KONSEP PEMROGRAMAN MODULAR

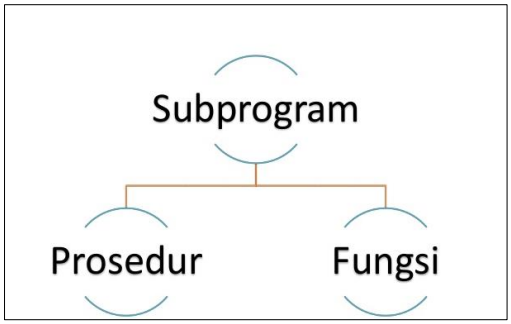
Pemrograman modular adalah sebuah metode pembuatan program dengan cara memecah masalah menjadi beberapa kelompok masalah yang lebih kecil. Dengan membagi masalah menjadi beberapa modul maka masalah tersebut akan menjadi lebih sederhana sehingga program dapat menjadi lebih mudah disusun dan dipahami. Untuk menyusun program modular dapat menggunakan konsep fungsi, prosedur ataupun *subroutine.*

Keuntungan pemrograman modular :

1. Program yang kompleks bisa dibuat lebih ringkas
2. Mudah dibaca dan dimengerti
3. Mudah didokumentasi
4. Mengurangi dan mudah mencari kesalahan program, karena kesalahan yang terjadi bersifat "lokal".
5. Membantu memahami algoritma yang dibuat dan mengembangkannya.
6. Memudahkan penulisan bagian program yang sama, modularisasi menghindari penulisan bagian program yang berulang. Sub program dapat dipakai berulang kali dengan hanya memanggilnya tanpa menuliskan banyak yang diulang-ulang.

Secara umum terdapat dua bentuk modul program yaitu procedure/prosedur dan function/fungsi, berikut perbedaan keduanya :

1. Procedure bisa mengembalikan atau tidak mengembalikan nilai/hasil, sedangkan function wajib mengembalikan nilai keluaran.
2. Procedure membutuhkan suatu variabel khusus untuk menampung hasil/nilai ketika terjadi suatu proses perhitungan, sedangkan function tidak membutuhkan karena berlaku ketentuan bahwa nama fungsi = nama/variabel proses.
3. Pada procedure pencetakan hasil/nilai berada dalam blok subrutinnya sendiri yang selanjutnya dipanggil nama procedurenya di dalam program utama, sedangkan pada function proses pencetakan hasil/nilai dibuat di program utama ketika pemanggilan function-nya.



Jenis Sub Program - Prosedur dan Fungsi (Sumber : e-materiku)

Modul pada bahasa C++ dikenal dengan nama fungsi (function). Bahasa C terdiri dari fungsi-fungsi, baik yang langsung dideklarasikan dalam program ataupun dipisah di dalam header file. Fungsi yang selalu ada pada program C++ adalah fungsi main().

Selain bersifat modular, fungsi merupakan salah satu dasar penyusunan blok pada C++. Sebuah program C++ minimal mengandung sebuah fungsi, yaitu fungsi main(). Fungsi ini menjadi awal dan akhir eksekusi program C++. Badan fungsi dimulai dari tanda kurawal pembuka { hingga tanda kurawal penutup }, semua yang terletak didalam tanda { } tersebut disebut blok. Tanda () digunakan untuk mengapit argumen fungsi, yaitu nilai yang akan dilewatkan / dimasukkan ke dalam fungsi.

