MAKALAH

ALGORITMA & STRUKTUR DATA

Untuk Memenuhi Tugas Mata Kuliah Algoritma & Struktur Data

Dosen : Heri Suhendar, S.T.,M.Kom.

Disusun oleh:

Melina Amelia (2206152)

Teknik Informatika - E

PROGRAM STUDI TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI GARUT

2023

PERTEMUAN I

PENGANTAR ALGORITMA

ALGORITMA ?

Algoritma adalah urutan langkah-langkah logis penyelesaian masalah yang disusun secara sistematis dan logis serta dapat ditentukan bernilai salah ataubenar.

Suatu Algoritma harus dinyatakan dalam bentuk yang dapat dimengerti olehpemroses.

- Mengerti setiap langkah dalam Algoritma.
- Mengerjakan operasi yang bersesuaian dengan langkah tersebut.Dalam membuat sebuah program, ada beberapa hal penting, yaitu:
- Tujuan pembuatan program.
- Algoritma.
- Bahasa pemrograman itu sendiri.

1. Masalah : Membuat algoritma diawali dengan adanya masalah
2. Algoritma : Membuat algoritma untuk menjadi pedoman pembuatan program
3. Program computer : Program komputer sesuai pedoman algoritma
4. Solusi : Solusi akan tercapai

Cara Mendeskripsikan Masalah

- Menjabarkan masalah
- Merinci masalah untuk menyelesaikan masalah
- Membuat sarana interaksi manusia – computer

Masalah > Program Komputer

- Bentuk Urutan Masalah
- Tentukan Bahasa Pemrograman
- Konsep Mesin Komputer

PERTEMUAN II
LANJUTAN DARI PENGANTAR ALGORITMA

CIRI-CIRI ALGORTIMA

- Algoritma mempunyai awal dan akhir.
- Setiap langkah harus didefinisikan dengan tepat sehingga tidak memilikiarti ganda.
- Memiliki masukan (input) atau kondisi awal.
- Memiliki keluaran (output) atau kondisi akhir.
- Algoritma harus efektif, bila digunakan benar-benar menyelesaikanpersoalan.

MEKANISME PELAKSANAAN ALGORITMA OLEH PEMROSES

Komputer hanyalah salah satu pemroses. Agar dapat dilaksanakan oleh komputer, algoritma harus ditulis dalam notasi bahasa pemrograman sehingga dinamakan program. Jadi program adalah perwujudan atau implementasi teknis Algoritma yang ditulis dalam bahasa pemrogaman tertentu sehingga dapat dilaksanakan olehkomputer.

Langkah-langkah penyelesaian dalam algoritma dapat ditulis dalam notasi apapun, asalkan mudah dibaca dan dimengerti, karena memang tidak ada notasi baku dalam penulisan algoritma. Tiap orang dapat membuat aturan penulisan dannotasi algoritma sendiri. Agar notasi algoritma mudah ditranslasi ke dalam notasibahasa pemrograman, maka sebaiknya notasi algoritma tersebut berkoresponden dengan notasi bahasa pemrograman secara umum.

NOTASI ALGORITMA

A. DESKRIPTIF

Langkah-langkah pemecahan masalah dituangkan secara narasi atau dengan untaian kalimat deskriptif.

B. FLOWCHART

Menggunkan simbol-simbol tertentu yang mnggambarkan urutan prosesnya (Diagram Alir)

C. PSEUDOCODE

Struktur bahasa yang mendekati notasi dari suatu bahasa pemrograman

STRUKTUR DASAR ALGORITMA

1. Struktur Runtunan (Sequence) yaitu digunakan untuk program yangpertanyaannya sequential atau urutan.
2. Struktur Pemilihan (Selection) yaitu digunakan untuk program yangmenggunakan pemilihan atau penyeleksian kondisi.
3. Struktur Perulangan (Iteration) yaitu digunakan untuk program yangpernyataannya dieksekusi berulang-ulang.

KARAKTERISTIK ALGORITMA

- Finiteness

  Proses dalam algoritma harus memiliki akhir dan jumlah langkahnya terbatas
- Definiteness

  Setiap langkah proses dalam algoritma harus jelas dan tidak ambigu
- Input / Output

  Algoritma harus menerima nol atau lebih input dan setidaknya menghasilkanoutput
- Efectiveness

  Langkah-langkah dalam algoritma harus efektif dan sesederhana mungkin

NOTASI ALGORITMA : DESKRIPTIF

- Judul Algoritma

  Bagian Judul, merupakan bagian yang terdiri atas nama algoritma dan penjelasanatau spesifikasi algoritma tersebut.
- Deklarasi

  Bagian Deklarasi, merupakan bagian untuk mendefinisikan semua nama yang digunakan pada algoritma dapat berupa variabel, konstanta, tipe ataupun fungsi.
- Deskripsi

  Bagian Deskripsi, merupakan bagian inti pada struktur algoritma yang berisiuraian langkah-langkah penyelesaian masalah.

  Contoh Notasi Deskripsi

  Algoritma Luas_Segitiga

  Menghitung luas segitiga dengan memasukan nilai alas dan tinggi

Deklarasi :

alas, tinggi, luas = bilangan bulatDeskripsi

1. Mulai
2. Masukan nilai alas dan tinggi
3. Hitung luas sama dengan setengah alas x tinggi
4. Tampilkan luas segitiga
5. Selesai

NOTASI ALGORITMA : FLOWCHART

A. Input : Flowchart merupakan penulisan algoritma dengan menggunakan notasigrafis.

B. Proses : Flowchart adalah bagan yang mempelihatkan tahapan dari suatu programdan hubungan antar proses beserta penyataannya.

C. Output : Ilustrasi ini dinyatakan dalam simbol, setiap simbol mempunyai makna tertentu untuk proses tertentu.

NOTASI ALGORITMA : PSEUDOCODE

Program

Pseudecode merupakan cara penulisan algoritma yang menyerupai bahasa pemrograman tingkat tinggi.

