STRUKTUR DATA

POKOK BAHASAN - 2
PEMROGRAMAN MODULAR

Oleh :

NAZARUDDIN AHMAD, S.T, M.T

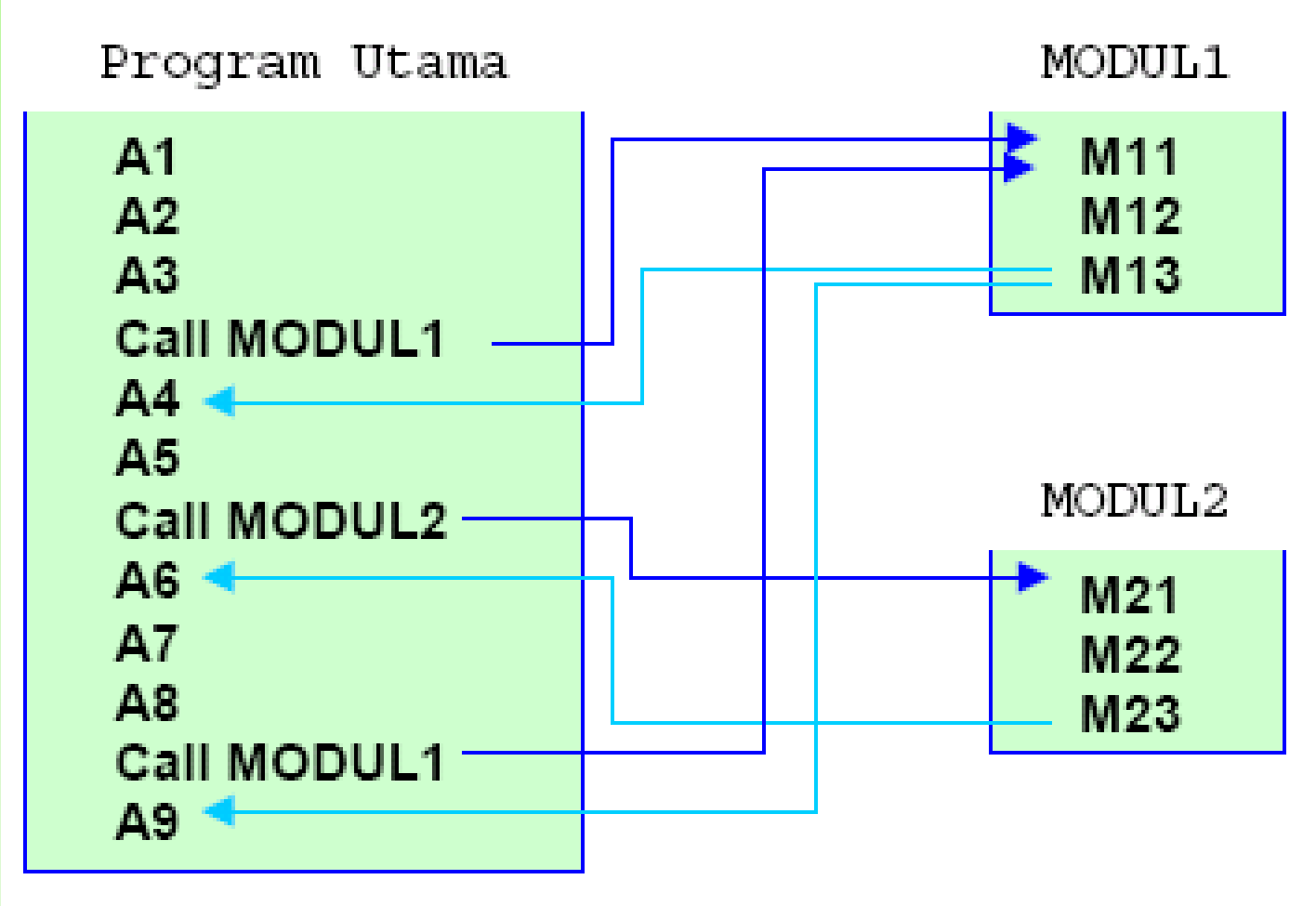
1. Pemrograman Modular

- Pemrograman Modular adalah pemrograman yang dilakukan dengan membuat subprogram-subprogram (Modul) diluar dari program utama.
- Modul yang sudah dirancang dapat dipasang ke dalam program lain yang membutuhkan.
- Teknik pemrograman modular (procedure dan function)

Modularisasi memberikan dua keuntungan :

a. Untuk aktifitas yang harus dilakukan lebih dari satu kali, modularisasi menghindari penulisan teks program yang sama secara berulang kali.

b. Kemudahan dalam menulis dan menemukan kesalahan (*debug*) program


Program Utama
A1
A2
A3
Call MODUL1
A4
A5
Call MODUL2
A6
A7
A8
Call MODUL1
A9
MODUL1
M11
M12
M13
MODUL2
M21
M22
M23

1.1 Procedure

Procedure

adalah suatu rutin yang melakukan proses tertentu tanpa adanya pengembalian nilai.

