FUNGSI

LEARNING OUTCOMES

Pada akhir pertemuan ini, diharapkan mahasiswa

akan mampu :

- Mendemonstrasikan penggunaan fungsi serta pengiriman parameter

OUTLINE MATERI

- Pemrograman Modular
- Library Function vs user-defined function
- Prototipe fungsi
- Jangkauan identifier
- Pengiriman parameter
- Iterasi vs rekursif

PEMROGRAMAN MODULAR

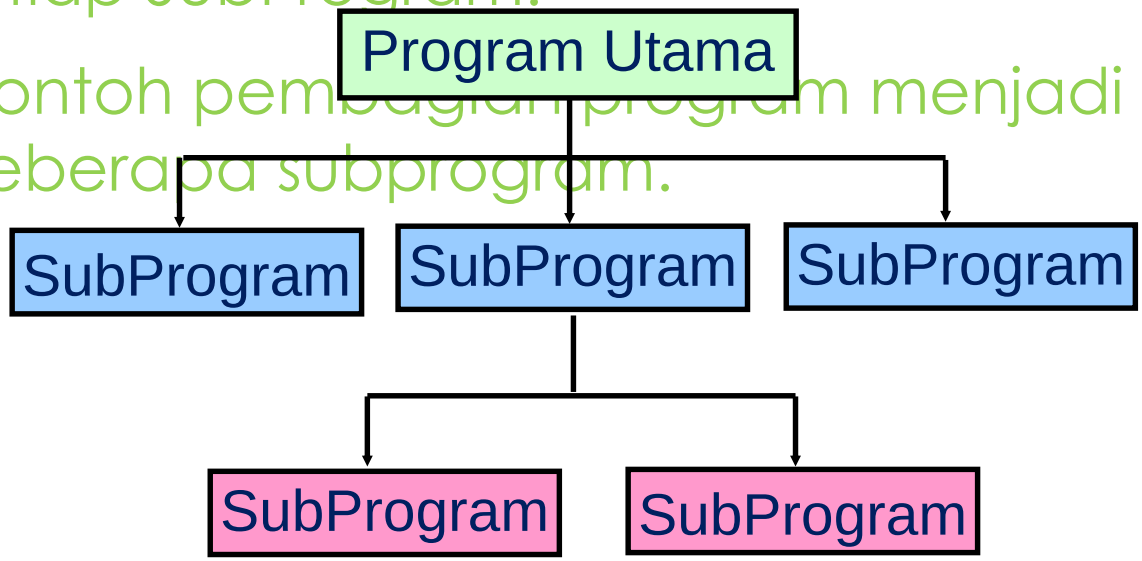
- Program dibagi-bagi menjadi Modul-Modul
- Modul dalam bahasa C di-implementasikan dengan Fungsi
- Fungsi dibentuk dengan mengelompokkan sejumlah perintah untuk menyelesaikan tugas tertentu.
- Modul diperlukan jika kelompok perintah tersebut kerap kali digunakan di tempat lain dalam program
- Modul sering disebut juga dengan Sub-Program

PEMROGRAMAN MODULAR

- **Keuntungan menggunakan modul :**
  1. Rancangan Top - down dengan teknik Sub goal, program besar dapat dibagi menjadi modul-modul yang lebih kecil.
  2. Dapat dikerjakan oleh lebih dari satu orang dengan koordinasi yang relatif mudah.
  3. Mencari kesalahan relatif lebih mudah karena alur logika lebih jelas, dan kesalahan dapat dilokalisir dalam satu modul.
  4. Modifikasi dapat dilakukan, tanpa menggangu program secara keseluruhan
  5. Mempermudah dokumentasi

PEMROGRAMAN MODULAR

- Bahasa C melengkapi fasilitas modular dengan menggunakan ***fungsi*** pada setiap SubProgram.
- Contoh pembagian program menjadi beberapa subprogram.



PEMROGRAMAN MODULAR

- Sifat-sifat modul yang baik adalah:
  - **Fan-In** yang tinggi, yaitu makin sering suatu modul dipanggil oleh pengguna, makin tinggi nilai fan-in.
  - **Fan-Out** yang rendah, makin sedikit tugas yang dilakukan oleh suatu modul makin rendah nilai fan-out. Dengan demikian, makin spesifik tugas yang dikerjakan oleh modul tersebut.
  - **Self-Contained**, atau memenuhi kebutu-hannya sendiri.