Deklarasi

Pada umumnya notasi pseudecode menggunakan bahasa yang mudahdimengerti secara umum dan juga lebih ringkas dari pada algoritma.

Deskripsi

Supaya notasi pseudecode dapat dimengerti oleh komputer maka harus diterjemahkan ke dalam sintaks bahasa pemrograman tertentu.

PARADIGMA PEMROGRAMAN

- Simulasi
- Analisis Kebutuhan (Requirement) dan Design
- Menulis Program
- Debugging dan Testing
- Membaca Program

  Membuktikan Kebenaran Program Secara Formal

Algoritma ?

- Penyusunan aspek proses logika dari suatu pemecahan masalah tanpa melihat karakteristik bahasa pemrograman yang akan digunakan.
- Urutan notasi logika yang merupakan hasil analisis dan rancangan sistematik dari strategi pemecahan masalah, untuk menggambarkan urutanlangkah kerja yang jika dikerjakan akan membawa ke tujuannya.
- Urutan logika langkah kerja untuk menyelesaikan suatu masalah.

PERTEMUAN III

VARIABEL & KONSTANTA, TIPE DATA

Variabel dan Konstanta

A. **Variabel :** adalah suatu elemen yang nilainya berubah-ubah atau tidak tetap. Setiap data yang disimpan dalam komputer memerlukan variabel sebagai sesuatu tempat untuk menyimpan nilai dari data tersebut, dan nilainya suatu variabel dapat berubah-ubah selama proses program.

B. **Konstanta :** adalah suatu elemen yang nilainya sudah di tetapkan. Konstanta adalah variabel yang memiliki nilai tetap, sekali konstanta diberi nilai maka selama proses program berjalan nilai konstanta tidak akan berubah. Konstantabiasanya digunakan untuk menyimpan nilai-nilai tertentu yang bersifat tetap.

Konsep Variabel dan Konstanta

- Konstanta & variabel hanya menyimpan data secara sementara (Volatile /RAM).
- Nilai konstanta tidak dapat diubah selama program dijalankan, sedangkan nilai variabel dapat diubah selama program dijalankan.
- Konstanta & variabel hanya dapat menyimpan satu data.
- Nilai dari suatu konstanta & variabel adalah nilai yang terakhir.

Aturan Variabel dan Konstanta

- Harus dimulai huruf alfabet
- Huruf kecil atau huruf kapital dibedakan
- Karakter penyusun variabel hanya boleh huruf alfabet, angka dan garis bawah (underscore)
- Tidak boleh menggunakan spasi (karakter putih)
- Penamaan konstanta & variabel sebaiknya mencerminkan nilai yangdikandungnya.

Syarat-syarat penamaan variabel antara lain :

- **Tidak boleh diawali dengan angka**

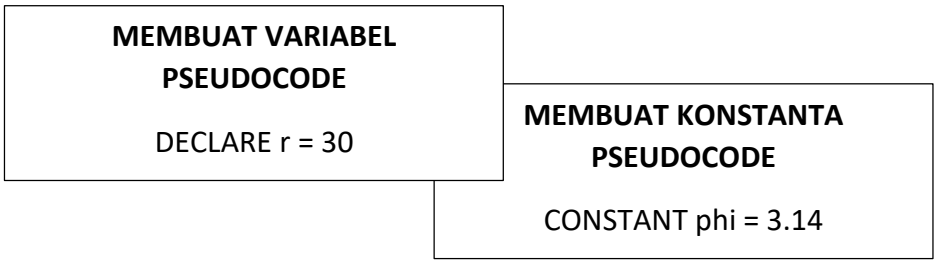
  contoh : *3example* (salah)

  solusi : *example3* atau *tigaexample*

- **Tidak boleh menggunakan spasi**

  contoh : *total harga* (salah)

  solusi : *total_harga* atau *totalHarga*
- **Tidak boleh menggunakan simbol khusus(kecuali underscore "_")**



TIPE DATA

Tipe data digunakan untuk merepresentasikan nilai suatu data (sesuai denganbahasa pemrograman yang digunakan).

Tipe data merupakan klasifikasi jenis dari data yang kita ingin simpan dalamsebuah variabel.

A. **Jenis-jenis tipe data:**

1. **Bilangan Bulat**

   Digunakan untuk menyatakan bilangan yang tidak mempunyai pecahan desimal, misalnya: 6, 18, 2500, -9, 67845, dll.

2. **Bilangan Pecahan**

   Digunakan untuk menyatakan bilangan yang mempunyai pecahan desimal, misalnya 9.25, 0.067, 22.5, dll.

3. **Karakter/Teks**

   Karakter digunakan untuk menyatakan simbol (angka,huruf, special character). **Teks** merupakan kumpulan dari karakter**. Boolean** Digunakan untuk menyatakan nilai Benar/ True/ 1 atau Salah/ False/ 0

PERTEMUAN IV

LARIK (ARRAY)

Variabel Larik atau lebih dikenal dengan ARRAY adalah Tipe terstruktur yangterdiri dari sejumlah komponen-komponen yang mempunyai tipe yang sama.

Suatu Array mempunyai jumlah komponen yang banyaknya tetap. Banyaknya komponen dalam suatu larik ditunjukan oleh suatu indeks untuk membedakanvariabel yang satu dengan variabel yang lainnya.