Pendefenisian prosedur meliputi :

Menuliskan nama prosedur, mendeklarasikan nama-nama konstanta, peubah dan tipe (jika ada) dan menjabarkan rangkaian aksi yang dilakukan.

Variabel Lokal dan Variabel Global

Variabel Lokal :

- Variabel yang terdapat pada bagian deklarasi prosedur.
- Bersifat lokal, hanya dapat digunakan di dalam prosedur yang melingkupinya.

Variabel Global :

- Variabel yang dideklarasikan di program utama.
- Bersifat global, dapat digunakan di bagian manapun dalam program, baik di program utama maupun di prosedur.

Passing Parameter

Kebanyakan program memerlukan pertukaran informasi antara prosedur (atau fungsi) dan titik di mana ia dipanggil.

→ parameter berfungsi sebagai media komunikasi antara modul dengan program pemanggil.

Tiap item data ditransferkan antara parameter aktual dan parameter formal.

- **Parameter aktual** : parameter yang disertakan pada waktu pemanggilan.
- **Parameter formal** : parameter yang dideklarasikan di bagian *header* prosedur itu sendiri.
- Saat prosedur dipanggil, parameter aktual menggantikan parameter formal.
- Tiap parameter aktual berpasangan dengan parameter formal yang bersesuaian.

Passing Parameter

Aturan penting yang harus diperhatikan:

- Jumlah parameter aktual pada pemanggilan prosedur harus sama dengan jumlah parameter formal pada deklarasi prosedurnya.
- Tiap parameter aktual harus bertipe sama dengan tipe parameter formal yang bersesuaian.
- Tiap parameter aktual harus diekspresikan dalam cara yang sesuai dengan parameter formal yang bersesuaian, bergantung pada jenis parameter formal.

Procedure dibagi 2 :

a. Procedure tanpa parameter

BU :

```
Program <Nama_Program>

Var
    <menentukan variabel yang digunakan>

    Procedure <Nama_Procedure>
    Begin
        <statement program procedure>
    End;

Begin
    <statement program utama>
End.
```

Contoh 1: (Pascal)

```
Program Hitung;

uses wincrt;

var
   x,y:integer;

    Procedure Tambah;
    var
        p:integer;

    Begin

          p:=x+y;

          write('x+y=',p);
    end;

Begin
       write('x=');read(x);
       write('y=');read(y);
       Tambah;

end.
```

Variabel Global

Variabel Lokal

Deklarasi Prosedur

Pemanggilan Prosedur

Program Utama

Contoh 2: (C++)

```
#include<stdio.h>
#include<conio.h>

int a,b;

void Tambah()
{
     int p;

     p=a+b;
     printf("x+y =%d",p);
}

int main()
{
     printf("x =");scanf("%d", &a);
     printf("y =");scanf("%d", &b);
     Tambah();

     getch();
     return 0;
}
```

Variabel Global

Variabel Lokal

Deklarasi Prosedur

Pemanggilan Prosedur

Program Utama

b. Procedure dengan parameter

BU :

```
Program <Nama_Program>

Var
    <menentukan variabel yang digunakan>

    Procedure <Nama_Procedure>(<Daftar Parameter>);
    Begin
        <deklarasi procedure>;
    end;
Begin
    <Statement program utama>
End.
```

Contoh 3: (Pascal , with parameter)

```
Program Hitung;

uses wincrt;

var
   p,x,y:byte;

   Procedure Tambah(a,b:byte);
   Begin
        p:=a+b;
        write('x+y =',p);
   End;

Begin
     write('x =');read(x);
     write('y =');read(y);
     Tambah(x,y);
End.
```

parameter

Contoh 4: (C++ , with parameter)

```
#include<stdio.h>
#include<conio.h>

int x,y;

void Tambah(int a, int b)
{
     int p;

     p=a+b;
     printf("x+y = %d",p);
}

int main()
{
     printf("x = ");scanf("%d",&x);
     printf("y = ");scanf("%d",&y);
     Tambah(x,y);

     getch();
     return 0;
}
```

STRUKTUR DATA

DISKUSI

TUGAS 2

Buatlah program menggunakan procedure untuk mencari nilai rata-rata dari jumlah nilai yang diinput.

