MODUL I
JUDUL MODUL

1.1 TUJUAN

Tujuan dari praktikum ini adalah:

1. Mahasiswa dapat memahami konsep *array* dan *single linked list.*
2. Mahasiswa dapat mengimplementasikan *single linked list* ke dalam bahasa pemrograman.

1.2 DASAR TEORI

1.2.1 *Array*

Array atau larik adalah kelompok peubah tunggal atau multidimensi. Larik merupakan sejumlah nilai-nilai bertipe data sama yang bisa dirujuk menggunakan nama peubah yang sa a (posisi nilai tertentu dalam larik). mudah kan larik adalah dengan menggunakan awah ini yang memberi contoh suatu larik bilangan-bilangan bulat yang semuanya dirujuk dengan nama larik yang sama [1].

2	34	19	65	7	34	212	78	67	10
0	1	2	3	4	5	6	7	8	9

Gambar 1.1 Larik bilangan

Perhatikan larik pada **Gambar 1.1** tersebut. Nilai-nilai pada baris yang lebih atas (2, 34, 19, 65, dan seterusnya) adalah elemen-elemen dari larik yang masing-masing dapat diakses dengan menyebutkan nama larik diikuti dengan indeksnya (posisi nilai tertentu dalam larik) (0, 1, 2, 3, dan seterusnya). Berikut ini yaitu jenis-jenis *array*:

1. *Array* Satu Dimensi

   Array satu dimensi berupa vektor. Sesuai dengan namanya, elemen *array* satu dimensi dapat diakses melalui 1 buah indek eklarasian *array* satu dimensi:

```
tipe_data[] nama_array;
```

2. *Array* Dua Dimensi

Array dua dimensi berupa matriks/tabel yang jumlah datanya ditentukan oleh jumlah baris dan jumlah kolom. Sesuai dengan namanya, elemen *array* dua dimensi dapat diakses melalui 2 buah indeks, yaitu indeks baris dan indeks kolom (Elemen pada baris ke-i, kolom ke-j, dengan i indeks baris dan j indeks kolom). Bentuk umum pendeklarasian *array* satu dimensi:

```
tipe_data[][] nama_array = new type_data[][];
```

3. *Array* Tiga Dimensi

Array tiga dimensi dapat digambarkan sebagai suatu benda ruang. Deklarasi pada *array* tiga dimensi tidak berbeda pada *array* satu dimensi dan dua dimensi yang telah dijelaskan sebelumnya, kecuali pada indeks *array*.

```
tipe_data[][][] nama_array = new type_data[][][];
```

Indeks pertama menunjukkan baris, indeks kedua menunjukkan banyak isi baris, indeks ketiga menunjukkan banyak kolom [2].

1.2.2 *Single Linked List*

Single linked list atau biasa disebut *linked list* terdiri dari elemen-elemen individu, di mana masing-masing dihubungkan dengan *pointer* tunggal. Masing-masing elemen terdiri dari dua bagian, yaitu sebuah data dan sebuah *pointer* yang disebut dengan *pointer next*. Dengan menggunakan struktur *two-member* seperti ini, *linked list* dibentuk dengan cara menunjuk *pointer next* suatu elemen ke elemen yang mengikutinya. *Pointer next* pada elemen terakhir merupakan NULL, yang menunjukkan akhir dari suatu *list*. Elemen pada awal suatu *list* disebut *head*, dan elemen terakhir dari suatu *list* disebut *tail*. Berikut operasi-operasi yang terdapat *single linked list*:

1. Operasi Sisip
   a. Penyisipan di depan

      Penyisipan di depan pada *single linked list* adalah dengan cara menyisipkan data pada elemen awal *list*, sehingga *pointer* awal menunjuk *list* baru

   b. Penyisipan di tengah

      Penyisipan di tengah pada *single linked list* adalah dengan cara menyisipkan data baru setelah elemen yang ditunjuk oleh variabel bantu pada *list*, kemudian *pointer* variabel baru menunjuk *pointer* variabel tertentu.

