First** untuk memudahkan dalam pemanggilan variabel.

# INSERT DEPAN



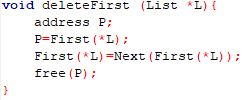
1. Fungsi **insertFirst** bertipe voiddengan parameter **List \*L, address P** untuk memasukan nilai pada index pertama. Alamat **Next(P)** memiliki nilai alamat **First(\*L)** karena fungsi ini untuk memasukan nilai pertama.
2. Fungsi **insertValueFirst** bertipe void dengan parameter **List \*L, Infotype X** berisi deklarasi variabel **P** bertipe address, lalu nilai **P** berisi nilai dari fungsi **Alokasi** berparameter **X** (Infotype).
3. Jika variabel **P** tidak sama dengan **Nil** maka akan memanggil fungsi **insertFirst** untuk memasukan nilai kedalam variabel **\*L** dengan alamat **P.**

# INSERT BELAKANG



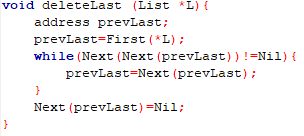
1. Fungsi **insertLast** bertipe void dengan parameter **List \*L, address P**  berisi deklarasi variabel **Last** bertipe address dan mengambil nilai dari variabel **First(\*L).**
2. Ketika variabel **Next** dengan alamat **Last** tidak sama dengan **Nil** maka variabel **Last** menjadi berisi nilai dari variabel **Next** dengan alamat **Last**. Lalu memanggil fungsi **insertAfter** yang akan dimasukan kedalam **L (pointer)** beralamat **P** dan didefinisikan sebagai nilai **Last** (terakhir atau belakang)
3. Fungsi **insertValueLast** bertipe void dengan parameter **List \*L, Infotype X** berisi deklarasi **address P** dengan nilai dari fungsi **Alokasi** berparameter **X** (index).
4. Jika variabel **P** tidak sama dengan **Nil** maka memanggil fungsi **insertLast** untuk memasukan nilai.

# HAPUS DEPAN



1. Fungsi **deleteFirst** bertipe void dengan parameter **List \*L** (sebagai penampung data) berisi deklarasi variabel **address P** bernilai dari variabel **First(\*L)** yang adalah nilai pertama**.**
2. Lalu **First(\*L)** diisi dengan variabel **Next** yang mengandung nilai **First(\*L).**
3. Setelah itu memanggil fungsi **free** yang digunakan untuk menghapus nilai berdasarkan alamat dari variabel **P.**

# HAPUS BELAKANG



1. Fungsi **deleteLast** bertipe void dengan parameter **List(\*L)** sebagai penampung data. Berisi deklarasi variabel **address prevLast** yang diisi dengan nilai dari variabel **First(\*L).**
2. Ketika variabel **Next** yang mengandung variabel **Next** dengan nilai terakhir yang dideklarasikan melalui variabel **prevLast** tidak sama dengan variabel **Nil,** maka variabel **prevLast** berisi nilai **Next** dengan nilai **prevLast** dan variabel **Next(prevLast)** menjadi berisi Nil atau bisa disebut kosong.

# REFERENSI

Sucianti, N. (2014, 3). *STRUKTUR DATA - Linked List*. Diambil kembali dari Catatan Kuliah TI: http://suciantinovi.blogspot.com/2014/03/linked-list-i\_14.html

Xanny. (2011, 3). *Perbedaan Array dengan Linked List pada struktur data*. Diambil kembali dari Satu Cinta Sejuta Makna: http://xannia.blogspot.com/2011/03/perbedaan-array-dengan-linked-list-pada.html