LIBRARY VS USER-DEFINED FUNCTION

- Fungsi dalama bahasa C terbagi dalam dua jenis :
  - Library function
  - User-defined function

- Library function, adalah fungsi-fungsi standard yang sudah disediakan oleh bahasa C. Fungsi-fungsi tersebut dideklarasikan dalam file header (.h), contohnya clrscr() ada di file conio.h, sqrt() dalam math.h, printf() dalam stdio.h

- User-define function, adalah fungsi yang didefinisikan sendiri oleh pemrogram.

KONSTRUKSI FUNGSI

- Konstruksi fungsi

```
return-value-type  function-name( parameter-list )
{
    statements;
}
```

- return-value-type: tipe data yang dikembalikan oleh fungsi
- Jika tidak diisi maka dianggap tipenya integer (default int)
- Jika return-value-type diganti void maka fungsi tidak mengembalikan nilai
- Parameter-list: berisi daftar nilai yang dikirimkan dari fungsi pemanggil

Konstruksi Fungsi

formal parameter

Fungsi

```
int maksimum (int x, int y){
        int maks = x;
        if ( y > maks) maks = y;
        return maks;
}
```

Pemanggil

```
void main () {
      int a,b;
      cou<<"Input 2 bilangan bulat : ";
      cin>>a;
      cin>>b;
      cout<<"Bilangan yg lebih besar “<<maksimum(a,b);
}
```

Actual parameter

PROTOTIPE FUNGSI

- Penulisan fungsi pada bahasa C pada dasarnya diletakkan diatas pemanggil (blok *main,* atau blok fungsi lainnya*)*. Namun adakalanya blok fungsi diletakkan setelah blok pemanggil. Pada kondisi tersebut perlu digunakan prototipe fungsi.
- Tujuan dari prototipe fungsi :
  - Meyakinkan sebuah fungsi dikenal oleh pemanggilnya
  - *Compiler* akan memvalidasi parameter
- Sintaks

  return-value-type function-name(parameter-list);

PROTOTIPE FUNGSI

Contoh:

```
#include<iostream.h>

int maksimum (int x, int y) {
      int maks = x;
      if ( y > maks)  maks = y;
      return maks
}

void main () {
     int a,b;
     cout"Input 2 bilangan bulat : “;
     cin>>a>>b;
     cout<<“Bilangan yg lebih besar:”<<maksimum(a,b));
}
```

Karena fungsi maksimum diletakkan di atas pemanggilnya (main program), maka tidak perlu prototipe fungsi

PROTOTIPE FUNGSI

- Contoh:

```
#include<stdio.h>

int maksimum(int, int);

void main () {
      int a,b;
     cout"Input 2 bilangan bulat : “;
     cin>>a>>b;
     cout<<“Bilangan yg lebih besar:”<<maksimum(a,b));
}

int maksimum (int x, int y){
      int maks = x;
      if ( y > maks)  maks = y;
      return maks
}
```

Prototipe Fungsi

Karena fungsi maksimum diletakkan di bawah pemanggilnya (main), maka perlu diletakkan prototipe fungsi diatas,supaya dikenal oleh pemanggilnya

PROTOTIPE FUNGSI

- Penulisan Prototipe Fungsi seperti di atas bisa ditambah nama parameternya sbb :

```
int maksimum(int a, int b);
```

- Yang dipentingkan dalam prototipe fungsi adalah tipe parameter, jumlah parameter dan urutannya.

Lingkup Identifier

- Lingkup identifier meliputi bagian-bagian program dimana sebuah identifier masih bisa diakses.
- Lingkup identifier meliputi :
  - Local
  - Global
- Local identifier
  - Identifier yang dideklarasikan di dalam fungsi, termasuk daftar parameter.
  - Lingkupnya terbatas pada fungsi tempat dideklarasikan.

Lingkup Identifier

- Global declaration
  - Identifier yang dideklarasikan di luar fungsi.
  - Ruang lingkupnya meliputi seluruh program.
  - Identifier global, dapat digunakan untuk identifier lokal.
  - Disarankan tidak banyak menggunakan identifier global karena:
    - Jika program semakin besar, kecenderungan error semakin besar .
    - Sulit melacak bila terjadi kesalahan.
    - Data tidak terjaga dengan baik, setiap fungsi dapat mengubah nilai variabel tanpa sepengetahuan fungsi lainnya.

