**LAPORAN PRAKTIKUM 6**

**PEMROGRAMAN TERSTRUKTUR**

**FUNGSI 2**



Disusun Oleh:

Alfian Karim Fathur Rahman

3.34.18.1.04

IK-1B

**PROGRAM STUDI D3 TEKNIK INFORMATIKA**

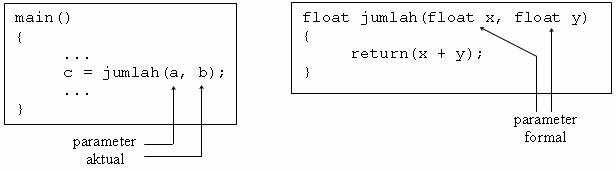
**JURUSAN TEKNIK ELEKTRO**

**POLITEKNIK NEGERI SEMARANG**

**2018/2019**

1. **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.



* 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

1. **Pengiriman Parameter secara p*ass 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 tersebut dipanggil

Contoh pass by value

#include <stdio.h>

void tukar(int, int);

main(){

int a=5, b=3;

printf("Sebelum pemanggilan fungsi\n");

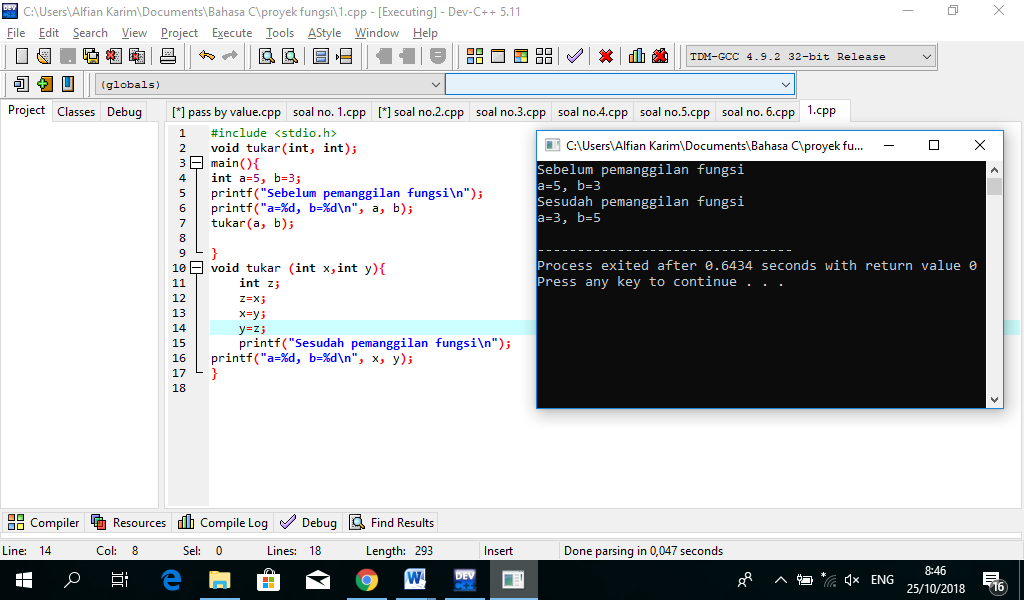
printf("a=%d, b=%d\n", a, b);

tukar(a, b);

printf("Sesudah pemanggilan fungsi\n");

printf("a=%d, b=%d\n", a, b);

}



1. **Penggolongan Variabel berdasarkan Kelas 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

1. Variabel Lokal (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 :

#include <stdio.h>

void fung\_1(void);

main(){

int i = 20;

fung\_1();

