**PRAKTIKUM 16**

**Fungsi : Penggolongan Variabel & Pemrograman Terstruktur**

1. **DASAR TEORI**

**Penggolongan Variabel berdasarkan Kelas Penyimpanan**

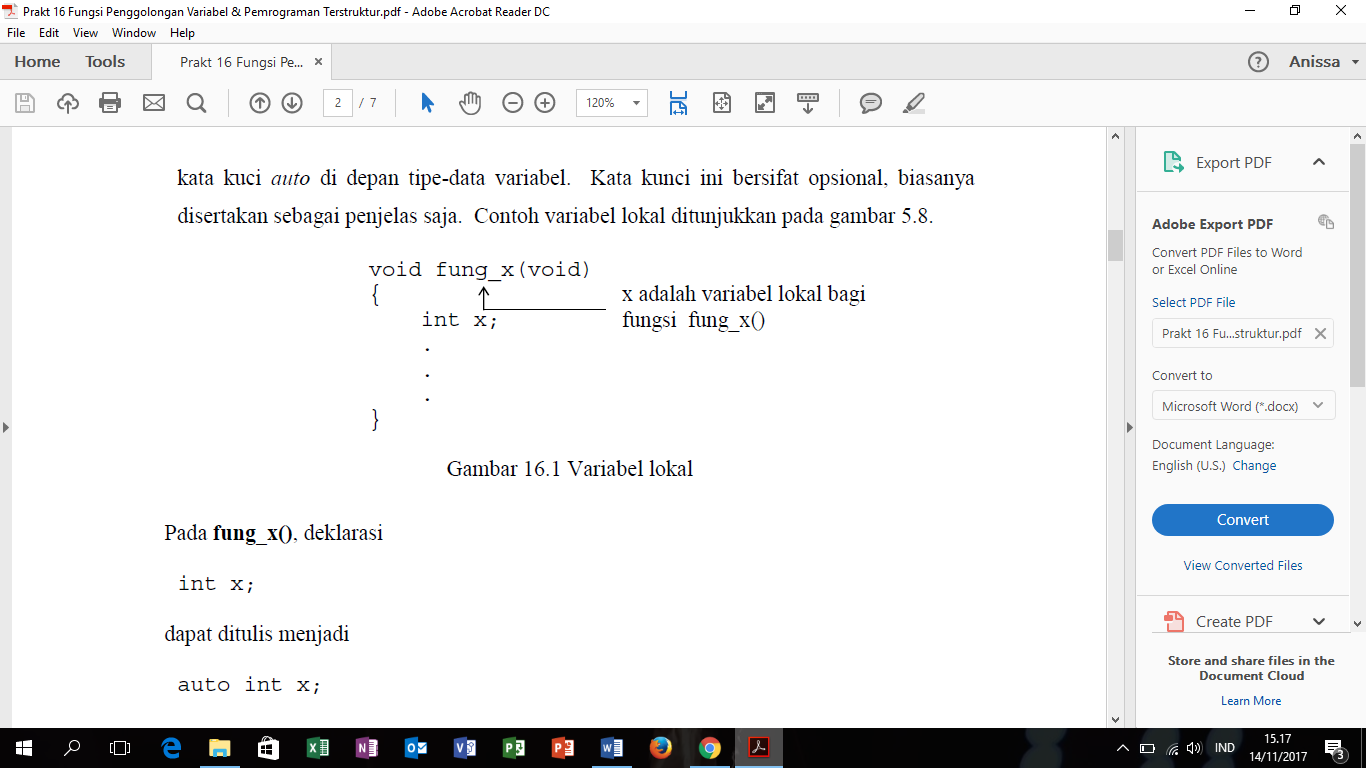
Suatu variabel, di samping dapat digolongkan berdasarkan jenis/tipe data juga dapat diklasifikasikan berdasarkan kelas penyimpanan *(storage class).* Penggolongan berdasarkan kelas penyimpanan berupa :

* variabel lokal
* variabel eksternal
* variabel statis
* variabel register

**Variabel Lokal**

Variabel lokal adalah variabel yang dideklarasikan dalam fungsi, dengan sifat :

* Secara otomatis diciptakan ketika fungsi dipanggil dan akan sirna (lenyap) ketikA eksekusi terhadap fungsi berakhir.
* Hanya dikenal oleh fungsi tempat variabel tersebut dideklarasikan
* Tidak ada inisialisasi secara otomatis (saat variabel diciptakan, nilainya tak menentu).

