**MAKALAH**

**ALGORITMA SEMESTER SATU**

**Makalah Ini Disusun untuk Memenuhi UAS Algoritma Semester Satu**

****

**Disusun Oleh :**

**Nama : Farkhan**

**NPM : 20081010060**

**Kelas : B**

**PROGRAM STUDI INFORMATIKAFAKULTAS ILMU KOMPUTER**

**UNIVERSITAS PEMBANGUNAN NASIONAL “VETERAN”**

**JAWA TIMUR**

**2021**

# BAB I

**PENDAHULUAN**

# BAB II

**PEMBAHASAN**

## **Algoritma dan Program**

* + 1. Algoritma

Algoritma merupakan sekumpulan langkah-langkah atau instruksi-instruksi yang terbatas untuk menyelesaikan suatu masalah. Algoritma digunakan untuk mengolah data menjadi sebuah informasi (input menjadi output), pemrogram wajib menyusun langkah detail (runutan) bagaimana komputer akan menyelesaikan masalah-masalah tersebut, langkah detail ini disebut algoritma. Kata algoritma pertama kali diperkenalkan oleh seorang ilmuan matematika yang berasal dari Persia yang bernama Al Khawarizmi, beliau hidup pada tahun 780-850 masehi.

Berikut ini merupakan suatu contoh algoritma yang digunakan untuk menghitung luas lingkaran.

* Dapatkan jari-jari lingkaran
* Hitung luas lingkaran dengan menggunakan rumus 3.14 x jari-jari x jari-jari
* Tampilkan nilai luas lingkaran

Selain menggunakan kalimat, algoritma juga bisa dituliskan menggunakan pseuducode. Contoh algoritma dalam bentuk pseuduode ialah sebagai beikut.

* Jari-jari 🡨 20
* Luas 🡨 3.14 \* (jari-jari2)
* Print (Luas)
  + 1. Program

Program adalah kumpulan instruksi yang digunakan untuk mengatur komputer agar melakukan suatu tindakan tertentu. Pada dasarnya, komputer hanyalah mesin kosong yang tidak bisa apa-apa atau tidak berfungsi tanpa adanya program. Komputer memiliki beberapa komponen, di antaranya adalah (1) perangkat keras (*hardware*), (2) perangkat lunak (*software*), dan manusia yang mengendalikannya atau disebut juga perangkat otak (*brainware*). Orang yang membuat program biasa disebut dengan pemrogram atau *programmer*. Aktivitas yang dilakukan ketika membuat program disebut dengan pemrograman atau *programming/coding*.

Program dibuat dari kumpulan baris kode-kode yang ditulis menggunakan kaidah/aturan bahasa pemrograman tertentu. Bahasa pemrograman memiliki fungsi dan peranannya masing-masing yang berbeda dengan bahasa pemrograman lainnya. Oleh karena itu, bahasa pemrograman yang digunakan untuk membuat suatu program harus sesuai dengan program seperti apa yang akan dibuat. Sama halnya manusia yang saling berkomunikasi, komputer dapat menjalankan pekerjaannya sesuai dengan instruksi yang mengikuti kaidah/aturan tertentu.

Secara garis besar, bahasa pemrograman dapat dibedakan menjadi dua bagian, yaitu sebagai berikut.

* Bahasa Tingkat Rendah (*Low-Level Language*)

Bahasa tingkat rendah ialah bahasa pemrograman yang berorientasi pada mesin. Bahasa tingkat rendah memberikan perintah ke komputer dengan menggunakan kode biner (0 atau 1). Kode biner ialah kode yang sangat sederhana yang berfungsi untuk menggantikan kode-kode tertentu dalam sistem biner. Bahasa yang berorientasi pada mesin membuatnya menjadi sulit dipahami oleh orang awam, bahkan oleh pemrogram sekalipun. Kelebihan dari bahasa tingkat rendah, yaitu sangat cepat dieksekusi karena dekat dengan mesin, sehingga komputer tidak perlu menerjemahkannya terlalu jauh. Berikut adalah contoh bahasa pemrograman tingkat rendah.

* B402 atau 1011 0100 0000 0010 artinya : masukkan angka 2 ke register AH
* B22A atau 1011 0010 0010 1010 artinya : muatlah angka A2 hex ke register DL
* CD21 atau 1100 1101 0010 0001 artinya : jalankan interupsi 21 heksadesimal

Ketiga perintah di atas akan dieksekusi secara berurutan dan menghasilkan karakter \* pada layar.

* Bahasa Tingkat Tinggi (*High-Level Language*)

Bahasa tingkat tinggi ialah bahasa pemrograman yang berorientasi pada bahasa manusia. Bahasa tingkat tinggi dirancang untuk dapat mudah dipahami oleh manusia, termasuk oleh pemula. Contoh bahasa pemrograman tingkat tinggi ialah Java, C, C++, Pascal, Basic, PHP, dan masih banyak lagi. Contoh penulisan kode dalam bahasa pemrograman tingkat tinggi adalah sebagai berikut.

