SEVEN
PROCUDURE & FUNCTION

Mengapa Menggunakan Fungsi?

- Pemrograman yang baik harus bersifat modular agar suatu masalah program yang besar dan kompleks dapat dipecah-pecah menjadi bagian-bagian yang lebih kecil dan sederhana.
- Di dalam bahasa C modul-modul yang berisi bagian program yang bersifat spesifik dapat dituangkan ke dalam suatu fungsi.
- Fungsi/function adalah bagian dari program yang memiliki nama tertentu, digunakan untuk mengerjakan suatu pekerjaan tertentu, serta letaknya dipisahkan dari bagian program yang menggunakan fungsi tersebut.
- Keuntungan menggunakan fungsi:
    - Dapat melakukan pendekatan top-down dan divide-and-conquer: program besar dapat dipisah menjadi program-program kecil.
    - Dapat dikerjakan oleh beberapa orang sehingga koordinasi mudah.
    - Kemudahan dalam mencari kesalahan-kesalahan karena alur logika jelas dan kesalahan dapat dilokalisasi dalam suatu modul tertentu saja.
    - Modifikasi program dapat dilakukan pada suatu modul tertentu saja tanpa mengganggu program keseluruhan.
    - Mempermudah dokumentasi.
    - Reusability: Suatu fungsi dapat digunakan kembali oleh program atau fungsi lain
- Sifat-sifat modul/fungsi yang baik:
    - Nilai fan-in tinggi, artinya semakin sering suatu modul dipanggil oleh pengguna semakin tinggi nilai fan-in
    - Fan-out rendah, artinya semakin spesifik fungsi suatu modul akan semakin rendah nilai fan-out

- Self-contained tinggi: artinya kemampuan untuk memenuhi kebutuhannnya sendiri.

Kategori Function dalam C

1. **Standard Library Function**

   Yaitu fungsi-fungsi yang telah disediakan oleh C dalam file-file header atau librarynya.

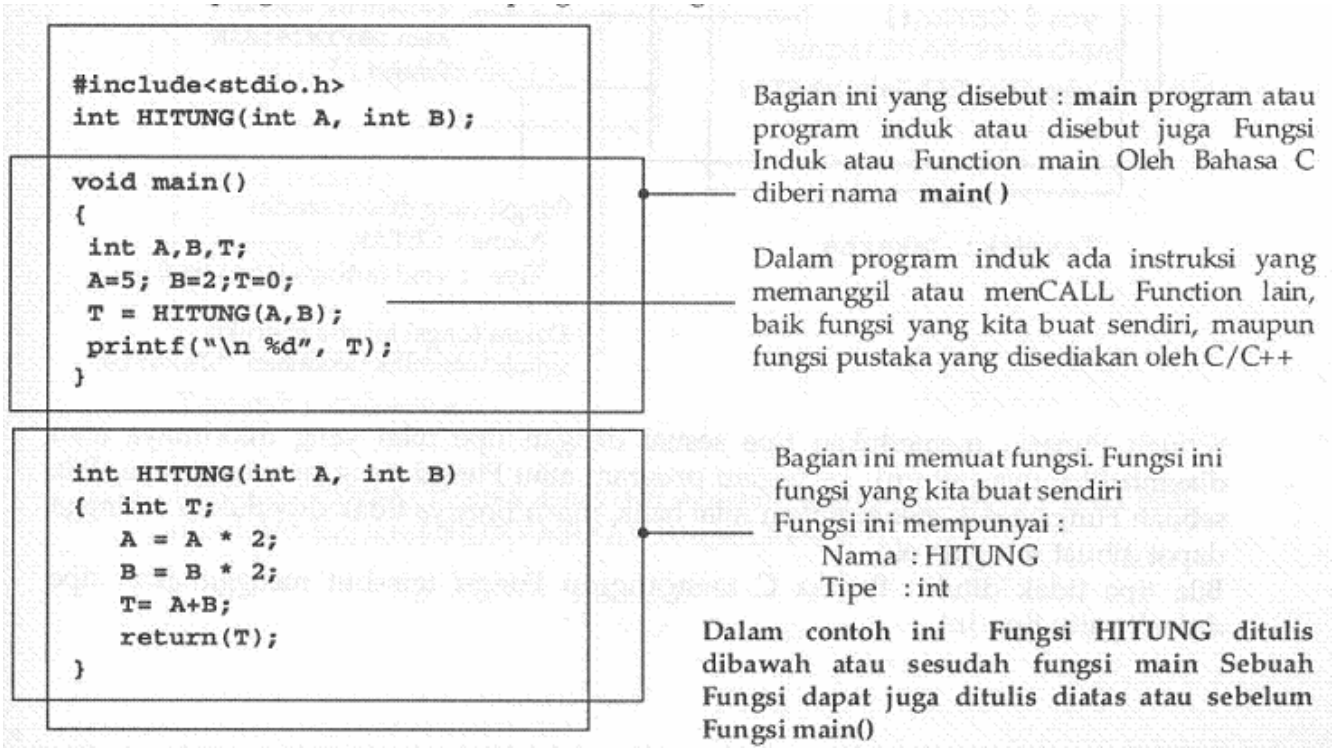
   Misalnya: `clrscr(), printf(), getch()`