Dalam banyak literatur, variabel lokal disebut juga dengan variabel otomatis. Variabel yang termasuk dalam golongan ini bisa dideklarasikan dengan menambahkan kata kuci *auto* di depan tipe-data variabel. Kata kunci ini bersifat opsional, biasanya disertakan sebagai penjelas saja. Contoh variabel lokal

Pada **fung\_x()**, deklarasi

int x;

dapat ditulis menjadi

auto int x;

Penerapan variabel lokal yaitu bila variabel hanya dipakai oleh suatu fungsi (tidak dimaksudkan untuk dipakai oleh fungsi yang lain). Pada contoh berikut, antara variabel **i** dalam fungsi **main()** dan **fung\_1()** tidak ada kaitannya, sebab masing-masing merupakan variabel lokal.

**Variabel Eksternal**

Variabel eksternal merupakan variabel yang dideklarasikan di luar fungsi, dengan sifat :

* dapat diakses oleh semua fungsi
* kalau tak diberi nilai, secara otomatis diinisialisasi dengan nilai sama dengan nol.

Variabel eksternal haruslah dideklarasikan sebelum definisi fungsi yang akan mempergunakannya. Untuk memperjelas bahwa suatu variabel dalam fungsi merupakan variabel eksternal, di dalam fungsi yang menggunakannya dapat mendeklarasikan variabel itu kembali dengan menambahkan kata kunci *extern* di depan tipe data variabel.

Kalau dalam suatu program terdapat suatu variabel eksternal, suatu fungsi bisa saja menggunakan nama variabel yang sama dengan variabel eksternal, namun diperlakukan sebagai variabel lokal.

**Variabel Statis**

Variabel statis dapat berupa variabel internal (didefinisikan di dalam fungsi) maupun variabel eksternal. Sifat variabel ini :

