POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

DIKTISAINTEK BERDAMPAK KAMPUS INOVASI



FUNGSI 2

Konsep Pemrograman

Yunia Ikawati

Referensi: Umi Sa'adah, Entin Martiana Kusumaningtyas, Tri Hadiah Muliawati

Teknik Informatika-PENS

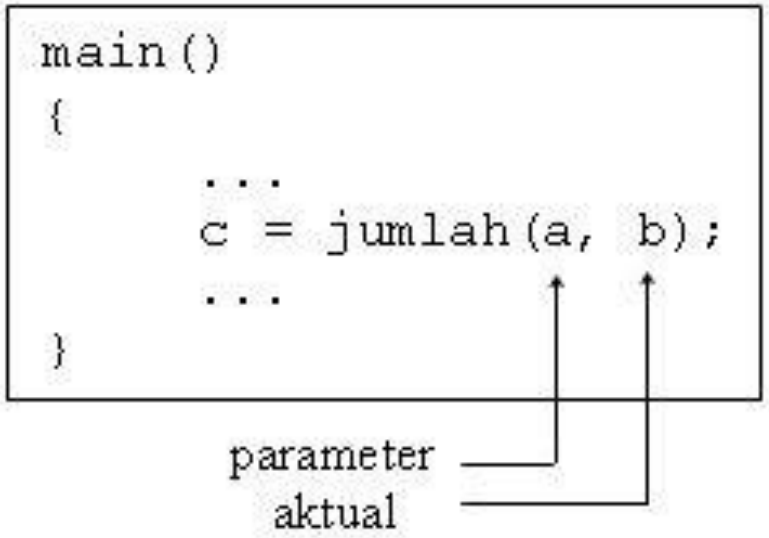
Overview

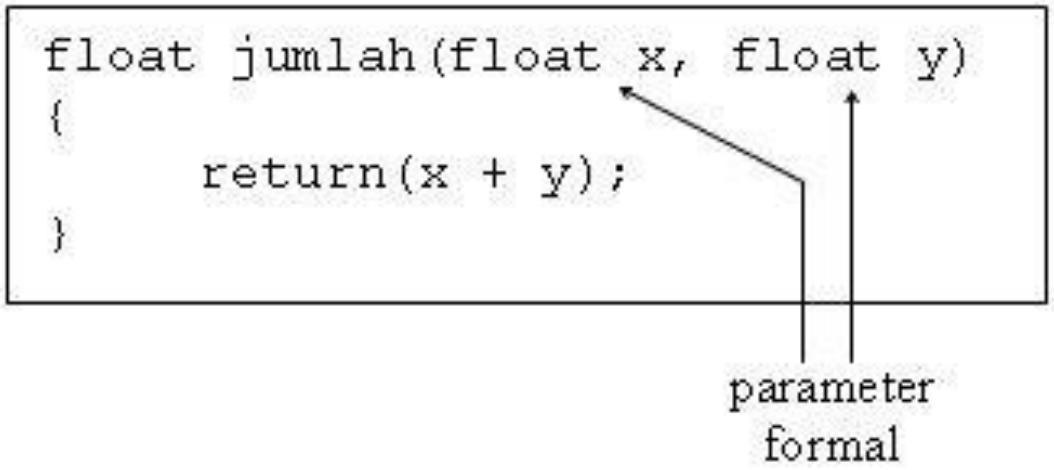
- Parameter Aktual dan Parameter Formal
- Pengiriman parameter secara *pass by value*
- Penggolongan Variabel berdasarkan Kelas Penyimpanan
  - Variabel lokal/auto
  - Variabel global/eksternal
  - Variabel statis
  - Variabel register
- Pengenalan Konsep Pemrograman Terstruktur

Parameter Formal dan Parameter Aktual

- Parameter formal adalah variabel yang ada pada daftar parameter dalam definisi fungsi.
- Parameter aktual adalah parameter (tidak selalu berupa variabel) yang dipakai dalam pemanggilan fungsi.