* Write (‘\*’) -- PASCAL B22A
* Display ‘\*” – COBOL
* PRINT “\*” – BASIC
* Printf (“\*”) – C

Semua perintah di atas akan menampilkan hasil yang sama, yaitu menampilkan karakter \* pada layar.

Bahasa tingkat tinggi membutuhkan penerjemah atau *translator* untuk menerjemahkan bahasa pemrograman ke bahasa mesin agar kemudian bisa dieksekusi oleh komputer. Berdasarkan urutan kerjanya, *translator* dapat dibedakan menjadi dua bagian, yaitu sebagai berikut.

* Compiler

Compiler bekerja dengan menerjemah seluruh kode bahasa tingkat tinggi secara utuh terlebih dahulu dalam suatu excutable code, yang kemudian dieksekusi oleh mesin. File yang dihasilkan dari *compiler* ialah berupa .exe file. Contoh bahasa pemrograman yang menggunakan *compiler* ialah bahasa C, C++, dan Pascal.

Kelebihan Compiler :

1. Pengerjaan instruksi dilakukan dengan lebih cepat karena telah diterjemahkan ke dalam bahasa mesin secara utuh.
2. Kode sumber tidak perlu didistribusikan ke pengguna yang menjalankannya, sehingga kerahasiaannya terjamin.

Kekurangan Compiler :

Kode sumber program harus benar sepenuhnya secara sintak agar program dapat dijalankan.

* Interpreter

Pada interpreter, setiap baris instruksi bahasa tingkat tinggi diterjemahkan menjadi *intermediate code* per baris, untuk kemudian tiap baris *intermediate code* tersebut dieksekusi oleh mesin. Kekurangan dari interpreter ialah proses eksekusinya lebih lambat. Contoh bahasa pemrograman yang menggunakan interpreter ialah Python, Matlab, dan Java.

Kelebihan Interpreter :

Mudah untuk mencari kesalahan andaikan terdapat kesalahan pada program (debugging). Oleh karena itu, program dapat terus berjalan hingga akhirnya komputer menemukan kesalahan.

Kekurangan Interpreter :

1. Kode sumber harus selalu tersedia – isu pencurian hak cipta.
2. Proses eksekusi program berjalan dengan lebih lambat.

Tujuan dari dibuatnya program ialah untuk menyelesaikan suatu masalah atau tugas yang ada. Dalam membuat program, ada beberapa tahap yang dikerjakan, yaitu (1) melakukan analisis terhadap masalah yang akan diselesaikan, (2) membuat algoritma dari program yang akan dibuat, (3) mengimplementasikan algoritma ke dalam instruksi program, dan (4) mengeksekusi dan menjalankan program.

Analisis masalah merupakan tahapan yang sangat penting, diperlukannya pengalaman, pengetahuan, imajinasi, kreativitas, dan kecerdasan untuk menganalisis informasi apa yang akan menjadi masukan dan luaran, serta bagaimana mengolahnya.

**Gambar 2.1. Alur membuat program**

Di bawah ini adalah contoh kode program untuk menghitung luas lingkaran.

#include <stdio.h>

int main ( ) {

double jari\_jari;

double luas;

jari\_jari = 20;

luas = 3.14 \* jari\_jari \* jari\_jari;

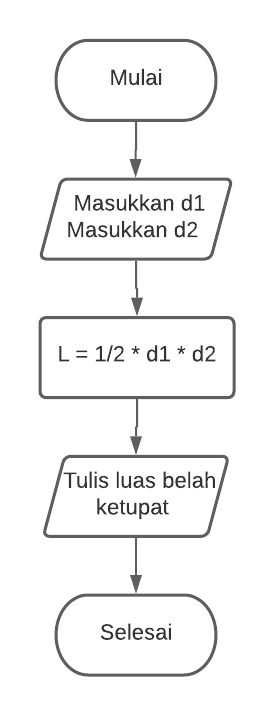
printf(“Luas lingkaran = %lf”, luas); return 0;

}

* + 1. Flowchart

*Flowchart* adalah algoritma yang dipresentasikan dalam bentuk diagram alir. Berikut ini adalah simbol *flowchart* beserta dengan kegunaannya.

**Gambar 2.2. Simbol flowchart**

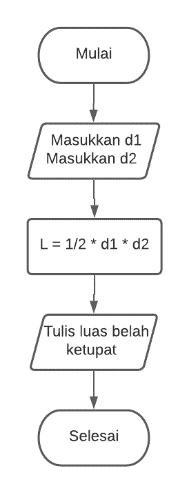
**Berikut ini adalah contoh *flowchart* untuk menghitung luas bela ketupat.