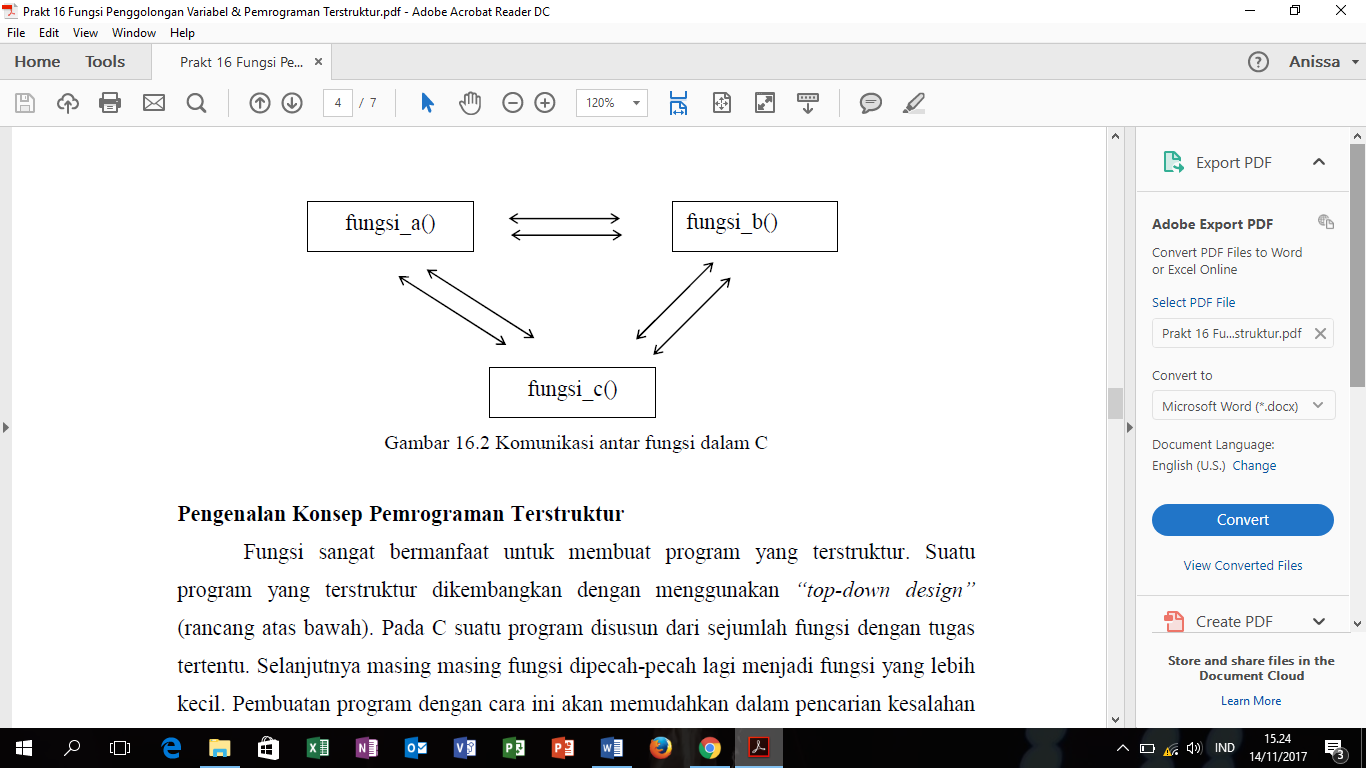
* Kalau variabel statis bersifat internal, maka variabel hanya dikenal oleh fungsi tempat variabel dideklarasikan
* Kalau variabel statis bersifat eksternal, maka variabel dapat dipergunakan oleh semua fungsi yang terletak pada file yang sama, tempat variabel statis dideklarasikan
* Berbeda dengan variabel 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.

