**LAPORAN JOBSHEET 6**

**Fungsi**



Disusun Oleh :

DWI RINAYATUS SA’ADAH

3.34.18.1.10

IK- 1B

**PROGRAM STUDI TEKNIK INFORMATIKA**

**JURUSAN TEKNIK ELEKTRO**

**POLITEKNIK NEGERI SEMARANG**

**TAHUN 2018/2019**

**FUNGSI**

1. **Pendahuluan**

* Fungsi adalah :
* suatu bagian dari program
* yang dirancang untuk melaksanakan tugas tertentu
* letaknya dipisahkan dari program yang menggunakannya.
* Macam fungsi:
* standard : sudah disediakan oleh compiler, tinggal dipakai dengan menyebutkan headernya (kamusnya) pada preprosessor include, misalnya fungsi : printf() stdio.h; exit() stdlib.h
* user defined : didefinisikan oleh user, disesuaikan dengan kebutuhan user ybs

1. **Tujuan Fungsi**

* Fungsi banyak digunakan dengan tujuan :
* Membuat program menjadi terstruktur, sehingga mudah dipahami dan mudah dikembangkan. Dengan memisahkan langkah-langkah detail ke satu atau lebih fungsi-fungsi, maka fungsi utama (main()) menjadi lebih pendek, jelas dan mudah dimengerti.
* Mengurangi pengulangan (duplikasi) kode, langkahlangkah program yang sama dan dipakai berulang-ulang di program dapat dituliskan sekali saja secara terpisah dalam bentuk fungsi-fungsi. Selanjutnya bagian program yang membutuhkan langkah-langkah ini tidak perlu selalu menuliskannya, tetapi cukup memanggil fungsi-fungsi tersebut.

1. **Dasar Fungsi**

* Fungsi standar C yang mengemban tugas khusus contohnya adalah ;
* printf() , yaitu untuk menampilkan informasi atau data ke layar.
* scanf() , yaitu untuk membaca kode tombol yang diinputkan.
* Pada umumnya fungsi memerlukan nilai masukan atau parameter yang disebut sebagai argumen yang akan diolah oleh fungsi.
* Hasil akhir fungsi berupa sebuah nilai (disebut sebagai return value atau nilai keluaran fungsi).
* Oleh karena itu fungsi sering digambarkan sebagai "kotak gelap" seperti ditunjukkan pada gambar berikut ini.

**Keluaran Fungsi**

**Parameter**

**FUNGSI**

* Parameter bisa diatikan sebagai “bahan baku”yang akan diproses dalam fungsi dan dikirim dari tempat fungsi tsb dipanggil
* Keluaran fungsi (return value) bisa diartikan sebagai “oleholeh”yang akan dibawa ketika proses kembali ke tempat asal fungsi tsb dipanggil
* Bentuk umum dari definisi sebuah fungsi adalah sebagai berikut ;

tipe-keluaran-fungsi nama-fungsi (deklarasi argumen)

{

tubuh fungsi

}

Keterangan :

1. tipe-keluaran-fungsi, dapat berupa salah satu tipe data C, misalnya char atau int . Kalau penentu tipe tidak disebutkan maka dianggap bertipe int (secara default).
2. tubuh fungsi berisi deklarasi variabel (kalau ada) dan statemenstatemen yang akan melakukan tugas yang akan diberikan kepada fungsi yang bersangkutan. Tubuh fungsi ini ditulis di dalam tanda kurung kurawal buka dan kurung kurawal tutup.
3. **Jenis Fungsi**

Berdasarkan keberadaan return value-nya, maka fungsi dibagi menjadi 2 jenis, yaitu :

1. Punya return value (RV), ciri-cirinya :

* ada nama tipe data di depan nama fungsi, kecuali jika RVnya integer, boleh tidak ditulis karena merupakan tipe default
* ada statemen return di dalam body fungsi 2.

1. Tidak punya return value, ciri-cirinya :

* ada tipe void di depan nama fungsi
* tidak ada statemen return di dalam body fungsi

1. Fungsi dengan Return Value integer

Fungsi yang memiliki RV integer, maka di depan nama fungsi boleh dituliskan tipe int atau tanpa tipe sama sekali.

int minimum (int x, int y) minimum (int x, int y)

{ {

if (x < y) if (x < y)

return (x); return (x);

else else

return (y); return (y);

} }

Contoh Program

#include <stdio.h>

int minimum (int, int);

main()

{

int a, b, kecil;

printf("Masukkan nilai a : ");

scanf("%d", &a);

printf("Masukkan nilai b : ");

scanf("%d", &b);

kecil = minimum(a, b);

printf("\nBilangan terkecil antara %d dan %d adalah %d\n\n",a,b,kecil);

}

minimum(int x, int y)

{

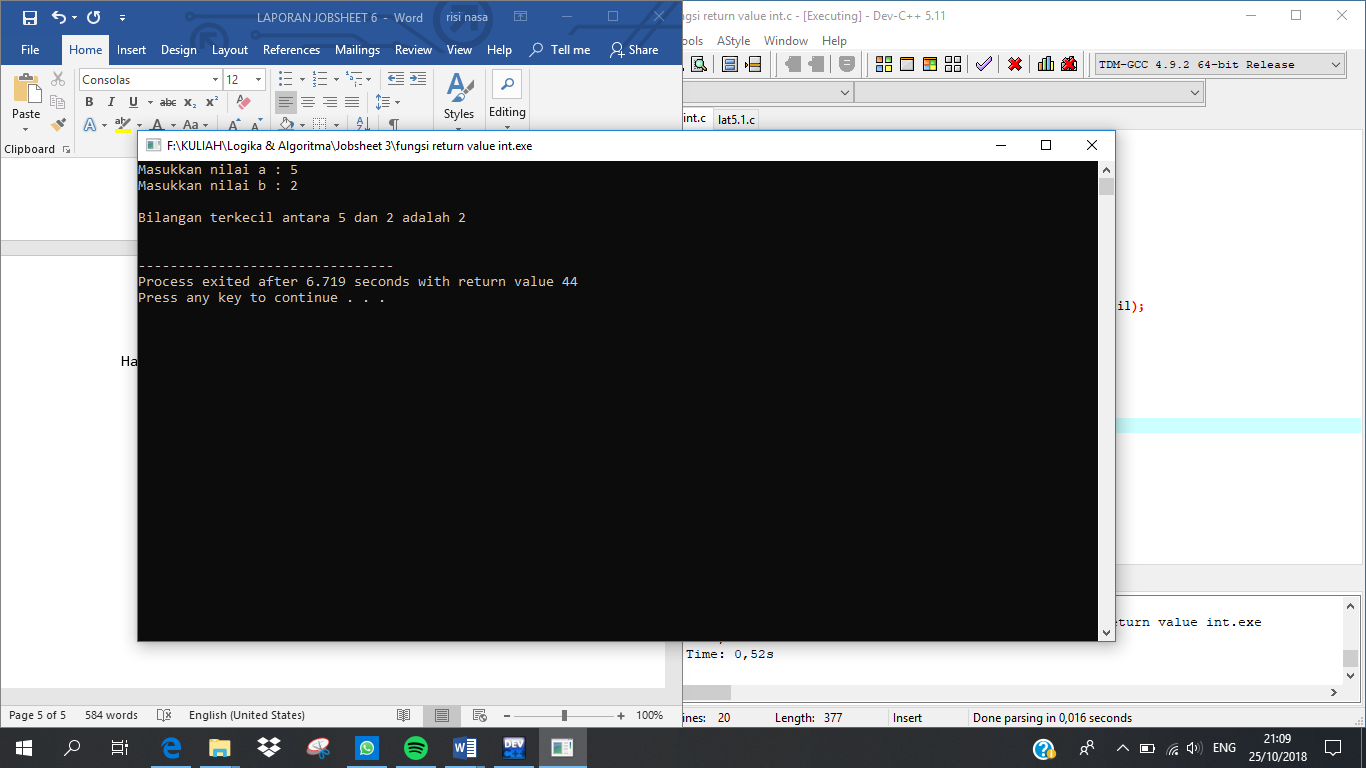
if (x < y)

return(x);

else

return(y);

}

Hasil Run Program

1. Fungsi dengan Return Value bukan integer

* Untuk fungsi yang mempunyai RV bertipe bukan integer, maka fungsi HARUS didefiniskan dengan diawali tipe RV-nya (ditulis di depan nama fungsi).
* Contoh fungsi jumlah() memiliki RV bertipe float sbb :

float jumlah(float x, float y)