**Gambar 2.3. Flowchart luas lingkaran**

## **Struktur Dasar Algoritma**

* + 1. Runutan (*Sequential*)

Algoritma runutan adalah algoritma yang langkah-langkahnya dikerjakan atau dieksekusi secara berurutan dari awal hingga akhir sesuai dengan urutan program. Setiap instruksi dikerjakan tepat satu kali tanpa ada intruksi yang dikerjakan berulang. Akhir dari instruksi terakhir merupakan akhir dari algoritma.

Contoh *sequential* (1)

Algoritma menghitung luas belah ketupat :

1. Masukkan diagonal 1 (d1)
2. Masukkan diagonal 2 (d2)
3. Hitung luas (L = 1/2 \* d1 \*d2)
4. Tulis luas

Contoh *sequential* (2)

Masukkan alas, tinggi

Mulai

Tampilkan sisimiring

Selesai

Pseuducode menghitung sisi miring segitiga, dengan asumsi bahwa panjang alas dan tinggi sebuah segitiga siku-siku telah diketahui.

1. Masukkan (alas, tinggi)
2. Sisimiring 🡨 akarpangkat (alas \* alas + tinggi \* tinggi)
3. Tampilkan (sisimiring)

Contoh *sequential* (3)

Berikut ini adalah contoh diagram alir untuk menghitung konversi suhu dari Fahrenheit ke Celcius.

c 🡨 (f-32) \* 5/9

Masukkan f

c ß (f-32) \* 5/9

Tampilkan c

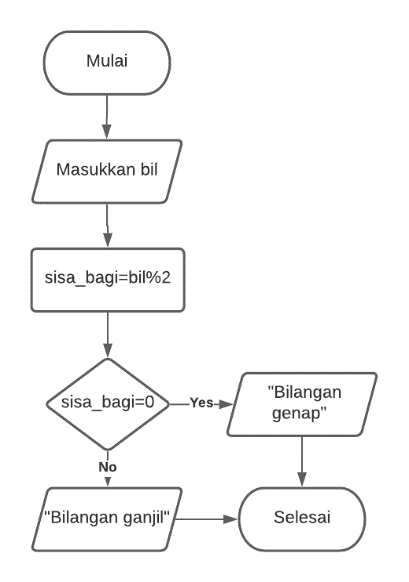
Mulai

Selesai

* + 1. Pemilihan (*Selection/Branching*)

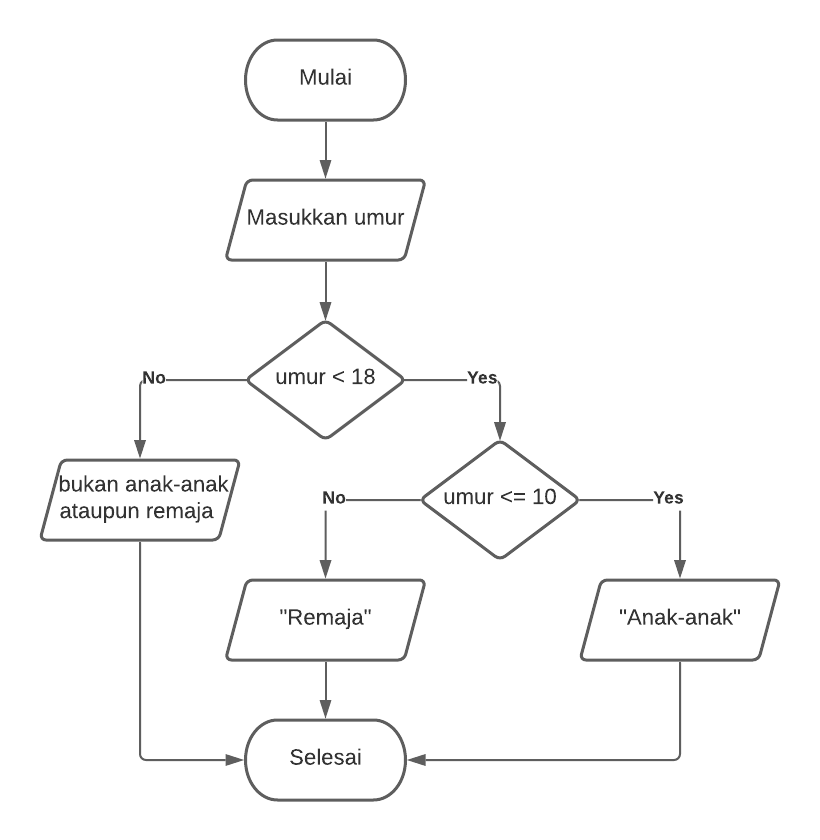
Struktur pemilihan adalah struktur program yang melakukan proses pengujian untuk mengambil suatu keputusan apakah suatu baris program atau blok kode akan dieksekusi atau tidak. Struktur pemilihan memungkinkan suatu instruksi dijalankan jika suatu kondisi terpenuhi atau tidak terpenuhi. Struktur pemilihan mampu memungkinkan pemroses mengikuti jalur aksi yang berbeda berdasarkan kondisi yang ada. Hanya instruksi program yang terpenuhi saja yang akan dijalankan, sehingga tidak semua baris kode akan dijalankan.