   Untuk function ini kita harus mendeklarasikan terlebih dahulu library yang akan digunakan, yaitu dengan menggunakan preprosesor direktif: `#include <conio.h>`

2. Programmer-Defined Function

   Adalah function yang dibuat oleh programmer sendiri. Function ini memiliki nama tertentu yang unik dalam program, letaknya terpisah dari program utama, dan bisa dijadikan satu ke dalam suatu library buatan programmer itu sendiri yang kemudian juga di-includekan untuk penggunaanya.

Contoh program C yang menggunakan function



Contoh-02.

```
#include<stdio.h>
void main()
{
 printf("Jakarta");
}
```

Program ini **tidak** menggunakan Funtion lain selain main function

Tercetak : **Jakarta**

Struktur Function

- **Deklarasi function (function prototype/declaration)**: yang terdiri dari judul fungsi dan tipe data yang akan dikembalikan (dapat berupa tipe data tertentu atau bersifat void) tanpa adanya kode implementasi function tersebut.

  Bentuk umum function prototype:

  ```
  Tipe_data/void nama_fumgsi([arguman 1, argument 2,….])
  ```

  - Deklarasi fungsi tidak diakhiri dengan titik koma
  - Tipe_data dapat berupa segala tipe data yang dikenal C, namun tipe data dapat juga tidak ada dan digantikan dengan void yang berarti fungsi tersebut tidak mengembalikan nilai apapun
  - Nama fungsi adalah nama yang unik
  - Argumen dapat ada atau tidak (opsional) yang digunakan untuk menerima parameter-parameter dalam fungsi. Antar argumen-argumen dipisahkan dengan menggunakan tanda koma.