Empat sifat dasar dari sebuah array adalah :

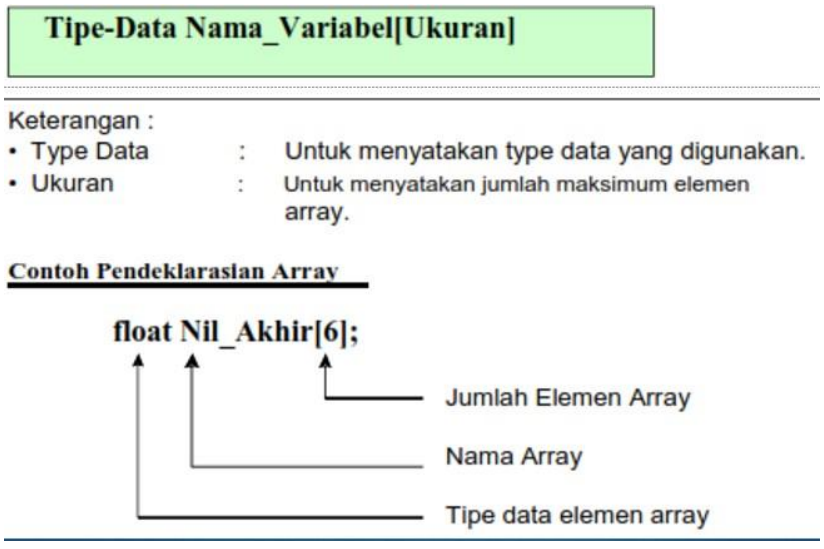
- Item-item data individual dalam array disebut elemen.
- Semua elemen harus berasal dari jenis data yang sama.
- Semua elemen disimpan secara berdampingan dalam memori komputer, dan subskrip (atau indeks) dari elemen pertama adalah nol.
- Nama array adalah nilai konstanta yang merepresentasikan alamat dari elemen pertama dalam array tersebut.

Jenis-Jenis Larik (Array)

1. **Array Berdimensi Satu**

   Pendeklarasian Array Berdimensi Satu

   Sebelum digunakan, variabel array perlu dideklarasikan terlebih dahulu. Cara mendeklarasikan variabel array sama seperti deklarasi variabel yang lainnya, hanya saja diikuti oleh suatu indek yang menunjukan jumlah maksimum data yang disediakan.

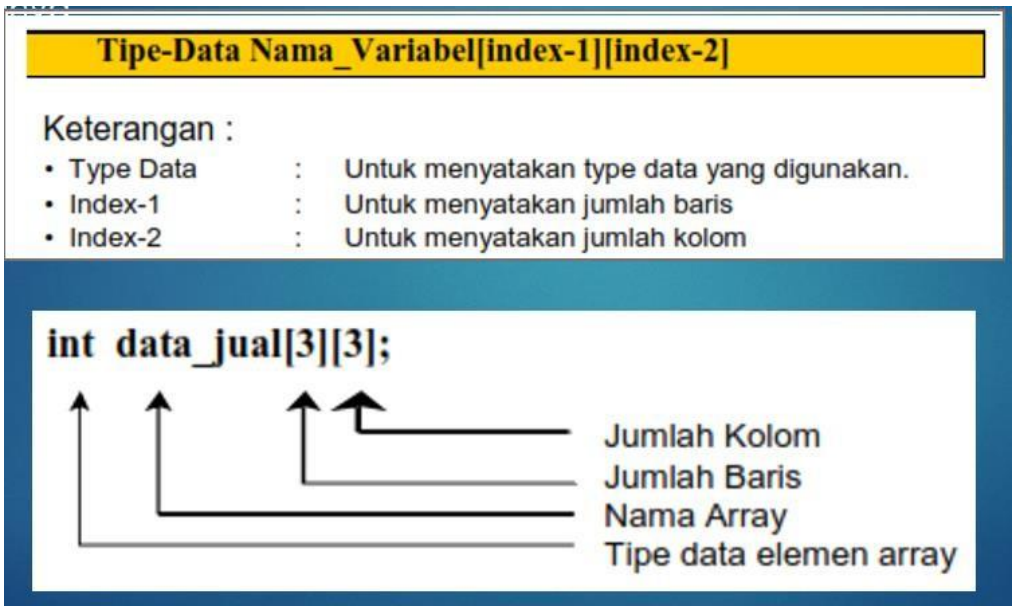


Contoh Program (Bahasa C):

```
/*.....*/
/* Array Berdimensi Satu  */
/*.....*/#include "stdio.h"
#include "conio.h" #include "iostream"void main()
{
int index,nilai[10];
printf("Masukkan 10 nilai Mahasiswa: \n");
for(index=0;index<10;index++)
{
printf("Mahasiswa %i: ",index+1);
scanf("%i",&nilai[index]);
}
printf("Nilai Mahasiswa yang telah dimasukkan : \n");
for(index=0;index<10;index++)
{
printf("%5.0i",nilai[index]);
```

2. Array Berdimensi DuaPendeklarasian Array Berdimensi Dua

Array dimensi dua tersusun dalam bentuk baris dan kolom, dimana indeks pertama menunjukan baris dan indeks kedua menunjukan kolom. Array dimensi dua dapat digunakan seperti pendatan penjualan, pendataan nilai dan lain sebagainya.



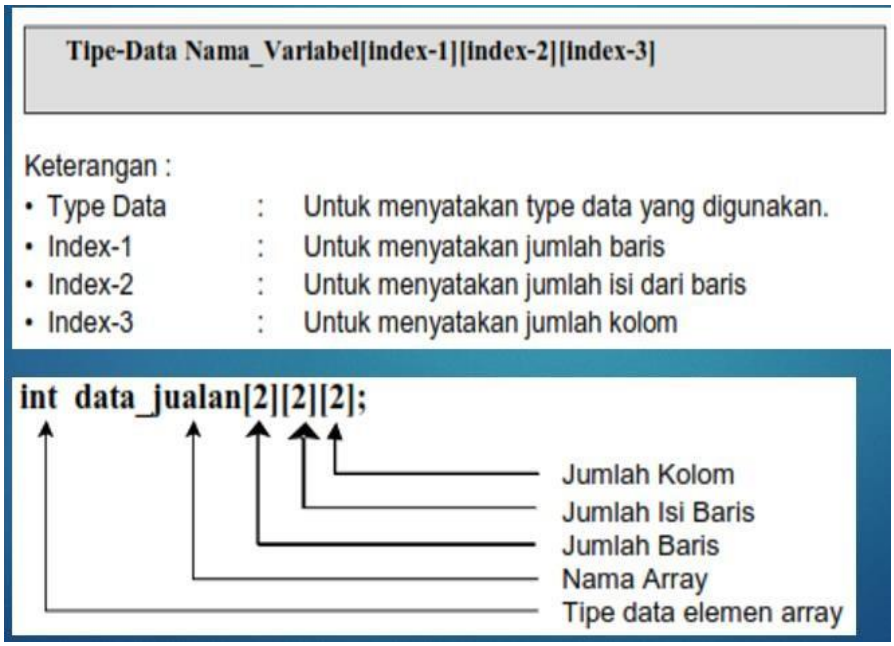
Contoh Program (Bahasa C):

```
/*-----------------------------*/
/* Array Berdimensi Dua  */
/*-----------------------------*/          #include<stdio.h>
#include<conio.h> #include<iostream.h>
void main()
{
int i, j;
int data[2][5] = {{2, 3, 4, 5, 2},{4, 2, 6, 2, 7}};
clrscr(); for(i=0;i<2;i++)
{
for(j=0;j<5;j++)
{
cout<<data[i][j];cout<<" ";
}
cout<<endl;
}
```

```
getch();
}
```