{

return(x+y);

}

Contoh Program

#include <stdio.h>

float jumlah(float, float);

main()

{

float a, b,c;

printf("Masukkan nilai a : ");

scanf("%f", &a);

printf("Masukkan nilai b : ");

scanf("%f", &b); c = jumlah(a, b);

printf("\nHasil penjumlahan a + b = %g\n", c);

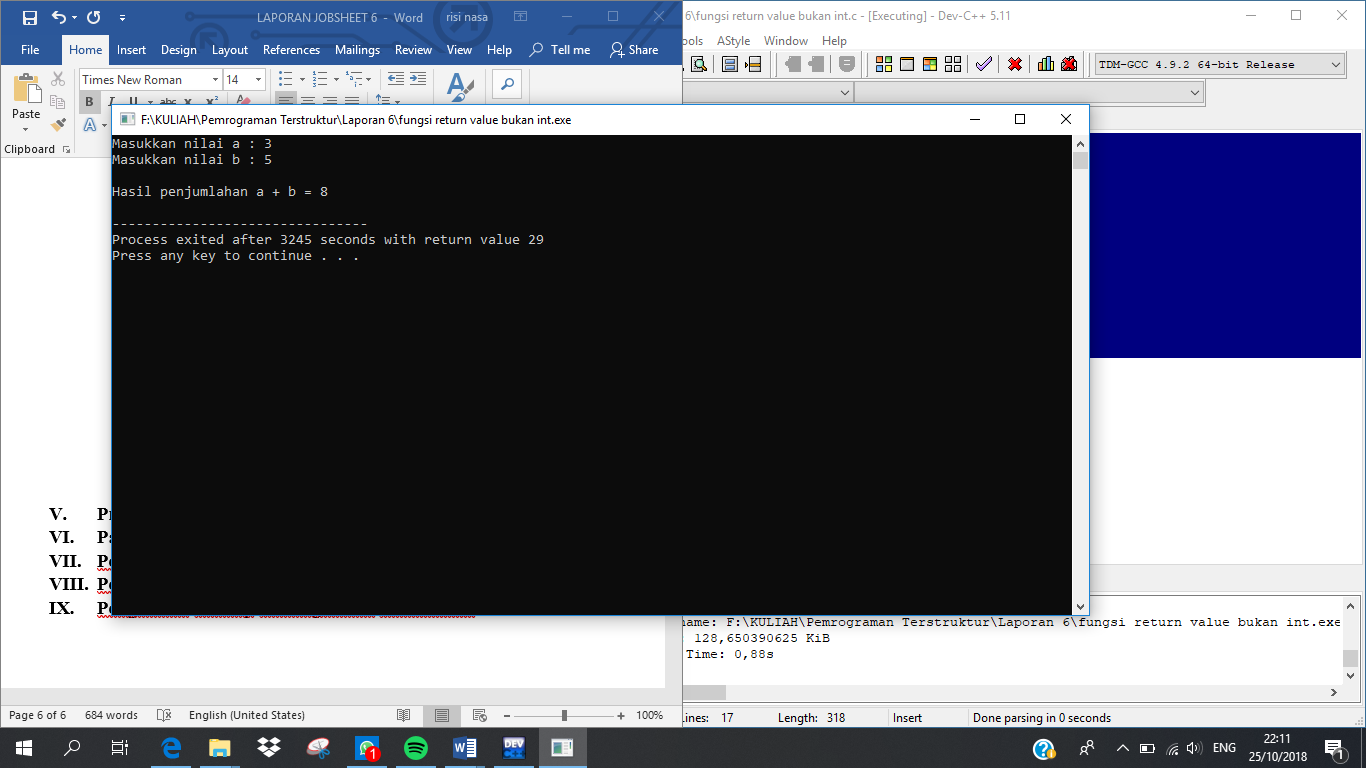
}

float jumlah(float x, float y)

{

return(x + y);

}

**Hasil Run Program**

* Khusus untuk fungsi yang dirancang tanpa memberikan RV (melainkan hanya menjalankan suatu tugas khusus) biasa didefinisikan dengan diawali kata kunci void (di depan nama fungsi).
* Contoh fungsi info\_program() yang tugasnya menampilkan informasi tentang sebuah program

Void info\_program()

{

puts (“========================”);

puts(“Program dibuat oleh Moh.Izzudin”);

puts(“Tanggal : 5 Marey 2007 ”);

puts(“ “);

puts(“Selamat menggunakannya…… “);

puts(“========================”);

}

**Contoh Program**

#include <stdio.h>

void info\_program();

main()

{

info\_program();

}

void info\_program()

{

puts("==================================");

puts("Progam dibuat oleh Moh. Izzuddin ");

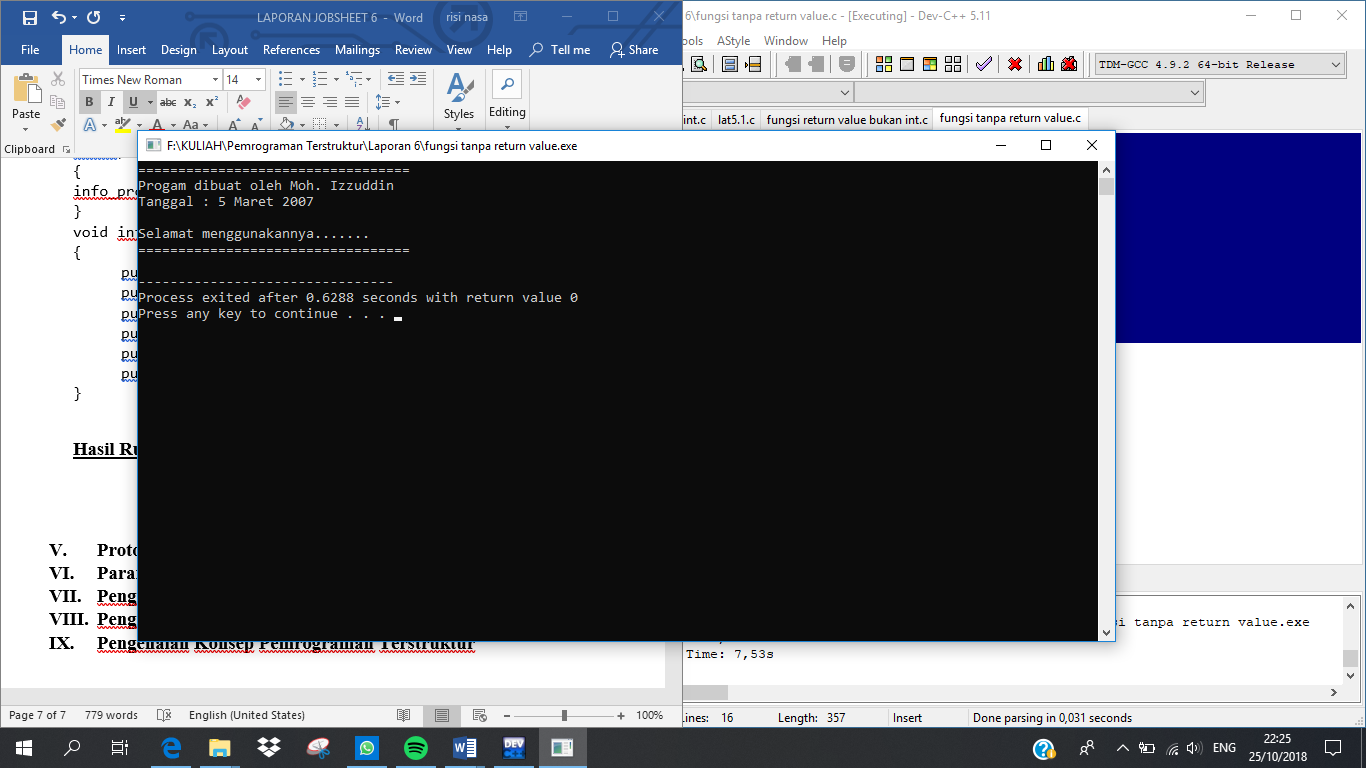
puts("Tanggal : 5 Maret 2007 ");

puts(" ");

puts("Selamat menggunakannya....... ");

puts("==================================");

}

 **Hasil Run Program**

1. **Prototype/Deklarasi Fungsi**

* Prototipe fungsi digunakan untuk menjelaskan kepada kompiler mengenai :
* tipe keluaran fungsi
* jumlah parameter
* tipe dari masing-masing parameter
* Bagi kompiler, informasi tsb akan dipakai untuk memeriksa keabsahan (validitas) parameter dalam pemanggilan fungsi.
* Salah satu keuntungannya adalah, kompiler akan melakukan konversi seandainya antara tipe parameter dalam fungsi dan parameter saat pemanggilan fungsi tidak sama, atau akan menunjukan kesalahan bila jumlah parameter dalam definisi dan saat pemanggilan berbeda.