Contoh diagram alir pemilihan biasa secara umum.

Berikut ini adalah contoh diagram alir pemilihan/percabangan untuk menentukan bilangan ganjil atau genap.

Algoritma :

* Masukkan bilangan.
* Bilangan dimodulus 2 untuk mencari sisa baginya.
* Apabila sisa baginya sama dengan nol, maka bilangan tersebut adalah bilangan genap. Namun, jika sisa baginya tidak sama dengan nol, berarti bilangan tersebut adalah bilangan ganjil.

Pemilihan bersarang adalah seleksi bertingkat. Seleksi ini dilakukan apabila ingin menyeleksi kondisi secara lebih detail, sampai suatu kondisi yang dikehendaki tercapai. Contoh diagram alir pemilihan bersarang adalah sebagai berikut.

* + 1. Pengulangan (*Looping*)

Perulangan menyatakan suatu tindakan atau langkah yang dijalankan beberapa kali. Perulangan adalah instruksi yang dapat mengulang sederet instruksi secara berulang-ulang sesuai persyaratan yang ditetapkan. Sebagai contoh, jika ingin menampilkan tulisan “Selamat Belajar” sebanyak 10 kali, maka caranya adalah sebagai berikut.

Cara pertama :

printf(”Selamat Belajar”);

printf(”Selamat Belajar”);

…

printf(”Selamat Belajar”);

// hingga 10 kali.

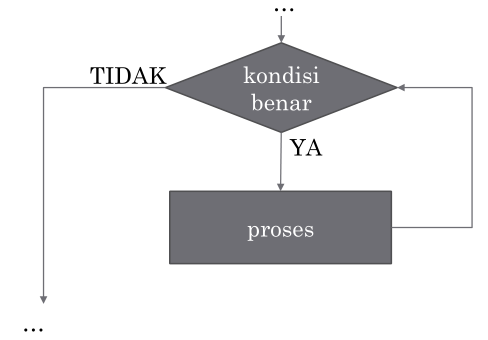
Cara kedua (struktur perulangan) :

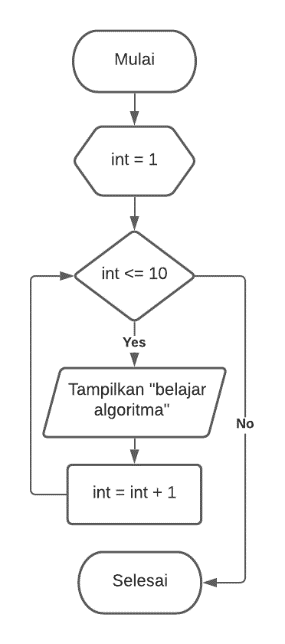
for (i=1; i<=10; i++)

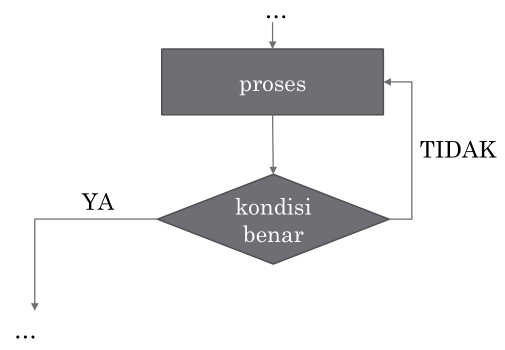
{

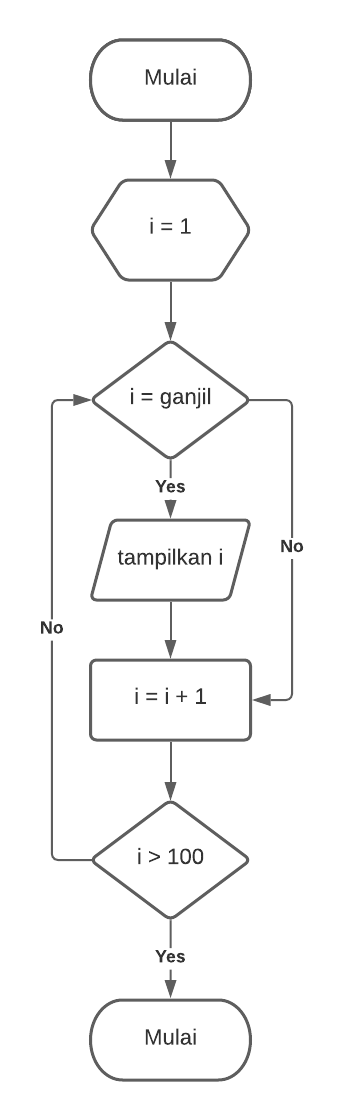
printf(“Selamat Belajar”);

}

Pada struktur perulangan, proses dapat dilakukan dengan satu atau beberapa langkah. Pada bentuk ini, proses akan dijalankan berulang-ulang selama suatu kondisi benar terjadi.