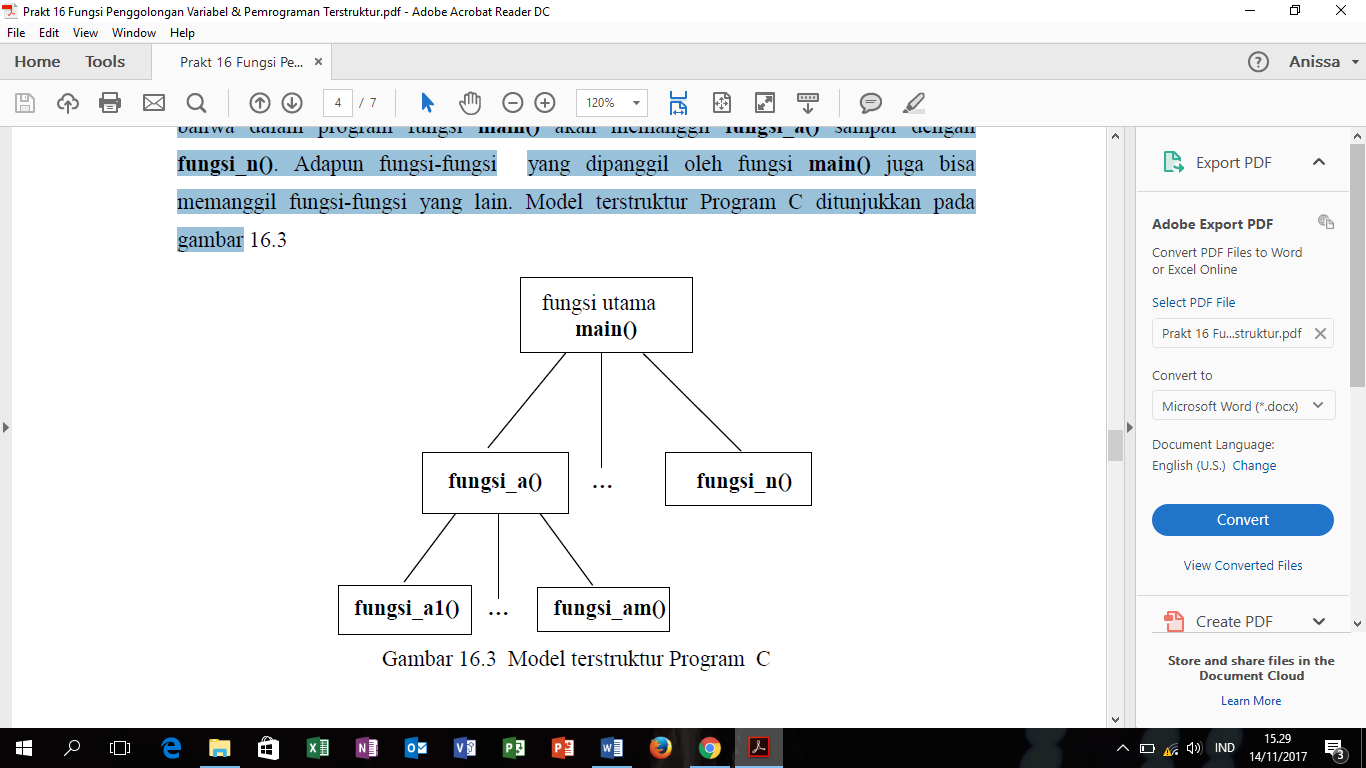
**Variabel Register**

Variabel register adalah variabel yang nilainya disimpan dalam register 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.

**Menciptakan Sejumlah Fungsi**

Pada C, semua fungsi bersifat sederajat. Suatu fungsi tidak dapat didefinisikan di dalam fungsi yang lain. Akan tetapi suatu fungsi diperbolehkan memanggil fungsi yang lain, dan tidak tergantung kepada peletakan definisi fungsi pada program. Komunikasi antara fungsi dalam C ditunjukkan dalam gambar 16.2. Gambar tersebut menjelaskan kalau suatu fungsi katakanlah **fungsi\_a()** memanggil **fungsi\_b()**, maka bisa saja **fungsi\_b()** memanggil **fungsi\_a()**. Contoh program yang melibatkan fungsi yang memanggil fungsi yang lain ada pada program **kom\_fung.c**, yaitu **fungsi\_1()** dipanggil dalam **main()**, sedangkan **fungsi\_2()** dipanggil oleh **fungsi\_1()**.

**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. Dalam bentuk diagram, model suatu program C yang terstruktur adalah seperti yang tertera pada bagan berikut ini. Namun sekali lagi perlu diketahui, bahwa pada C semua fungsi sebenarnya berkedudukan sederajat. Fungsi **main()** terdiri dari **fungsi\_a()** sampai dengan **fungsi\_n()**, menegaskan bahwa dalam program fungsi **main()** akan memanggil **fungsi\_a()** sampai dengan **fungsi\_n()**. Adapun fungsi-fungsi yang dipanggil oleh fungsi **main()** juga bisa memanggil fungsi-fungsi yang lain. Model terstruktur Program C ditunjukkan pada gambar di bawah ini

1. **PERCOBAAN**
2. Adakah sesuatu yang salah pada sebuah fungsi yang tidak mempunyai return value ? Jelaskan analisismu tentang sebuah fungsi yang tidak memiliki return value!

Jawab : Tidak ada yang salah pada fungsi yang tidak memiliki return value, karena fungsi bisa melakukan tugas khusus tanpa memakai return value, tetapi harus memakai tipe data void dimana tipe data ini berfungsi mendeklarasikan fungsi yang tidak memiliki return value.

1. Apakah yang terjadi jika sebuah fungsi memberikan return value tetapi tidak diassign ke variabel apapun ?

Jawab : Tidak terjadi masalah atau error. Hasil return value akan tetap muncul apabila tidak ada variable yang meassign return valuenya tetapi fungsi tersebut memiliki nilai yang sama dengan return valuenya, atau apabila terdapat variabel tetapi tidak diassignkan, maka pemanggilan fungsi tersebut akan sia-sia karena tidak bisa mengolah hasil return value tersebut.