Parameter Formal dan Parameter Aktual





- Pada contoh program di atas misalnya, maka dalam fungsi `jumlah()` variabel **x** dan **y** dinamakan sebagai parameter formal, sedangkan variabel a dan b adalah parameter aktual

Pengiriman Parameter secara p*ass by value*

- Adalah cara pengiriman parameter pada semua contoh yang telah dibahas sebelumnya (Fungsi – Part 1)
- Yang dikirim sebagai paramenter aktual adalah value/nilainya
- Parameter aktual akan dicopy oleh parameter formal
- perubahan apapun yang terjadi pada parameter formal tidak akan berpengaruh kepada parameter aktual
  - perubahan di dalam fungsi tidak bisa terbaca di tempat fungsi tsb dipanggil

Contoh pass by value

```
#include <stdio.h>
void tukar(int, int);
main(){
  int a=5, b=3;

  printf("Sebelum pemanggilan fungsi\n");
  printf("a=%d, b=%d\n", a, b);
  tukar(a, b);
  printf("Sesudah pemanggilan fungsi\n");
  printf("a=%d, b=%d\n", a, b);
  return 0;
}
```

```
Sebelum pemanggilan fungsi
a=5, b=3

Nilai di akhir fungsi tukar()
x = 3   y = 5
Sesudah pemanggilan fungsi
a=5, b=3
```

```
void tukar(int x, int y) {
   int z;

   z = x;
   x = y;
   y = z;

   printf("\nNilai di akhir fungsi tukar()\n");
   printf("x = %d    y = %d\n", x, y);
}
```

Penggolongan Variabel berdasarkan Kelas Penyimpanan

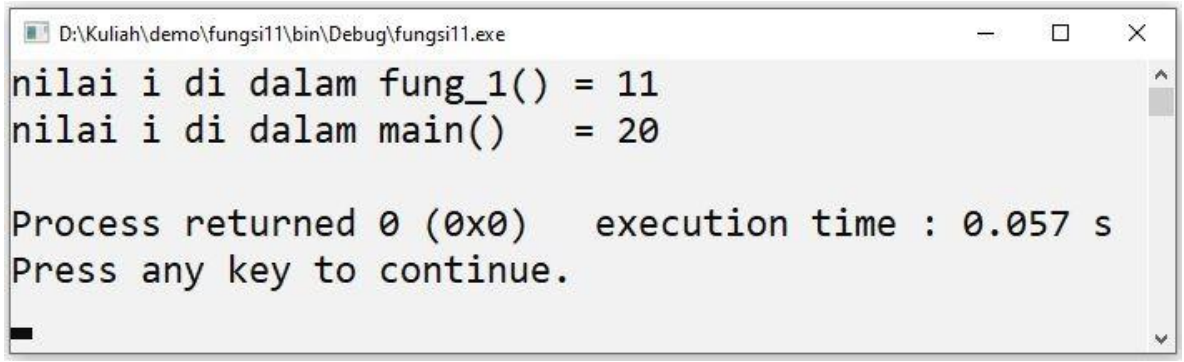
- Suatu variabel, di samping dapat digolongkan berdasarkan jenis/tipe data juga dapat diklasifikasikan berdasarkan kelas penyimpanan *(storage class).*
- Penggolongan berdasarkan kelas penyimpanan berupa :
  - variabel lokal/auto
  - variabel eksternal/global
  - variabel statis
  - variabel register

Variabel Lokal(auto)

- Variabel lokal adalah variabel yang dideklarasikan dalam sebuah fungsi
- Karakteristik variabel lokal adalah sbb :
  - Hanya dikenal oleh fungsi tempat variabel tersebut dideklarasikan
  - secara otomatis diciptakan ketika fungsi dipanggil dan akan sirna (lenyap) ketika eksekusi terhadap fungsi berakhir.
- Dalam banyak literatur, variabel lokal disebut juga dengan variabel otomatis, sehingga bisa dideklarasikan dengan menambahkan kata kuci *auto* di depan tipe-data variabel.
- Kata kunci ini bersifat opsional, biasanya disertakan sebagai penjelas saja.