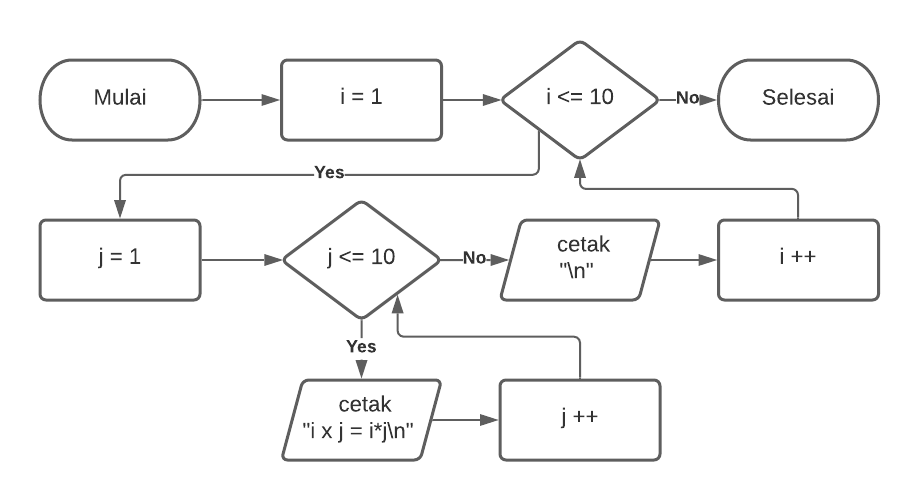
Pada diagram alir di samping, menunjukkan perulangan untuk menampilkan “Belajar Algoritma” sebanyak 10 kali.

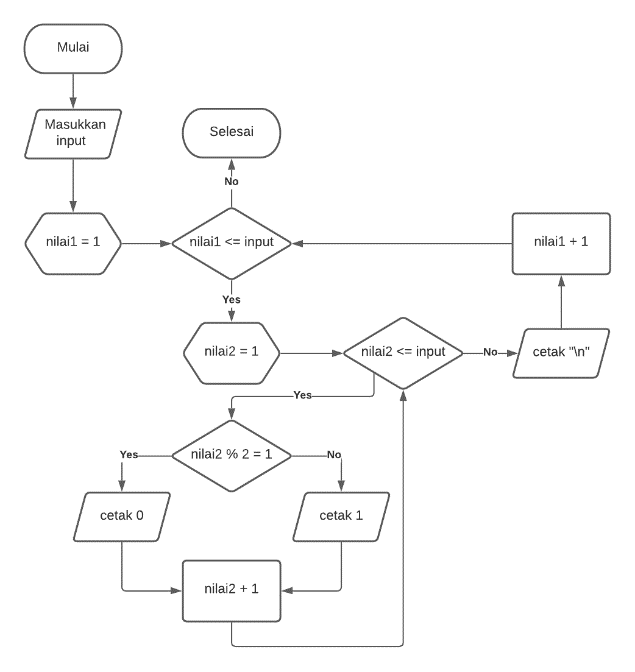
Selanjutnya, ada pula perulangan yang akan menjalankan instruksi selama kondisinya bernilai salah (*false*). Perulangan tersebut memiliki bentuk seperti pada gambar di bawah ini.

Contohnya, diagram alir di bawah ini menunjukkan perulangan yang menambahkan 1 angka ke sebuah variabel apabila kondisinya bernilai salah (*false*).

## **Kombinasi Struktur Dasar Algoritma**

Algoritma memiliki 3 (tiga) struktur dasar, yang sebelumnya telah dibahas pada subbab sebelumnya. Ketiga struktur dasar tersebut bisa dikombinasikan dengan struktur dasar yang lainnya. Untuk membuat diagram alir dari suatu program yang kompleks, sering kali dua atau bahkan tiga struktur dasar yang berbeda digunakan secara bersamaan dalam satu diagram alir yang sama.

Diagram alir di bawah ini menunjukkan program untuk membuat tabel perkalian, yang menggabungkan pemilihan dengan pengulangan. Hasil yang ditampilkan merupakan perkalian 1 hingga perkalian 10.

Ada pula diagram alir yang menggabungkan ketiga struktur dasar algoritma, yaitu runutan, pemilihan, dan pengulangan. Diagram alir di bawah ini menunjukkan program yang akan menampilkan angka 0 jika nilai yang dimasukkan *user* adalah ganjil dan menampilkan angka 1 jika nilai yang dimasukkan *user* adalah genap.

* + 1. Pseuducode

*Pseuducode* atau disebut juga kode semu merupakan deskripsi tingkat tinggi informal dan ringkas atas algoritma pemrograman komputer yang menggunakan konvensi struktural (suatu bahasa pemrograman) dan ditujukan untuk dibaca oleh manusia dan bukan oleh mesin.