3. Array Berdimensi TigaPendeklarasian Array Berdimensi Tiga

Array dimensi tiga tersusun dalam bentuk baris, kolom dan isi dari baris, dimana indeks pertama menunjukan baris, indeks kedua menunjukan kolom dan indeks ketiga menunjukan isi dari baris.



Contoh Program Bahasa Python:

```
#  array 3D untuk menyimpan data nilai mahasiswa
data_nilai = []

# Fungsi untuk menambahkan data nilai mahasiswa
def tambah_data_nilai():
    # Input jumlah kelas
    jml_kelas = int(input('Masukkan jumlah kelas: '))

    # perulangan/loopong untuk input data nilai mahasiswa tiap kelas
    for i in range(jml_kelas):
        kelas = input(f'Masukkan nama kelas {i+1}: ')

        # Input jumlah mahasiswa
        jml_mhs = int(input(f'Masukkan jumlah mahasiswa kelas {kelas}: '))

        # array untuk menyimpan data nilai mahasiswa tiap kelas
        arr_kelas = [['NIM', 'Nama', 'Kelas', 'Nilai Tugas', 'Nilai UTS', 'Nilai UAS', 'Nilai Akhir']]

        # Looping untuk input data nilai mahasiswa
        for j in range(jml_mhs):
            nim = int(input(f'Masukkan NIM mahasiswa ke-{j+1} kelas {kelas}: '))
            nama = input(f'Masukkan nama mahasiswa ke-{j+1} kelas {kelas}: ')
            n_tugas = int(input(f'Masukkan nilai tugas mahasiswa ke-{j+1} kelas {kelas}: '))
            n_uts = int(input(f'Masukkan nilai UTS mahasiswa ke-{j+1} kelas {kelas}: '))
            n_uas = int(input(f'Masukkan nilai UAS mahasiswa ke-{j+1} kelas {kelas}: '))

            # Menambahkan data nilai mahasiswa ke array kelas
            arr_kelas.append([nim, nama, kelas, n_tugas, n_uts, n_uas, 0])

        # Menambahkan array kelas ke array data_nilai
```

```
        data_nilai.append(arr_kelas)

# Fungsi untuk menghitung nilai akhir
def hitung_nilai_akhir(array):
    for i in range(len(array)):
        for j in range(1, len(array[i])):
            nilai_akhir = 0.3 * array[i][j][3] + 0.3 * array[i][j][4] + 0.4 * array[i][j][5]
            array[i][j][6] = nilai_akhir

# function untuk menampilkan data nilai mahasiswa
def tampilkan_data_nilai(array):
    for i in range(len(array)):
        print(f'Data Nilai Mahasiswa Kelas {array[i][1][2]}:')
        for j in range(1, len(array[i])):
            print('NIM: ', array[i][j][0])
            print('Nama: ', array[i][j][1])
            print('Nilai Tugas: ', array[i][j][3])
            print('Nilai UTS: ', array[i][j][4])
            print('Nilai UAS: ', array[i][j][5])
            print('Nilai Akhir: ', array[i][j][6])
            print()

# Menambahkan data nilai mahasiswa
tambah_data_nilai()

# Function untuk menghitung nilai akhir dan menampilkan data nilai mahasiswa
hitung_nilai_akhir(data_nilai)
tampilkan_data_nilai(data_nilai)
```

PERTEMUAN V

FUNGSI MODULAR

Pengertian Fungsi Modular :

- Suatu teknik dimana program yang biasanya cukup besar dibagi-bagi menjadi beberapa bagian yang lebih kecil
- Modul atau function dideklarasikan dalam program ataupun terpisah
- Fungsi digunakan jika kita dihadapkan oleh suatu permasalahan yang besar, sehingga perlu di pecah untuk mempermudah penyelesaiannya
- Fungsi dibagi 2

1. Fungsi bawaan (standar library function)
2. Fungsi yang dibuat programmer (programmer defined function)

```
1   #include <iostream>
2   using namespace std;
3
4   void luas(int &ls, int p, int l) //definisi fungsi luas
5   { ls = p*l; }                    //isi fungsi
6
7   int main(){
8
9    int pj,lb, hsl;
10   cout<<"Panjang = ";cin>>pj;
11   cout<<"Lebar = ";cin>>lb;
12   luas(hsl,pj,lb);          //pemanggilan fungsi
13   cout<<"\nLuasnya = "<<hsl;
14   return 0;}
```

Standar Library Function

Fungsi-fungsi yang telah disediakan oleh C dalam file-file header atau *library*nya.