Jangkauan Identifier

```
int x;
fungsi1()
{
        -
        -
}
```

```
int y;
fungsi2(){
        int z;
        -
}
```

```
main(){
   int z, y;
   -
}
```

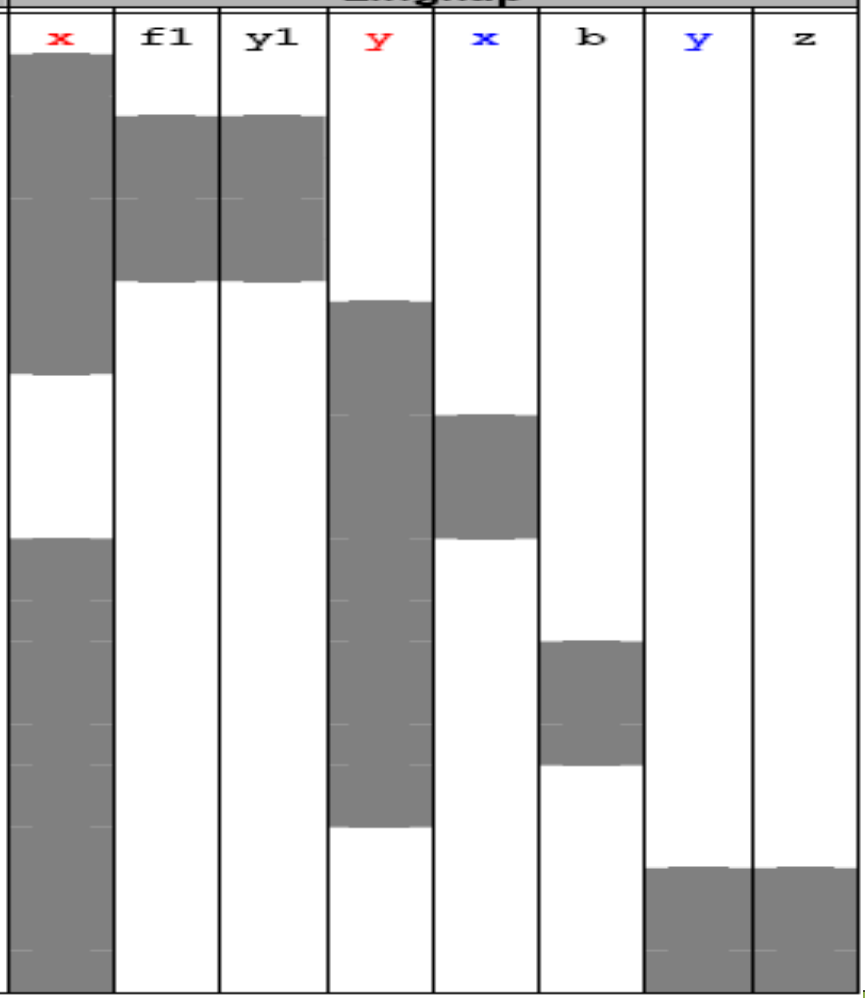
scope dari variabel x

scope dari variabel y

z dan y hanya dikenal oleh main()
z di main() berbeda dgn di fungsi2()
y di main() berbeda dgn di fungsi2()

Program	Lingkup

```
# include <stdio.h>
int x;

float funct_1(int f1){
   int y1;
   ...
}

char y;

int funct_2( ) {
   int x;
   ...
}

char funct_3( ) {
   ...
   { int b;
      ...
   }
}

void main() {
   char y; int z;
   ...
}
```



Parameter Fungsi

- Pengiriman nilai data antar fungsi dapat dilakukan melalui penggunaan parameter fungsi.
- Parameter merupakan 'interface' antara suatu fungsi dengan fungsi lain.
- Pengiriman nilai data melalui parameter dapat berupa:
  - **By-Value**
    Yang dikirim ke fungsi lain adalah nilai datanya.
  - **By Location / by reference**
    Yang ditransfer ke fungsi lain adalah alamat memorinya.