Variabel Lokal (auto)

```
#include <stdio.h>
void fung_1(void);
main(){
 int i = 20;

 fung_1();
 printf("nilai i di dalam main()    = %d\n", i);
 return 0;
}

void fung_1(void) {
 int i = 11;

 printf("nilai i di dalam fung_1() = %d\n", i);
}
```

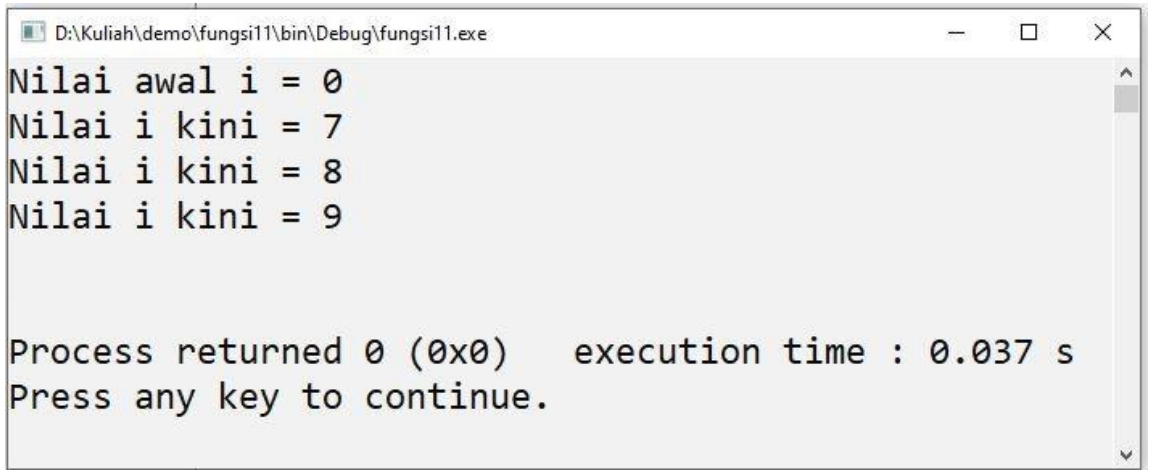


Variable Eksternal (global)

- Variabel eksternal merupakan variabel yang dideklarasikan di luar fungsi
- Karakteristiknya adalah sbb :
  - dapat diakses oleh semua fungsi
  - kalau tak diberi nilai, secara otomatis diinisialisasi dengan nilai sama dengan nol.
- variabel eksternal haruslah dideklarasikan sebelum definisi dari fungsi yang akan mempergunakannya.
- Untuk memperjelas bahwa suatu variabel dalam fungsi merupakan variabel eksternal, di dalam fungsi yang menggunakannya dapat mendeklarasikan ulang variabel tersebut dengan menambahkan kata kunci *extern* di depan tipe data variabel

Variable Eksternal (global)

```
int i;
void tambah(void);
main(){
 extern int i;
  printf("Nilai awal i = %d\n", i);
  i += 7;
  printf("Nilai i kini = %d\n", i);
  tambah();
  printf("Nilai i kini = %d\n", i);
  tambah();
  printf("Nilai i kini = %d\n", i);
  puts("");
  return 0;
}
void tambah(void){
 extern int i;
  i++;
}
```

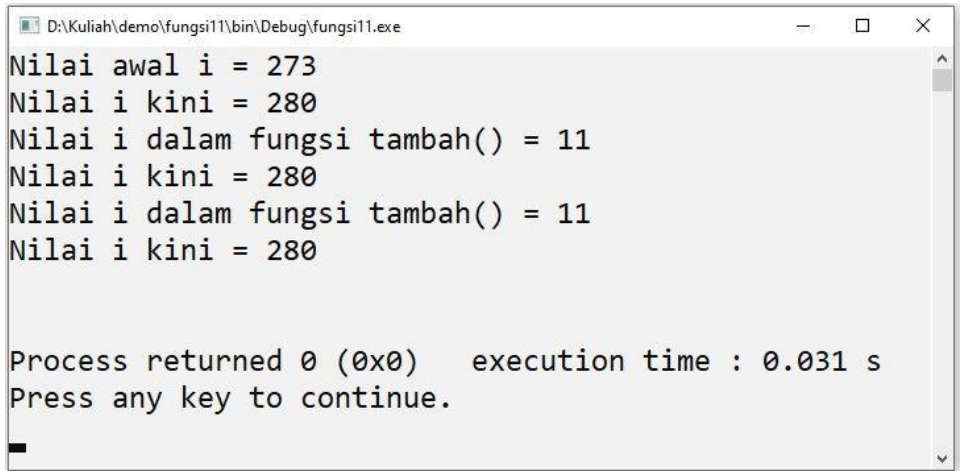


Variable Eksternal (global)

- Kalau dalam suatu program terdapat suatu variabel eksternal, maka suatu fungsi (yang ada pada program yang sama) bisa saja menggunakan nama variabel yang sama dengan variabel eksternal, namun diperlakukan sebagai variabel lokal.
- Untuk lebih jelasnya perhatikan contoh program berikut ini.