Deklarasikan dahulu library untuk bisa menggunakan fungsi tersebutContoh

#include <conio.h>

Programmer Defined Function

- fungsi yang dibuat oleh programmer sendiri. Fungsi ini memiliki namatertentu yang unik dalam program
- letaknya terpisah dari program utama, dan bisa dijadikan satu ke dalamsuatu *library* buatan

Sorting Pendahuluan

- Suatu proses menyusun data / himpunan menggunakan aturan tertentu

- Menurut Microsoft book-self algoritma pengurutan adalah algoritma untuk meletakkan kumpulan elemen data ke dalam urutan tertentu berdasarkan satu atau beberapa kunci dalam tiap-tiap elemen
- Urut naik (ascending) yaitu data yang mempunyai nilai paling kecilsampai paling besar
- Urut menurun (descending) yaitu data yang mempunyai nilai paling besarsampai paling kecil
- Metode untuk pengurutan yaitu :

1. Algoritma Bubble Sort
2. Algoritma Selection Sort
3. Algoritma Insertion Sort

Algoritma Bubble Sort

- Disebut juga pengurutan gelembung karena diinspirasi oleh gelembung sabun yang ada di permukaan air.
- Elemen yang berharga paling kecil "diapungkan" / diangkat ke ujung paling kiri melalui pertukaran

  Proses pengapungan ini dilakukan N kali langkah. Pada langkah ke-I, Larik[1..N] akan terdiri dari 2 bagian yaitu:

  – Bagian yang sudah terurut yaitu L[1]..L[i].
  – Bagian yang belum terurut L[i+1]..L[n].

Contoh Program Bubble Sort Ascending by Python

```python
def bubble_sort(arr):
    n = len(arr)

    # melakukan iterasi sebanyak (n-1) kali
    for i in range(n-1):

        # membandingkan elemen ke-i dengan elemen berikutnya
        for j in range(0, n-i-1):

            # jika elemen ke-j lebih besar dari elemen berikutnya, tukar posisinya
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]

    return arr

# contoh penggunaan
array = [1, 50, 10, 3, 2]
print("Array yang belum terurut:", array)
sorted_array = bubble_sort(array)
print("Array yang sudah terurut:", sorted_array)
```

```
Hasil Urut Ascending :
1
2
3
10
50
```

Contoh Program Bubblesort Descending C++

```cpp
#include <iostream>
using namespace std;

void bubbleSort(int arr[], int n) {
    // melakukan iterasi sebanyak (n-1) kali
    for (int i = 0; i < n-1; i++) {

        // membandingkan elemen ke-i dengan elemen berikutnya
        for (int j = 0; j < n-i-1; j++) {

            // jika elemen ke-j lebih kecil dari elemen berikutnya, tukar posisinya
            if (arr[j] < arr[j+1]) {
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}

int main() {
    int arr[] = {1, 50, 10, 3, 2};

    int n = sizeof(arr)/sizeof(arr[0]);

    // mencetak array sebelum diurutkan
    cout << "Array sebelum diurutkan: ";
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;

    bubbleSort(arr, n);

    // mencetak array setelah diurutkan
    cout << "Array setelah diurutkan secara descending: ";
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;

    return 0;
}
```

```
Hasil Urut Descending :
50
10
3
2
1
```

Contoh Program Bubblesort Descending Python

```python
def bubbleSort(arr):
    n = len(arr)

    # melakukan iterasi sebanyak (n-1) kali
    for i in range(n-1):

 # membandingkan elemen ke-i dengan elemen berikutnya
        for j in range(n-i-1):

            # jika elemen ke-j lebih besar dari elemen berikutnya, tukar posisinya
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
    return arr
arr = [1, 50, 10, 3, 2]

# mencetak array sebelum diurutkan
print("Array sebelum diurutkan:", arr)

# memanggil fungsi bubbleSort untuk mengurutkan array
arr = bubbleSort(arr)

# mencetak array setelah diurutkan
print("Array setelah diurutkan:", arr)
```

```
Hasil Urut Descending :
50
10
3
2
1
```

Algoritma Selection Sort

- Metode pengurutan maksimum/minimum karena berdasarkan pada pemilihan elemen maksimum/minimum kemudian ditukarkan ke elemenlarik diujung kiri atau kanan
- Selanjutnya elemen terujung di “isolasi” dan tidak diikut sertakan pada proses selanjutnya

Contoh Program selection Sort Python

```python
def selectionSort(arr):
    n = len(arr)

    # melakukan iterasi sebanyak (n-1) kali
    for i in range(n-1):

        # mencari nilai minimum di antara elemen yang belum
diurutkan
        min_idx = i
        for j in range(i+1, n):
            if arr[j] < arr[min_idx]:
                min_idx = j

        # menukar elemen ke-i dengan elemen minimum
        arr[i], arr[min_idx] = arr[min_idx], arr[i]

    return arr

# meminta input elemen dari user
arr = list(map(int, input("Masukkan elemen array (pisahkan
dengan spasi): ").split()))

# mencetak array sebelum diurutkan
print("Array sebelum diurutkan:", arr)

# memanggil fungsi selectionSort untuk mengurutkan array
arr = selectionSort(arr)

# mencetak array setelah diurutkan
print("Array setelah diurutkan:", arr)
```

Algoritma Insertion Sort

- Metode pengurutan dengan cara menyisipkan elemen larika pada posisi yang tepat
- Pencarian posisi yang tepat dilakukan dengan pencarian beruntun
- Selama pencarian posisi yang tepat dilakukan pergeseran elemen larik

- Pengurutan dimulai dari elemen ke dua

  Langkah-Langkah

  ▶ **L[1]** dianggap sudah pada tempatnya.

- **Langkah 2:**

  L[2] harus dicari tempatnya yang tepat pada L[1..2] dengan cara menggeser elemen L[1] ke kanan bila L[1] lebih besar dari L[2]. Misalkan posisi elemen yang tepat adalah K sisipkan L[2] pada K.

- **Langkah 3:**

  L[3] harus dicari tempatnya yang tepat pada L[1..3] dengan cara menggeser elemen L[1..2] ke kanan bila L[1..2] lebih besar dari L[3]. Misalkan posisi elemen yang tepat adalah K sisipkan L[3] pada K.