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

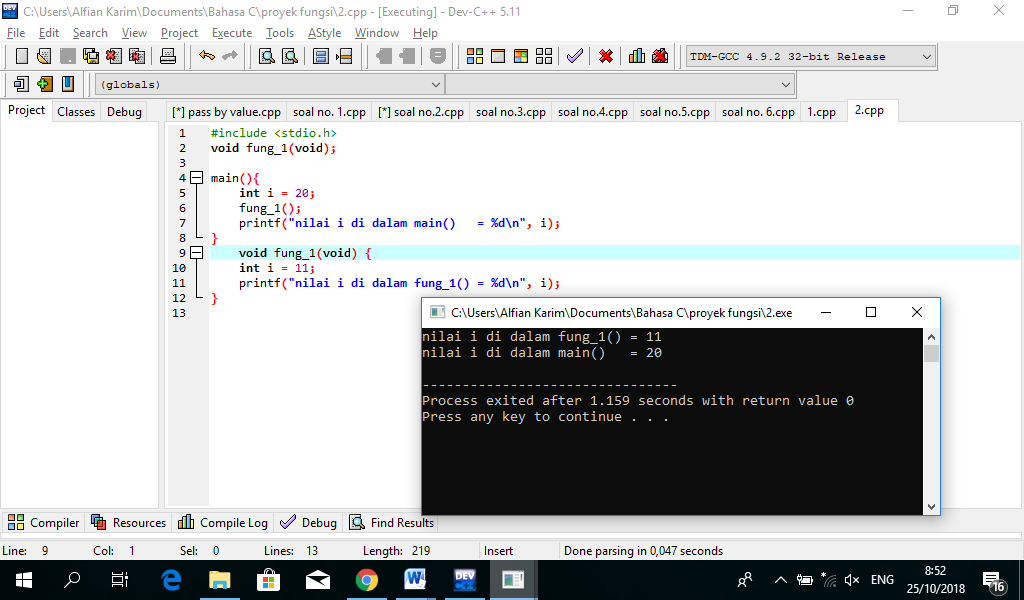
}

void fung\_1(void) {

int i = 11;

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

}



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

Contoh :

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("");

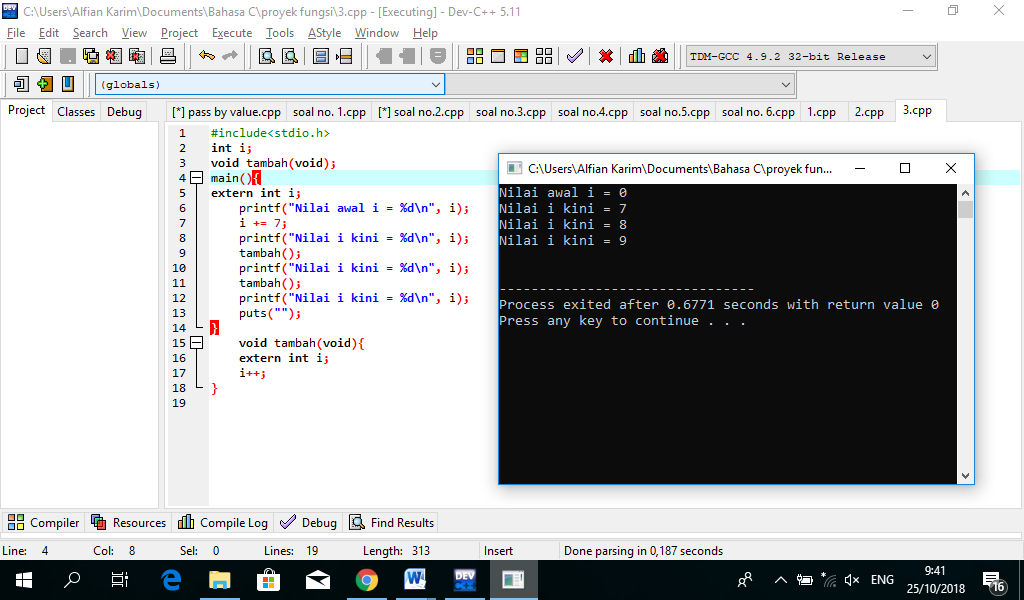
}

void tambah(void){

extern int i;

i++;

}



* 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.