Variable Eksternal (global)

```
int i = 273;          //variabel eksternal
void tambah(void);
main(){
 extern int i;          //variabel eksternal
  printf("Nilai awal i = %d\n", i);
  i += 7;
  printf("Nilai i kini = %d\n", i);
  tambah();
  printf("Nilai i kini = %d\n", i);
  tambah();
  printf("Nilai i kini = %d\n\n", i);
 }
 void tambah(void){
  int i = 10;          //variabel lokal
  i++;
  printf("Nilai i dalam fungsi tambah() = %d\n", i);
 }
```

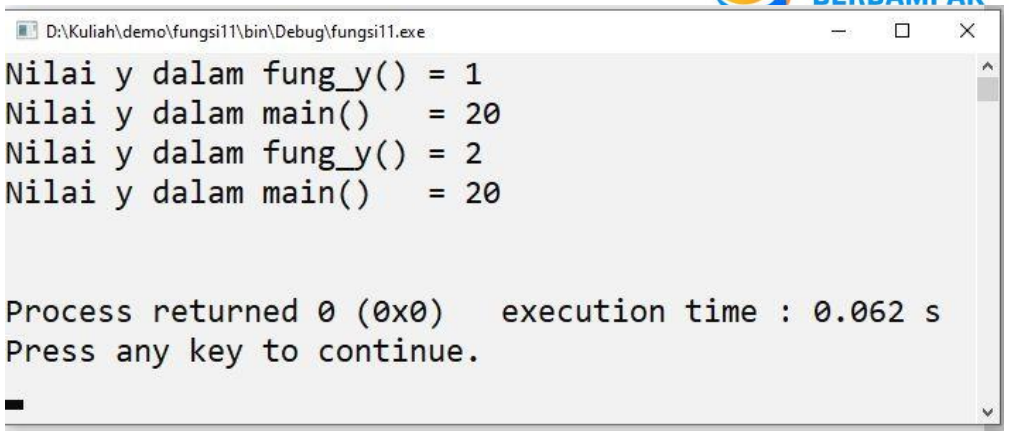


Variabel Statis

- Variabel statis dapat berupa variabel internal (didefinisikan di dalam fungsi) maupun variabel eksternal.
- Karakteristiknya adalah sbb :
  - Kalau variabel statis bersifat internal, maka variabel hanya dikenal oleh fungsi tempat variabel dideklarasikan
  - Kalau variabel statis bersifat eksternal, maka variabel dapat dipergunakan oleh semua fungsi yang terletak pada file yang sama, tempat variabel statis dideklarasikan
  - Berbeda dengan variabel lokal, variabel statis tidak akan hilang sekeluarnya dari fungsi (nilai pada variabel akan tetap diingat).
  - Inisialisasi akan dilakukan hanya sekali, yaitu saat fungsi dipanggil yang pertama kali. Kalau tak ada inisialisasi oleh pemrogram secara otomatis akan diberi nilai awal nol
- Variabel statis diperoleh dengan menambahkan kata kunci *static* di depan tipe data variabel.

Variabel Statis

```
#include <stdio.h>
void fung_y(void);
main(){
 int y = 20;

 fung_y();
 printf("Nilai y dalam main()    = %d\n", y);
 fung_y();
 printf("Nilai y dalam main()    = %d\n\n", y);
 return 0;
}

void fung_y(void){
 static int y;

 y++;
 printf("Nilai y dalam fung_y() = %d\n", y);
}
```

