**Bab 7**

**Studi Kasus Percabangan dan**

**Perulangan**

**Pokok Bahasan**

5. Kalkulator sederhana

6. Tumpukan bilangan

**Tujuan**

1. Memahami logika proses kalkulator sederhana

2. Memahami logika proses untuk menampilkan tumpukan bilangan

3. Menjelaskan pemakaian operasi percabangan untuk pembuatan kalkulator sederhana

4. Menjelaskan pemakaian operasi perulangan untuk membuat tumpukan bilangan

**7.1 Contoh Kasus 1: Kalkulator sederhana**

Buatlah programkalkulator sederhana yang dapat melakukan operasi +, -, \* dan /

terhadap 2 bilangan.

***Contoh***

Input: Masukkan bilangan 1: 16

Masukkan bilangan 2: 4

Masukkan operator: /

Output: Hasilnya dari 16 / 4 adalah 4

**7.1.1 Cara Penyelesaian Masalah:**

Program akan menerima input berupa bilangan 1, bilangan 2, dan operator. Kemudian program akan mengecek nilai operator, jika operator yang dimasukkan adalah ***‘+’*** maka operasinya adalah penjumlahan, jika operator yang dimasukkan adalah ***‘-’*** maka operasinya adalah pengurangan, jika operator yang dimasukkan adalah ***‘\*’*** maka operasinya adalah perkalian, dan jika operator yang dimasukkan adalah ***‘/’*** maka operasinya adalah pembagian.

**7.1.2 Input:**

Input yang dibutuhkan untuk program ini adalah:

- Bilangan pertama dalam bentuk angka (integer)

- Bilangan kedua dalam bentuk angka (integer)

**7.1.3 Output:**

Program ini akan menghasilkan output yang berupa:

- Nilai hasil perhitungan yang bertipe floating point>

**7.1.4 Struktur Data yang Dibutuhkan:**

- Variabel bertipe integer untuk menyimpan data bilangan 1 --> variabel ***bil1***

- Variabel bertipe integer untuk menyimpan data bilangan 2 --> variabel ***bil2***

- Variabel bertipe char untuk menyimpan jenis operator --> variabel ***op***

- Variabel bertipe float untuk menyimpan hasil perhitungan --> variabel ***hasil***

**7.1.5 Logika Pemrograman:**

- Operasi percabangan dengan menggunakan **‘IF ... THEN ... ELSE ...’**

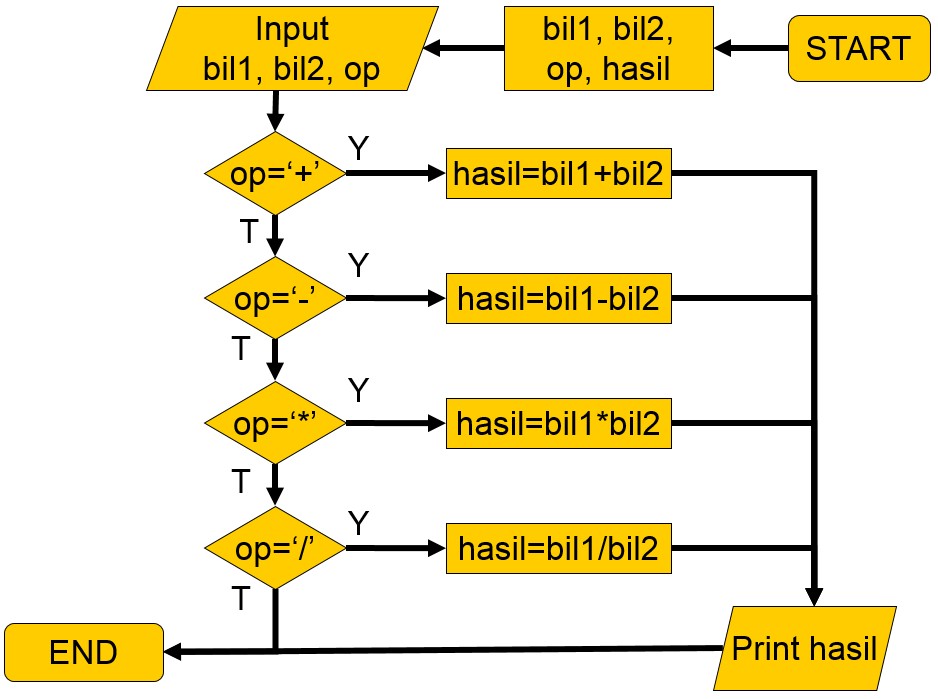
- Operasi penjumlahan menggunakan operator ‘+’

- Operasi pengurangan menggunakan operator ‘-’

- Operasi perkalian menggunakan operator ‘\*’

- Operasi pembagian menggunakan operator ‘***/***’.

**7.1.6 Flowchart:**



**7.2 Contoh Kasus 2: Tumpukan bilangan**

**7.2.1 Permasalahan**

Buatlah program untuk menampilkan tumpukan bilangan.

***Contoh***

|  |  |
| --- | --- |
| Input:  Output: | Masukkan jumlah tumpukan: 55  1 |
|  | 222 |
|  | 33333 |
|  | 4444444 |
|  | 555555555 |

**7.2.2 Cara Penyelesaian Masalah:**

Untuk membuat tumpukan bilangan seperti pada contoh kasus diatas, diperlukan dua level perulangan. Perulangan pertama untuk tingkatan, sedangkan perulangan kedua untuk mencetak angka pada tiap tingkatan.

**7.2.3 Input:**

Input yang dibutuhkan untuk program ini adalah:

- Jumlah tumpukan dalam bentuk angka (integer)

**7.2.4 Output:**

Program ini akan menghasilkan output yang berupa:

- Tumpukan angka dengan jumlah tumpukan sesuai nilai angka yang dimasukkan

**7.2.5 Struktur Data yang Dibutuhkan:**

- Variabel bertipe integer untuk menyimpan data jumlah tumpukan --> variabel ***n***

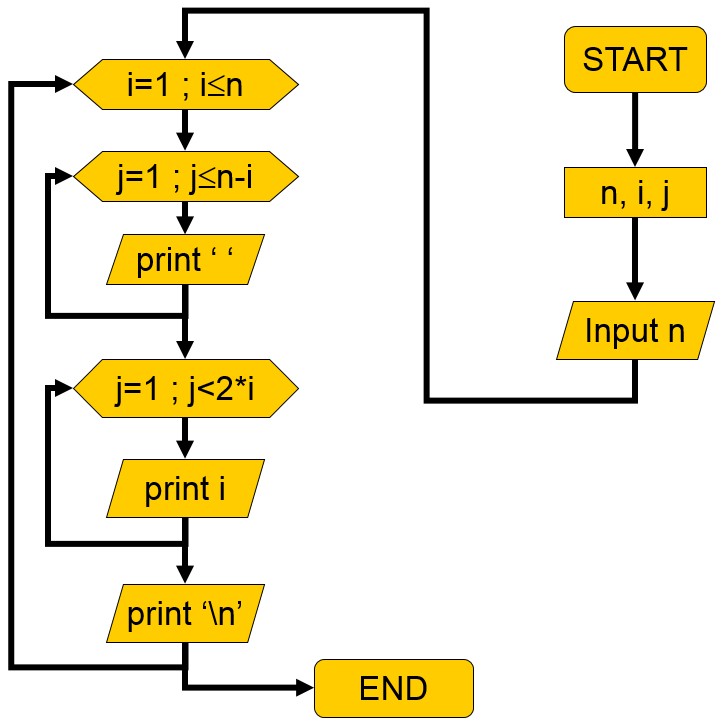
- Variabel bertipe integer untuk digunakan pada perulangan pertama --> variabel ***i***

- Variabel bertipe integer untuk digunakan pada perulangan kedua --> variabel ***j***