**Contoh prototype fungsi :**

float jumlah (float x, float y);

atau

float jumlah (float, float);

Penjelasannya adalah sbb :

float jumlah (float, float);

Nama fungsi

Diakhiri dengan titik koma

Tipe parameter kedua

Tipe parameter pertama

Tipe keluaran fungsi

* Untuk fungsi yang tidak memiliki argument (contoh program void.c) maka deklarasinya adalah

void info\_program(void);

menyatakan bahwa info\_program() tidak memiliki parameter)

Catatan :

* Untuk fungsi-fungsi pustaka, prototipe dari fungsi-fungsi berada di file-file judulnya (header file). Misalnya fungsi pustaka *printf()* dan *scanf()* prototipenya berada pada file dengan nama **stdio.h**
* Untuk fungsi pustaka pencantuman pada prototipe fungsi dapat dilakukan dengan menggunakan processor directive **#include**.

1. **Parameter Aktual dan Parameter Formal**

* 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.

float jumlah(float x, float y))

{

return( x + y);

}

main()

{

. . .

c= jumlah (a,b);

. . .

}

Parameter formal

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 actual

1. **Pengiriman parameter secara pass by value**

* Adalah cara pengiriman parameter pada semua contoh yang telah dibahas sebelumnya (Bab 7 Fungsi 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);

}

1. **Penggolongan Variabel berdasarkan Kelas**

* 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

1. Variabel local (auto)

* Variabel lokal adalah variabel yang dideklarasikan dalam sebuah fungsi
* Karakteristik variabel lokal adalah sbb :

– 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, sehingga bisa dideklarasikan dengan menambahkan kata kuci auto di depan tipe-data variabel.
* Kata kunci ini bersifat opsional, biasanya disertakan sebagai penjelas saja.

**Contoh Program**

#include <stdio.h>

void fung\_1(void);

main()

{

int i = 20;

fung\_1();

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

}

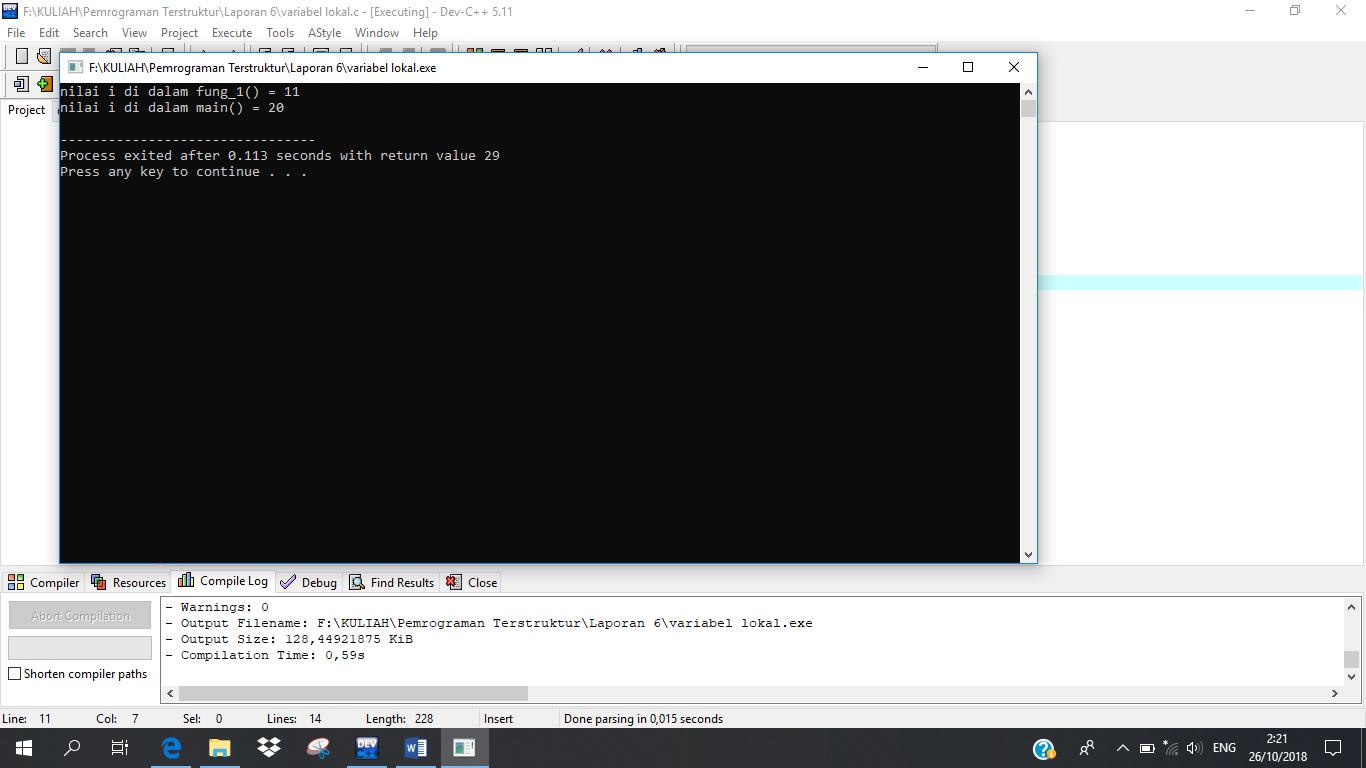
void fung\_1(void)

{

int i = 11;

printf("nilai i di dalam fung\_1() = %d\n", i);

}

**Hasil Run Program**

1. Variabel 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 variabel itu kembali dengan menambahkan kata kunci extern di depan tipe data variable

**Contoh Program**

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("");

}

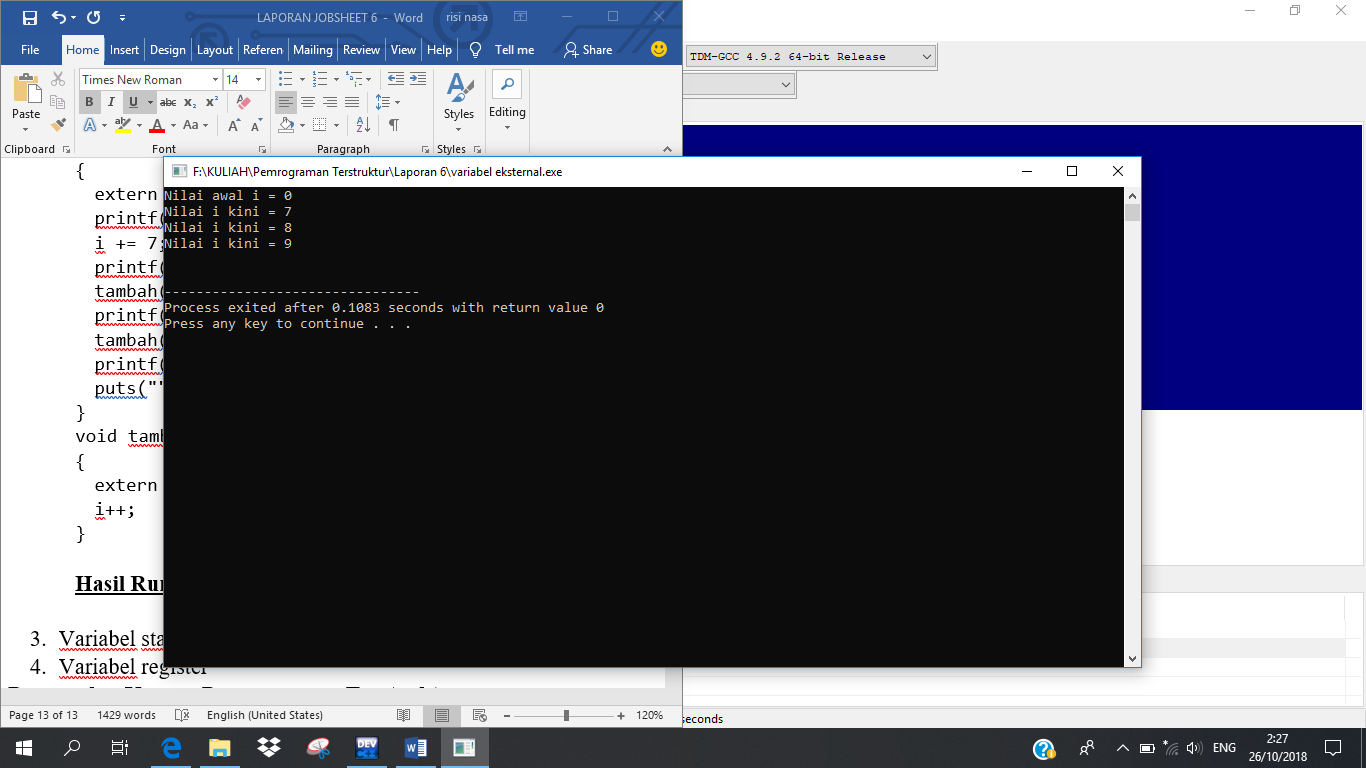
void tambah(void)

{

extern int i;

i++;

}

**Hasil Run Program**

* 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.

