**Review Matakuliah Pratikum Struktur Data**

**FITRI WALDI**

Fakultas Teknik, Universitas Negeri Padang

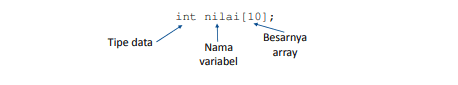
[Fitriwaldi02@sma.belajar.id](mailto:Fitriwaldi02@sma.belajar.id)

**I. Array**

Array atau larik adalah variabel yang terdiri dari sekumpulan data terstruktur yang mempunyai tipe data yang sama. Setiap data disimpan di alamat memori yang berbeda-beda yang disebut dengan elemen array. Setiap elemen mempunyai nilai index sesuai urutannya. melalui index inilah kita dapat mengakses data-data tersebut. Umumnya index dari elemen array dimulai dari 0.

Array juga bersifat statis, dimana alokasi memori dilakukan saat deklarasi source code.

Contoh deklarasi array:



* Dalam contoh diatas terdapat 10 variabel dengan tipe yang sama yaitu integer.

Untuk mengakses array harus ditentukan index yang diakses.

Contoh:

#include <stdio.h>

void main() {

int bilangan[10], i;

for(i=0;i<10;i++){

bilangan[i] = 2\*i;

printf("%d %d\n", i, bilangan[i]);

}

}

/\*

Output:

7 14

8 16

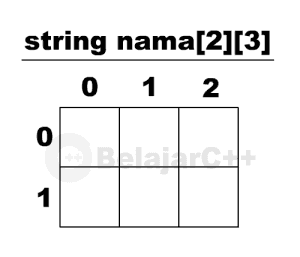
9 18

\*/

* Contoh diatas kita mengakses elemen array berdasarkan looping(perulangan) kemudian menginisiasikan valuenya dan kembali mengakses arraynya untuk dicetak kedalam output.

Terdapat juga jenis array yang lainnya yaitu:

* Array 2 Dimensi  
  Deklarasi array 2 dimensi: string nama[2][3]  
  Array dua dimensi memungkinkan kita untuk menggunakan array di dalam array, setiap array memiliki array di dalamnya.

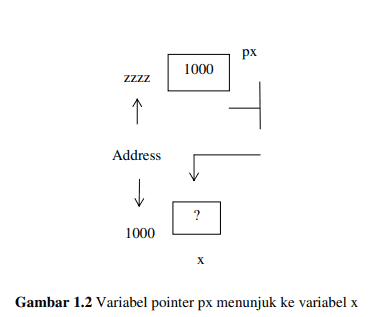
Array dua dimensi bisa digambarkan seperti sebuah tabel yang mempunyai baaris dan kolom array pertama adalah sebuah baris dan array kedua adalah sebuah kolom berikut ilustrasinya:  


* Array Multidimensi  
  Array multidimensi merupakan array yang mempunyai ukuran lebih dari dua. Jika kita sudah mengerti mengenai konsep dari array, kita akan dengan mudah untuk memodifikasi array, menambahkan beberapa subscript untuk menjadikanya array multidimensi. Secara garis besar, pembuatan, penggunaan, pengaksesan array multi dimensi memiliki konsep yang sama seperti apa yang telah penulis jelaskan di atas pada array dua dimensi.

**II. Pointer**

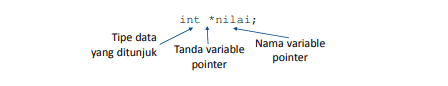
Pointer adalah variabel yang menunjuk ke lokasi alamat memori tertentu. Variabel pointer sering dikatakan sebagai variabel yang menunjuk ke obyek lain. Pada kenyataan yang sebenarnya, variabel pointer berisi alamat dari suatu obyek lain (yaitu obyek yang dikatakan ditunjuk oleh pointer). Sebagai contoh px adalah variabel pointer dan x adalah variabel yang ditunjuk oleh px. Kalau x berada pada alamat memori (alamat awal) 0x1000, maka px akan berisi 1000.

Lebih jelasnya mari kita lihat ilustrasi berikut:



Variabel pointer juga bersifat dinamis (kebalikan dari array), dimana alokasi memori dilakukan pada saat program dijalankan.

Contoh deklarasi pointer:



* Variabel nilai menunjuk ke variable lain yang bertipe integer

Berikut ini adalah contoh cara mengakses pointer:

#include <stdio.h>

void main() {

int a, \*b;

a = 5;

b = &a;

printf("%d %d\n", a, \*b);

}

/\*

Output:

5 5

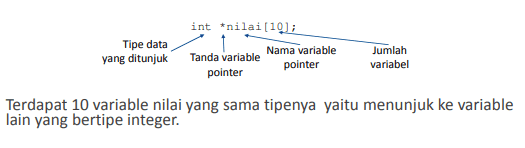
\*/

* variabel b menampung alamat memori variabel a

**Array of Pointer**

Array of pointer adalah sekumpulan array dengan tipe data pointer.