**7.2.6 Logika Pemrograman:**

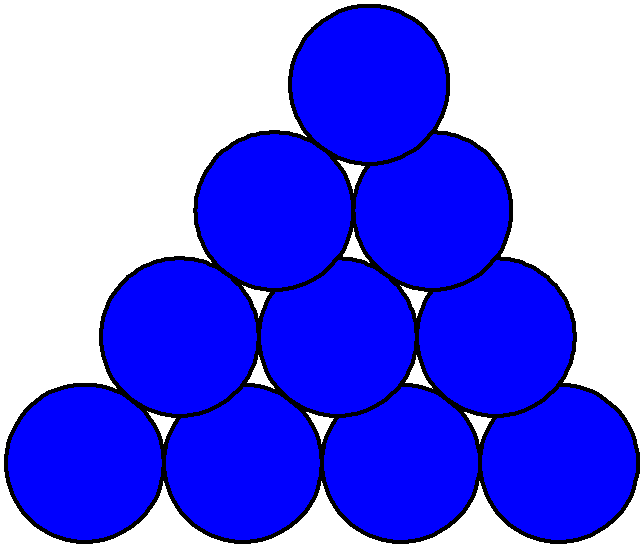
- Operasi perulangan menggunakan statement **For**

**7.2.7 Flowchart:**



**7.3 Soal Latihan**

Buatlah program untuk menghitung jumlah bola dalam suatu triangular. Triangular adalah suatu susunan benda (contoh: bola) yang disusun sedemikian rupa sehingga menyerupai segitiga. Dengan mengetahui jumlah bola yang paling bawah, maka dapat dihitung jumlah seluruh bola yang menyusun triangular tersebut.



**Contoh:**

Input: Masukkan jumlah triangular: 4

Output: Jumlah bola: 10

*Halaman ini sengaja dikosongkan*

65

**Bab 8**

**Studi Kasus Tumpukan (Stack)**

**Pokok Bahasan**

7. Membalik kalimat

8. Membalik bilangan

**Tujuan**

5. Memahami logika proses pembalikan kalimat

6. Memahami logika proses pembalikan bilangan

7. Menjelaskan pemakaian stack untuk pembalikan kalimat

8. Menjelaskan pemakaian stack untuk pembalikan bilangan

**8.1 Contoh Kasus 1: Membalik kalimat**

**8.1.1 Permasalahan**

Buatlah flowchart untuk membalik kalimat.

***Contoh***

Input: Masukkan kalimat = “APA KABAR” Output: “RABAK APA”

**8.1.2 Cara Penyelesaian Masalah:**

Untuk membalik kalimat dapat diselesaikan menggunakan dua cara. Pertama dengan menggunakan array. Sedangkan yang kedua dengan memanfaatkan stack. Stack (tumpukan) merupakan struktur data yang dalam pengelolaan datanya bersifat Last In First Out (LIFO), yaitu data yang terakhir dimasukkan akan dikeluarkan pertama kali. Data yang dimasukkan kedua dari terakhir akan dikeluarkan yang kedua. Demikian seterusnya, sehingga data yang pertama kali dimasukkan akan dikeluarkan terakhir.

**8.1.3 Input:**

Input yang dibutuhkan untuk program ini adalah:

- Kalimat yang akan dibalik dalam bentuk array of character (string)

**8.1.4 Output:**

Program ini akan menghasilkan output yang berupa:

- Kalimat yang sudah dibalik dalam bentuk string

**8.1.5 Struktur Data yang Dibutuhkan:**

- Variabel bertipe string (array of character) untuk menyimpan kalimat awal -->

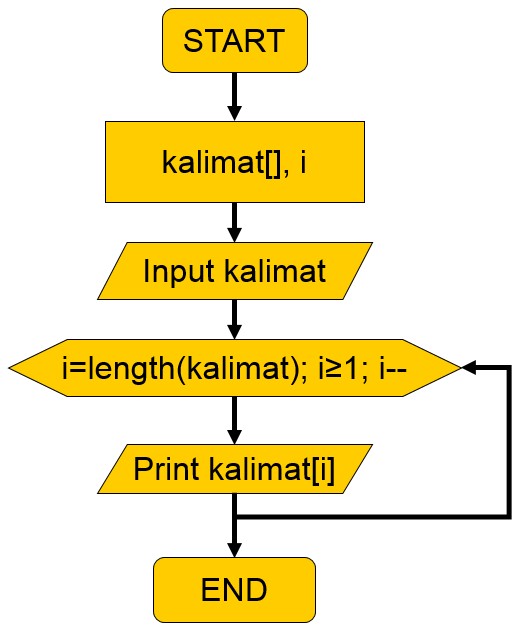
variabel ***kalimat***

**8.1.6 Logika Pemrograman:**

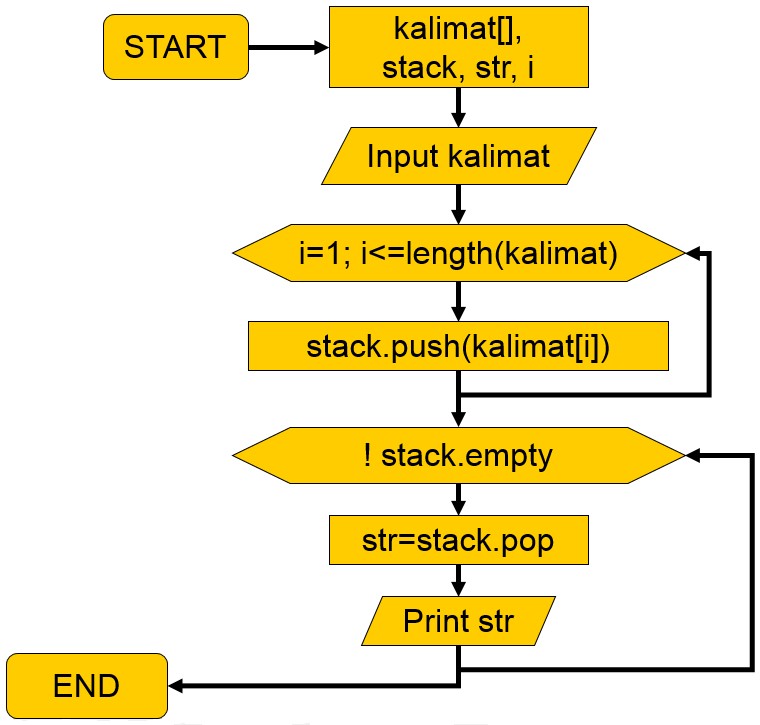
- Operasi perulangan dengan menggunakan **‘DO ... WHILE’** atau **‘WHILE ... DO’**

**8.1.7 Flowchart**

Flowchart untuk penyelesaian menggunakan array of character



Flowchart untuk penyelesaian menggunakan stack



**8.2 Contoh Kasus 2: Membalik bilangan**

**8.2.1 Permasalahan**

Buatlah flowchart untuk menampilkan bilangan dalam urutan terbalik.

***Contoh***

Input: Masukkan kalimat : 1234

Output: Hasil : 4321

**8.2.2 Cara Penyelesaian Masalah:**

Proses untuk membalik urutan bilangan berbeda dengan membalik kalimat. Suatu kalimat bisa dianggap sebagai array dari karakter, sedangkan bilangan tidak bisa.

Cara yang bisa dilakukan adalah dengan menggunakan operator modulus untuk mengambil angka terakhir dari bilangan. Kemudian dengan membagi bilangan tersebut dengan 10 menggunakan operator pembagian ‘/’ kita akan mendapatkan sisa bilangan setelah bilangan yang terakhir diambil.

Contoh, misalkan kita mempunyai bilangan 1234. Dengan melakukan operasi 1234 % 10, kita akan mendapatkan angka terakhir yaitu 4. Kemudian dengan operasi 1234 / 10 kita akan mendapatkan bilangan 123. Berikutnya kita dapat mengulangi lagi mulai langkah yang pertama (menggunakan operator modulus).

**8.2.3 Input:**

Input yang dibutuhkan untuk program ini adalah:

- Bilangan yang akan dibalik dalam bentuk integer

**8.2.4 Output:**

Program ini akan menghasilkan output yang berupa:

- Bilangan dengan urutan angka yang berupa kebalikan dari urutan angka pada bilangan hasil input.