Pengiriman Parameter

- Contoh: Pengiriman parameter by value

```
#include <stdio.h>

void Garis (char x ) {                    // x sbg Parameter Formal
{   int i;                                // i, x adalah Local Variabel
    for (i = 1; i<=10; i++) printf("%c",x);
}

/*Program Utama*/
void main()
{   char A = '-';
    Garis(A);                             // A disebut Parameter Aktual
}
```

Pengiriman Parameter

- Contoh Pengiriman parameter by reference

```
#include <stdio.h>
void Hitung (int X, int Y, int *P, int *Q)
{     *P = X + Y;
      *Q = X * Y;
}

void main()
 {    int X, Y, P, Q;       /*local variabel*/
       printf(“ X=”); scanf(“%d”,&X);
       printf(“ Y=”); scanf(“%d”,&Y);
       Hitung(X,Y,&P,&Q);
       printf(”X + Y = %d\n”, P);
       printf(”X * Y = %d\n”, Q);
}
```

Array Dimensi-1 Sebagai Parameter

- Jika array digunakan sebagai parameter dalam suatu fungsi, maka passing parameter harus by location.
- Contoh:

```
#include <stdio.h>
void cetak_array(int index, int *A) {
        printf("A[%d]=%d\n",index, A[index]);
}

void main() {
        int A[ ]={1,6,2,8,12};
        cetak_array(2, A);
}
```

- Contoh diatas: A pada fungsi main adalah pointer constant, sedangkan A pada fungsi cetak_array adalah pointer variable.

Array Dimensi-2 sbg Parameter

- Deklarasi fungsinya dapat berupa:

```
void isimatriks(int a[10][10], int b, int k)
```

atau

```
void isimatriks(int a[][10], int b, int k)
```

- tetapi ***TIDAK*** bisa berupa:

```
void isimatriks(int a[10][], int b, int k)
```

atau

```
void isimatriks(int a[][], int b, int k)
```

Array Dimensi-2 sbg Parameter

- Contoh:

```
#include <iostream.h>

void cetak(int A[3][4])
{     int row,col;
      for(row=0; row<3; row++){
          for(col=0; col<4; col++)
            cout<<"X["<<row<<"]"<<
printf("X[%d][%d]=%d",row,col,A[row][col]);
          printf("\n");
      }
}

int main()
{   int x[3][4] = {{1,2,3,4},  {8,7,6,5},  {9,10,11,12}};
    cetak(x);
    return(0);
```

Pengiriman Parameter

```
int main()
{   char ss[20]="KASUR";
    balik(ss);
    printf("%s\n",ss);
    getch();
    return(0);
}
```

Untuk string pada formal parameter bisa :

char[ ] atau char *

```
void balik( char ss[ ] )
{   int c,i,j;
    for(i=0, j=strlen(ss)-1; i<j; i++, j--){
        c=ss[i];
        ss[i]=ss[j];
        ss[j]=c;
    }
}
```

```
void balik( char *ss )
{   int c,i,j;
    for(i=0, j=strlen(ss)-1; i<j; i++, j--){
        c=ss[i];
        ss[i]=ss[j];
        ss[j]=c;
    }
}
```

Fungsi Rekursif

- Fungsi rekursif adalah fungsi yang di dalamnya terdapat statement yang memanggil dirinya sendiri.
- Fungsi rekursif, sangat berguna dalam pemecahan masalah yang dapat didefinisikan secara rekursif pula.
- Contoh :

  Faktorial (n) atau n! didefinisikan sebagai berikut :

  n! = 1, untuk n = 0;

  *n! = n * (n-1)!, untuk n > 0*

  4! = 4 * 3!

  3! = 3 * 2!

  2! = 2 * 1!

  1! = 1* 0!

  0! = 1

  Bila ditelusur mundur : 4! = 1*2*3*4 = 24

Fungsi Rekursif

- Contoh sederhana fungsi faktorial

5!
(5 * 4!)
(5 * (4 *3!))
(5 * (4 * (3 * 2!)))
(5 * (4 * (3 * (2 * 1!))))
(5 * (4 * (3 * (2 * (1 * 0!)))))
(5 * (4 * (3 * (2 * (1 * 1)))))
(5 * (4 * (3 * (2 * 1))))
(5 * (4 * (3 * 2)))
(5 * (4 * 6))
(5 * 24)
120