1. Apakah yang terjadi jika sebuah fungsi diassign ke sebuah variabel padahal fungsi tersebut tidak memiliki return value ?

Jawab : Apabila suatu fungsi bertipe void dan tidak memiliki return value namun harus diassign pasti akan error karena fungsi bertipe void sama saja dengan fungsi yang tidak memiliki tipe keluaran. Namun, akan berbeda jika fungsi tersebut memiliki tipe keluaran selain void, karena fungsi tersebut memiliki tipe keluaran sehingga nilai dari fungsi tersebut bisa diassign ke suatu variable meskipun tidak memiliki return value.

1. Trace secara manual semua program di bawah ini baris per barisnya, dan tampilkan nilai semua variabel pada setiap baris prosesnya. Selain itu, tebaklah tampilkan keluaran programnya
2. 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 |
| 5 | 1 | 5 |

1. void demo(void);

main()

{

int i=0;

while(i < 3) {

demo();

i++;

}

void demo(void)

{

auto int var\_auto = 0;

static int var\_static = 0;

printf("auto = %d, static = %d\n",

var\_auto, var\_static);

++var\_auto;

++var\_static;

}

|  |  |  |
| --- | --- | --- |
| i | var\_auto | var\_static |
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 2 | 0 | 2 |

1. void fung\_a(void);

|  |
| --- |
| x |
| 20 |
| 22 |
| 5 |
| 6 |
| 21 |
| 7 |
| 20 |
| 20 |

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);

}

1. Definisikanlah fungsi main(), masukan() dan average()

* 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.

1. Listing

#include <stdio.h>

#include <stdlib.h>

float masukan(int);

float average(int, float);

int main()

{

puts("Program Menentukan Rata-Rata Data\n");

int n;

float total,rata;

printf("Masukkan jumlah data yang diinputkan : ");

scanf("%d", &n);

total = masukan(n);

rata = average(n, total);

printf("\nTotal dari seluruh data = %g", total);

printf("\nRata-rata dari seluruh data = %g ", rata);

return 0;

}

float masukan(int data)

{

int x;

float jumlah=0, nilai;

for(x=1; x<=data; x++)

{

printf("Masukkan data ke-%d : ", x);

scanf("%f", &nilai);

fflush(stdin);

jumlah = jumlah + nilai;

}

return jumlah;

}

float average(int a, float sum)