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);

}

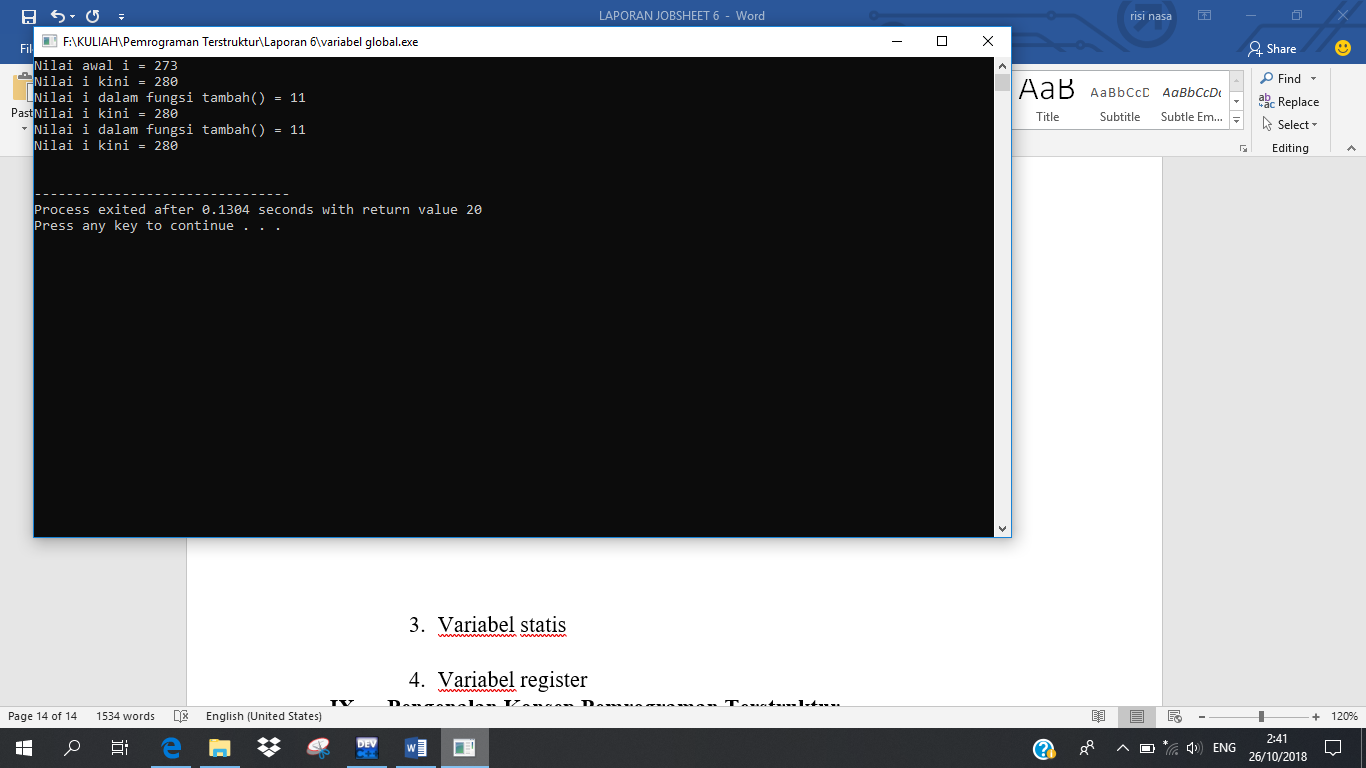
void tambah(void){

int i =10; //Variabel lokal

i++;

printf("Nilai i dalam fungsi tambah() = %d\n", i);

}

**Hasil Run Program**

1. 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.

**Contoh Program**

#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);

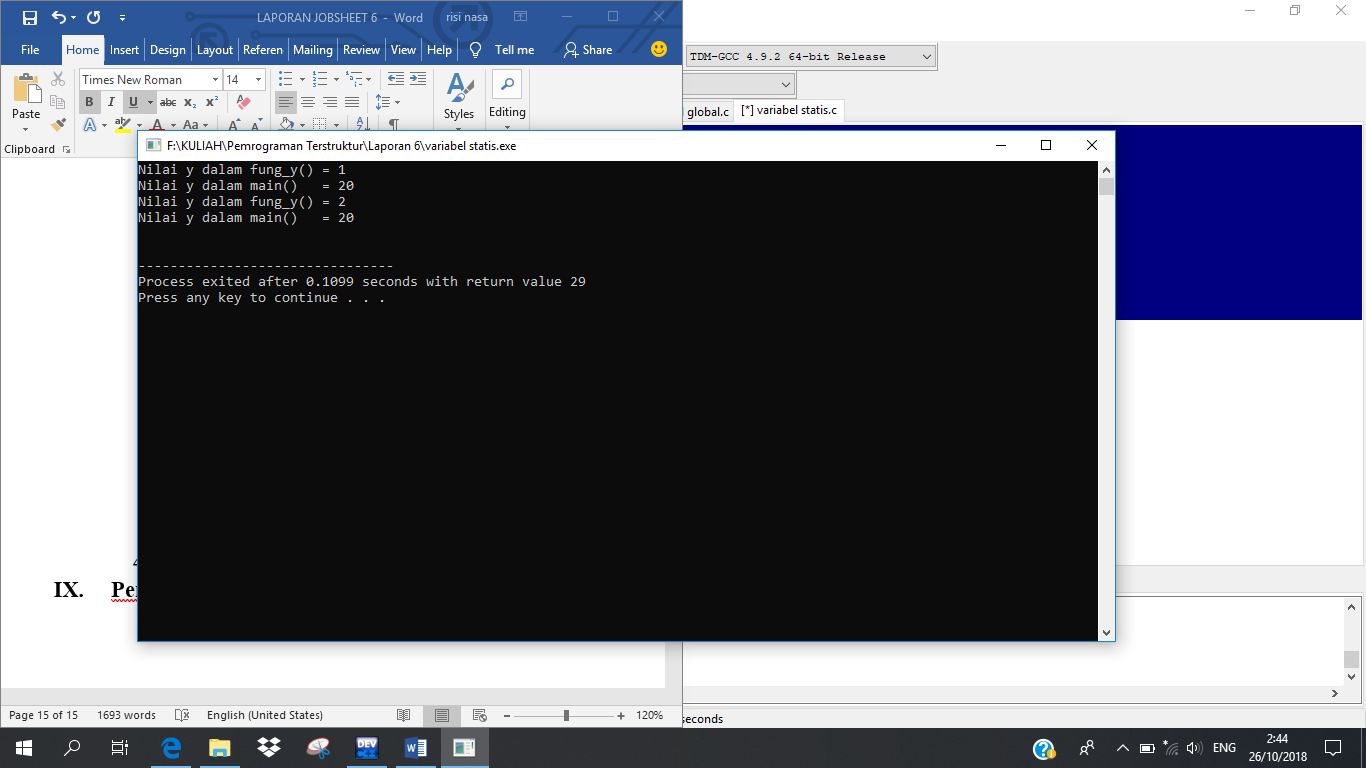
}

void fung\_y(void)

{ static int y;

y++; printf("Nilai y dalam fung\_y() = %d\n", y);