**8.2.5 Struktur Data yang Dibutuhkan:**

- Variabel bertipe integer untuk menyimpan bilangan awal --> variabel ***bil***

- Variabel bertipe integer untuk menyimpan bilangan hasil --> variabel ***hasil***

- Variabel untuk menyimpan stack --> variabel ***stack***

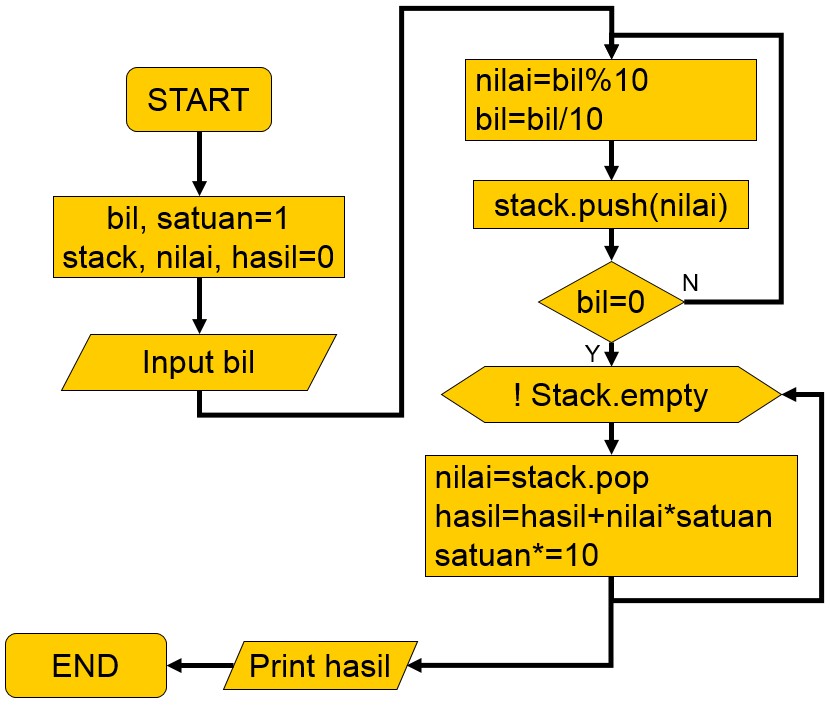
- Variabel bertipe integer untuk menyimpan nilai sementara --> variabel ***nilai***

- Variabel bertipe integer untuk menyimpan angka satuan --> variabel ***satuan***

**8.2.6 Logika Pemrograman:**

- Operasi perulangan dengan menggunakan **‘DO ... WHILE’** atau **‘WHILE ... DO’**

**8.2.7 Flowchart**



**8.3 Soal Latihan**

Buatlah flowchart untuk mengecek suatu kalimat termasuk palindrom apa bukan. Kalimat palindrom adalah kalimat yang susunannya sama dengan keadaan terbaliknya.

**Contoh:**

Input: Masukkan kalimat = “KASUR RUSAK” Output: Kalimat termasuk palindrom

Input: Masukkan kalimat = “MAKAN MALAM” Output: Kalimat tidak termasuk palindrom

*Halaman ini sengaja dikosongkan*

71

**Bab 9**

**Studi Kasus Konversi Bilangan**

**Pokok Bahasan**

1. Konversi bilangan biner ke desimal

2. Konversi bilangan desimal ke biner

**Tujuan**

1. Memahami logika melakukan konversi bilangan

2. Memahami algoritma konversi bilangan

3. Menjelaskan pemakaian looping untuk konversi bilangan.

4. Menjelaskan pemakaian operasi stack untuk konversi bilangan.

**9.1 . Studi Kasus 1: Konversi bilangan biner ke desimal**

**9.1.1 Permasalahan**

Buatlah flowchart untuk konversi bilangan biner ke desimal (maksimum bilangan=11111111).

Contoh:

Masukkan bilangan biner : 00110101

Bilangan desimal : 53

**9.1.2 Cara Penyelesaian Masalah**

Konversi bilangan biner ke desimal dilakukan dengan cara mengalikan setiap elemen bilangan biner (bi) dengan deretan bilangan 2 yang berpangkat sebagai berikut:

b1 \* 2¬n-1 + b2 \* 2¬n-2 + ... + bn-2 \* 22 + bn-1 \* 21 + bn \* 20

atau : b1 \* 2¬n-1 + b2 \* 2¬n-2 + ... + bn-2 \* 4 + bn-1 \* 2 + bn \* 1

Dalam kasus permasalahan diatas, input bilangan biner yang dimasukkan sepanjang 8 elemen bilangan dapat diselesaikan sebagai berikut:

00110101

0\*2*7* + 0\*2*6* + 1\*2*5* + 1\*2*4* + 0\*2*3* + 1\*2*2* + 0\*2*1* + 1\*2*0*

= 0\*128 + 0\*64 + 1\*32 + 1\*16 + 0\*8 + 1\*4 + 0\*2 + 1\*1

= 0 + 0 + 32 + 16 + 0 + 4 + 0 + 1

= 53

**9.1.3 Struktur Data Yang Dibutuhkan**

• n sebagai variabel banyaknya elemen bilangan (dalam hal ini n=8)

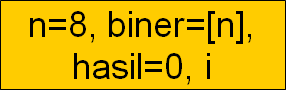
• biner sebagai array bilangan biner dengan panjang n elemen

• hasil sebagai variabel penampung hasil konversi biner ke desimal

• i sebagai variabel penunjuk elemen array

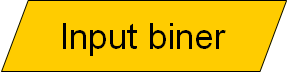
**9.1.4 Deklarasi dan Inisialisasi**

Variabel terdiri dari n, biner, hasil dan i. Untuk n, pada permasalahan n=8. Sedangkan hasil, karena dipakai sebagai penampung hasil konversi, saat awal hasil harus diset menjadi 0. Dari sini, deklarasi dan inisialisasi variabel pada flowchart dapat dibuat sebagai berikut:



**9.1.5 Input**

Bilangan biner yang terdiri dari 8 elemen, sehingga input pada flowchart dapat dibuat sebagai berikut:



**9.1.6 Output**

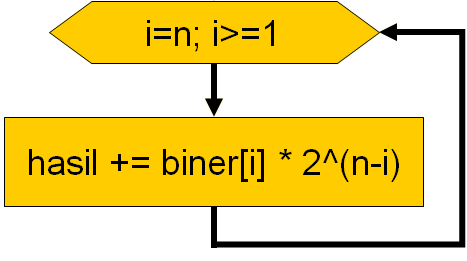
Bilangan desimal hasil konversi dari bilangan biner yang disimpan pada variabel

*hasil*, sehingga output pada flowchart dapat dibuat sebagai berikut:

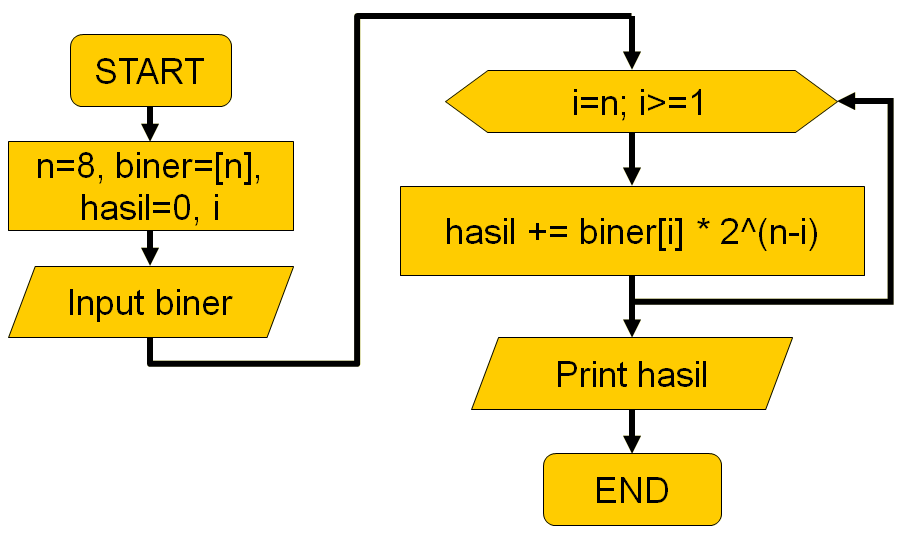


**9.1.7 Proses Penyelesaian**

Karena proses perhitungan konversi dilakukan dari kanan ke kiri, maka looping juga dilakukan menurun dari *i*=*n* sampai *i*=1. Saat awal *i*=*n*, dilakukan perhitungan pada *bn* \* 2*0* dan hasilnya diakumulasikan pada *hasil*. Pada iterasi berikut nilai *i* menurun menjadi *i*=*n*-1, dilakukan perhitungan pada *bn-1* \* 2*1* dan hasilnya diakumulasikan pada *hasil*. Dari sini, dapat diambil kesimpulan bahwa pada setiap iterasi *i*, *hasil* dapat dihitung dengan rumusan *hasil* = *hasil* + *bi* \* 2*n-i* . Dengan demikian, untuk proses dapat direpresentasikan dengan flowchart sebagai berikut:



**9.1.8 Flowchart Keseluruhan**



**9.2 Studi Kasus 2: Konversi bilangan desimal ke biner**

**9.2.1 Permasalahan**

Buatlah flowchart untuk konversi bilangan desimal ke biner (maksimum bilangan=255)

Contoh:

Masukkan bilangan desimal : 53

Bilangan biner : 110101

**9.2.2 Cara Penyelesaian Masalah**

Konversi bilangan desimal ke biner dilakukan dengan cara melakukan pembagian dengan nilai 2 dan mengambil sisa baginya. Sisa bagi tersebut kemudian disimpan pada suatu tempat kumpulan sisa bagi. Bilangan hasil bagi tersebut dibagi lagi dan diambil sisa baginya untuk kemudian disimpan pada tempat kumpulan sisa bagi. Demikian seterusnya sampai didapatkan hasil baginya sama dengan nol. Setelah itu, baru diambil sisa bagi dalam keadaan terbalik (yang masuk terakhir, diambil pertama kali) pada kumpulan sisa bagi. Proses pembacaan terbalik ini membutuhkan operasi stack.

Dalam kasus permasalahan diatas, dapat diselesaikan sebagai berikut:

|  |  |  |  |
| --- | --- | --- | --- |
| **Bilangan** | **Hasil bagi dengan 2** | **Sisa bagi dengan 2** | **Kumpulan sisa bagi** |
| 53 | 26 | 1 | 1 |
| 26 | 13 | 0 | 10 |
| 13 | 6 | 1 | 101 |
| 6 | 3 | 0 | 1010 |
| 3 | 1 | 1 | 10101 |
| 1 | 0 | 1 | 101011 |

Bilangan biner yang didapatkan adalah dari pembacaan kumpulan sisa bagi yang dibaca terbalik, sehingga hasilnya adalah 110101.

**9.2.3 Struktur Data Yang Dibutuhkan**

• bil sebagai variabel nilai bilangan

• sisa sebagai variabel nilai sisa bagi dengan 2

• stack sebagai penampung sisa bagi sehingga dapat dibaca terbalik

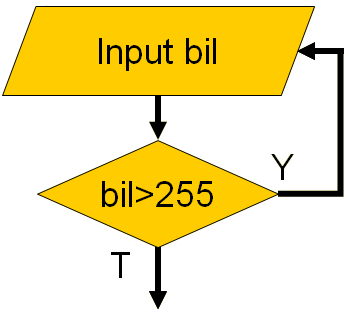
**9.2.4 Deklarasi dan Inisialisasi**

Variabel terdiri dari bil, sisa, dan stack. Dari sini, deklarasi dan inisialisasi variabel pada flowchart dapat dibuat sebagai berikut:



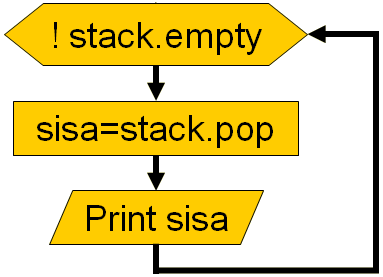
**9.2.5 Input**

Saat pertama kali, user diminta untuk memasukkan nilai bilangan bil yang akan dikonversi ke biner. Karena pada permasalahan terdapat syarat bahwa nilai bilangan tidak boleh lebih dari 255, maka perlu diberikan pengecekan apakah bil sesuai dengan syarat. Dari sini, flowchart Input dapat dibuat sebagai berikut:



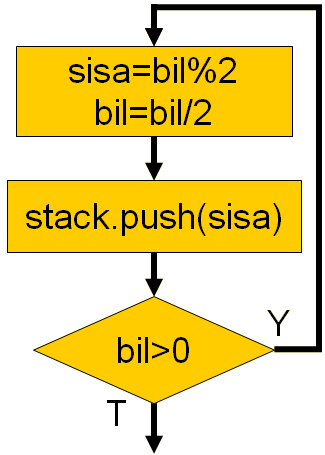
**9.2.6 Output**

Bilangan desimal hasil konversi dari bilangan desimal yang dikeluarkan dari *stack* dengan operasi pop sehingga dapat dibaca dalam keadaan terbalik. Setiap hasil pembacaan kemudian disimpan pada variabel *sisa*, sehingga output pada flowchart dapat dibuat sebagai berikut:

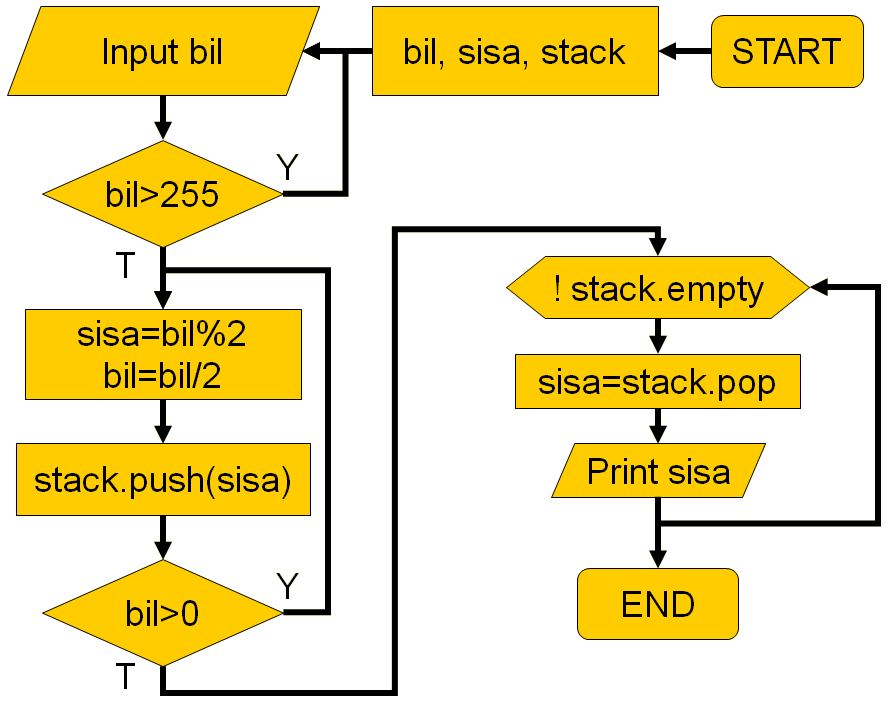


**9.2.7 ProsesPenyelesaian**

Sebagai dijelaskan pada sebelumnya, konversi bilangan desimal ke biner dilakukan dengan cara melakukan pembagian dengan nilai 2 dan mengambil sisa baginya, dimana proses ini dilakukan secara iteratif dan berhenti saat bilangan hasil bagi sama dengan 0. Setiap kali didapatkan sisa bagi, nilai tersebut dimasukkan ke dalam stack melalui operasi push. Adapun tujuan dipergunakannya stack adalah agar data yang masuk ke dalam stack, dapat dibaca dalam keadaan terbalik, sehingga ini berguna untuk membaca terbalik nilai sisa bagi yang dimasukkan ke dalam stack. Dari sini, flowchart Proses dapat direpresentasikan sebagai berikut:



**9.2.8 Flowchart Keseluruhan**



**9.3 Latihan**

**9.3.1 Permasalahan**

Buatlah flowchart untuk konversi bilangan desimal negatif ke biner (maksimum bilangan=-127).