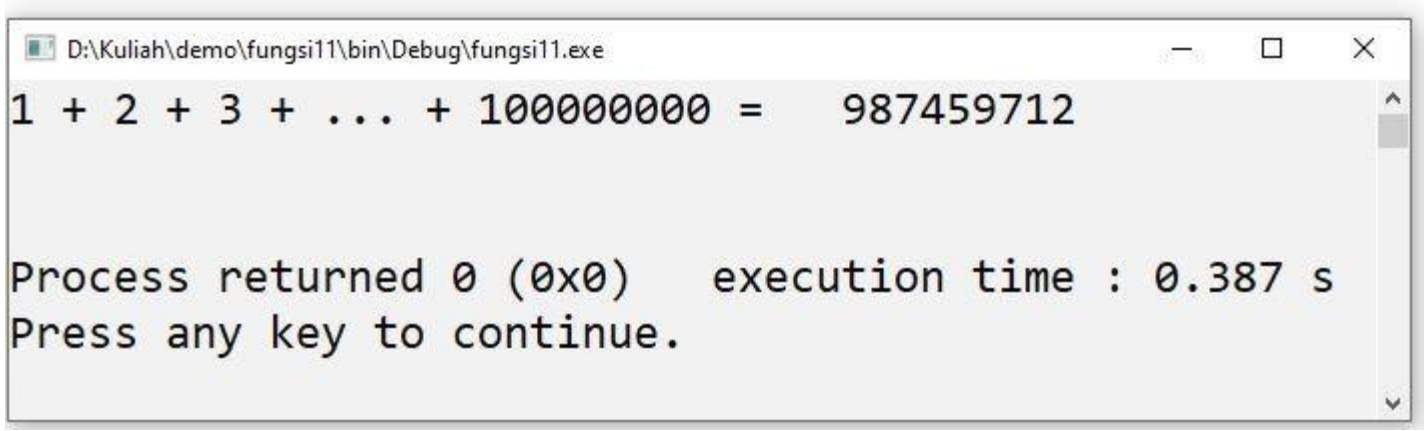


Variabel Register

- Variabel register adalah variabel yang nilainya disimpan dalam register CPU dan bukan dalam memori RAM.
- Variabel yang seperti ini hanya bisa diterapkan pada variabel yang lokal atau parameter formal, yang bertipe ***char*** atau ***int***.
- Variabel register biasa diterapkan pada variabel yang digunakan sebagai pengendali *loop*.
- Tujuannya untuk mempercepat proses dalam *loop,* sebab variabel yang dioperasikan pada register memiliki kecepatan yang jauh lebih tinggi daripada variabel yang diletakkan pada RAM.

Variabel Register

```
#include <stdio.h>

main(){
 register int i;
 int jumlah = 0;

 for(i = 1; i <= 100000000; i++)
   jumlah = jumlah + i;

 printf("1 + 2 + 3 + ... + 100000000 =    %d\n\n", jumlah);
 return 0
}
```