Fungsi Rekursif

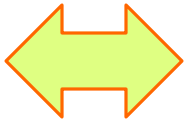
- Fungsi rekursif mempunyai dua komponen yaitu:
  - **Base case**: mengembalikan nilai tanpa melakukan pemanggilan rekursi berikutnya.
  - **Reduction step**: menghubungkan fungsi di suatu nilai input ke fungsi yang dievaluasi di nilai input yang lain. Sekuen nilai input harus konvergen ke base case.
- Fungsi faktorial
  - Base case: n = 0
  - Reduction step: f(n) = n * f(n-1)

Fungsi Rekursif vs Iterasi

- Contoh

- ***Faktorial - Rekursif***

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



- ***Faktorial - Iteratif***

```
long faktor(int n)
{   long i, fak = 1;
    for(i=1; i<=n; i++)  fak *= i;
    return (fak);
}
```

Kekurangan Rekursif

- Meskipun penulisan program dengan cara rekursif bisa lebih pendek, namun function rekursif memerlukan :
  - Memori yang lebih banyak, karena perlu tambahan untuk mengaktifkan Stack.
  - Waktu lebih lama, karena perlu menjejaki setiap pemanggilan rekursif melalui stack.

Kapan Menggunakan Rekursif?

- Secara umum gunakan penyelesaian secara rekursif, hanya jika :
  - Penyelesaian sulit dilaksanakan secara iteratif
  - Efisiensi dengan cara rekursif sudah memadai
  - Efisiensi bukan masalah dibandingkan dengan kejelasan logika program
  - Tidak mempertimbangkan faktor penghematan memori dan kecepatan eksekusi program
- Pertimbangan antara aspek kecepatan dan penghematan memori menggunakan iteratif, dibanding perancangan logika yang baik menggunakan rekursif

Bilangan Fibonacci

- Urutan bilangan 0, 1, 1, 2, 3, 5, 8, 13 … disebut bilangan Fibonacci. Hubungan antara satu angka dengan angka berikutnya didefinisikan secara rekursif sebagai berikut :
  - Fib(N) = N jika N = 0 atau 1
  - Fib(N) = Fib(N-2) + Fib(N-1) jika N >= 2

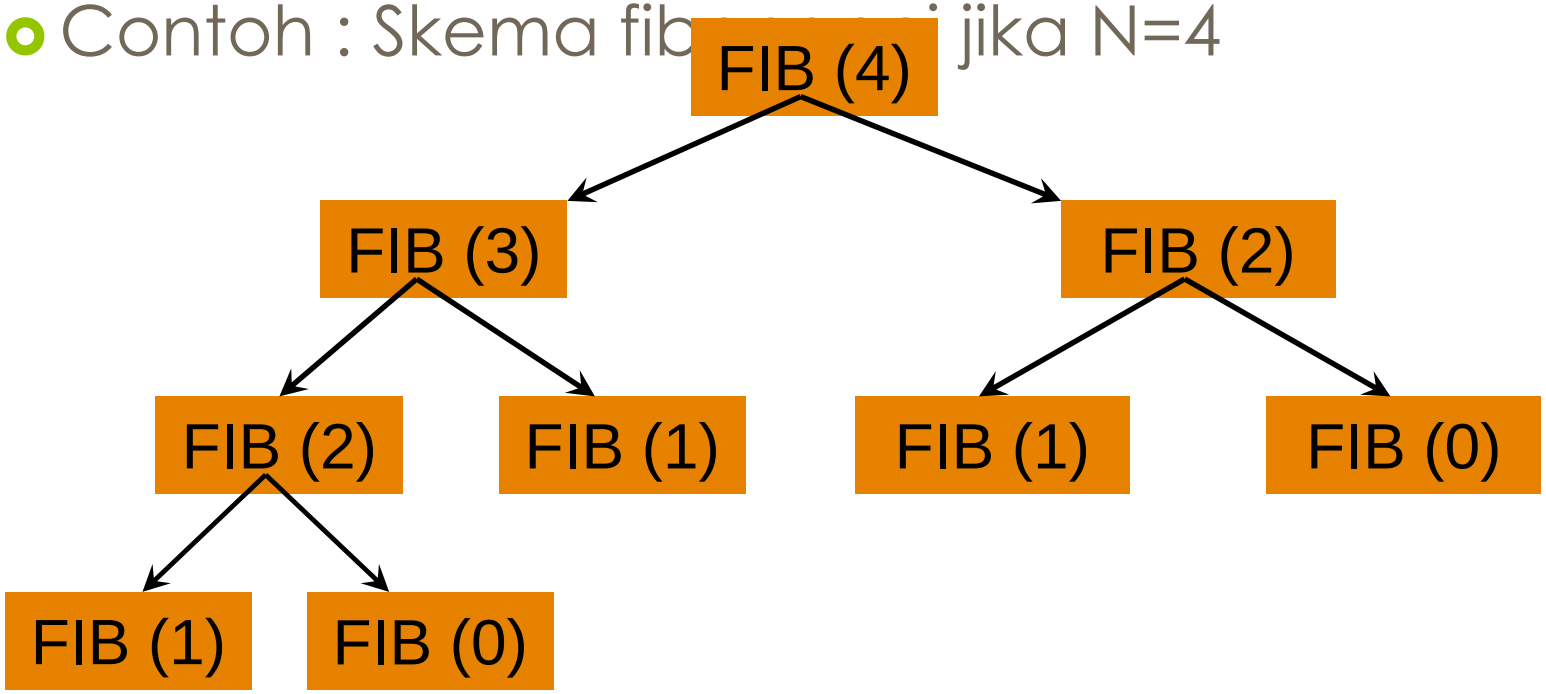
Bilangan Fibonacci

```
int Fib(int n) {
   int f;
   if(n==0) f = 0;
      else if(n==1) f = 1;
         else f = Fib(n-2) + Fib(n-1);
   return f;
}
```