A. PROSEDUR

Bentuk Umum Prosedur :

```
void nama_prosedur (daftar_-parameter)
{
  /*kumpulan kode program di dalam prosedur*/
}
```

Contoh prosedur lengkap dengan program utamanya :

```
#include <iostream>
using namespace std;
// Deklarasi dua prosedur, int a sebagai parameter formal
void ContohProsedur(int a);
void ContohProsedur1(int a);

// Fungsi Utama atau main program
int main(){
    int panjang = 5;


//memanggil dua prosedur di atas, panjang sebagai parameter aktual
    ContohProsedur(panjang);
    ContohProsedur1(panjang);
    return 0;
}

//prosedur untuk menghitung dan mencetak luas persegi panjang
//int panjang sebagai parameter input, dan var panjang sudah diberi harga awal = 5 pada
main program di atas

void ContohProsedur(int panjang)
{
    int lebar, luas;
    cout<<"\n\nMasukkan Lebar Persegi Panjang : ";cin>>lebar;
    luas=panjang*lebar;
    cout<<"Luas Persegi Panjang = “<<luas<<endl;
}

//prosedur untuk menghitung dan mencetak keliling persegi panjang
//int panjang sebagai parameter input, dan var panjang sudah diberi harga awal = 5 pada
main program di atas

void ContohProsedur1(int panjang){
    int lebar, keliling;
    cout<<"\n\nMasukkan Lebar Persegi Panjang = ";cin>>lebar;
    keliling=(panjang+lebar)*2;
    cout<<"Keliling Persegi Panjang = “<<keliling<<endl;
}
```

Contoh output prosedur di atas :
(ingat, nilai panjang sudah ditentukan di dalam program utama = 5)

Masukkan Lebar Persegi Panjang = 4
Luas Persegi Panjang = 20

Masukkan Lebar Persegi Panjang = 6
Keliling Persegi Panjang = 22

B. FUNGSI
Bentuk Umum Fungsi :
Bentuk umum Fungsi adalah sebagai berikut:

```
TipeData NamaFungsi (DaftarParameter){
   /*kode program di dalam fungsi*/
   return nilaireturn;
}
```

Contoh fungsi lengkap dengan program utamanya:

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

// Deklarasi Fungsi, int a sebagai parameter formal
int ContohFungsi(int a);

// Fungsi Utama
int main(){
   int luas1, luas2, totalluas;
   int panjang = 5;

   // memanggil fungsi ContohFungsi, panjang sebagai parameter aktual
   luas1 = ContohFungsi(panjang);
   luas2 = ContohFungsi(panjang);
   total_luas = luas1 + luas2;

   cout<<"\n\nLuas Gabungan Kedua Persegi Panjang adalah = "<<total_luas<<endl;
   return 0;
}

// Contoh Fungsi, int panjang sebagai parameter input
int ContohFungsi(int panjang){
   int lebar, luas;
   cout<<"\n\nMasukkan Lebar Persegi Panjang = ";cin>>lebar;
   luas=panjang*lebar;
   cout<<"Luas Persegi Panjang adalah "<<panjang<<" x "<<lebar<<" = "<<luas;
   return luas;
}
```

Contoh output fungsi di atas :
(ingat, nilai panjang sudah ditentukan di dalam program utama = 5)

Masukkan Lebar Persegi Panjang = 2
Luas Persegi Panjang adalah 5 x 2 = 10

Masukkan Lebar Persegi Panjang = 5
Luas Persegi Panjang adalah 5 x 5 = 25

Luas Gabungan Kedua Persegi Panjang adalah = 35

C. REKURSIF

Rekursif adalah sub program yang memanggil dirinya sendiri selama kondisi pemanggilan dipenuhi. Rekursif umumnya dipakai untuk permasalahan yang memiliki langkah penyelesaian yang terpola atau langkah-langkah yang teratur. Bila ada suatu permasalahan dan sudah diketahui algoritma penyelesaiannya, maka sub program rekursif dapat menjadi pilihan untuk digunakan.

Fungsi-fungsi yang dapat diubah ke bentuk rekursif antara lain perhitungan faktorial bilangan bulat positif n sebagai berikut :

Jika n > 1 → n! = n (n-1)!
Jika n=0 atau 1 → n! = 1

Kode programnya :