#include<stdio.h>

int i = 273; //variabel eksternal

void tambah(void);

main(){

extern int i; //variabel eksternal

printf("Nilai awal i = %d\n", i);

i += 7; //i=i+7 i=280;

printf("Nilai i kini = %d\n", i);

tambah();

printf("Nilai i kini = %d\n", i);

tambah();

printf("Nilai i kini = %d\n\n", i);

}

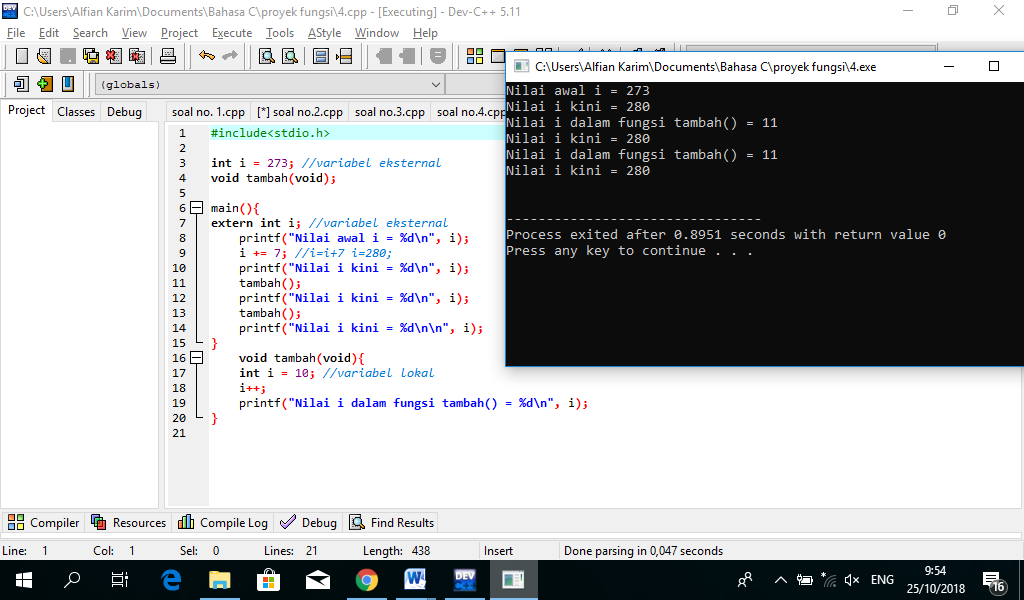
void tambah(void){

int i = 10; //variabel lokal

i++;

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

}



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 :

#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);