Contoh:

Masukkan bilangan desimal : -1

Bilangan biner : 11111111

Masukkan bilangan desimal : -13

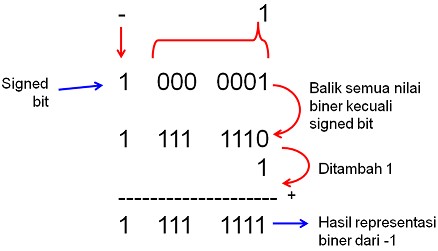
Bilangan biner : 11110011

**9.3.2 Cara Penyelesaian Masalah**

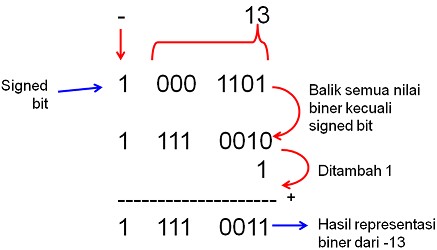
Karena batasan permasalahan bahwa maksimum bilangan adalah -127, maka total bit yang dipakai adalah sebanyak 8 bit. Pertama kali, signed bit (bit pada depan) diset dengan nilai 1 sebagai tanda negatif. Sisa 7 bit yang lainnya adalah nilai bit hasil konversi dari input bilangan desimal yang positif. Selanjutnya, 7 bit sisa tersebut (tidak termasuk signed bit) dibalik nilainya (0 menjadi 1, 1 menjadi 0). Kemudian 7 bit sisa ditambahkan dengan nilai

bit1. Susunan bit keseluruhannya adalah hasil dari konversi desimal negatif. Untuk lebih mudahnya, perhatikan ilustrasi berikut:

Masukkan bilangan desimal : -1



Masukkan bilangan desimal : -13



**Bab 10**

**Studi Kasus Operasi Matriks**

**Pokok Bahasan**

1. Operasi Matriks

**Tujuan**

1. Memahami logika operasi matriks

2. Memahami penggunaan looping bertingkat untuk menyelesaikan operasi matriks

**10.1 Studi Kasus: Operasi penambahan matriks**

**10.1.1 Permasalahan**

Buatlah suatu flowchart untuk meyelesaikan operasi penambahan matriks

Contoh:

Masukkan Matriks 1 dan Matriks 2 :

Matriks 1 : 5 2 Matriks 2 : 1 0



1 7 1 1



Hasil : 6 2



2 8

**10.1.2 Cara Penyelesaian Masalah**

Operasi penambahan matriks dilakukan dengan cara menambahkan setiap elemen Matriks 1 dengan elemen Matriks 2 pada posisi baris dan kolom yang sama untuk kedua matriks.

Pada permasalahan diatas, operasi penambahan matriks dilakukan sebagai berikut:

5 2 + 1 0



1 7 1 1



= 5+1 2+0



1+1 7+1



= 6 2



2 8

**10.1.3 Struktur Data Yang Dibutuhkan**

***brs*** sebagai variabel panjang baris untuk matriks

***klm*** sebagai variabel panjang kolom untuk matriks

***M1*** sebagai input matriks pertama bertipe array 2 dimensi

***M2*** sebagai input matriks kedua bertipe array 2 dimensi

***M3*** sebagai matriks dimensi

penampung hasil operasi penambahan matriks bertipe array 2

***i*** sebagai variabel penunjuk baris saat proses penambahan matriks

***j*** sebagai variabel penunjuk kolom saat proses penambahan matriks

**10.1.4 Deklarasi dan Inisialisasi**

Variabel terdiri dari *brs*, *klm*, *M1*, *M2*, *M3*, *i* dan *j*. Dari sini, deklarasi dan inisialisasi variabel pada flowchart dapat dibuat sebagai berikut:

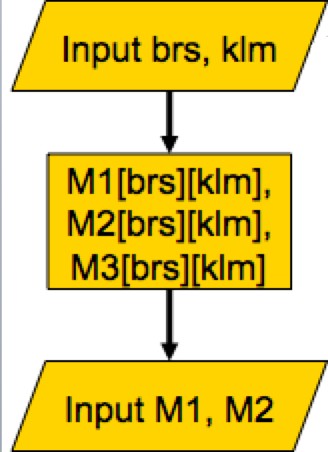


**10.1.5 Input**

Pertama kali, user harus memasukkan panjang baris *brs* dan kolom *klm* untuk matriks. Dari panjang baris dan kolom tersebut, kemudian dibuat suatu 2 buah matriks input *M1* dan

*M2* dengan panjang baris dan kolom yang diinginkan. Kemudian, user memasukkan nilai dari

2 matriks input. Flowchart untuk input ini dapat dilihat sebagai berikut:



**10.1.6 Output**

Matriks *M3* sebagai hasil operasi penambahan matriks *M1* dan *M2*, sehingga output pada flowchart dapat dibuat sebagai berikut:



**10.1.7 Proses Penyelesaian**

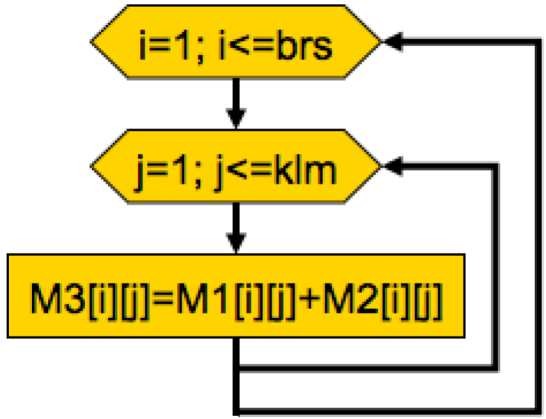
Proses operasi penambahan matriks dilakukan dengan menambahkan setiap elemen kedua matriks input pada posisi yang sama. Jika proses penambahan dimulai dari baris, untuk

baris pertama, dilakukan penambahan nilai pada kolom pertama untuk kedua matriks.

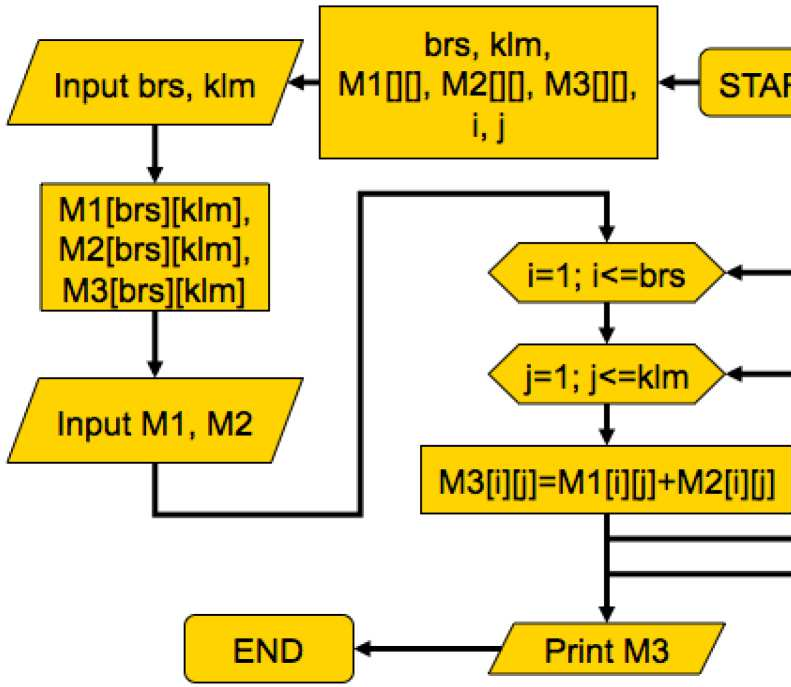
Selanjutnya berganti pada kolom kedua. Setelah baris pertama selesai,

kemudian berganti

pada baris kedua, dilakukan penambahan nilai pada kolom pertama untuk kedua matriks, dan selanjutnya dilakukan hal serupa pada kolom kedua. Pada setiap kali operasi penambahan, hasil penambahan ditampung pada matriks M3. Dengan demikian, untuk proses dapat direpresentasikan dengan flowchart sebagai berikut:



**10.1.8 Flowchart Keseluruhan**



**10.2 10.2. Latihan**

**10.2.1 Permasalahan**

Buatlah suatu flowchart untuk meyelesaikan operasi penambahan ma

Contoh:



Masukkan Matriks 1 dan Matriks 2 :

1



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Matriks 1 : | 5 2 3 | Matriks 2 : | 1 | 0 |
|  | 1 7 1 |  | 1 | 1 |
|  |  |  | 2 | 4 |

|  |  |  |
| --- | --- | --- |
| Hasil : | 3 | 14 |
|  | 10 | 11 |

**10.2.2 Cara Penyelesaian Masalah**



Operasi penambahan matriks dilakukan dengan cara mengalikan setiap elemen baris pada Matriks 1 dengan elemen kolom pada Matriks 2 secara berturut-turut, dan kemudian masing-masing dari hasil perkalian tersebut diakumulasikan.

Pada permasalahan diatas, operasi perkalian matriks dilakukan sebagai berikut:

5



|  |  |  |  |
| --- | --- | --- | --- |
| 5 2 3 | x | 1 | 0 |
| 1 7 1 |  | 1 | 1 |
|  |  | 2 | 4 |

|  |  |  |
| --- | --- | --- |
| = | x1 + 2x1 + 3x2 | 5x0 + 2x1 + 3x4 |
|  | 1x1 + 7x1 + 1x2 | 1x0 + 7x1 + 1x4 |

= 3 1

1

4



10 11

*Halaman ini sengaja dikosongkan*

85

**Bab 11**

**Studi Kasus Shortest Path Problem**

**Pokok Bahasan**

1. Shortest Path Problem (Permasalahan Jalur Tercepat)

**Tujuan**

1. Memahami logika shortest path problem

2. Memahami algoritma Dijkstra untuk menyelesaikan shortest path problem

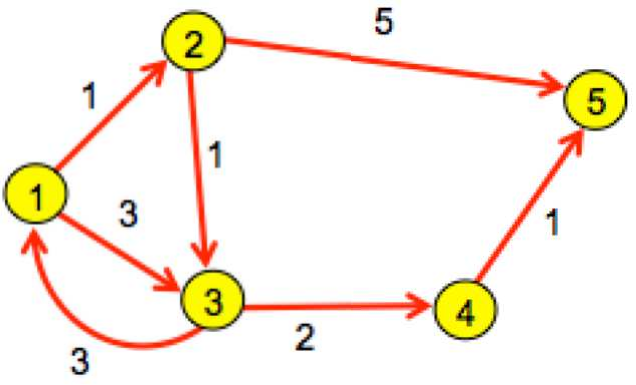
3. Memahami penggunaan matrik untuk manipulasi variabel pada shortest path problem

4. Memahami penggunaan queue dan stack untuk menyelesaikan shortest path problem

**11.1 Studi Kasus: Shortest Path Problem dengan Algoritma Dijkstra**

**11.1.1 Permasalahan**

Buatlah flowchart untuk menghitung jarak minimal dan rutenya dari titik 1 ke titik 5 dari jalur berikut ini:



**11.1.2 Cara Penyelesaian Masalah**

Untuk menyelesaikan

shortest path problem satu arah seperti yang ada pada

permasalahan, dapat digunakan algoritma Dijkstra. Pertama kali tentukan titik asal dan titik tujuan pada permasalahan. Pada permasalahan, titik asal adalah titik 1 dan titik tujuan adalah

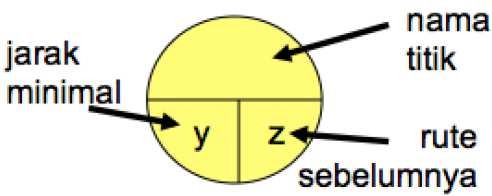
titik 5. Karena yang dicari adalah jarak terpendek dan rute pada jarak

terpendek tersebut,

maka setiap titik perlu memuat jarak terpendek dan rutenya terhadap

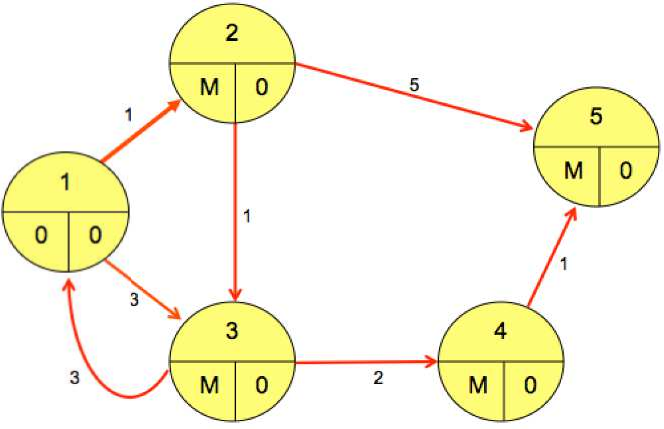
sehingga setiap titik terdiri dari informasi berikut:

titik sebelumnya,



Saat awal, pada setiap titik selain titik asal, nilai y diinisialisasi dengan nilai yang besar (diasumsikan sebagai M) dan nilai z diinisialisasi dengan titik 0, seperti yang tampak

pada gambar berikut:



Setelah itu, proses dimulai dari titik asal yaitu titik 1 sebagai

titik aktif. Titik 1

mempunyai jalur ke 2 titik, yaitu titik 2 dan titik 3. Saat mendapatkan ada jalur dari titik 1 ke titik 2, maka nilai y pada titik 1 ditambahkan dengan jarak dari titik 1 ke titik 2. Jika nilai

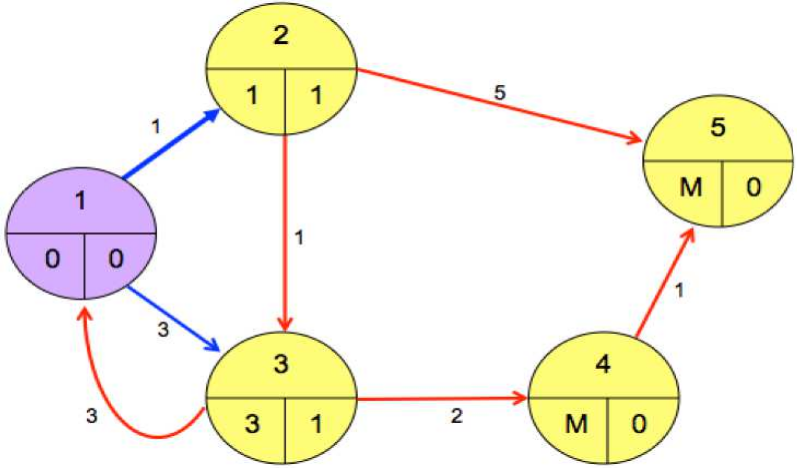
penambahan ini lebih kecil dari nilai y pada titik 2, maka akan menyebabkan 3 hal:

1. nilai y pada titik 2 digantikan dengan nilai penambahan tersebut

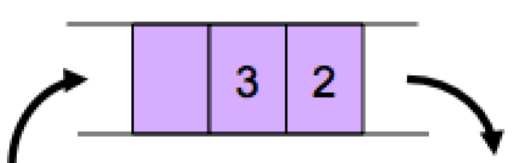
2. nilai z pada titik 2 digantikan dengan titik 1.