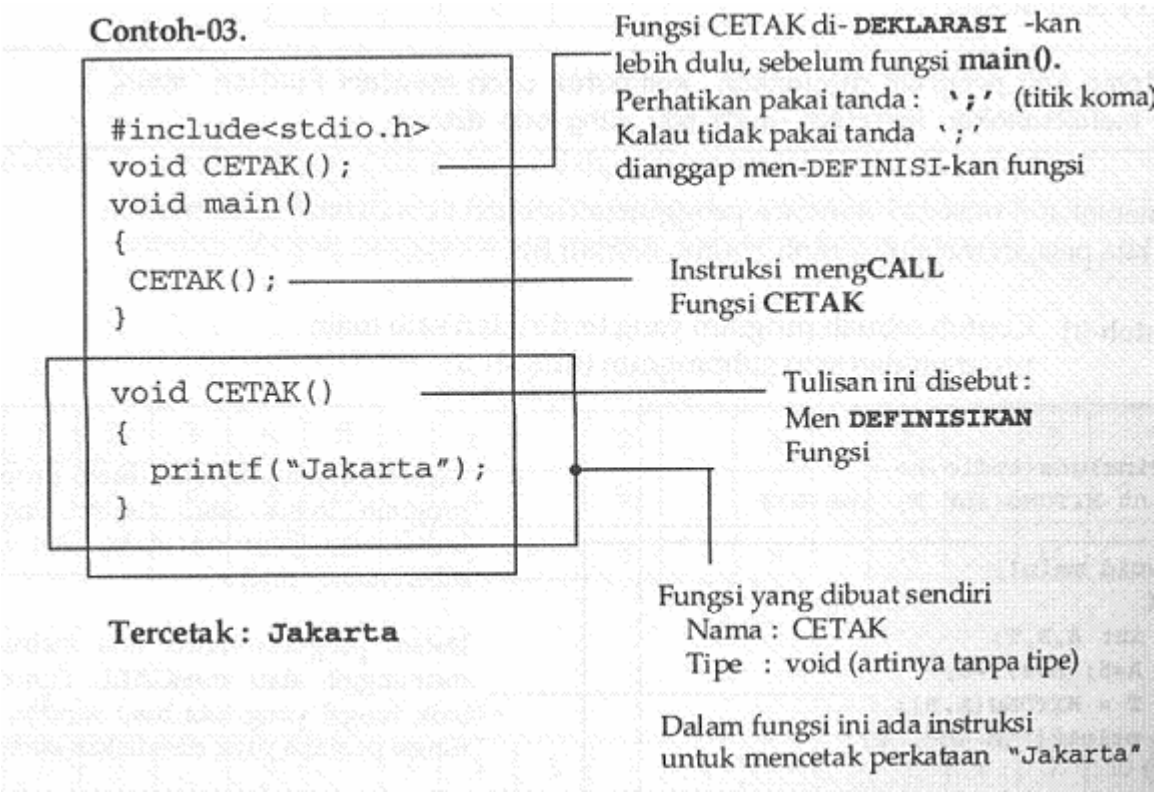
- **Tubuh Function/Definisi Function (Function Definition)**: yang terdiri dari function prototype yang disertai dengan kode implementasi dari function tersebut, yang berisikan statemen-statemen yang akan melakukan tugas yang diberikan oleh fungsi tersebut.

  Bentuk umum function definition:

```
Tipe_data/void nama_fumgsi([arguman 1, argument 2,….]) //funciton prototype
{
      //bagian ini merupakan tubuh fungsi.
      [Variabel_lokal;]

      [Statement_1;]
      [Statement_2;]
      ...
```

```
        [Statement_3;]
        [return (variabel)];
}
```

- Tubuh function dapat berisi segala perintah yang dikenal oleh C, pada dasarnya tubuh fungsi sama dengan membuat program seperti biasa.
- Return adalah keyword pengembalian nilai dari fungsi ke luar fungsi, return wajib jika fungsi tersebut mengembalikan nilai berupa tipe data tertentu, sedangkan return tidak wajib jika fungsi tersebut bersifat void.



FUNGSI DAN PROSEDUR

- Dalam PASCAL dikenal istilah procedure dan function, dalam Basic dikenal sub dan function, sedangkan dalam C, Java, PHP, dan keturunan C lainnya dikenal hanya istilah function.
- Procedure/Sub dalam Pascal/Basic adalah suatu kumpulan program yang mengerjakan suatu tugas spesifik tertentu yang tidak mengembalikan nilai kembalian ke luar procedure tersebut.
- Function dalam C adalah suatu kumpulan program yang mengerjakan suatu tugas spesifik tertentu yang bisa mengembalikan nilai atau tidak mengembalikan nilai (sama dengan procedure di Pascal/sub di Basic).

- Dalam C fungsi ada 2 jenis:
  - Fungsi yang tidak mengembalikan nilai (void)
  - Fungsi yang mengembalikan nilai (non-void)

Function yang Void

- Fungsi yang void sering disebut juga function
- Disebut void karena fungsi tersebut tidak mengembalikan suatu nilai keluaran yang didapat dari hasil proses fungsi tersebut.
- Ciri: tidak adanya keyword `return`.
- Ciri: tidak adanya tipe data di dalam deklarasi fungsi.
- Ciri: menggunakan keyword `void`.
- Tidak dapat langsung ditampilkan hasilnya
- Contoh: `clrscr(), printf()`

Contoh function void:

```
void tampilkan_jml(int a,int b){
      int jml;
      jml = a + b;
      printf("%d",jml);
}
```

Keyword `void` juga digunakan jika suatu function tidak mengandung suatu parameter apapun.