}

 **Hasil Run Program**

1. 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

**Contoh Program**

#include <stdio.h>

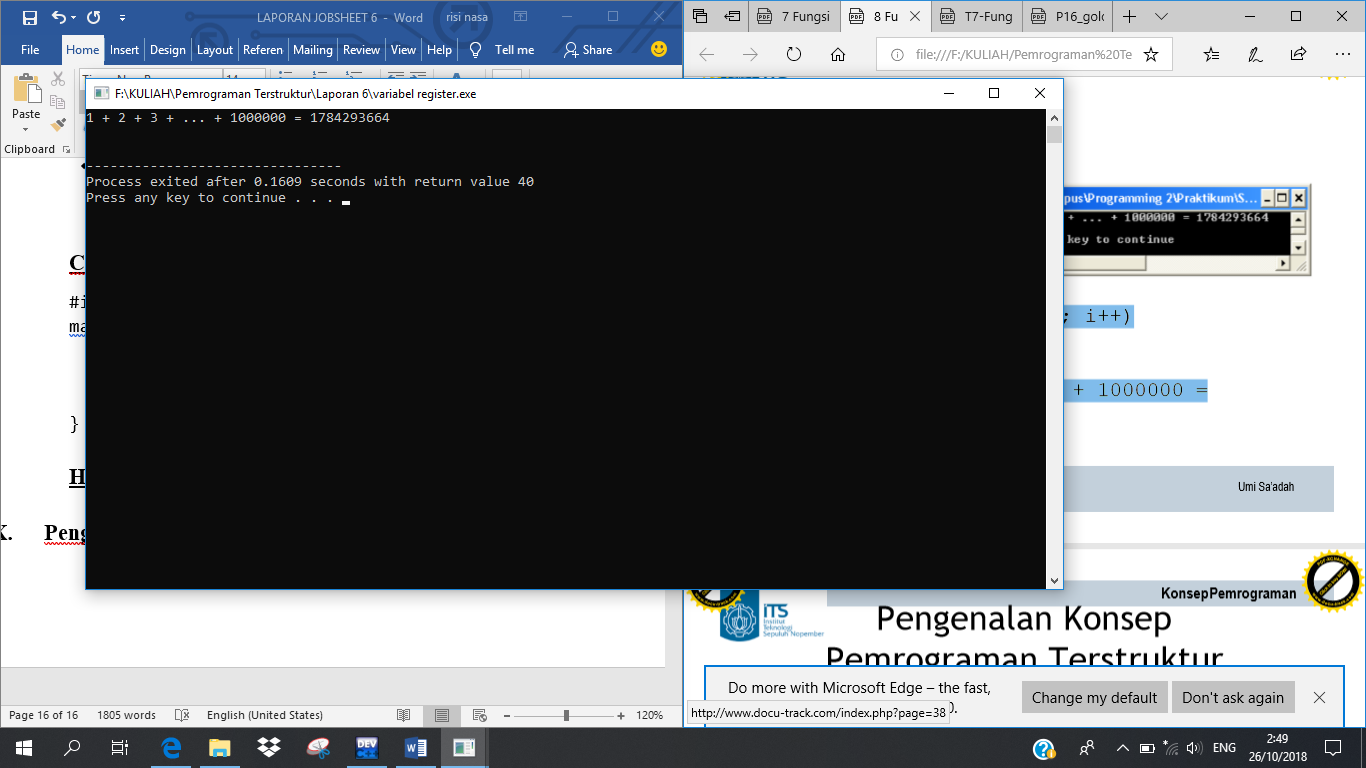
main(){

register int i; int jumlah = 0;

for(i = 1; i <= 1000000; i++) jumlah = jumlah + i;

printf("1 + 2 + 3 + ... + 1000000 = %d\n\n", jumlah);

}

**Hasil Run Program**

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 dipecahpecah 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 di halaman berikut.
* Namun sekali lagi perlu diketahui, bahwa pada C semua fungsi sebenarnya berkedudukan sederajat.

. . .

Fungsi\_a1()

Fungsi\_am()

Fungsi\_n()

Fungsi\_a()

. . .

Fungsi utama

Main()

* Fungsi main() terdiri atas fungsi\_a() sampai dengan fungsi\_n(), menegaskan bahwa dalam program dalam 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.

1. **Latihan**
2. 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.

#include <stdio.h>

int terbesar (int, int);

main()

{

int a, b, hasil=0;

printf("Masukkan bilangan ke-1 : ");

scanf("%d", &a);

printf("Masukkan bilangan ke-2 : ");

scanf("%d", &b);

terbesar (a, b);

}

int terbesar (int a, int b)

{

int temp;

if (a<b)

{

temp = b;

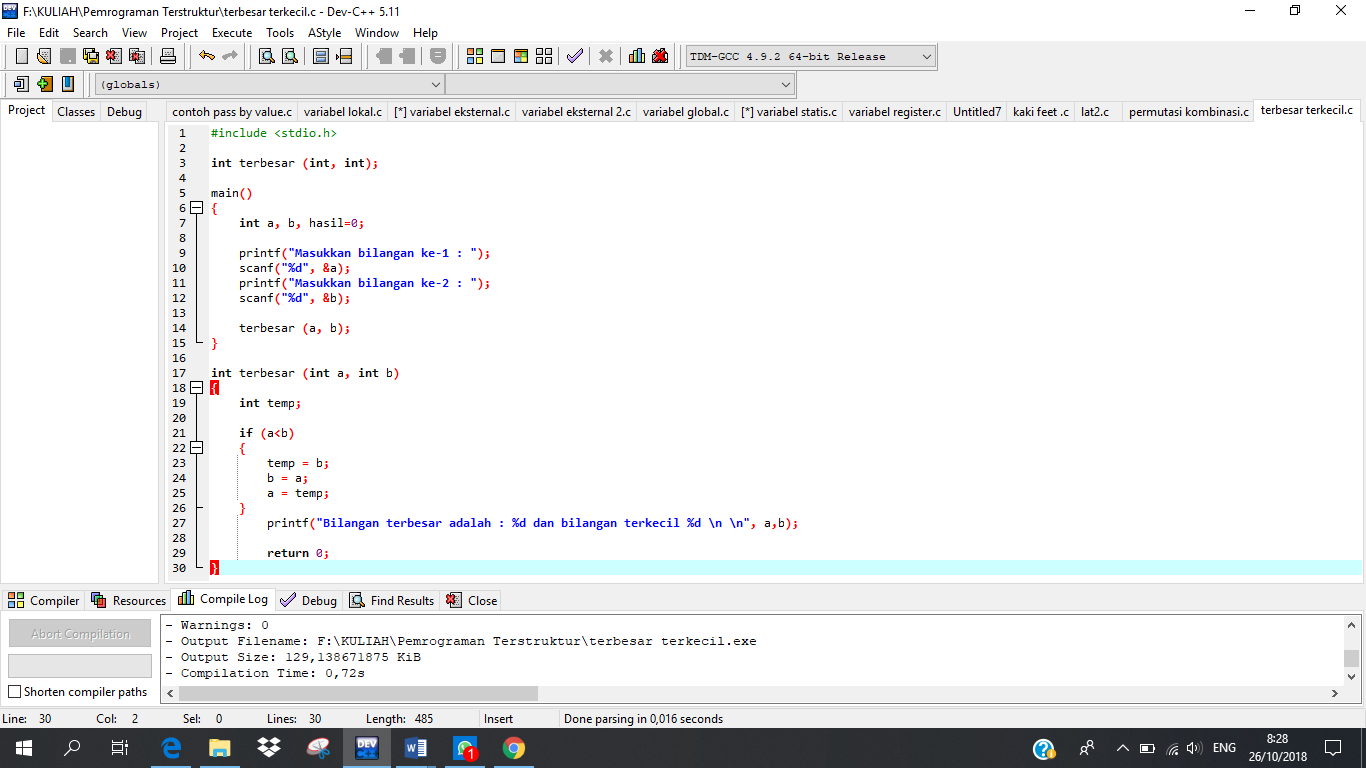
b = a;