Contoh deklarasi array of pointer:



#include <stdio.h>

void main() {

int \*bil[3], a, b, c, i;

a = 5;

b = 10;

c = 15;

bil[0] = &a;

bil[1] = &b;

bil[2] = &c;

for(i=0;i<3;i++)

printf("%d\n", \*bil[i]);

}

/\*

Output:

5

10

15

\*/

* kita liat pada source code diatas setiap elemen arraynya menampung alamat memori dari variabel yang didefinisikan sebelumnya.
* kemudian kita tampilkan value dari array bil dengan method print maka valuenya akan berisi value dari alamat memori variabel yang direferensikan.

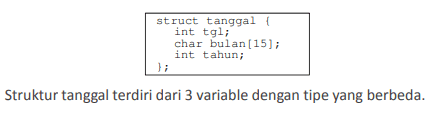
Hubungan antara pointer dan array pada C sangatlah erat. Sebab sesungguhnya array secara internal akan diterjemahkan dalam bentuk pointer.

Misalnya dideklarasikan di dalam suatu fungsi  
static int tgl\_lahir[3] = { 01, 09, 64 }; dan int \*ptgl;  
Kemudian diberikan instruksi  
ptgl = &tgl\_lahir[0];  
maka ptgl akan berisi alamat dari elemen array tgl\_lahir yang berindeks nol. Instruksi di atas bisa juga ditulis menjadi  
ptgl = tgl\_lahir;  
sebab nama array tanpa tanda kurung menyatakan alamat awal dari array. Sesudah penugasan seperti di atas,  
\*ptgl dengan sendirinya menyatakan elemen pertama (berindeks sama dengan nol) dari array tgl\_lahir.

**III. Structures / Struktur**

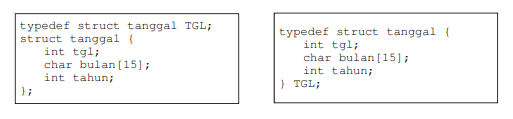
Struktur adalah sekumpulan variabel yang mempunyai tipe yang berbeda. Struktur dapat digunakan untuk mengelompokkan beberapa informasi yang berkaitan menjadi sebuah satu kesatuan. Struktur merupakan user defined variabel, yang artinya variabel yang didefinisikan sendiri oleh user.

Contoh deklrasi structure:



Deklarasi Struktur dengan typedef

* typedef berfungsi sebagai nama lain / alias dari tipe data.
* typedef digunakan untuk meringkas tipe data struktur sehingga mudah diakses.



Element dalam structures dapat diakses dengan bentuk berikut variabel\_struktur.nama\_field.

#include <stdio.h>

typedef struct tanggal {

int tgl;

char bulan[15];

int tahun;

} TGL;

void main(){

TGL a;

a.tgl = 17;

strcpy(a.bulan,"Agustus");

a.tahun = 1945;

printf("%d %s %d\n", a.tgl, a.bulan, a.tahun);

}

/\*

Output:

17 Agustus 1945

\*/

**Array Of Structures**

Elemen-elemen dari suatu array juga dapat berbentuk sebuah struktur. Misalnya array yang dipakai untuk menyimpan sejumlah data mahasiswa. Array struktur berdimensi satu ini membentuk suatu tabel, dengan barisnya menunjukkan elemen dari array-nya dan kolomnya menunjukkan elemen dari struktur. Dalam hal ini maka deklarasi yang dibutuhkan adalah sebagai berikut :

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct mahasiswa {

char nama[10];

int nim;

} MHS;

void main()

{

MHS mahasiswa\_baru[2];

int i;

strcpy(mahasiswa\_baru[0].nama, "Luffy");

mahasiswa\_baru[0].nim = 281820;

strcpy(mahasiswa\_baru[1].nama, "Sanji");

mahasiswa\_baru[1].nim = 210290;

printf("%s %s\n", "Nama","Nim");

for(i=0;i<2;i++)

printf(" %s %d \n",mahasiswa\_baru[i].nama, mahasiswa\_baru[i].nim);

}

/\*

Output:

Nam Nim

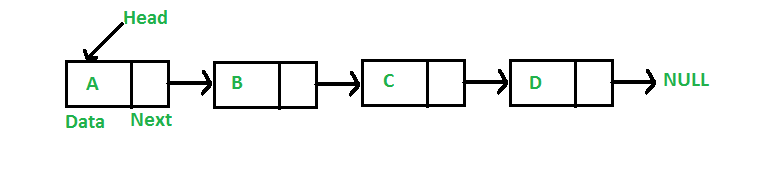
Luffy 281820

Sanji 210290

\*/

**IV. Linked List**

Linked list adalah [strukur data](https://www.trivusi.web.id/2022/06/mengenal-struktur-data.html) linier berbentuk rantai simpul di mana setiap simpul menyimpan 2 item, yaitu nilai data dan pointer ke simpul elemen berikutnya. Berbeda dengan [array](https://www.trivusi.web.id/2022/07/struktur-data-array.html), elemen linked list tidak ditempatkan dalam alamat memori yang berdekatan melainkan elemen ditautkan menggunakan pointer.



|  |
| --- |
|  |

Simpul pertama dari linked list disebut sebagai head atau simpul kepala. Apabila linked list berisi elemen kosong, maka nilai pointer dari head menunjuk ke NULL. Begitu juga untuk pointer berikutnya dari simpul terakhir atau simpul ekor akan menunjuk ke NULL.