{

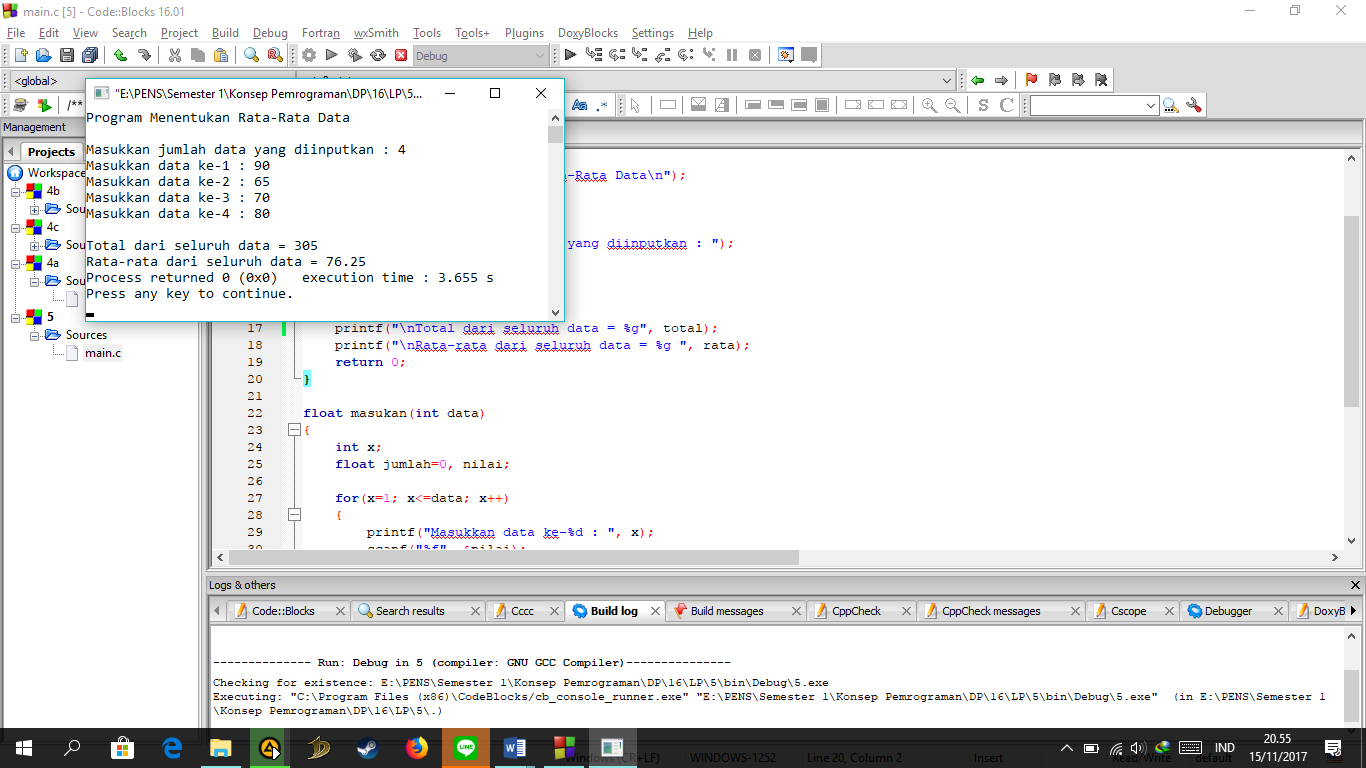
float akhir;

akhir = sum / a;

return akhir;

}

1. Output



1. 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

1. Listing

#include <stdio.h>

#include <stdlib.h>

float f\_to\_i(float);

float i\_to\_cm(float);

float cm\_to\_m(float);

int main()

{

float kaki, inchi, cm, m;

puts("Program Konversi Ukuran Kaki Menjadi Meter\n");

printf("Masukkan ukuran kaki anda : ");

scanf("%f", &kaki);

inchi = f\_to\_i(kaki);

cm = i\_to\_cm(inchi);

m = cm\_to\_m(cm);

printf("%g feet = %g inchi\n", kaki, inchi);

printf("%g inchi = %g cm\n", inchi, cm);

printf("%g cm = %g m\n", cm, m);

return 0;

}

float f\_to\_i(float x)

{

float y;

y = x \* 12;

return y;

}

float i\_to\_cm(float a)

{

float b;

b = a \* 2.54f;

return b;

}

float cm\_to\_m(float c)