- **Langkah 4:**

  L[4] harus dicari tempatnya yang tepat pada L[1..4] dengan cara menggeser elemen L[1..4] ke kanan bila L[1..4] lebih besar dari L[4]. Misalkan posisi elemen yang tepat adalah K sisipkan L[4] pada K.

- **Langkah N:**

  L[N] harus dicari tempatnya yang tepat pada L[1..N] dengan cara menggeser elemen L[1..N ke kanan bila L[1..N] lebih besar dari L[N]. Misalkan posisi elemen yang tepat adalah K sisipkan L[N] pada K.

```cpp
#include <iostream>
using namespace std;

void insertionSort(int arr[], int n) {
    int i, key, j;
    for (i = 1; i < n; i++) {
        key = arr[i];
        j = i - 1;

        /* Pindahkan elemen arr[0..i-1] yang lebih besar dari
           kunci ke posisi yang satu nilai lebih tinggi dari posisi
sebelumnya */
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}
int main() {
    int n;
    cout << "Masukkan jumlah elemen array: ";
    cin >> n;
    int arr[n];
    cout << "Masukkan elemen array:\n";
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    // mencetak array sebelum diurutkan
    cout << "Array sebelum diurutkan: ";
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
    // memanggil fungsi insertionSort untuk mengurutkan array
    insertionSort(arr, n);
    // mencetak array setelah diurutkan
    cout << "Array setelah diurutkan: ";
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
```

```
Data setelah diurutkan :
1 2 10 25 30
```

PERTEMUAN VI

LINKED LIST

PENDAHULUAN

- Linked List adalah salah satu bentuk struktur data, berisi kumpulan data (node) yang **tersusun** secara sekuensial, **saling sambung-menyambung, dinamis** dan **terbatas**.
- Linked List sering disebut juga Senarai Berantai atau Senarai sendiri berarti daftar dan berantai bisa berarti sesuatu hal yang saling terhubung.
- Linked List saling terhubung dengan bantuan variabel pointer
- Masing-masing data dalam Linked List disebut dengan node (simpul) yang menempati alokasi memori secara dinamis dan biasanya berupa struct yang terdiri dari beberapa field.

Sederhananya, *linked list* adalah sebuah struktur data *llinear* yang setiap datanya terhubung satu sama lain.

Istilah dalam Linked List

- Node adalah simpul pada linked list yang menampung elemen/data dan pointer, *node* ditujukan untuk mewakili satuan data di dalam *linked list.*
- Pointer adalah bagian dari node yang menyimpan lokasi atau alamat memory node yang lain. Berfungsi menyambungkan setiap simpul yang ada antar node.
- Head adalah elemen yang berada pada posisi pertama / paling depan dalam suatu linked list
- Tail adalah elemen yang berada pada posisi terakhir / paling belakang dalam suatu linked list

Mengapa Linked List

- Mudah untuk menambahkan dan menghapus elemen（pada array tidak mungkin menambahkan elemen, karena banyaknya elemen sudah ditentukan dari awal）
- Panjang list bisa diubah dengan bebas (panjang array fixed)
- Mudah untuk menyambungkan beberapa list, maupun memutuskannya (array tidak bisa)
- Memungkinkan user mendesain struktur data yang kompleks

Struktur Linked List

- Node (elemen) *linked list* saling berkait melalui pointer. Bagian next sebuah node menunjuk alamat node selanjutnya
- *pHead*: pointer yang menunjuk node pertama

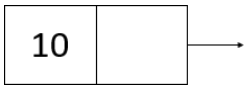
- Node terakhir menunjuk NULL
- Setiap node terdiri atas
  - Isi data
  - Next, yaitu pointer ke node selanjutnya pada list

Cara Kerja Link List

Misalkan ada dua buah simpul yang masuk secara beurutan, simpul pertama berisi angka 10 dan simpul kedua dengan 20.

Di saat program menerima angka 10, program akan mengalokasi memori yang nantinya akan bertugas sebagai wadah dari angka 10 tersebut. Kemudian, wadah ini juga akan menyimpan satu data lagi yang nantinya akan berisikan data berupa *pointer*.

Baik angka 10 dan data *pointer* ini akan terbungkus ke dalam satu buah *node* atau simpul. Karena simpul saat ini masih berjumlah satu, maka nilai dari *pointer* di dalam simpul pertama masih *NULL* atau tidak menunjuk ke alamat tertentu.



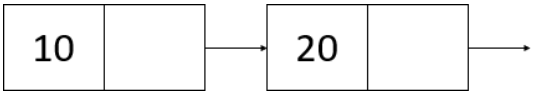
Kemudian simpul dengan data 20 datang dan program akan kembali mengalokasi memori untuk menjadi simpul yang baru.

Masih sama dengan yang terjadi pada simpul pertama, *pointer* pada simpul ini masih bernilai *NULL*.

Jadi sekarang sudah ada dua buah simpul yang tidak belum terhubung dan masing-masing memiliki alamatnya sendiri di memori komputer.

Agar dapat terhubung, kita akan mengubah nilai dari variabel *pointer* pada simpul yang pertama. Nilai yang awalnya *NULL* akan berubah menjadi alamat dari simpul kedua.

Maka sekarang kedua simpul sudah saling terhubung melalui nilai dari *pointer* simpul pertama.