Fungsi fib() di-samping ditulis secara rekursi dan disebut sebagai slow_Fib() tulislah fast_Fib() menggunakan iterasi.

Bilangan Fibonacci

- Contoh : Skema fib[illegible] jika N=4



Function Parameter Declaration

- Classic Function Parameter declaration:
- Contoh:

```
int fungsi1(a)
int a;
{ a++;
  return a;
}

int fungsi2(b)
int b;
{ b = b * b;
  return b;
}
```

```
#include <stdio.h>

int main()
{ int x;
  x=fungsi1(3);
  printf("x=%d\n",x);
  x=fungsi2(13);
  printf("x=%d\n",x);
  return(0);
}
```

Function Parameter Declaration

- Modern Function Parameter declaration
- Contoh:

```
int fungsi1(int a)
{ a++;
  return a;
}

int fungsi2(int b)
{ b = b * b;
  return b;
}
```

```
#include <stdio.h>

int main()
{ int x;
  x=fungsi1(3);
  printf("x=%d\n",x);
  x=fungsi2(13);
  printf("x=%d\n",x);
  return(0);
}
```

Latihan

- Buatlah program dengan fungsi sbb:
  - Fungsi untuk meng-input 10 bilangan ke dalam array
  - Fungsi untuk mencari bilangan terbesar dalam array tersebut
  - Fungsi untuk mencari bilangan terkecil dalam array tersebut
  - Fungsi untuk menampilkan :
    - 10 bilangan tersebut
    - Bilangan terbesar dan terkecil

Latihan

- Perbaiki program berikut sehingga bisa digunakan untuk menukar 2 buah karakter

```
void Tukar(char A, char B )
{      char C ;
       C = A;   A = B,   B = C;
}

void main()
{     char X, Y ;
      X = 'S';    Y = 'D';
      Tukar(X, Y);
      printf("X = %c  Y= %c", X, Y);
}
```

Latihan

```
#include <stdio.h>

void bagi(float x, int y, float *z)
{   if(y==0) return;
     *z=x/y;
}

float div(float x, int y)
{   if(y!=0) return(x/y);
}

void main()
{   float f,a=12.75;    int b=5;
    bagi(a,b,&f);
    printf("%f dibagi %d = %f\n",a,b,f);
    b=3;
    f=div(a,b);
    printf("%f dibagi %d = %f\n",a,b,f);
}
```

Fungsi tidak mengembalikan nilai

Fungsi yang mengembalikan nilai

Jelaskan apa perbedaan keyword return yang ada pada fungsi bagi dengan return yang ada pada fungsi div ?