Times New Roman, 11, tanpa titik diakhir

Tabel 1.1 Tabel perbedaan *array* dan *linked list*

No.	*Linked List*	*Array*
1.	Setiap elemen *linked list* terdiri dari 2 bagian, data dan *pointeraddress*.	Setiap elemen *array* hanya berisi data saja.
2.	Pengalokasian ruang memori dilakukan tanpa pendeklarasian sebelumnya dan terbatas pada jumlah ruang memori yang tersisa (dapat dipakai).	Pengalokasian ruang memori terbatas pada jumlah ruang yang dideklarasikan sebelumnya.
3	Elemen data selalu menggunakan RECORD.	Elemen data bisa menggunakan RECORD.
4	Bersifat Dinamis : • ukurannya berubah-ubah disesuaikan dengan kebutuhan. • alokasi memori ditentukan pada saat data baru dibuat. • pembebasan memori dilakukan setiap ada penghapusan data.	Bersifat Statis : • volumenya selalu tetap tidak tergantung pada jumlah data. • alokasi memori dilakukan pada saat *array* didefinisikan. • pembebasan memori dilakukan pada saat program berhenti.
5	Cara akses ke masing-masing *class* data dilakukan secara linier (selalu dimulai dari elemen pertama).	Cara akses bersifat random dengan menggunakan nomor indeks.

1. Pertama (tanpa paragraf dan untuk sub-sub seterusnya)
   a. Kedua
      1) Ketiga
         a) Keempat
            i. Kelima

Contoh Sitasi

"...didapati sekian [13]."

"Teorin ini muncul pada 2008 [1]."

"Lebah Ganteng [2] mengatakan bahwa..."

"Beberapa pembelajaran [3], [4], [15], [16] sudah membuktikan bahwa..."

"Sebagai contoh, lihatlah [7]."

1.3 PERMASALAHAN

1. Membuat *array* dinamis 2 dimensi.
2. Terdapat ilustrasi sebagai berikut:

   Di sebuah supermarket yang ramai, terdapat 6 orang pengunjung yang sedang mengantre untuk membayar di kasir, dengan nomor kode: 222, 231, 234, 275, 276, 280. Setelah 10 menit orang pertama selesai membayar dan meninggalkan supermarket. Kemudian orang yang berada di posisi pertama pada antrian terbaru meninggalkan antrian untuk pergi ke toilet, sehingga posisinya diambil oleh orang yang berada di belakangnya.

   Tiga menit kemudian, orang yang terdepan dalam antrian terbaru selesai melakukan pembayaran, kemudian antrian tersebut bertambah 2 orang dengan kode 282 dan 283 dan orang yang pergi ke toilet kembali. Melihat posisinya diambil oleh orang lain, orang tersebut marah-marah kepada orang yang mengambil posisinya sehingga orang yang mengambil posisinya memberikan posisinya pada orang yang marah, lalu ia mundur ke posisi belakang.

   Manajer supermarket memberi penghargaan kepada orang ke 1.000.000 dari supermarket tersebut jika orangterdepan pada antrian 999.995, berapa nomor kode pelanggan ke-1.000.000 tersebut?

   Namun, sebelum manajer menemui pelanggan ke-1.000.000 terjadi kebakaran sehingga orang yang mengantri kaburdengan urutan kabur yaitu orang terakhir, orang terakhir kedua, dan yang secara bersamaan.

1.4 HASIL DAN PEMBAHASAN

1.4.1 Judul Program

1. Algoritma
   a. Langkah pertama
   b. Langkah kedua
   c. Langkah ketiga
   d. Langkah keempat
   e. Langkah kelima
   f. Dan seterusnya
2. *Source code*

```
package modul.pkg1;
import java.util.Scanner;
/**
 *
 * @author Direct
 */
public class Array2 {

    public static void main(String[] args) {

        Scanner temp = new Scanner(System.in);
        int i,j;
        System.out.print("MASUKAN BARIS = ");
        i = temp.nextInt();
        System.out.print("MASUKAN KOLOM = ");
        j = temp.nextInt();
        int a[][] = new int[i][j];

        for (int z=0; z<i; z++){
            for (int y=0; y<j; y++) {
                System.out.println("Baris ke - "+ z +"\t kolom
ke - "+ y);
                a[z][y] = temp.nextInt(); }
        }
        System.out.println(" ");
     for (int z=0; z<i; z++) {
         for (int y=0; y<j; y++) {

             System.out.print("\t"+a[z][y]);
         }   System.out.println(" \n");
        }
    }
}
```