Deklarasi Linked List

Pada C Deklarasi sebuah node dapat dilakukan dengan struct mhs{

//bagian data

tipedata data 1; int nim; tipedata data 2; string nama;

…

tipedata data n;

//pointer ke node selanjutnya struct node *next;

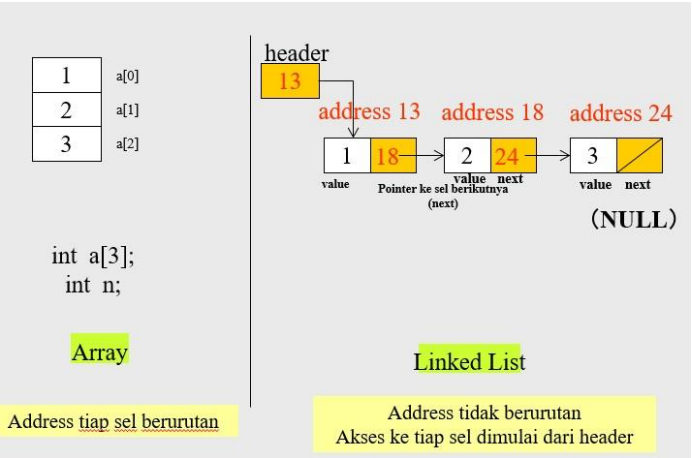
};

typedef struct node node;

Array VS Linked List

ARRAY	LINKED LIST
Statis	Dinamis
Penambahan / penghapusan data terbatas	Penambahan / penghapusan data tidak terbatas
Random access	Sequential access
Penghapusan array tidak mungkin	Penghapusan linked list mudah

Array Vs List



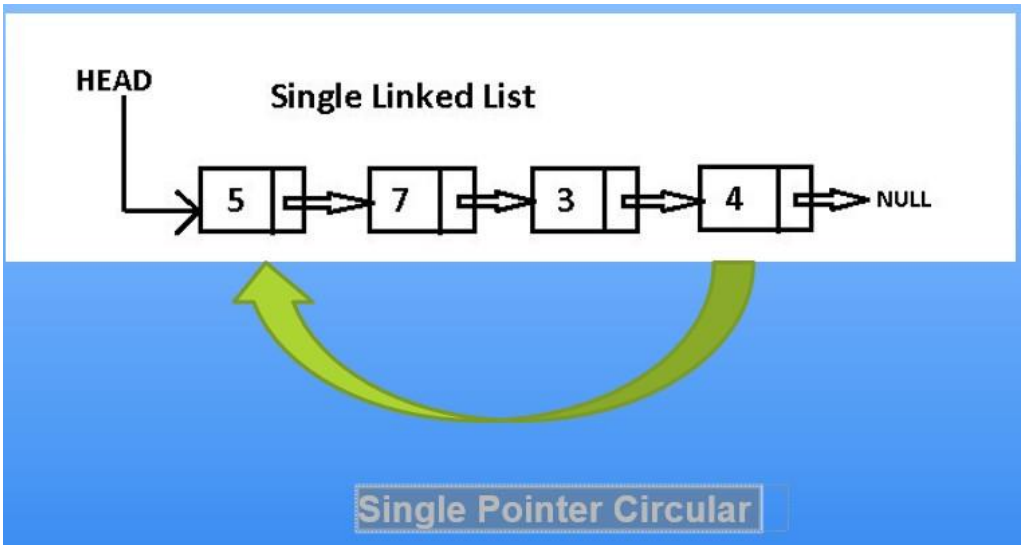
Jenis Linked List

1. Linier
   - Single Pointer
   - Double Pointer
2. Circular
   - Single Pointer
   - Double Ponter

Single Pointer

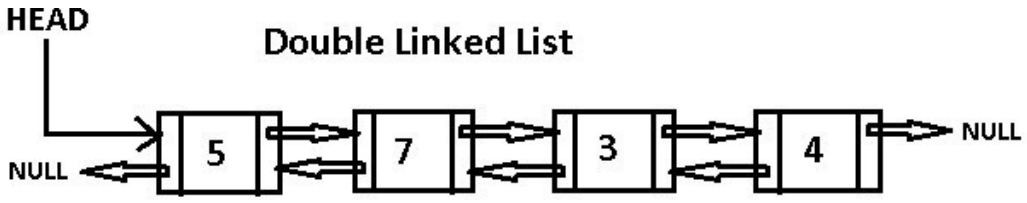
- Sebuah linked list yang hanya menggunakan sebuah variable pointer untuk menghubungkan setiap node (satu arah)
- Karena arahnya hanya satu maka, pada single linked list pencarian data hanya bisa bergerak maju atau mundur, kekanan atau kekiri

  Single Pointer Linear



Double Linked List

- Double Linked List (DLL) berisi pointer tambahan, biasanya disebut pointer previous, bersama dengan pointer next dan data yang ada di single linked list
- Dengan adanya pointer tambahan tersebut disetiap nodenya, maka proses pencarian elemen dapat dilakukan dalam dua arah.