Pengenalan Konsep Pemrograman Terstruktur

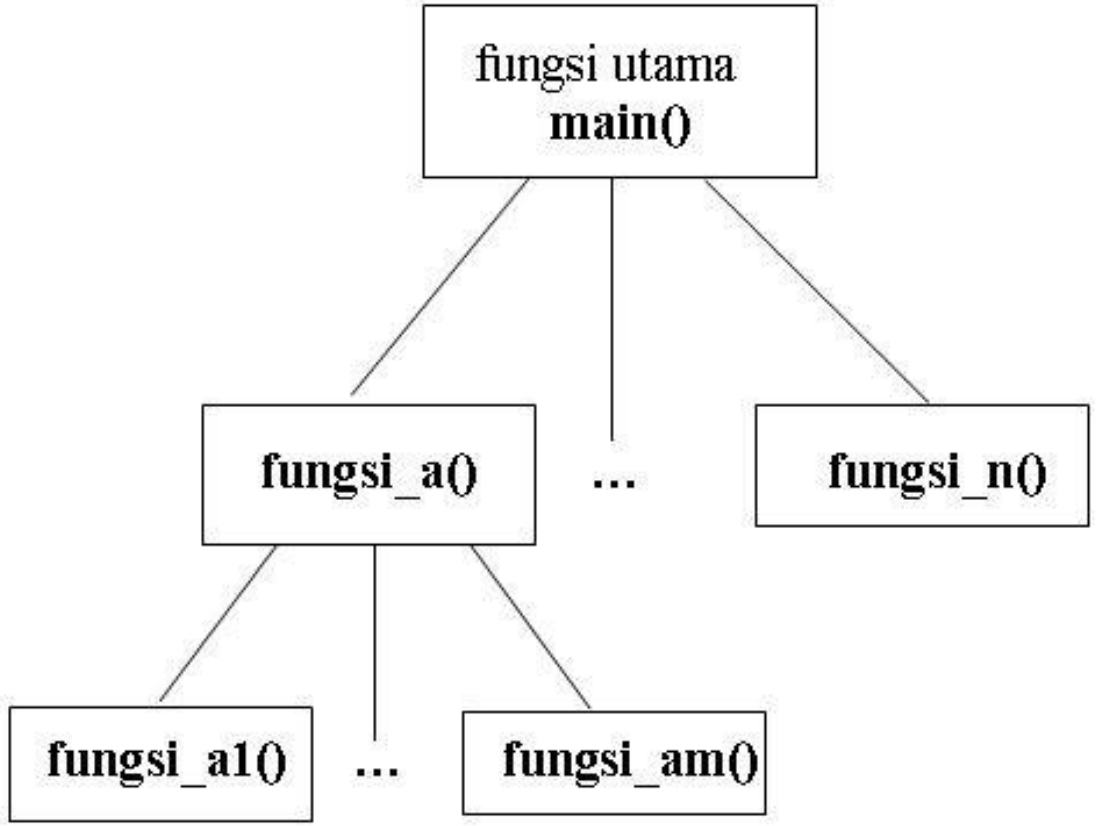
- Fungsi sangat bermanfaat untuk membuat program yang terstruktur
- Suatu program yang terstruktur dikembangkan dengan menggunakan *"top-down design"* (rancang atas bawah).
- Pada C suatu program disusun dari sejumlah fungsi dengan tugas tertentu, selanjutnya masing masing fungsi dipecah-pecah lagi menjadi fungsi yang lebih kecil
- Pembuatan program dengan cara ini akan memudahkan dalam pencarian kesalahan ataupun dalam hal pengembangan dan tentu saja mudah dipahami/dipelajari

Pengenalan Konsep Pemrograman Terstruktur

- Dalam bentuk diagram, model suatu program C yang terstruktur adalah seperti yang tertera pada bagan di halaman berikut.
- Namun sekali lagi perlu diketahui, bahwa pada C semua fungsi sebenarnya berkedudukan sederajat.

Pengenalan Konsep Pemrograman Terstruktur

- Fungsi `main()` terdiri atas `fungsi_a()` sampai dengan `fungsi_n()`
- Menegaskan bahwa fungsi `main()` akan memanggil `fungsi_a()` sampai dengan `fungsi_n()`
- Adapun fungsi-fungsi yang dipanggil oleh fungsi `main()` tsb juga bisa memanggil fungsi-fungsi yang lain.



Latihan

1. Definisikanlah function untuk menentukan bilangan terbesar dari 2 bilangan yang diinputkan di `main()`. Function mempunyai parameter berupa 2 buah bilangan yang akan dibandingkan dan memberikan *return value* berupa bilangan yang terbesar. Sertakan pula prototype function tsb.
2. Buatlah suatu fungsi `permutasi()` dan `kombinasi()` untuk menghitung permutasi dan kombinasi dari suatu pasangan bilangan, yang dinyatakan dengan formula sbb:

$$\text{Permutasi} \quad : \quad P(n,r) = \frac{n!}{(n-r)!}$$

$$\text{Kombinasi} \quad : \quad C(n,r) = \frac{n!}{r!(n-r)!}$$

Latihan

Untuk program-program di bawah ini :

- Trace secara manual semua program di bawah ini baris per barisnya, dan tampilkan nilai semua variabel pada setiap baris prosesnya.
- Tebaklah tampilkan keluaran programnya