3. Hasil program

```
run:
MASUKAN BARIS = 2
MASUKAN KOLOM = 2
Baris ke - 0     kolom ke - 0
10
Baris ke - 0     kolom ke - 1
11
Baris ke - 1     kolom ke - 0
12
Baris ke - 1     kolom ke - 1
100

        10      11

        12      100
```

Gambar 1.2 Hasil *run* program *array* dinamis 2 dimensi

Pada **Gambar 1.2** membuktikan bahwa *array* 2 dimensi memiliki 2 indeks. Indeks pertama merupakan baris dan membentuk hasil *output* menjadi barisan dan indeks kedua merupakan kolom dan membentuk hasil *output* menjadi bentuk kolom.

1.5 ANALISA

1.5.1 Judul Program

```
public class Node{
  int data;
  Node next;
Node(int data){
  this.data=data;
  next=null;
}
```

1 cm

Pendeklarasian subkelas dalam kelas "`xxi`" dengan nama "`Node`" yang bersiat "`public`". Selain itu, juga dilakukan pendeklarasian atribut untuk kelas tersebut berupa sebuah variabel bertipe data "`int`" dengan nama "`data`" juga sebuah variabel bertipe data "`Node`" dengan nama "`next`".

Script "`Node(int data)`" merupakan kosntruktor untuk kelas "`Node`" dengan parameter sebuah variabel bertipe data "`int`" dengan nama "`data`". Lalu terdapat kumpulan instruksi didalamnya dimana nilai pada parameter (nilai variabel "`n`") dimasukkan pada "`this.data`". "`this`" merupakan ponter dimana menunjuk bahwa variabel "`data`" yang dimaksud ialah atribut kelas bukanlah parameter konstruktor itu sendiri. Setelah itu *field* untuk variabel "`next`" diberi "`null`".

1.6 KESIMPULAN

Dari praktikum yang telah di laksanakan dapat diambil beberapa kesimpulan, yaitu sebagai berikut:

1. *Array* merupakan kumpulan data yang memiliki tipe data yang sama. tiap data tersebut di bedakan dengan indeks dalam *array* tersebut, indeks dari tiap *array* dimulai dari "`[0]`". *Array* digunakan untuk mengurangi jumlah penggunaan variabel dalam program. dan *array* terdiri dari satu dimensi, dua, tiga dimensi dan seterusnya tergantung kebutuhan. Sementara *Linked list* (senarai bertautan; daftar bertautan) adalah salah satu cara untuk menyimpan sekumpulan elemen. Sama halnya dengan *array*, elemen yang disimpan dapat berupa karakter atau *integer*. Masing-masing elemen dalam *linked list* disimpan dalam bentuk sebuah node. *Linked list* terbentuk saat terdapat banyak node yang saling tertaut dan membentuk sebuah rantai. Setiap node menunjuk pada node selanjutnya sesuai urutan. Node pertama yang selalu digunakan sebagai referensi saat melakukan *traversing* pada *list* (senarai; daftar) disebut *head*. Node terakhir akan menunjuk ke *null*.
2. *Linked List* adalah koleksi data item yang tersusun dalam sebuah barisan secara linear, dengan penyisipan dan pemindahan dapat dilakukan dalam semua tempat di LL tersebut. *Array* adalah koleksi dari objek yang mempunyai tipe identik / sama, *array* dapat disebut juga koleksi data dengan setiap elemen data menggunakan nama yang sama dan masing-masing elemen mempunyai tipe data sama. *Array* dapat di-*loop* dengan memberi indeks setiap item di dalamnya, dan setiap komponen / *item array* dapat diakses dan dibedakan melalui indeks *array*. Setiap elemen *linked list* terdiri dari 2 bagian, data dan *pointer address*. Setiap elemen *array* hanya berisi data saja serta pengalokasian ruang memori dilakukan tanpa pendeklarasian sebelumnya dan terbatas pada jumlah ruang memori yang tersisa (dapat dipakai). Pengalokasian ruang memori terbatas pada jumlah ruang yang dideklarasikan sebelumnya. *LinkedList* dan *ArrayList* adalah dua implementasi yang berbeda dari interface *List*. *LinkedList* mengimplementasikannya dengan ganda-*linked list*. Seperti standar *linked list* dan berbagai operasi, berbagai metode akan memiliki *runtimes* algoritmik yang berbeda.