3. Informasi titik 2 dimasukkan ke dalam queue dengan proses enqueue

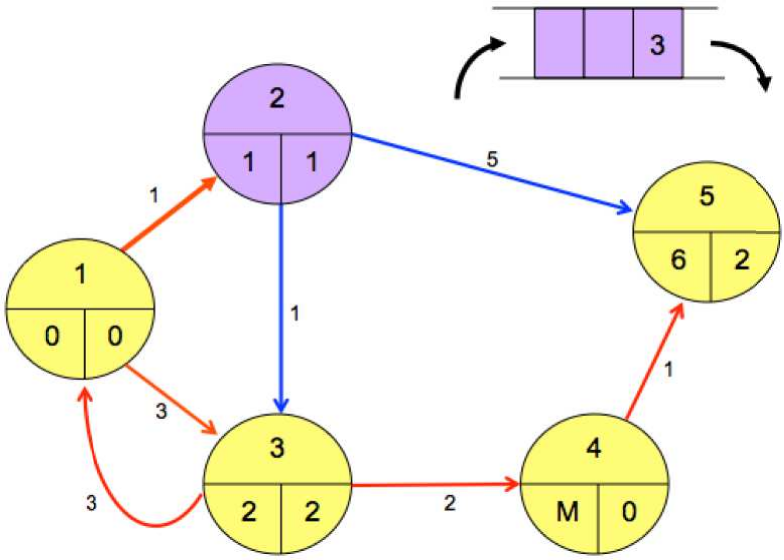
Hal yang sama dilakukan terhadap titik 1 ke titik 3. Ini membuat adanya perubahan nilai pada titik 2 dan titik 3, seperti yang terlihat sebagai berikut.



Adapun kondisi queue, karena titik 2 dan titik 3 dimasukkan ke dalam queue, maka queue terlihat dibawah ini.

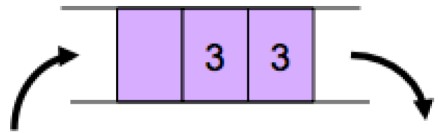


Selanjutnya, isi queue dikeluarkan satu nilai (proses dequeue) sebagai titik aktif. Hasil dari dequeue adalah nilai 2, sehingga sekarang titik aktif adalah titik 2. Titik 2 mempunyai jalur ke titik 3 dan titik 5. Hal yang dilakukan seperti yang sebelumnya, sehingga kondisinya sekarang menjadi seperti berikut:

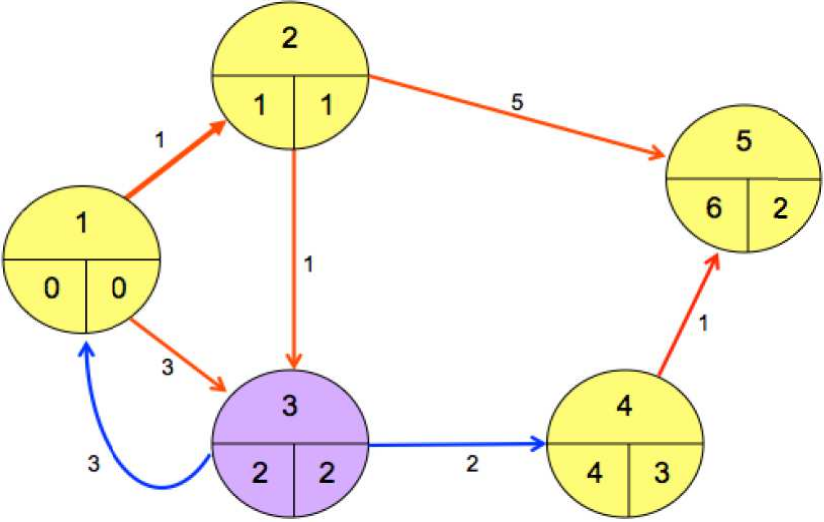


Informasi jalur dari titik 2 ke titik 3 dan titik 5 ini kemudian akan dimasukkan ke dalam queue. Informasi titik 3 kemudian dimasukkan lagi ke dalam queue. Untuk titik 5,

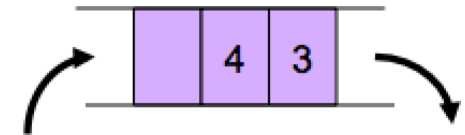
karena titik 5 adalah tujuan, maka tidak perlu dimasukkan ke dalam queue. Sekarang kondisi queue terlihat seperti dibawah ini:



Selanjutnya, isi queue dikeluarkan satu nilai (proses dequeue) sebagai titik aktif. Hasil dari dequeue adalah nilai 3, sehingga sekarang titik aktif adalah titik 3. Titik 3 mempunyai jalur ke titik 1 dan titik 4. Hal yang dilakukan seperti yang sebelumnya, sehingga kondisinya sekarang menjadi seperti berikut:



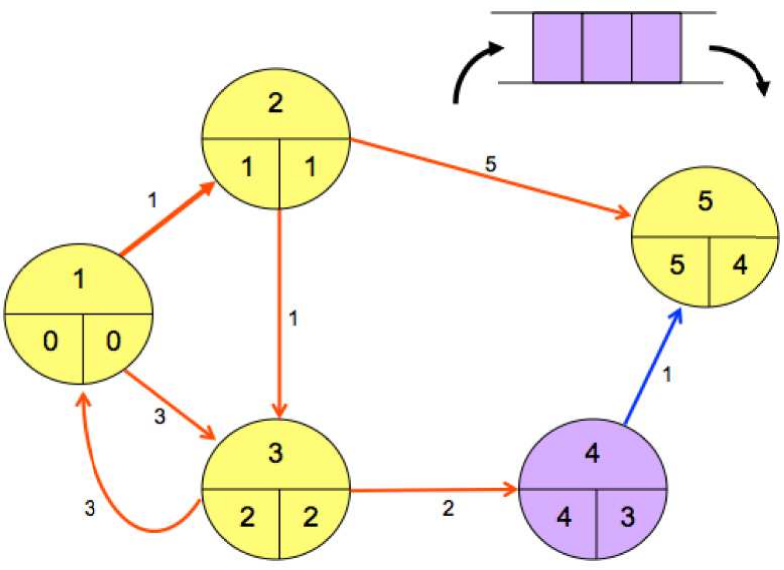
Proses berikutnya adalah enqueue untuk memasukkan titik 1 dan titik 4. Karena titik 1 adalah titik asal, maka tidak dimasukkan ke queue. Kemudian titik 4 dimasukkan ke dalam queue, sehingga kondisi queue sekarang tampak seperti dibawah ini:



Selanjutnya, isi queue dikeluarkan satu nilai (proses dequeue) sebagai titik aktif. Hasil dari dequeue adalah nilai 3, sehingga sekarang titik aktif adalah titik 3, sama seperti keadaan sebelumnya. Proses dilakukan sama, akan tetapi karena nilai penambahan nilai y dan nilai z pada titik 3 ke titik-titik jalurnya tidak ada yang lebih kecil dari nilai y dan nilai z pada titik- titik jalurnya, maka tidak ada perubahan nilai pada nilai y dan nilai z pada titik jalurnya dan

tidak ada proses enqueue titik-titik jalur dari titik 3.

Selanjutnya, isi queue dikeluarkan satu nilai (proses dequeue) sebagai titik aktif. Hasil dari dequeue adalah nilai 4, sehingga sekarang titik aktif adalah titik 4. Titik 4 mempunyai jalur ke titik 5. Hal yang dilakukan seperti yang sebelumnya, sehingga kondisinya sekarang menjadi seperti berikut:



Informasi jalur dari titik 4 ke titik 5 ini kemudian akan dimasukkan ke dalam queue.

Karena titik 5 adalah tujuan, maka tidak perlu dimasukkan ke dalam queue.

Untuk selanjutnya, kembali pada proses enqueue pada queue.

kosong, maka perubahan nilai y dan nilai y pada setiap titik sudah selesai.

Karena isi queue

Proses selanjutnya adalah pembacaan nilai jarak terdekat dan jalur terdekat sebagai solusi. Informasi jarak terdekat dapat diambil dari nilai y pada titik tujuan. Berarti, jarak terdekat pada permasalan tersebut adalah 5 sebagaimana terlihat pada nilai y di titik 5. Untuk mendapat informasi jalur terdekat, diperlukan operasi stack. Dimulai dari titik tujuan, titik 5 dimasukkan ke dalam stack. Kemudian, pada titik 5, nilai z=4, sehingga bergerak ke titik 4.