Latihan

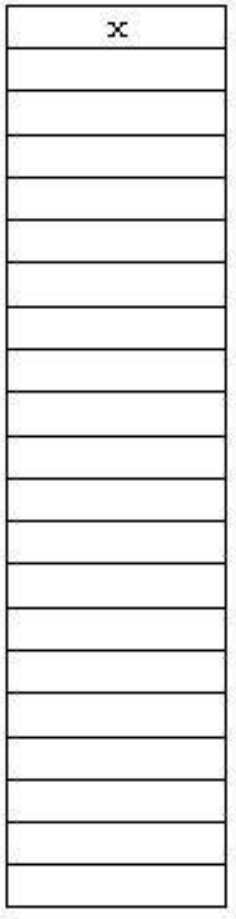
3.
```
int OddEvenTest(int);

main()
{
   int a, hasil;

   a = 5;
   hasil = OddEvenTest(a);
   printf("a=%d; hasil=%d\n",a,hasil);
}

OddEvenTest(int b)
{
   int a;

   a = b % 2;
   return a;
}
```

a	hasil	b

Latihan

4.

```
void fung_a(void);
void fung_b(void);

int x = 20;

main(){
   x += 2;
   fung_a();
   fung_a();
   printf("\nNilai x dalam main() = %d\n\n",x);
}

void fung_a(void){
   static x = 5;

   x++;
   printf("Nilai x dalam fung_a() = %d\n", x);
   fung_b();
}

void fung_b(void){
   x--;
   printf("Nilai x dalam fung_b() = %d\n", x);
}
```



Latihan

5. Definisikanlah fungsi `main()`, `masukan()` dan `average()`, sebagai berikut :

- Fungsi `masukan()` menerima satu parameter berupa jumlah data yang akan dimasukkan dan memberikan return value berupa nilai total dari seluruh data yang dimasukkan. Fungsi ini bertugas menerima masukan data sebanyak n kali dan sekaligus menghitung total nilai seluruh data.
- Fungsi `average()` menerima dua parameter berupa jumlah data yang telah dimasukkan dan nilai total seluruh data. Fungsi ini memberikan return value berupa nilai rata-rata dari seluruh data yang dimasukkan.
- Pada fungsi `main()` mintalah masukan jumlah data yang akan diinputkan. Selanjutnya lakukan pemanggilan fungsi `masukan()` dan `average()`, kemudian tampilkan nilai rata-rata dari seluruh datanya.

Latihan

6. Definisikanlah fungsi-fungsi sebagai berikut :
   - Fungsi `f_to_i()` untuk mengubah ukuran dari satuan kaki *(feet)* ke inci
   - Fungsi `i_to_cm()` untuk mengubah ukuran dari satuan inci ke centimeter
   - Fungsi `c_to_m()` untuk mengubah ukuran dari satuan centimeter ke meter
   - Dalam `main()` mintalah masukan ukuran dalam satuan kaki *(feet)* kemudian lakukan konversi sampai mendapatkan keluaran berupa ukuran dalam meter. Tentukan jumlah dan tipe parameter dan return value yang dibutuhkan

Keterangan :

1 kaki = 12 inchi, 1 inchi = 2.54 cm, 100 cm = 1 meter

Latihan

- 7. Apa hasil eksekusi dari program berikut:

```
/* File program : lat1.c
*/ #include
void
fung_a(void); void
fung_b(void);

int x =

20; main()
{
x += 2;
fung_a();
fung_a();
printf("\nNilai x dalam main() = %d\n\n", x);
}
```

```
void fung_a(void)
{
  static x = 5;
  x++;
  printf("Nilai x dalam fung_a() = %d\n", x);
  fung_b();
}
void fung_b(void)
{
  x--;
  printf("Nilai x dalam fung_b() = %d\n", x);
}
```

Referensi

1. Brian W. Kerninghan, Dennis M. Ritchie (2012): The C Programming Language : Ansi C Version 2 Edition, PHI Learning
2. Byron Gottfried (2010) : Programming with C, Tata McGraw - Hill Education
3. **Kochan Stephen** (20040 : Programming in C, 3rd Edition, Sams
4. K. N. King (2008) : C Programming: A Modern Approach, 2nd Edition, W.
   - W. Norton & Company
5. Abdul Kadir (2012) : Algoritma & Pemrograman Menggunakan C & C++, Andi Publisher, Yogyakarta
6. http://www.gdsw.at/languages/c/programming-bbrown/
7. https://www.petanikode.com/tutorial/c/
8. http://www.cprogramming.com/tutorial/c-tutorial.html