Format penulisan *pseuducode* dibagi menjadi 3 (tiga) bagian, yaitu sebagai berikut.

1. Judul

Bagian judul diawali dengan kata “PROGRAM” dan diikuti dengan nama algoritma. Umumnya, nama algoritma hanya terdiri dari satu kata. Jika terdapat lebih dari satu kata, maka penulisan nama algoritma dapat ditulis dengan menghilangkan spasi atau menggantinya seperti PROGRAM HitungSisiMiring, PROGRAM Pencarian\_Angka, dan sebagainya.

1. Deklarasi

Bagian deklarasi digunakan untuk mendeklarasikan nama variabel beserta dengan tipe datanya. Tipe data pada aplikasi komputer, yaitu bilangan bulat, bilangan pecahan, teks, tanggal, dan sebagainya. Format penulisan deklarasi variabel diawali dengan tipe data dan diikuti dengan nama variabelnya.

1. Isi

Bagian isi merupakan bagian utama dari jalannya algoritma. Bagian yang berisi perintah-perintah sesuai dengan logika algoritma. Perintah tersebut dapat berupa runutan, pemilihan, pengulangan, dan/atau kombinasinya.

Contoh penulisan algoritma dalam bentuk *pseuducode* untuk algoritma pencarian sisi miring segitiga adalah sebagai berikut.

PROGRAM CariSisiMiring

DEKLARASI

int a, b

float c

ALGORITMA

read(a, b)

c = sqrt(a\*a + b\*b)

write("Sisi miring: ", c)

Penjelasan *pseuducode* :

Judul dari algoritma di atas adalah CariSisiMiring yang dituliskan di awal *pseudocode.* Kemudian pada bagian deklarasi, terdapat 3 buah variabel, yaitu a, b dan c. Variabel a dan b dideklarasikan dengan tipe data int, yaitu bilangan bulat, sedangkan variabel c dideklarasikan dengan tipe data float, yaitu bilangan pecahan

Pada bagian isi, terdapat 3 buah perintah, yaitu sebagai berikut.

1. Perintah read(a, b), yang digunakan untuk meminta masukan dari pengguna. Pada aplikasi yang telah jadi, yang memberikan masukan adalah pengguna aplikasi. Masukkan berupa 2 buah angka dapat dimasukkan melalui papan kunci.
2. Perintah c = sqrt(a\*a + b\*b) adalah perintah untuk memberikan nilai variabel c. Nilai variabel c didapatkan dari perhitungan akar kuadrat dari a kuadrat ditambah b kuadrat.
3. Perintah write("Sisi miring: ", c) merupakan perintah untuk menampilkan teks “Sisi miring: ” diikuti dengan isi dari variabel c. Variabel c dicetak di luar tanda petik sebagai penanda bahwa yang dicetak bukan teks “c” melainkan isi dari variabel c.

## **Raptor**

Raptor merupakan pemrograman yang berbasis flowchart, raptor dirancang khusus untuk membantu memvisualisasikan algoritma yang telah kita buat. Program raptor diciptakan secara visual dan dieksekusi secara visual dengan menelusuri eksekusi melalui flowchart. Biasanya kita lebih suka menggunakan flowchart untuk mengekspresikan algoritma, dan lebih berhasil menciptakan algoritma menggunakan raptor daripada menggunakan bahasa tradisional atau menulis flowchart tanpa raptor.

Raptor ditulis dalam kombinasi dari bahasa C# dan A# (port dari Ada untuk .NET Framework.) dan hanya didukung pada Windows. Raptor telah bereksperimen dengan Mono di Mac OS X dan Ubuntu. Versi Mac tidak berjalan sama sekali, tetapi raptor dapat berjalan pada Ubuntu dengan beberapa fitur yang harus dihilangkan.

*Raptor* memiliki beberapa mode, secara default kita memakai mode *Novice*. Mode *Novice* memiliki global *namespace* tunggal untuk setiap variabel. Mode *Intermediate* digunakan untuk membuat prosedur yang memiliki ruang lingkup mereka sendiri (memperkenalkan gagasan lewat parameter dan mendukung rekurs). Mode baru *raptor* adalah mode yang berorientasi objek, yaitu versi 2009.

*Raptor* bebas untuk didistribusikan sebagai layanan kepada masyarakat. *raptor* pada awalnya dikembangkan oleh dan untuk US Air Force Academy, Departemen Ilmu Komputer, namun penggunaannya telah menyebar dan *raptor* sekarang digunakan untuk pendidikan di lebih 17 negara pada setidaknya 4 benua.

*Raptor* juga di lengkapi dengan proses *generate flowchart* ke beberapa sumber kode yang sudah banyak di kenal seperti C++, Java, C# dan lain-lain. Sehingga pengguna tidak perlu lagi membangun dari awal sebuah sumber kode, karena dari *flowchart* yang telah di buat langsung di terjemahkan ke sumber kode oleh *raptor*.