Implementasikan menggunakan :

Pascal dan C++

Format file :

Nama_NPM_Kelas.pas

Nama_NPM_Kelas.cpp

Kirim ke email : nazar.unigha08@gmail.com

Paling terlambat diterima : Kamis , jam 23.59 wib

1.2 Function (Fungsi)

- Modul program yang mengembalikan / memberikan (return) sebuah nilai yang bertipe sederhana.

  → tipe data sederhana : integer, real, boolean dan string

- Dalam matematika :

  $f(x,y) = 3x - y + xy$

  → f adalah sebuah fungsi dengan parameter x dan y.

- Nilai yang diberikan fungsi tergantung nilai parameter masukannya.

- Fungsi diakses dengan memanggil namanya (sama seperti prosedur).
- Fungsi dapat mengandung parameter formal berjenis parameter masukan
- Fungsi harus dideklarasikan dengan tipenya atau jenis hasilnya.

Function dibagi 2 :

a. Function tanpa parameter

BU :

```
Program <Nama_Program>

Var
    <menentukan variabel yang digunakan>

    Function <Nama_Function>(tipe_data);
    Begin
        <Statement Function>;
    end;
Begin
    <Statement program utama>;
End.
```

Contoh 1: Pascal

```
Program Hitung;

uses wincrt;

var
   x,y:byte;

   Function Tambah:integer;
   Begin
        Tambah:=x+y;
   end;
Begin
     write('x=');read(x);
     write('x=');read(y);
     write('x+y =',Tambah);
End.
```

Contoh 2: C++

```
#include<stdio.h>
#include<conio.h>

int x,y;

int Tambah()
{
     int p;
     p=x+y;
}

int main()
{
     int z;

     printf("x=");scanf("%d",&x);
     printf("y=");scanf("%d",&y);

     z=Tambah(x,y);

     printf("x+y = %d",z);

     getch();
     return 0;
}
```

```
Format Function dalam C++ :

type name(argument1, argument2, ...)
```

- type adalah:

  type data yang akan dikembalikan oleh function.

- name adalah:

  nama yang digunakan untuk memanggil function.

b. Function dengan parameter

BU :

```
Program <Nama_Program>

Var
    <menentukan variabel yang digunakan>

    Function <Nama_Function>(<Daftar Parameter>)(tipe_data);
    Begin
        <Statement Function>;
    end;
Begin
    <Statement program utama>;
End.
```

Contoh 3: Pascal (Function with parameter)

```
Program Hitung;

uses wincrt;

var
   x,y:integer;

   Function Tambah(a,b:integer):integer;
   Begin
        Tambah:=a+b;
   end;
Begin
     write('x=');read(x);
     write('y=');read(y);
     write('x+y=',Tambah(x,y));
End.
```

Contoh 4: C++ (Function with parameter)

```
#include<stdio.h>
#include<conio.h>

int x,y;

int Tambah(int a, int b)
{
     int p;
     p=a+b;
}

int main()
{
     int z;

     printf("x=");scanf("%d",&x);
     printf("y=");scanf("%d",&y);

     z=Tambah(x,y);

     printf("x+y = %d",z);

     getch();
     return 0;
}
```

Beda Function dan Procedure

1. Pada fungsi, nilai yang dikirimkan balik terdapat pada nama fungsinya. Pada contoh, nama fungsi tersebut adalah Tambah dan nilai yang dikirim balik berada pada nama fungsi tersebut. Sehingga nama fungsi ini harus digunakan untuk menampung hasil yang akan dikirimkan dari fungsi, sebagai berikut :

```
Tambah := x+y ;
```

Nama fungsi yang berisi nilai yang akan dikirimkan

Beda Function dan Procedure

2. Karena nilai balik berada di nama fungsi tersebut, maka fungsi tersebut dapat langsung digunakan untuk dicetak hasilnya, sebagai berikut :

```
write('x+y=',Tambah(x,y)
```

3. Atau nilai fungsi tersebut dapat juga langsung dipindahkan ke pengenal variabel yang lainnya, sebagai berikut :

```
z=Tambah(x,y);
printf("x+y = %d",z);  atau bisa langsung diketikkan
printf("x+y = %d", Tambah(x,y));
```

Sedang pada prosedur, nama prosedur tersebut tidak dapat digunakan langsung, yang dapat langsung digunakan adalah parameternya yang mengandung nilai balik.

STRUKTUR DATA

DISKUSI