Contoh penggunaan parameter void:

```
void print_error(void){
      printf("Error : unexpected error occurred!");
}
```

Fungsi yang Non-Void

- Fungsi non-void disebut juga procedure (terutama di bahasa pemrograman seperti Pascal dan Basic)
- Disebut non-void karena mengembalikan nilai kembalian yang berasal dari keluaran hasil proses function tersebut
- Ciri: ada keyword `return`
- Ciri: ada tipe data yang mengawali deklarasi fungsi

- Ciri: tidak ada keyword `void`
- Dapat dianalogikan sebagai suatu variabel yang memiliki tipe data tertentu sehingga dapat langsung ditampilkan hasilnya.
- Contoh: `sin(), getch()`

Contoh function non-void

```
int jumlah(int a,int b){
      int jml;
      jml = a + b;
      return jml;
}
```

Deklarasi Function/Prototype Function

Deklarasi fungsi harus ditulis di atas sebelum `void main()` dan tidak beisi kode sama sekali, hanya berupa judul fungsi

Contoh:

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

double Absolut(double X);       /*deklarsi fungsi Absolut */

void main()
{
    float Nilai;
    Nilai = -123,45;
    printf("%7.2f nilai mutlaknya adalah %7.2f\n",Nilai,Absolut(Nilai));
    getch();
}

/*----- Fungsi untuk memutlakkan nilai negatip- -----*/
double Absolut(double X)         /* definisi fungsi */
{
     if(X<0) X = -X;
     return(X);
}
```

Jika program ini dijalankan, akan didapatkan hasil :

Pada contoh program ini, terlihat bahwa bagian program yang menggunakan fungsi `absolut()` adalah sebagai berikut:

```
printf(“%7,2f nilai mutlaknya adalah %7,2f\n”,Nilai,Absolut(Nilai));
```

- Jika bagian dari program yang menggunakan fungsi diletakkan sebelum definisi dari fungsi, maka deklarasi dari fungsi diperlukan.
- Akan tetapi jika bagian dari program yang menggunakan fungsi terletak setelah definisi dari fungsi, maka deklarasi dari fungsi dapat tidak dituliskan.

Contoh :

Pada contoh ini, fungsi `absolut()` didefinisikan terlebih dahulu sebelum fungsi ini digunakan, sehingga deklarasi dari fungsi tidak diperlukan lagi.

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

/*----- Fungsi untuk memutlakan nilai negatif -----*/
double Absolut(double X)      /* definisi fungsi  */
{
       if(X<0) X= -X;
       return(X);
}

int main()
{
       float Nilai;

       Nilai = -123,45;
       printf("%7.2f nilai mutlaknya adalah %7.2f\n", Nilai,Absolut(Nilai));
       getch();
}
```

Hasilnya:

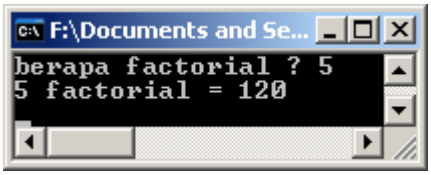
Walupun untuk contoh seperti ini deklarasi fungsi tidak diperlukan, tetapi untuk untuk praktek pembuatan program yang baik, **sebaiknya** deklarasi dari fungsi tetap dituliskan.

Contoh lain

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

int faktorial (int N);   /*prototype fungsi factorial*/

void main()
```

```
{
    int N;
    int fak;
    printf ("berapa factorial ? ");scanf("%d",&N);
    fak = faktorial(N);
    printf("%d factorial = %d\n",N,fak);
    getch();
}