{

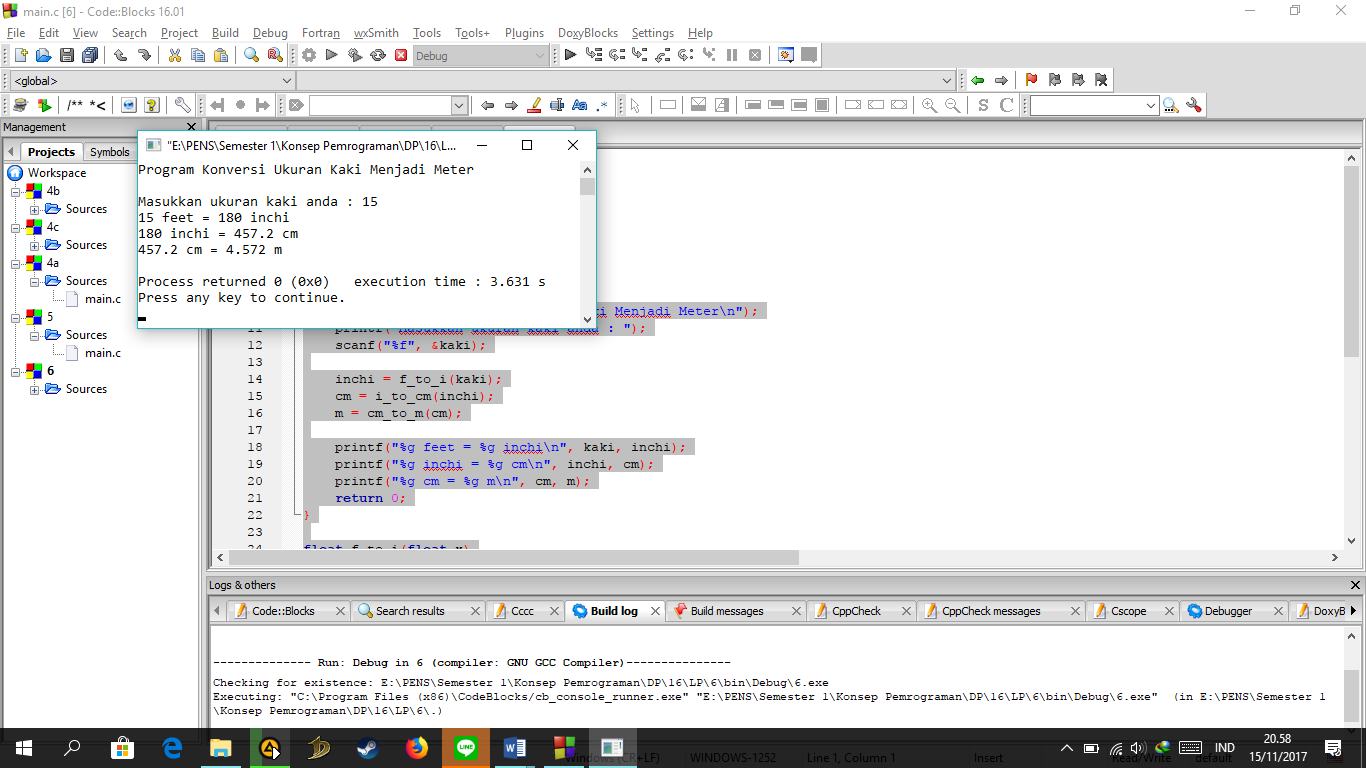
float d;

d = c / 100;

return d;

}

1. Output



1. **Kesimpulan**
2. Fungsi adalah potongan kecil dari program yang mempunyai tugas khusus dan bagiannya terpisah. Serta mempunyai tujuan agar suatu program lebih terstruktur dan mengurangi duplikasi dalam penulisan kode program
3. Fungsi memiliki parameter sesuai dengan kebutuhan
   * Parameter formal adalah variable yang ada pada daftar parameter dalam definisi fungsi
   * Parameter aktual adalah parameter(tidak selalu berupa variabel) yang dipakai dalam pemanggilan fungsi.
   * Dalam parameter ini type, jumlah, dan urutan harus sama, dan juga dalam penamaan sebaiknya diberi nama yang berbeda
4. Parameter dapat dikirim pada fungsi dengan *pass by value* dan *pass by reference*
   * *Pass by value* hanya mengirimkan nilai dari variabel yang dikirimkan
   * *Pass by reference* megirim alamat variabel yang dikirimkan, tetapi kami belum mempelajari ini lebih lanjut.
5. Terdapat beberapa variabel yang digolongkan berdasarkan tempat menyimpanannya, yaitu :
   * Variabel lokal, hanya dapat diakses dalam fungsi tempat dia dideklarasikan
   * Variabel eksternal, dapat diakses di fungsi mana saja di dalam program
   * Variabel statis, nilainya disimpan dan dapat diakses ketika fungsi dijalankan lagi
   * Variabel register, variabel yang nilainya disimpan dalam register bukan di RAM, sehingga membuat proses dalam program menjadi lebih cepat.