PERTEMUAN 2

KONSEP DASAR PEMROGRAMAN

I. Algoritma Pemrograman Yang Baik

Ciri-ciri algoritma pemrograman yang baik adalah:

1. Memiliki logika perhitungan/metode yang tepat dalam memecahkan masalah
2. Menghasilkan output yang tepat dan benar dalam waktu yang singkat
3. Ditulis dengan bahasa yang standar secara sistematis dan rapi sehingga tidak menimbulkan arti ganda.
4. Ditulis dengan format yang mudah dipahami dan diimplementasikan ke dalam bahasa pemrograman.
5. Semua operasi yang dibutuhkan terdefinisi dengan jelas.
6. Semua proses harus berakhir setelah sejumlah langkah dilakukan.

Berikut ini contoh program yang mempunyai algoritma yang TIDAK BAIK karena mengandung kesalahan logika.

```
#include <iostream>
void main ()
{
  int i ;
  i = 0 ;
  while (i <= 5)
    {
                cout<<"STMIK Swadarma"<<endl;
                i = i + i
    }
   return 0;
}
```

Berikut ini contoh program yang mempunyai algortitma yang BAIK karena mempunyai logika yang benar

```
#include <iostream>
void main ()
{
  int i;
  i = 0;
  while (i <= 5)
    {
      cout<<"STMIK Swadarma"<<endl;
      i = i + 1
    }
  return 0;
}
```

II. Standar Program Yang Baik

A. Standar Pemecahan masalah

Teknik untuk dapat membantu memecahkan masalah:

Top Down : program disusun dari yang paling umum (masalah yang komplek), kemudian dibagi menjadi masalah yang lebih kecil.

Bottom Up : program disusun dari pemecahan masalah yang kompleks, kemudian menggabungkan prosedur-prosedur yang ada menjadi satu kesatuan program sebagai penyelesaian masalah tersebut.

B. Standar Penyusunan Program

1. Kebenaran logika dan penulisan

- Program yang disusun harus memiliki logika dalam pemecahan masalah.
- Program yang dibuat harus memiliki ketepatan, ketelitian dan kebenaran sehingga menghasilkan program yang baik.

2. Waktu minimum penulisan dan eksekusi program.

a. Contoh Logika pengujian yang **TIDAK BAIK** karena pengujian yang berulang-ulang sehingga waktu eksekusi tidak efisien

```
if (kondisi1)
{
    pernyataan_1;
    ……….
}
if (kondisi2)
{
    pernyataan_2;
    ……….
}
if (kondisi3)
{
    pernyataan_3;
    ……….
}
```

b. Logika pengujian yang **BAIK** sehingga waktu lebih efisien:

```
if (kondisi_1)
{
    pernyataan_1;
    ……….
else
        if (kondisi_2)
          pernyataan_2;
       ……….
   else
       if (kondisi_3)
          pernyataan_3;
          ……….
}
```

3. Ekspresi Penggunaan Memori

Programer harus memperhatikan teknik-teknik pembuatan program yang nantinya akan meminimumkan penggunaan memori agar proses eksekusi program lebih cepat.

Hal-hal yang perlu diperhatikan yaitu:

a. Penggunaan tipe data yang cocok untuk kebutuhan program.

b. Hindari penggunaan berulang-ulang untuk variabel berindeks.

c. Penggunaan *record size* sesuai dengan kebutuhan dari data item. Misalnya untuk data item password tidak perlu mendeklarasikan dengan panjang (*Length*) 100 karakter, karena akan memboroskan memori.

4. Perawatan dan pengembangan program

Penyusunan program harus mempunyai sifat kesederhanaan dan kejelasan dari program yang nantinya akan dikembangkan dan membantu dalam perawatan.

5. Portabilitas

Bahasa pemrograman dan program yang disusun sebaiknya bisa dipakai pada berbagai tipe komputer yang berbeda-beda dan berbagai jenis sistem operasi.

6. User friendly

Program harus memiliki fasilitas yang memberikan kemudahan bagi user untuk pengoperasian, seperti menu tampilan yang informatif, pesan-pesan, dan lain-lain.

7. Pemrograman Modular

- **Modular,** yaitu program dipecah menjadi beberapa modul, dan setiap modul menunjukkan fungsi dan tugas tunggal.
- Modul program adalah sekumpulan instruksi yang memiliki operasi-operasi dan data yang didefinisikan.
- Modul memiliki struktur internal yang tidak tergantung dengan program yang lain, dan merupakan satu kesatuan yang utuh yang dapat digunakan secara berulang-ulang.
- **Standar Perawatan Program**

1. Dokumentasi

Merupakan catatan dari setiap langkah pekerjaan membuat program dari awal sampai akhir, dan biasanya digunakan untuk penelusuran kesalahan dan pengembangan program.

2. Penulisan Instruksi

a. Tulis satu baris untuk satu instruksi .
b. Awal dan akhir statemen dari sekumpulan statemen ditulis pada kolom yang sama (tabulasi)
c. Gunakan sebaris atau beberapa baris kosong sebagai pemisah.
d. Bedakan bentuk huruf dalam penulisan program, misalnya instruksi ditulis dengan huruf besar sedangkan variabel dengan huruf kecil.
e. Hindari pengunaan konstanta dalam penulisan rumus. Misalnya umur = 25 tahun yang ditulis pada saat pendaftaran pasien, dapat diambil dari tanggal lahir pasien.
f. Hindari pernyataan untuk Percabangan (**IF** statement) yang sangat rumit dan Nested Loop (Loop di dalam Loop lain) yang berlebihan.
g. Gunakan "kurung buka dan tutup" dalam menulis suatu ekspresi Aritmatika atau logika.
h. Gunakan "Spasi" dalam menulis statemen atau instruksi

III. Sifat Penulisan Program

a. Program Oriented

Penulisan program yang struktur programnya selalu berubah, apabila kondisi data yang diproses di dalam program tersebut bertambah volume datanya.

Selain itu penulisan program ini bersifat statis dan tidak fleksibel.

Contoh: Program Animasi

b. Data Oriented

Penulisan program yang struktur programnya tidak selalu berubah, walaupun volume data yang diproses di dalam program tersebut dalam jumlah besar. Selain itu pula penulisan program ini bersifat dinamis dan mempunyai tingkat fleksibilitas yang tinggi.

Contoh : Program aplikasi

- **Program Interaktif**

Penulisan program yang terstruktur, dimana programnya dapat dipergunakan oleh pengguna secara mudah dan dapat dimengerti tentang proses yang sedang dilakukan oleh program tersebut serta dapat mengatur kebutuhan akan peranti masukan dan keluaran.