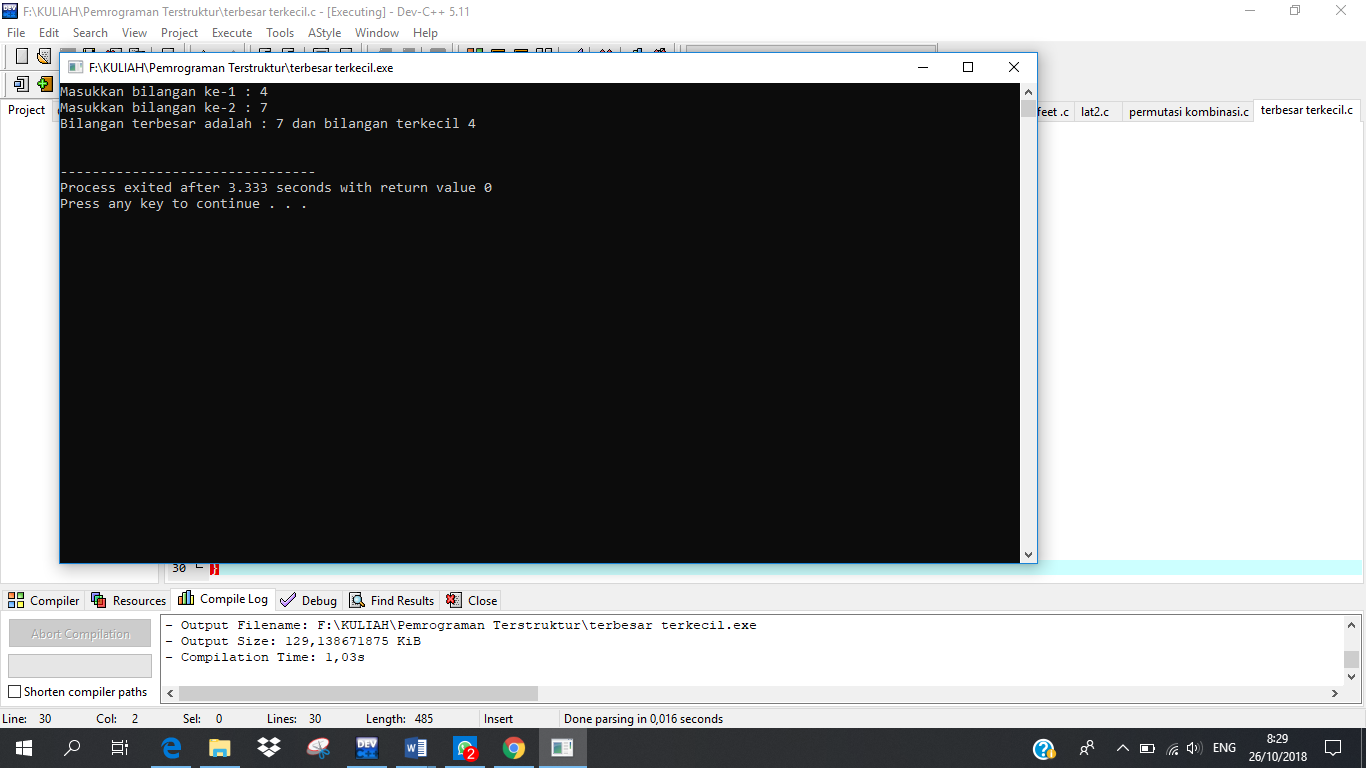
a = temp;

}

printf("Bilangan terbesar adalah : %d dan bilangan terkecil %d \n \n", a,b);

return 0;

 }

 **Hasil Run Program**

1. Buatlah suatu fungsi permutasi() dan kombinasi() untuk menghitung permutasi dan kombinasi dari suatu pasangan bilangan, yang dinyatakan dengan formula sbb:

Permutasi : P(*n,r) = n!*

*(n – r)!*

Kombinasi : C(n,r) = *n!*

*r!(n – r)!*

#include <stdio.h>

int permutasi (int, int);

int kombinasi (int, int);

main()

{

int n=0, r=0;

printf("Masukkan n : "); scanf("%d", &n);

printf("Masukkan r : "); scanf("%d", &r);

printf("\n");

printf("====PERMUTASI====\n");

printf("Hasil permutasi : %d \n\n", permutasi (n, r));

printf("====KOMBINASI====\n");

printf("Hasil kombinasi : %d \n", kombinasi (n, r));

}

int permutasi (int n, int r)

{

int hasil1=1, hasil2=1, hasil3, x, y, z;

//n!//

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

{

hasil1 \*= x;

}

printf("n! = %d\n", hasil1);

//(n-r)!//

z=n-r;

for (y=1; y<=z; y++)

{

hasil2 \*= y;

}

printf("(n-r)! = %d\n", hasil2);

hasil3 = hasil1/hasil2;

return (hasil3);

}

int kombinasi (int n, int r)

{

int hasil1=1, hasil2=1, hasil3=1, hasil4, x, y, z, a;

//n!//

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

{

hasil1 \*= x;

}

printf("n! = %d\n", hasil1);

//r!//

for (a=1; a<=r; a++)

{

hasil2 \*= a;

}

printf("r! = %d\n", hasil2);

//(n-r)!//

z=n-r;

for (y=1; y<=z; y++)

{

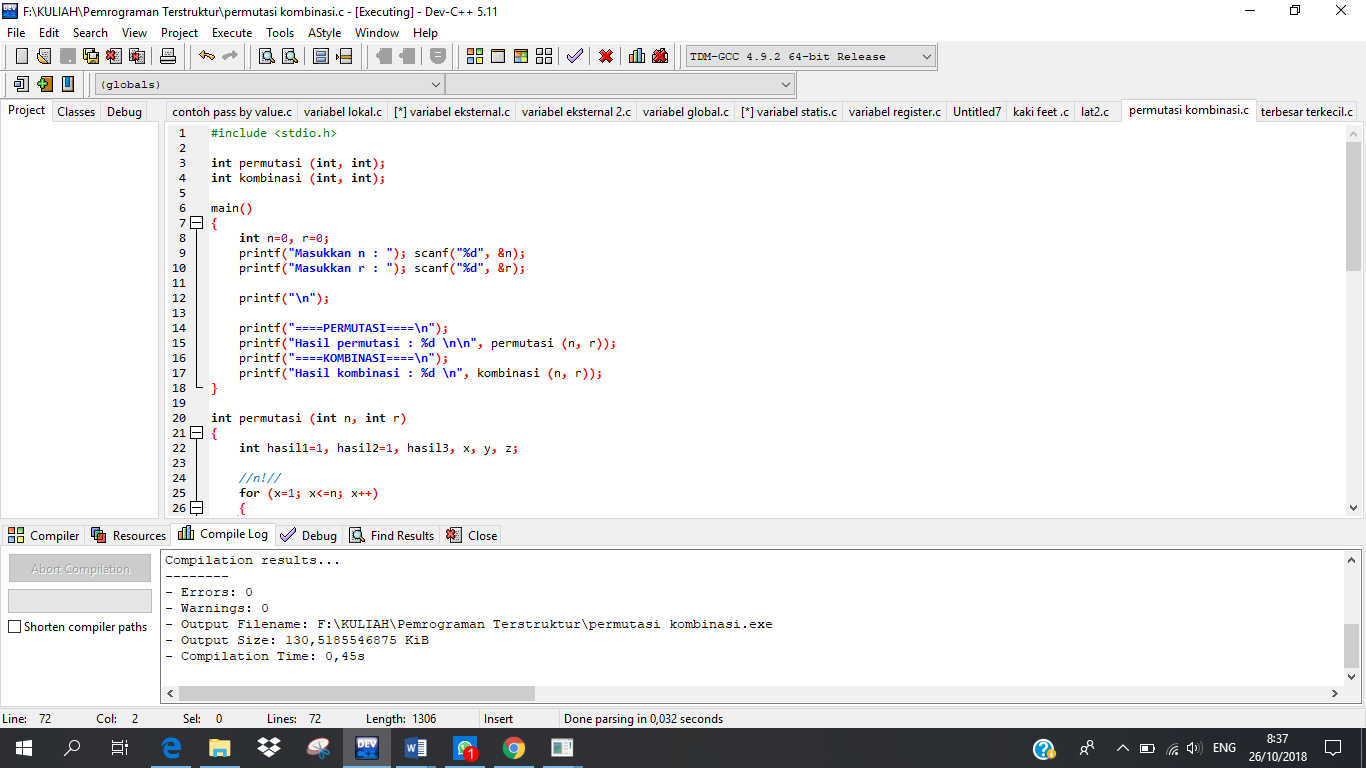
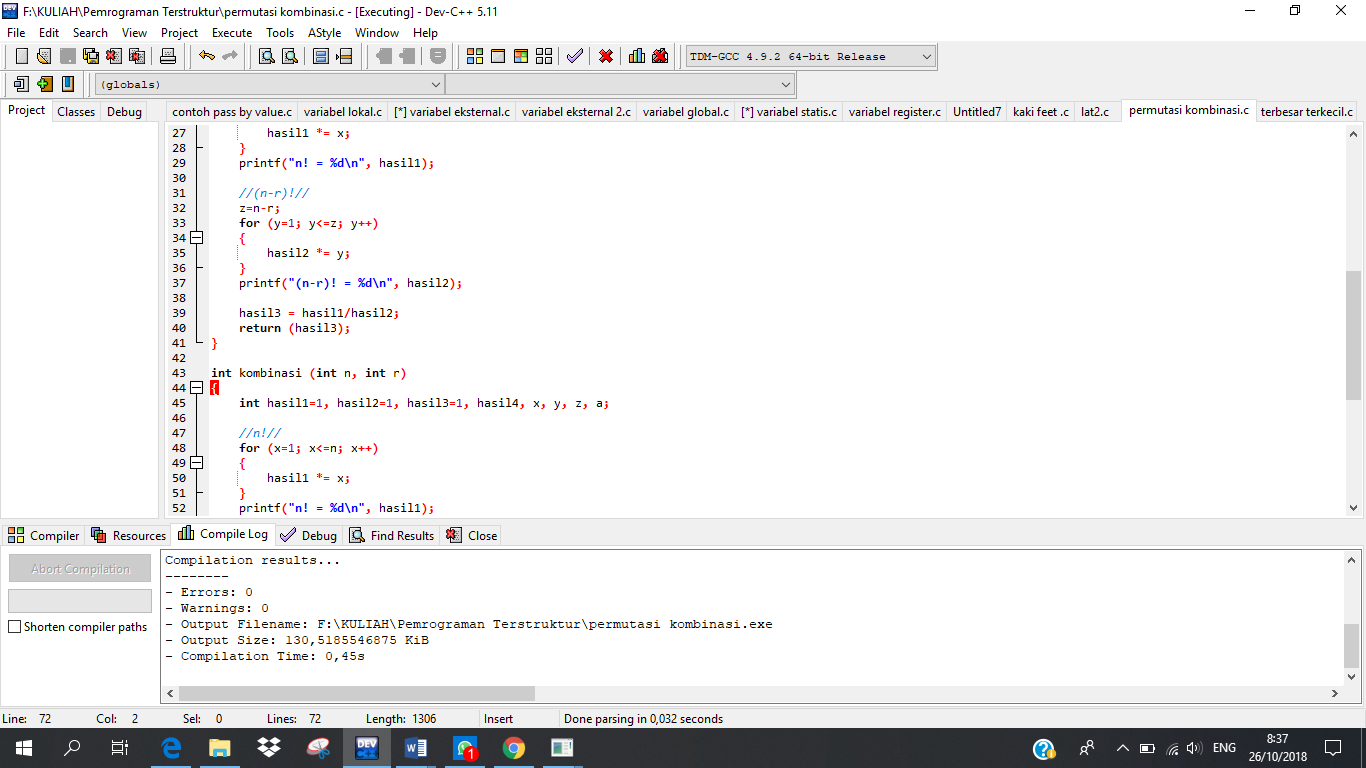
hasil3 \*= y;