Latihan

```
#include <stdio.h>

void bagi(float x, int y, float *z)
{   if(y==0) return;
     *z=x/y;
}

float div(float x, int y)
{    if(y!=0) return(x/y);
}

void main()
{   float f,a=12.75;    int b=5;
    bagi(a,b,&f);
    printf("%f dibagi %d = %f\n",a,b,f);
    b=3;
    f=div(a,b);
    printf("%f dibagi %d = %f\n",a,b,f);
}
```

1. Bolehkah pada fungsi bagi tidak menggunakan keyword return, jika boleh silahkan programnya dirubah ?
2. Bolehkan pada fungsi div tidak menggunakan keyword return ?

Latihan

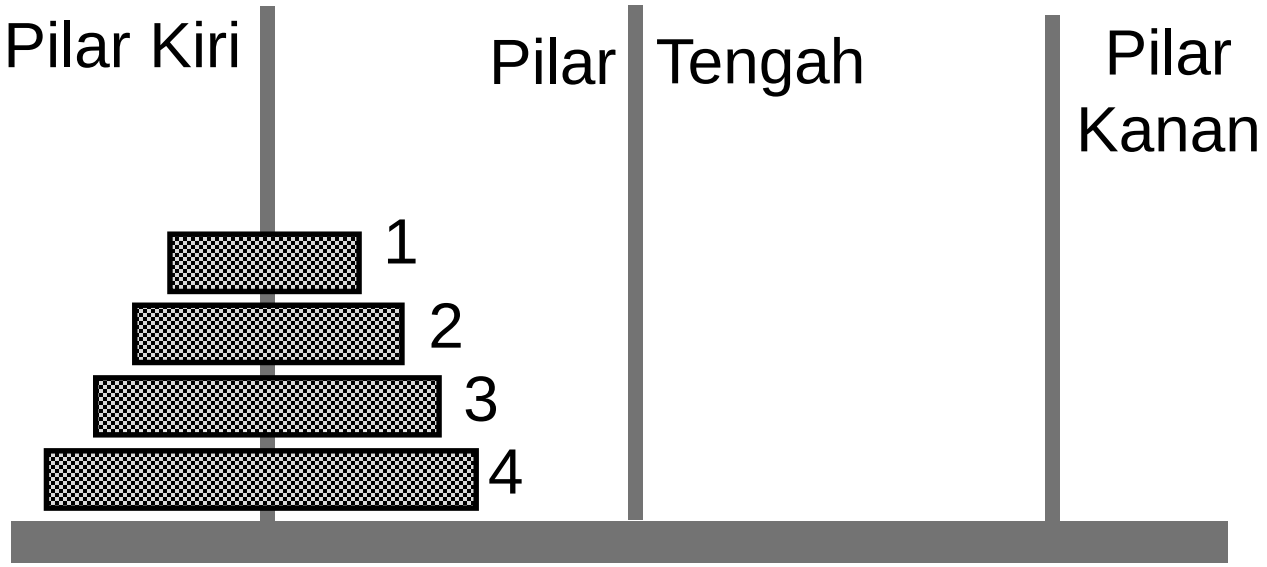
```
#include <stdio.h>
int main()
{   int x,y;
    for(x=1; x<=3; x++)
    {   int x=5;
        printf("x=%d ",x++);
        for(y=0; y<x; y++)
        { int x=20;
          printf("x=%d ",x++);
        }
        printf("\n");
    }
    return 0;
}
```

Perhatikan Lingkup variabel x pada program disamping.

Apa output dari program disamping ?

Latihan

- Menara Hanoi



Latihan

- Pindahkan n-piringan dari pilar-kiri ke pilar-kanan dengan pilar-tengah sebagai antara. Piringan yang berada dipilar kiri tersusun sedemikian rupa sehingga menyerupai menara, yaitu piringan yang lebih kecil selalu berada diatas piringan yang lebih besar. Pada proses pemindahan piringan-piringan tersebut, pola susunan menara harus selalu dijaga.
- Alur pemecahan secara rekursif :
  1. Pindahkan (n-1) piringan-piringan atas ke pilar antara.
  2. Pindahkan piringan terakhir ke pilar tujuan.
  3. Ulangi 2 dan 3, hingga selesai.

Latihan

- Simulasikan pemindahan dengan :
  - 3 piringan
  - 4 piringan
  - 5 piringan
- Buat programnya secara rekursif