Keunggulan dari perangkat lunak Raptor Interpreter Flowchart adalah dapat mengeksekusi *flowchart* yang telah di bangun menjadi sebuah visualisasi yang nyata, sehingga pengguna dapat mengetahui setiap tahapan *flowchart* yang mereka buat melalui eksekusi secara visual dalam tiap langkahnya. Selain itu kelebihan dari Raptor Interpreter Flowchart ini ada pada saat kita membuat *flowchart* dengan penulisan variabel, *raptor* di dukung dengan adanya fitur *auto complete* seperti layaknya pada Pemrograman Visual Basic.

Operator memerintahkan komputer untuk melakukan beberapa perhitungan pada data. Operator ditempatkan antara data yang dioperasikan (yaitu X / 3, Y +7, dan lain-lain), sedangkan fungsi menggunakan tanda kurung untuk menunjukkan data tersebut beroperasi pada (yaitu sqrt (4.7), sin (2,9)). Ketika dieksekusi, operator dan fungsi melakukan perhitungan dan mengembalikan hasil. RAPTOR memiliki operator dan fungsi sebagai berikut.

Basic math : +, -, \*, /, ^, \*\*, rem, mod, sqrt, log, abs , ceiling, floor.

Trigonometry : sin, cos, tan, cot, arcsin, arcos, arctan, arccot.

Relational : =, !=, /=, <, >, >=, <=.

Logical : and, or, not.

Miscellaneous : random, Length\_of.

Operator matematika dasar dan fungsi yang termasuk biasa (+, -, \*, /) serta beberapa yang tidak biasa “\*\*” dan “^” adalah *exponentiation*, seperti 2 \*\* 4 adalah 16 dan 3 ^ 2 adalah 9. Rem (*remainder*) dan mod (*modulus*) mengembalikan sisa (apa yang tersisa) ketika operan kanan membagi operan kiri, contoh : 10 rem 3 adalah 1, 10 mod 3 adalah 1 juga.

sqrt mengembalikan akar kuadrat contoh : sqrt (4) adalah 2.

log mengembalikan logaritma natural contoh : log (e) adalah 1.

abs mengembalikan nilai absolut contoh : abs (-9) adalah 9.

ceiling pada seluruh nomor contoh : ceiling (3,14159) adalah 4.

floor pada seluruh nomor contoh : floor (10/3) adalah 3.

Tanda “+” juga bekerja sebagai operasi *concatenation* untuk menggabungkan dua *string* atau *string* dan angka, contoh : “rata-rata adalah” + (Jumlah / Angka). length\_of mengembalikan jumlah karakter dalam sebuah variabel *string* (juga jumlah elemen dengan sebuah *array*), contoh : Nama ← “Stuff” diikuti dengan Length\_Of (Nama) adalah 5.

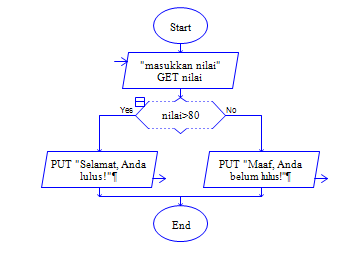
Kita terbiasa dengan fungsi trigonometri (sin, cos, tan, cot, arcsin, arcos, arctan, arccot). Mereka bekerja pada berbagai unit yang bernilai radian. (kita harus mengonversi dari derajat ke radian sebelum menggunakan fungsi tersebut.). arctan dan arccot adalah versi kedua parameter fungsi ini. (yaitu arctan (X / Y) ditulis dalam RAPTOR sebagai arctan (X, Y).

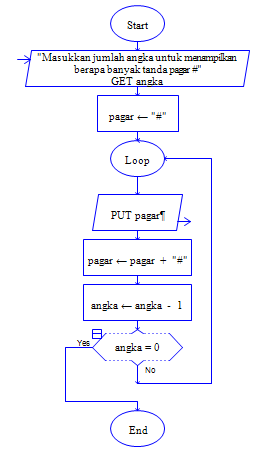
Dalam *raptor*, operator relasional dan operator logika hanya dapat digunakan dalam pengambilan keputusan sebagai bagian dari pemilihan dan pengulangan. Operator relasional adalah !== (tidak sama dengan), / = (tidak sama dengan), <, >, >=, dan <=. Operator relasional mengembalikan nilai “*Boolean*” dalam “*True*” atau “*False*” (ya atau tidak). Sebagai contoh, operasi X < Y akan mengembalikan TRUE jika nilai yang tersimpan dalam variabel X kurang dari nilai yang disimpan dalam variabel Y. Jika tidak, maka nilai FALSE dikembalikan. Hasil dari operasi relasional dapat digunakan oleh operator logika.