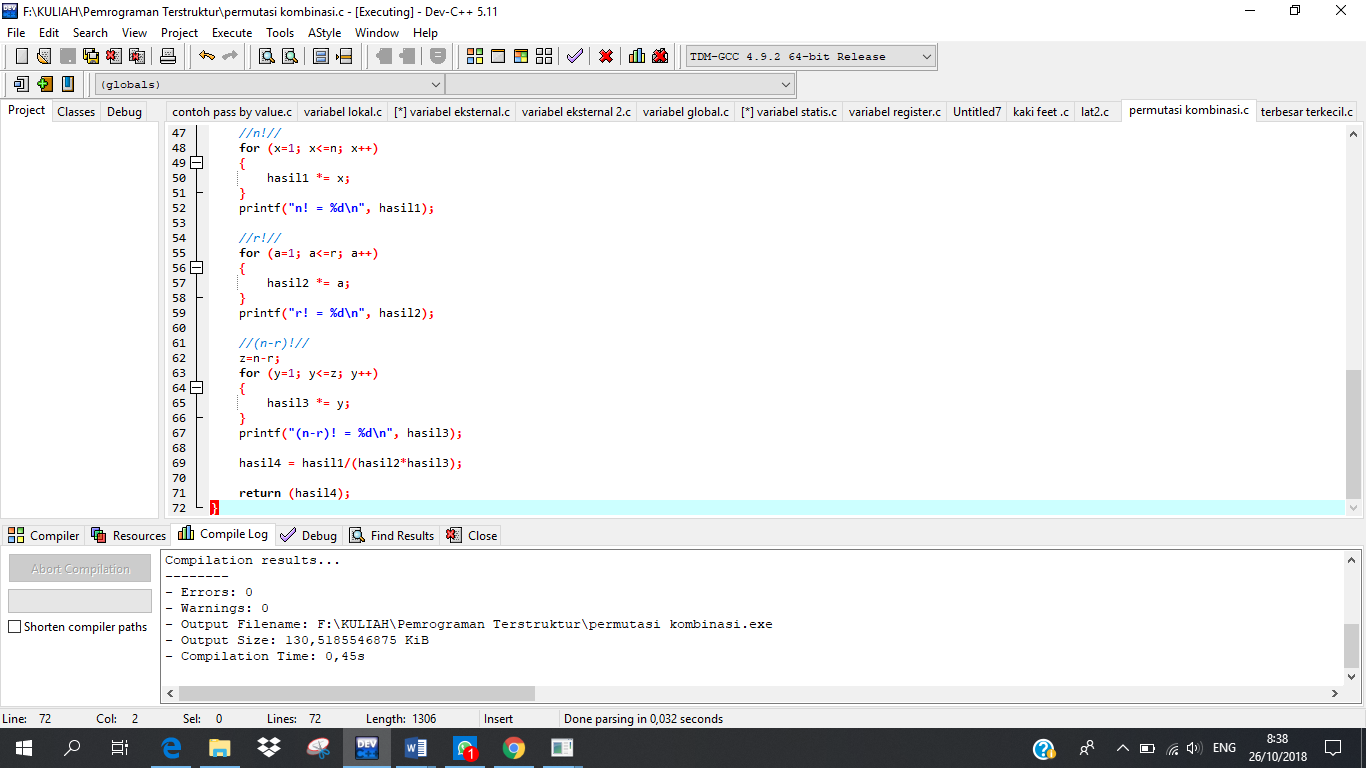
}

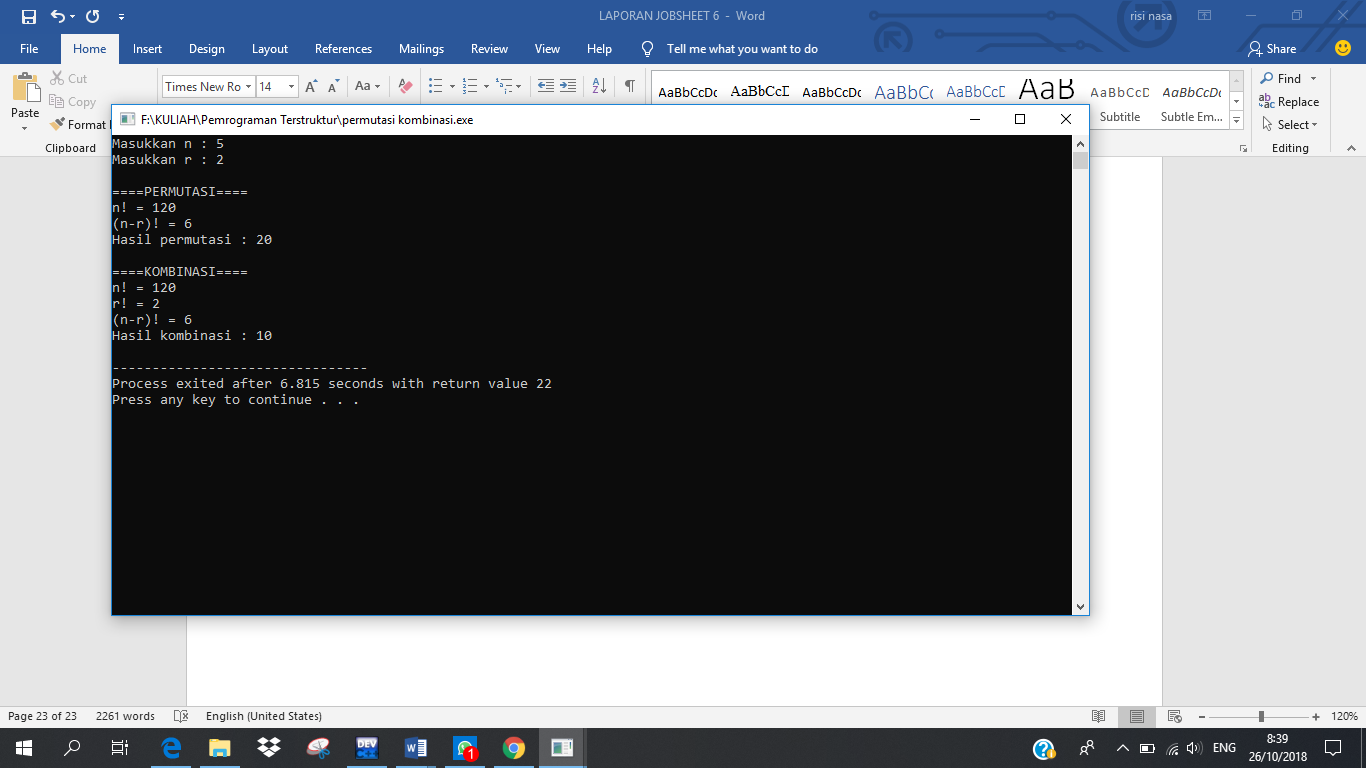
printf("(n-r)! = %d\n", hasil3);

hasil4 = hasil1/(hasil2\*hasil3);

return (hasil4);

}



 **Hasil Run Program**

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

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. 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.