Ukuran elemen dari linked list dapat bertambah secara dinamis dan mudah untuk menyisipkan dan menghapus elemen karena tidak seperti array, kita hanya perlu mengubah pointer elemen sebelumnya dan elemen berikutnya untuk menyisipkan atau menghapus elemen.

Linked list biasanya digunakan untuk membuat file system, adjacency list, dan hash table.

**Jenis Jenis Linked List**

Secara umum, linked list dapat dibagi ke dalam 4 jenis, yakni: Singly linked list, Doubly linked list, Circular linked list, dan Circular doubly linked list.

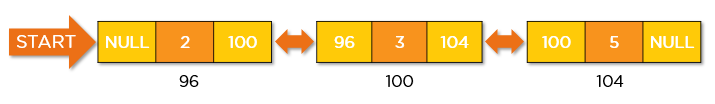
**1. Singly linked list**

Singly linked list adalah linked list unidirectional. Jadi, kita hanya dapat melintasinya dalam satu arah, yaitu dari simpul kepala ke simpul ekor.

|  |
| --- |
|  |
|  |

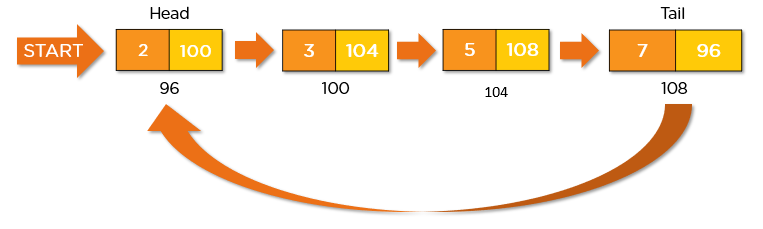
**2. Doubly linked list**

Doubly linked list adalah linked list bidirectional. Jadi, kita bisa melintasinya secara dua arah. Tidak seperti singly linked list, simpul doubly linked list berisi satu pointer tambahan yang disebut previous pointer. Pointer ini menunjuk ke simpul sebelumnya.



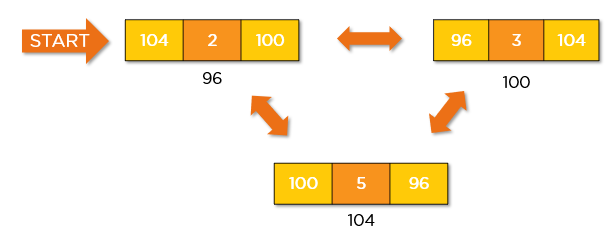
**3. Circular linked list**

Circular linked list adalah linked list unidirectional. Kita hanya dapat melintasinya dalam satu arah. Tetapi jenis linked list ini memiliki simpul terakhir yang menunjuk ke simpul kepala. Jadi saat melintas, kita harus berhati-hati dan berhenti saat mengunjungi kembali simpul kepala.



**4. Circular doubly linked list**

Circular doubly linked list adalah gabungan dari Doubly linked list dan Circular linked list. Seperti Doubly linked list, linked list ini memiliki pointer tambahan yang disebut previous pointer, dan mirip dengan Circular linked list, simpul terakhirnya menunjuk pada simpul kepala. Jenis linked list ini adalah bidirectional. Jadi, kita bisa melintasinya dua arah.



**Karakteristik Linked List**

Sebuah linked list memiliki beberapa karakteristik sebagai berikut:

* Linked list menggunakan memori tambahan untuk menyimpan link (tautan)
* Untuk inisialiasi awal linked list, kita tidak perlu tahu ukuran dari elemen.
* Linked list umumnya dapat digunakan untuk mengimplementasikan struktur data lain seperti stack, queue, ataupun graf
* Simpul pertama dari linked list disebut sebagai Head.
* Pointer setelah simpul terakhir selalu bernilai NULL
* Dalam struktur data linked list, operasi penyisipan dan penghapusan dapat dilakukan dengan mudah
* Tiap-tiap simpul dari linked list berisi pointer atau tautan yang menjadi alamat dari simpul berikutnya
* Linked list bisa menyusut atau bertambah kapan saja dengan mudah.

**Operasi-operasi pada Linked List**

Ada beberapa operasi yang bisa kita lakukan pada struktur data linked list. Misalnya, operasi insertion yaitu tindakan menambahkan elemen baru ke linked list.

Berikut adalah daftar operasi dasar pada linked list:

* Traversal - mengakses setiap elemen dari linked list
* Insertion - menambahkan elemen baru ke linked list
* Deletion - menghapus elemen yang ada
* Searching - menemukan simpul pada linked list
* Sorting - mengurutkan simpul dari struktur linked list

**DAFTAR PUSTAKA**

https://www.trivusi.web.id/2022/07/struktur-data-linked-list.html

https://muhammadfahri.com/array-pointer-struct/