SOURCE CODE SINGLE LINKED LIST

```
class Node:
    def __init__(self, data):
        self.data = data
```

```
        self.next = None
        self.prev = None

class Person:
    def __init__(self, name,age, height):
        self.name = name
        self.age = age
        self.height = height

class DoublyLinkedList:
    def __init__(self):
        self.head = None

    def append(self, data):
        new_node = Node(data)
        if self.head is None:
            self.head = new_node
        else:
            curr_node = self.head
            while curr_node.next is not None:
                curr_node = curr_node.next
            curr_node.next = new_node
            new_node.prev = curr_node

    def prepend(self, data):
        new_node = Node(data)
        if self.head is not None:
            new_node.next = self.head
            self.head.prev = new_node
        self.head = new_node
```

```
    def delete_first(self):
        if self.head is not None:
            self.head = self.head.next
            if self.head is not None:
                self.head.prev = None


    def delete_last(self):
        if self.head is None:
            return
        if self.head.next is None:
            self.head = None
            return
        curr_node = self.head
        while curr_node.next is not None:
            curr_node = curr_node.next
        curr_node.prev.next = None


    def delete(self, data):
        curr_node = self.head
        while curr_node is not None:
            if curr_node.data.name == data.name and curr_node.data.age == data.age and
curr_node.data.height == data.height:
                if curr_node.prev is not None:
                    curr_node.prev.next = curr_node.next
                else:
                    self.head = curr_node.next
                if curr_node.next is not None:
                    curr_node.next.prev = curr_node.prev
                return
            curr_node = curr_node.next
```

```
    def print_list(self):
        curr_node = self.head
        while curr_node is not None:
            print(f"Nama: {curr_node.data.name}, umur: {curr_node.data.age}, tinggi: {curr_node.data.height}")
            curr_node = curr_node.next


# Main program
linked_list = DoublyLinkedList()


while True:
    print("Menu:")
    print("1. Keluar program")
    print("2. Tambah awal list")
    print("3. Tambah akhir list")
    print("4. Hapus awal list")
    print("5. Hapus akhir list")


    choice = int(input("Pilih menu: "))


    if choice == 1:
        break
    elif choice == 2:
        name = input("Masukkan nama: ")
        age = int(input("Masukkan umur: "))
        height = float(input("Masukkan tinggi: "))
        person = Person(name, age, height)
        linked_list.prepend(person)
        print("List setelah ditambahkan elemen di awal:")
        linked_list.print_list()
    elif choice == 3:
```

```
        name = input("Masukkan nama: ")
        age = int(input("Masukkan umur: "))
        height = float(input("Masukkan tinggi: "))
        person = Person(name, age, height)
        linked_list.append(person)
        print("List setelah ditambahkan elemen di akhir:")
        linked_list.print_list()
    elif choice == 4:
        name = input("Masukkan nama: ")
        age = int(input("Masukkan umur: "))
        height = float(input("Masukkan tinggi: "))
        person = Person(name, age, height)
        linked_list.delete_first()
        print("List setelah dihapus elemen di awal:")
        linked_list.print_list()
    elif choice == 5:
        name = input("Masukkan nama: ")
        age = int(input("Masukkan umur: "))
        height = float(input("Masukkan tinggi: "))
        person = Person(name, age, height)
        linked_list.delete_last()
        print("List setelah dihapus elemen di akhir:")
        linked_list.print_list()
    else:
        print("Pilihan tidak valid, silakan coba lagi")
```

SOURCE CODE DOUBLE LINKED LIST

```
/*      ================================================
========== PROGRAM DOUBLE LINKED LIST ==========
===== Sisip Depan, Hapus Tengah, Hapus Belakang =====
================ Tanpa Pointer Bantu ================
================ BY : LAMHOT SITORUS ================
================================================ */



#include<iostream.h> #include<conio.h> #include<stdlib.> #define true 1
#define false 0
typedef struct node *simpul;



struct node
{
char Isi;

simpul kanan; simpul kiri;
};



//======================
//==Prototype Function==
//======================
void Sisip_Depan(simpul &DL, char elemen); void Hapus_Belakang(simpul &DL);
void Hapus_Tengah(simpul &DL, char elemen); void Cetak(simpul DL);

//=================
//==Function Main==
//=================
main()
{
char huruf;
```

```
simpul DL = NULL;   //Pastikan bahwa DL kosong int i;
cout<<"\t\t==OPERASI PADA DOUBLE LINKED LIST==\n\n";
//===============
//==Sisip Depan==
//===============
cout<<"Penyisipan Simpul Di Depan\n\n"; for(i=1; i<=6; i++)
{
cout<<"Masukkan Huruf :"; cin>>huruf; Sisip_Depan(DL,huruf);
}
Cetak(DL);
//=========================
//==Hapus Simpul Belakang==

//=========================
cout<<"\nSetelah Hapus Simpul Belakang \n"; Hapus_Belakang(DL);
Cetak(DL);
cout<<"\nSetelah Hapus Simpul Belakang \n"; Hapus_Belakang(DL);
Cetak(DL);
//=======================
//==Hapus Simpul TENGAH==
//=======================
cout<<"\nMasukkan Huruf Tengah Yang Akan Dihapus      :"; cin>>huruf;
Hapus_Tengah(DL,huruf); Cetak(DL);
cout<<"\nMasukkan Huruf Tengah Yang Akan Dihapus      :"; cin>>huruf;
Hapus_Tengah(DL,huruf); Cetak(DL);
getch();
}



//**********************************
//**FUNCTION SISIP SIMPUL DI DEPAN**
//**********************************
void Sisip_Depan(simpul &DL, char elemen)
{
```

```
simpul baru; //simpul baru = new simpul; maka baris berikut dihapus
baru = (simpul) malloc(sizeof(simpul)); baru->Isi = elemen;
baru->kanan = NULL; baru->kiri = NULL; if(DL == NULL)
DL=baru;

else
{
baru->kanan = DL; DL->kiri = baru; DL=baru;
}
}



//************************************
//**FUNTION MENCETAK ISI LINKED LIST**
//************************************ void Cetak(simpul DL)
{
if(DL==NULL)
cout<<"Linked List Kosong   "<<endl;
else
{
cout<<"Isi Linked List : "; while (DL->kanan != NULL)
{
cout<<DL->Isi<<"<==>";
DL=DL->kanan;
}
cout<<DL->Isi;
}
}

//**********************************
//**FUNCTION HAPUS SIMPUL BELAKANG**
//********************************** void Hapus_Belakang(simpul &DL)
{
simpul Hapus; if(DL==NULL)
```

STRUCT

Structur yaitu pengelompokan dari variable-variabel yang bernaung dalam satu sama lain. Struktur biasanya dipakai untuk mengelompokkan beberapa informasi yang berkaitan dengan sebuah kesatuan, ataubiasanya disebut dengan record.

Dalam bahasa C++, struct adalah tipe data bentukan yang terdiri dari kumpulan tipe data lain.

Struct mirip seperti array, tapi struct bisa menampung lebih dari 1 jenis tipe data.Cara Pendeklarasian :

Deklarasi Struct	**Contoh**

//Deklarasi objek langsungdidalam struct
 	
struct nama_struct {
/*deklarasi variable
 	
*..
*/
} nama_objek;
 	
//deklarasi objek diluarstruct
 	
int main(){
nama_struct nama_objek;
}
	//Deklarasi objek langsungdidalam struct
	
	struct Mahasiswa {char
	nama[30];
	int nim;
	} novan;
	
	//Deklarasi objek diluarstruct
	
	int main(){ Mahasiswa
	novan;
	}