/*-----fungsi untuk menghitung nilai N factorial -----*/
int faktorial(int N)  /*definisi fungsi*/
{
    int I;
    int F=1;
    if(N<=0)
    return(0);
    for(I=2;I<=N;I++) F *= I;
    return(F);
}
```

Hasilnya:



The main function

- Di dalam C, sebuah program yang paling sederhana AGAR DAPAT DIEKSEKUSI harus terdiri dari minimal 1 buah fungsi, yaitu function **`main()`**
- Tanpa function main, program C dapat dicompile tapi tidak dapat dieksekusi (harus dengan flag parameter –c, jika di UNIX)
- Pada saat program C dijalankan, maka compiler C akan mencari function main() dan melaksanakan instruksi-instruksi yang ada di sana.
- Di dalam function main, sering dideklarasikan dalam 2 bentuk:
  - **`int main()`**
  - **`void main()`**

int main()

- int main() berarti di dalam function main tersebut harus terdapat keyword return di bagian akhir fungsi dan mengembalikan nilai bertipe data int,
- Mengapa hasil return harus bertipe int juga? karena tipe data yang mendahului fungsi main() diatas dideklarasikan int

- Jika sebuah program C dieksekusi, maka akan dikembalikan status eksekusi program, jika “terminated successfully” maka, akan dikembalikan status 0, sedangkan jika “terminated unsuccessfully” akan dikembalikan nilai status tidak 0

void main()

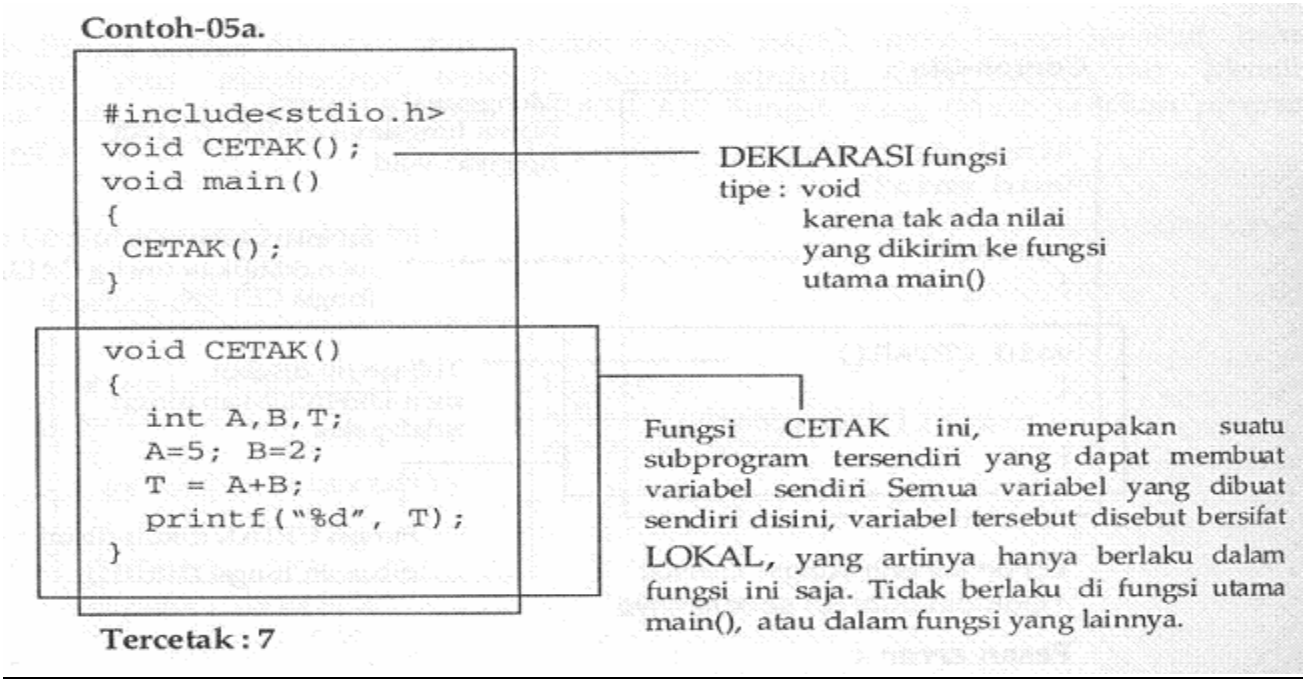
- void main() berarti function yang void dan tidak mengembalikan nilai status program tidak bisa diketahui

SCOPE VARIABLE

- Sebuah variabel di dalam sebuah program memiliki skop jangkauan variabel tertentu.
- Skop variabel terdiri dari:
  - Variabel lokal
  - Variabel global

Variabel lokal

- Variabel yang hanya dikenal di daerah yang lokal saja, misalnya di dalam sebuah fungsi/prosedur tertentu saja dan tidak dikenal di daerah lainnya.
- Harus dideklarasikan di dalam blok yang bersangkutan
- Variabel lokal akan dihapus dari memori bila proses sudah meninggalkan blok statemen letak variabel lokalnya.



Contoh:

```
#include <stdio.h>
void CETAK();

void main(){
      int A,B,T;
      A=5; B=2;
      T=A+B;
      CETAK();
}

void CETAK(){
      printf(“%d”,T);   //terjadi error, T tidak dikenal
}
```

Contoh:

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

int TAMBAH(int A,int B);

int main(){
      int hasil;
      hasil = TAMBAH(2,3);
      printf("Hasil = %d",hasil);
      getch();
}

int TAMBAH(int A,int B){
      int C;
      C = A + B;
      {
            float C;
            C = 100;
      }
      return(C);
}
```

Hasilnya: **5**

Mengapa tidak bernilai 100? Hal ini karena variabel C di deklarasikan di dalam blok sendiri sehingga dianggap berbeda dengan variabel C yang berisi nilai 5

Variabel global

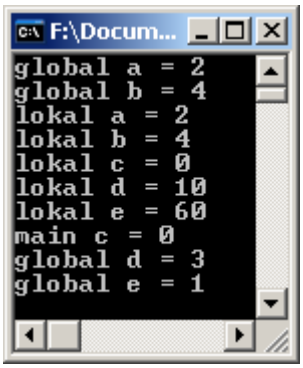
- Variabel yang dikenal diseluruh daerah di dalam program, di dalam dan luar fungsi.
- Dideklarasikan di luar suatu blok statemen atau di luar fungsi-fungsi yang menggunakannya.

Contoh skop variabel 1:

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

```
int d=3,e=1;

void coba_lokal(int a,int b){
      int c = 0;
      int d = 10;
      int e;
      e = (a+b) * (c+d);
      printf("lokal a = %d\n",a);
      printf("lokal b = %d\n",b);
      printf("lokal c = %d\n",c);
      printf("lokal d = %d\n",d);
      printf("lokal e = %d\n",e);
}

void main(){
      int a=2;
      int b;
      b = 4;
      int c=0;

      printf("global a = %d\n",a);
      printf("global b = %d\n",b);
      coba_lokal(a,b);
      printf("main c = %d\n",c);
      printf("global d = %d\n",d);
      printf("global e = %d\n",e);
      getch();
}
```

Hasilnya:



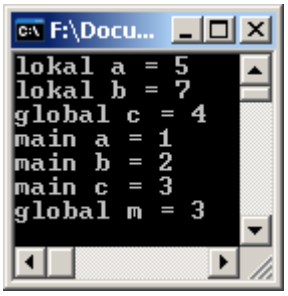
Jika dalam sebuah fungsi terdapat variabel a dan di dalam program utama juga terdapat variabel a juga (nama sama), maka variabel yang dipakai tergantung dari skop pengaksesnya.

Jika yang mengakses adalah fungsi, maka variabel yang dipakai adalah variabel lokal, jika yang mengakses adalah program utama, maka yang dipakai adalah variabel dalam program utama.

Contoh skop variabel 2:

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

int c = 4;
int m = 3;

void lokal(){
      int a = 5;
      int b = a + 2;
      printf("lokal a = %d\n",a);
      printf("lokal b = %d\n",b);
      //karena tidak ada c, maka ambil global
      printf("global c = %d\n",c);
}