Titik 4 dimasukkan ke dalam stack. Nilai z pada titik 4 adalah 3, sehingga kemudian bergerak

ke titik 3. Titik 3 dimasukkan ke dalam stack. Nilai z pada titik 3

adalah 2, sehingga

kemudian bergerak ke titik 2. Titik 2 dimasukkan ke dalam stack. Kemudian, nilai z pada titik

2 adalah 1, sama dengan titik asal. Titik asal dimasukkan ke dalam

stack. Selanjutnya,

informasi jalur tercepat dapat dibaca dari hasil operasi pop terhadap stack, sehingga didapatkan jalur tercepatnya adalah 1-2-3-4-5.

**11.1.3 Struktur Data Yang Dibutuhkan**

***n*** sebagai variabel banyaknya titik ***asal*** sebagai variabel titik asal ***tujuan*** sebagai variabel titik tujuan

***relasi*** sebagai matriks jalur dari satu titik ke titik lain

***jarak*** sebagai matriks jarak dari satu titik ke titik lain

***y*** sebagai array untuk menyimpan nilai y pada setiap titik

***z*** sebagai array untuk menyimpan nilai z pada setiap titik

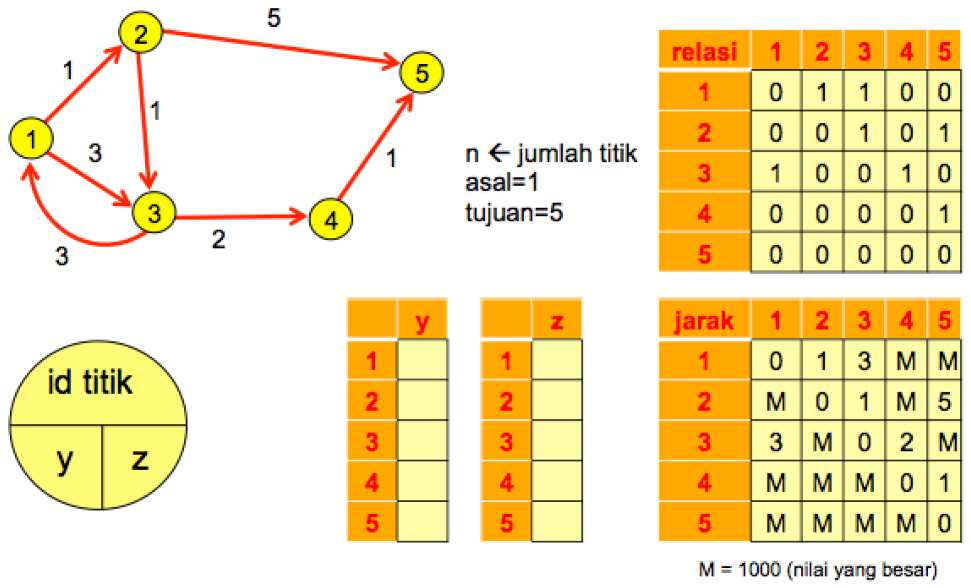
***Q*** sebagai queue penampung titik akan akan diproses sebagai titik aktif

***S*** sebagai stack penampung titik jalur terpendek

***M*** sebagai variabel nilai yang sangat besar

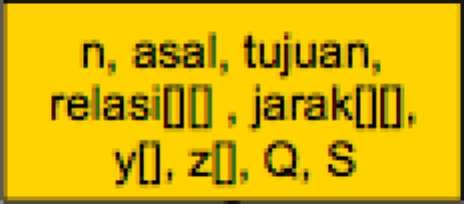
***i*** dan ***j*** sebagai variabel yang dipakai untuk proses looping

Ilustrasi pemakaian struktur data diatas dapat dilihat pada gambar dibawah ini:



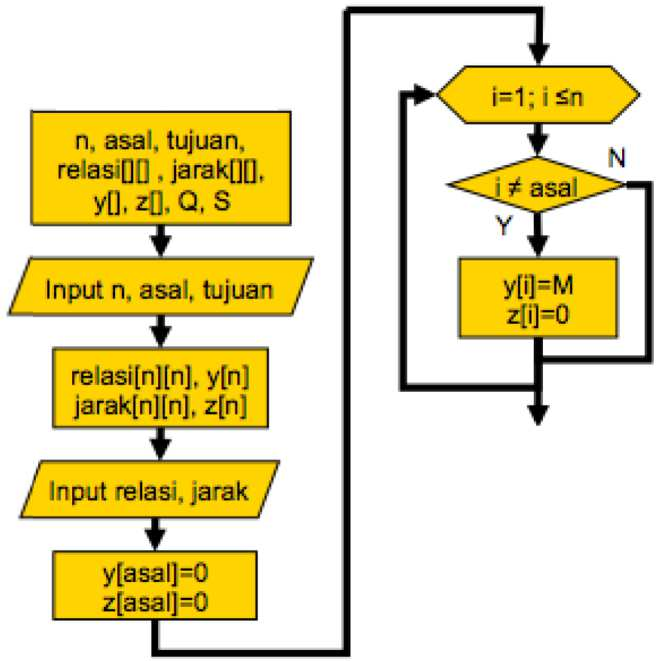
**11.1.4 Deklarasi dan Inisialisasi**

Variabel terdiri dari *n*, *asal*, *tujuan*, *relasi*, *jarak*, *y*, *z*, *Q*, *S*, *i* dan *j*. Dari sini, deklarasi dan inisialisasi variabel pada flowchart dapat dibuat sebagai berikut:



**11.1.5 Input**

Pertama kali, user harus memasukkan banyaknya titik *n*, titik asal dan titik tujuan. Dari banyaknya titik tersebut, dibuatlah matriks input *relasi* dan *jarak*, array *y* dan *z*. Kemudian, user memasukkan nilai dari 2 matriks input. Selanjutnya dilakukan prose inisilisasi nilai y dan z. Flowchart untuk input ini dapat dilihat sebagai berikut:



**11.1.6 Output**

Solusi yang ingin didapatkan pada permasalahan shortest path

problem tersebut

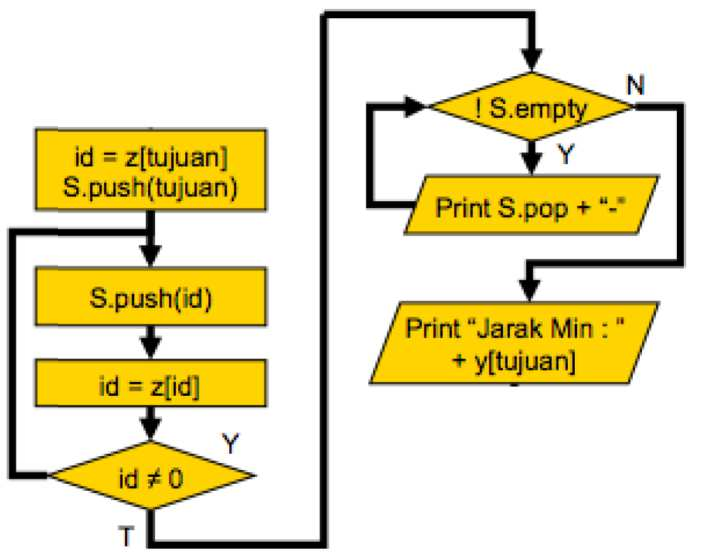
adalah jarak terpendek dan jalur untuk jarak terpendek. Informasi jarak terpendek terdapat pada nilai z di titik tujuan. Sedangkan untuk mendapatkan informasi jalur terpendek, harus

dilakukan operasi pop pada

stack, sebagaimana yang telah dijelaskan

pada bagian Cara

Penyelesaian. Flowchart untuk output terlihat seperti dibawah ini:



**11.2 Latihan**

Buatlah flowchart dari permasalahan shortest path problem diatas. Deskripsikan proses penyelesaian dengan menggunakan struktur data yang ada, dan kemudian buatlahlah flowchart keseluruhan.