Fungsi secara acak mengembalikan angka antara 0 dan 1, contoh : X ← secara acak bisa menjadi 0, 0,23, 0,46578, dll. Jika kita memerlukan nomor acak dalam kisaran yang berbeda maka kita bisa menggabungkan fungsi acak dengan operasi lain. Misalnya, random \* 100 akan mengevaluasi ke angka antara 0 dan 100. ceiling (random \* 100) akan mengevaluasi ke seluruh nomor antara 1 dan 100.

Berikut ini adalah contoh *flowchart* yang dibuat menggunakan raptor.

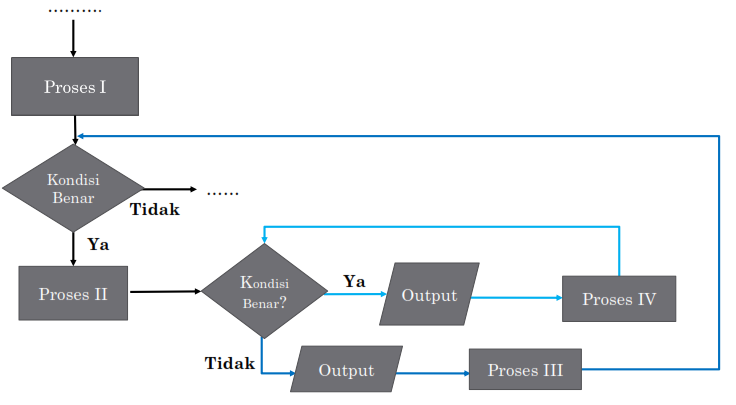
1. Program menentukan kelulusan berdasarkan nilai.



1. Program menampilkan tanda # (pagar) sebanyak angka yang dimasukkan.

## **Perulangan Bersarang**

Perulangan bersarang merupakan perulangan yang memiliki perulangan lagi di dalamnya. Perulangan For bersarang adalah perulangan For yang berada pada perulangan For lainnya. Perulangan yang lebih dalam akan diproses terlebih dahulu sampai habis, kemudian perulangan yang lebih luar baru akan bertambah, mengerjakan perulangan yang lebih dalam lagi mulai dari nilai awalnya dan seterusnya. Contoh perulangan bersarang dalam kehidupan sehari-hari bisa berupa pergi ke sekolah, bekerja, memasak, dan lain sebagainya.

Secara umum, struktur perulangan bersarang dapat digambarkan sebagai berikut.

Salah satu contoh program perulangan bersarang, ialah membuat segitiga siku-siku yang disusun dari karakter \* (bintang), dengan tinggi segitiga ditentukan melalui papan kunci.

Algoritma membuat segitiga siku-siku dengan karakter \* (bintang).

1. Tentukan Variabel

Tinggi : Tinggi Segitiga

i : variabel pencacah 1 (jumlah baris)

j : variabel pencacah 2 (jumlah kolom)

1. Input Nilai untuk menentukan jumlah baris bintang \* (Tinggi Segitiga)
2. Proses perulangan untuk menghitung jumlah bintang \*

for i = 1 to i < = Tinggi

for j = 1 to j < = i

1. Cetak Hasil \*

Sumber kode membuat segitiga siku-siku dengan karakter \* (bintang) :

#include <stdio.h>

main ( )

{

int i, j, tinggi;

printf(“Masukkan tinggi segitiga : “);

scanf(“%d”, &tinggi);

for(i=1; i <= tinggi; i++)

{

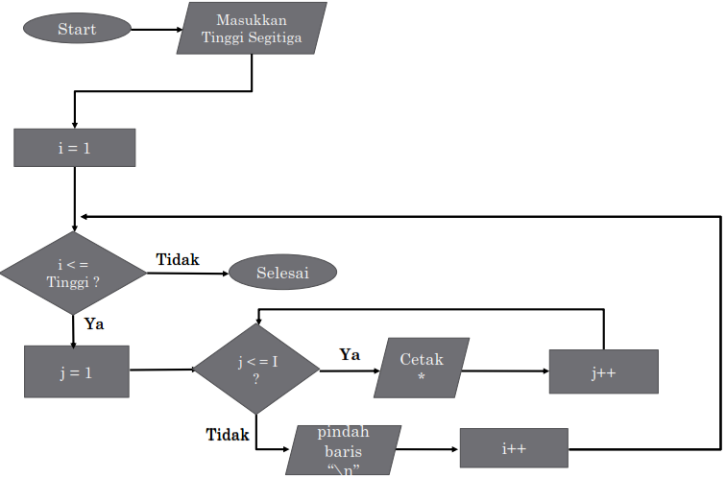
for(j=1; j <= i; j++)

printf(“\*”);

printf(“\n”);

}

}

*Flowchart* membuat segitiga siku-siku dengan karakter \* (bintang).

## **Sub Program**

## **Perulangan Bersarang**

# BAB III

**PENUTUP**