void main(){
      int a = 1;
      int b = 2;
      int c = 3;
      lokal();
      printf("main a = %d\n",a);
      printf("main b = %d\n",b);
      //walaupun global c ada tapi c yang digunakan yang di main
      printf("main c = %d\n",c);
      //karena tidak ada m, maka ambil global
      printf("global m = %d\n",m);
      getch();
}
```

Hasilnya:



Variabel global akan membuat sebuah fungsi menjadi tidak berfungsi sebagaimana mestinya, kurang robust, dan sulit untuk digunakan kembali.

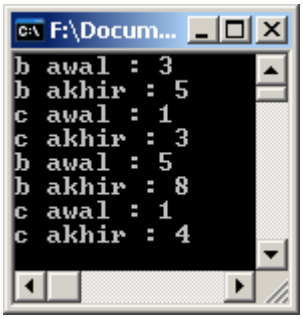
Variabel Static

Adalah variabel yang memiliki sifat statis, artinya nilai dari variabel tersebut akan tetap diingat oleh program, sehingga dapat digunakan untuk menyimpan state pada saat pemanggilan nilai variabel tersebut berikutnya. Nilai variabel statis akan bernilai sama dengan nilai terakhirnya.

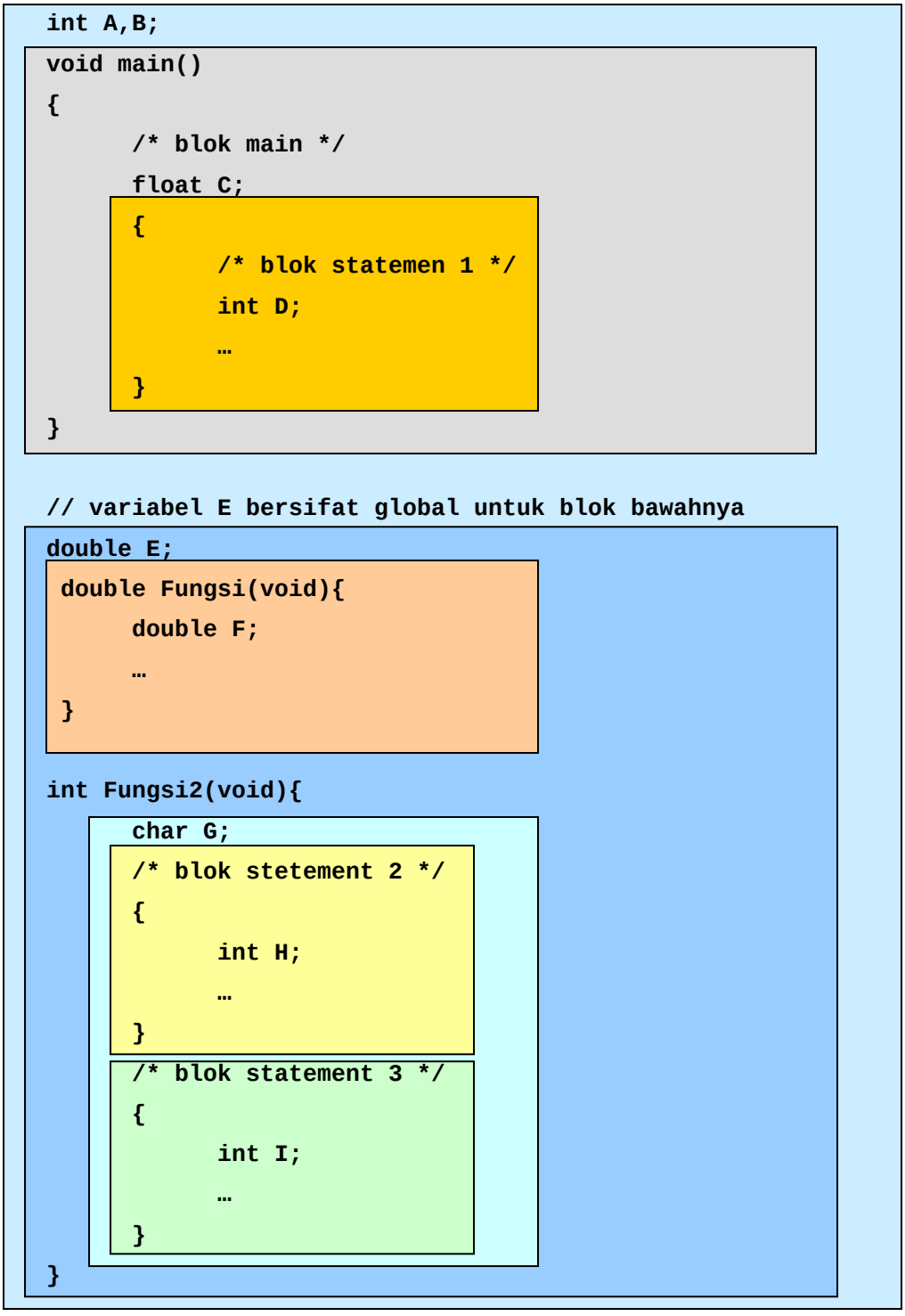
Contoh variable static:

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

void coba_static(int a){
     static int b=3;
     int c=1;

     printf("b awal : %d\n",b);
     b += a;
     printf("b akhir : %d\n",b);

     printf("c awal : %d\n",c);
     c += a;
     printf("c akhir : %d\n",c);
}

void main(){
    int a=2;
    coba_static(a);
    a=3;
    coba_static(a);
    getch();
}
```