```
int faktorial(int n)
{
 if ((n==0) || (n==1))
 return (1);
 else
 return (n * faktorial (n-1));
}
```

Penerapan lainnya pada perhitungan Fibonacci.Deret bilangan fibonacci adalah serangkaian deret angka yang susunannya merupakan penjumlahan dari dua angka sebelumnya.
Contoh deret Fibonacci :

0 1 1 2 3 5 8 13 21

Rumus deret Fibonacci dapat ditulis Fibo(n) = Fibo (n-1) + Fibo (n-2), artinya suku / deret ke-n perupakan penjumlahan dari dua suku sebelumnya.
Deret ke 0= Fibo(0) = 0
Deret ke 1 = Fibo(1) = 1
Deret ke 2 = Fibo(2) = 1
Deret ke 3 = Fibo(3) = 2
Deret ke 4 = Fibo(4) = Fibo(3) + Fibo(2) = 2 + 1 = 3, dan seterusnya

Berikut kode program untuk menampilkan deret bilangan Fibonacci menggunakan dua pendekatan yaitu non-rekursif dan rekursif. Bandingkan perbedaan di dalam kedua programnya.

```
//pendekatan non-rekursif
#include <iostream>
using namespace std;
```

```
int main()
{
   int n, f1 = 0, f2 = 1, lanjut = 0;
   cout << "Masukan jumlah deret Fibonacci :  ";
   cin >> n;
         cout<<endl;
   cout << "Deret Fibonacci: ";

   for (int i = 1; i <= n; ++i)
   {
      // mencetak dua deret Fibonacci pertama
      if(i == 1)
      {
         cout << " " << f1<<" ";
         continue;
      }
      if(i == 2)
      {
         cout << f2 << " ";
         continue;
      }
      lanjut = f1 + f2;
      f1 = f2;
      f2 = lanjut;
       // mencetak deret Fibonacci selanjutnya
      cout << lanjut << " ";
   }
   return 0;
}
```

```
//pendekatan rekursif
#include <iostream>
using namespace std;

//fungsi rekursif Fibo
int fibo(int m) {
   if (m == 0 || m ==1){
      return m;
    } else {
      return (fibo(m-1) + fibo(m-2));
    }
   }

int main() {

   int n, m= 0;
    cout << "masukan jumlah deret Fibonacci :  ";
   cin >> n;
   cout << "Deret Fibonacci: ";
```

```
   for (int i = 1; i <= n; i++){
      cout << fibo(m) <<" ";
      m++;
   }
         return 0;
}
```

Perhatikan fungsi rekursif yang diberi nama Fibo pada program di atas:

```
int fibo(int m) {
   if (m == 0 || m ==1){
      return m;
    } else {
      return (fibo(m-1) + fibo(m-2));
    }    }
```

Fungsi rekursif dengan nama Fibo tersebut membawa sebuah parameter yaitu variabel m. Di dalam fungsi tersebut terdapat percabangan if, jika nilai m = 0 atau 1 maka nilai yang kembali (return value) pada fungsi tersebut = nilai itu sendiri yaitu 0 dan 1.
Rumus deret bilangan Fibonacci merupakan penjumlahan dari dua bilangan sebelumnya, maka perlu memperoleh dua nilai awal yaitu 0 dan 1 agar dapat dijumlahkan, dan menjadi nilai pada deret selanjutnya. Apabila dijalankan akan menghasilkan output yang sama dengan program pertama (non-rekursif).

Penerapan fungsi rekursif lainnya dapat dilihat pada program di bawah ini, yaitu fungsi mencetak ke layar. Fungsi ini mencetak nilai dari paremeter yang dilempar kepadanya, jika nilai > 0 maka fungsi akan mencetak nilai dari parameter tersebut kemudian memanggil dirinya lagi, jika nilai <= 0 maka program berhenti.

```
void cetak(int n)
   {
      if (n>0)
      {
        printf ("\n cetak: %i",n);
       cetak(n-1);
      }
 }
```