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.

**Program**

#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)

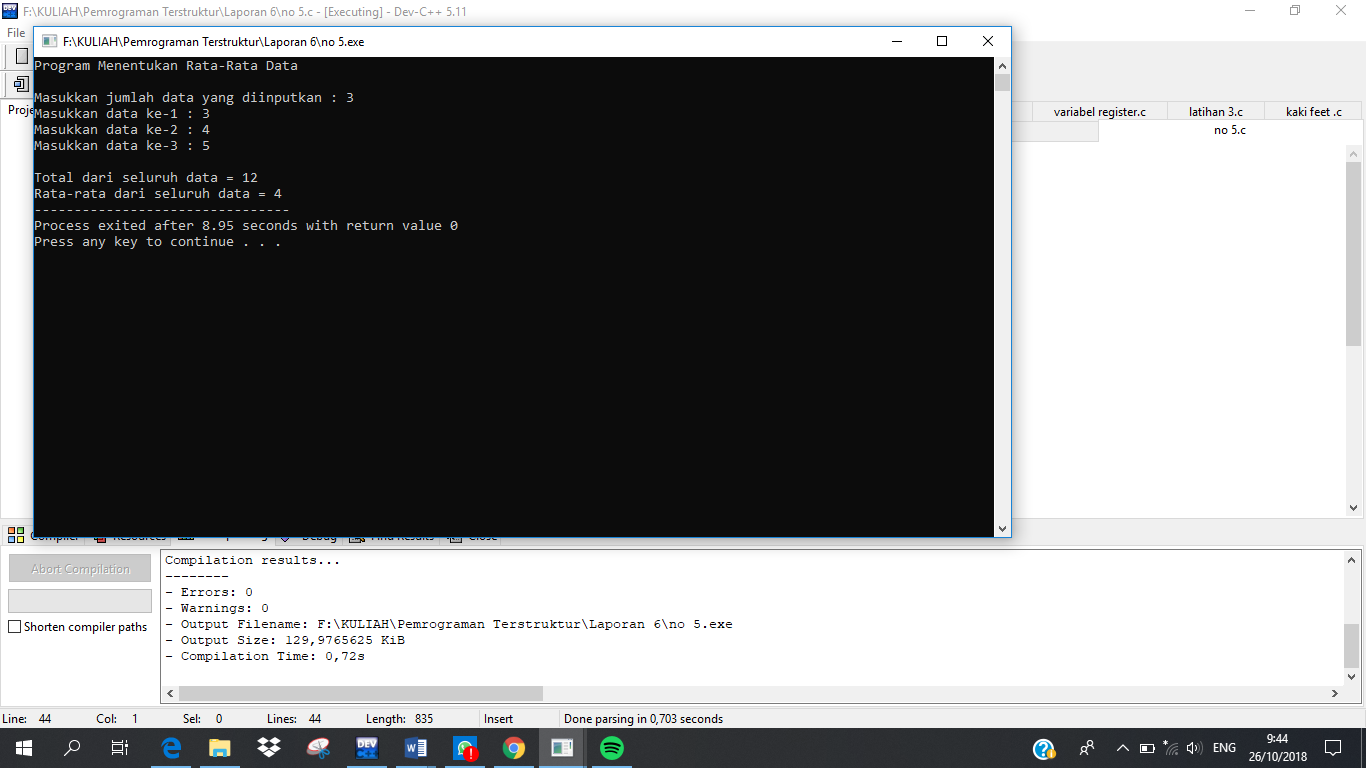
{

float akhir;

akhir = sum / a;

return akhir;

}

**Hasil Run Program**

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

**Program**

#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)

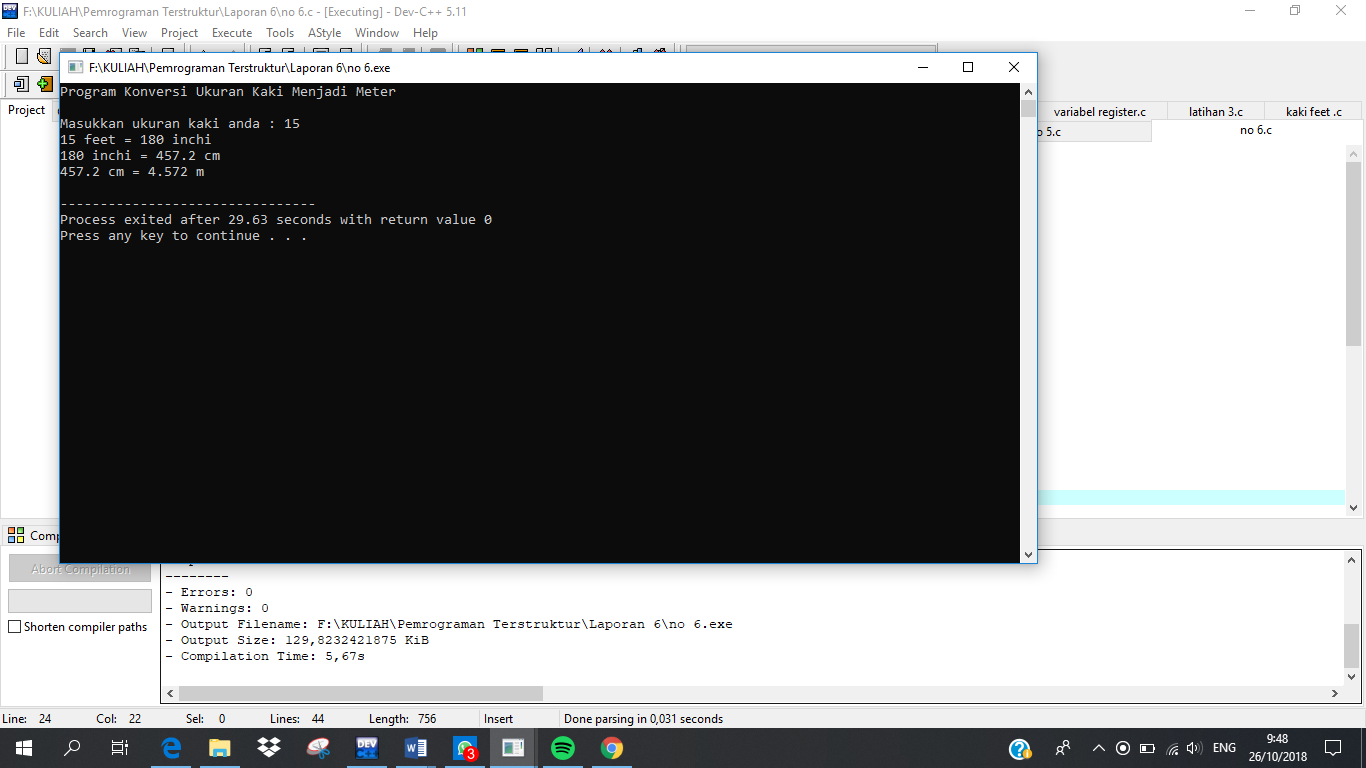
{

float d;

d = c / 100;

return d;

}

**Hasil Run Program**

1. **Kesimpulan**
2. Fungsi adalah suatu bagian dari program yang dirancang untuk melaksanakan tugas tertentu dan letaknya dipisahkan dari program yang menggunakannya.
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

1. 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