Hasilnya:



Bagan Skop Variabel

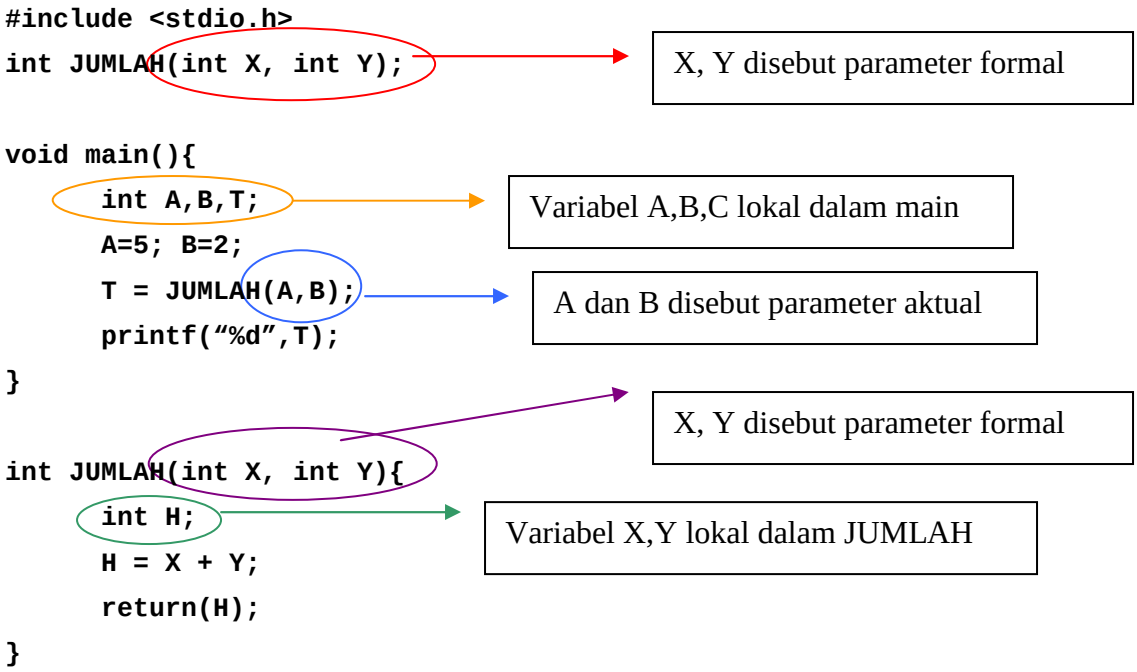


ARGUMEN/PARAMETER FUNCTION

- Sebuah fungsi bisa memiliki argumen-argumen yang bersifat opsional.
- Argumen-argumen tersebut berfungsi sebagai parameter inputan yang berupa variabel-variabel bagi fungsi tersebut (bersifat lokal).

- Argumen harus bertipe data tertentu.
- Terdapat 2 jenis parameter:
  - Parameter formal: parameter yang ditulis pada deklarasi fungsi.
  - Parameter aktual: parameter yang diinputkan dalam program pemanggil fungsi tersebut. Dapat berupa variabel atau langsung berupa nilai tertentu sesuai dengan tipe data yang dideklarasikan untuk masing-masing parameter fungsi.

Contoh:



Hal-hal penting lainnya

- Nilai variabel di dalam fungsi tidak dapat mengubah nilai dalam function main
- Sebuah function dapat meng-call function lain
- Dua atau lebih function dapat saling call
- Secara defautl sifat pemanggilan fungsi dalam C bersifat calling by value, artinya:
  - Yang dikirimkan ke fungsi adalah datanya, bukan alamat memorinya
  - Fungsi menerima data ini dan akan menyimpannya ke ke dalam alamat memori yang berbeda dari alamat asli datanya.
  - Karena itulah pengubahan nilai di fungsi tidak akan mengakibatkan perubahan pada nilai aslinya
  - Pengiriman by value adalah pengiriman searah: dari program yang memanggil fungsi ke fungsi yang dipanggil

- Pengiriman dapat dilakukan dengan suatu statement, tidak hanya untuk variabel atau nilai saja.

NEXT WEEK: Tipe Data Bentukan (Abstract Data Type)