}

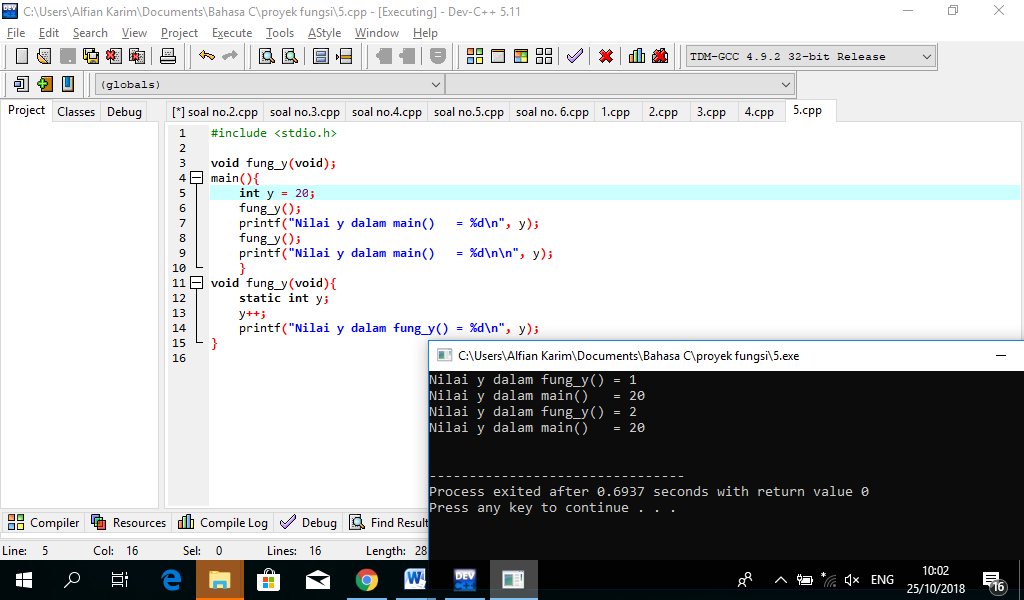
void fung\_y(void){

static int y;

y++;

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

}



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 :

#include <stdio.h>

main(){

register int i;

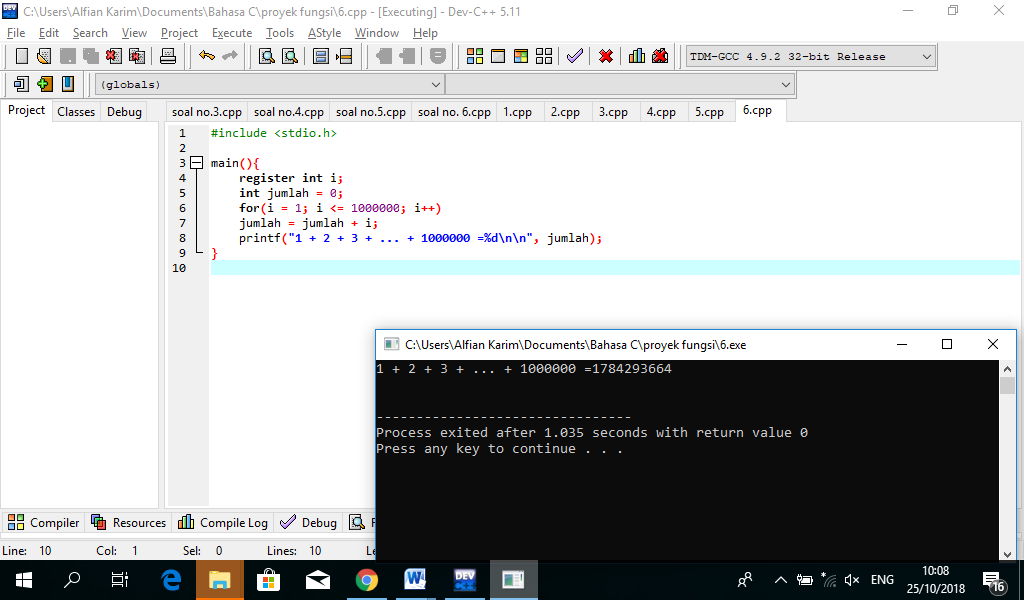
int jumlah = 0;

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

jumlah = jumlah + i;

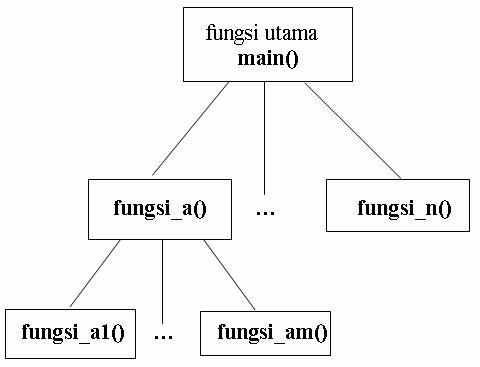
printf("1 + 2 + 3 + ... + 1000000 =%d\n\n", jumlah);

}



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



* 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 dan terkecil 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 tersebut

Jawab :

#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;

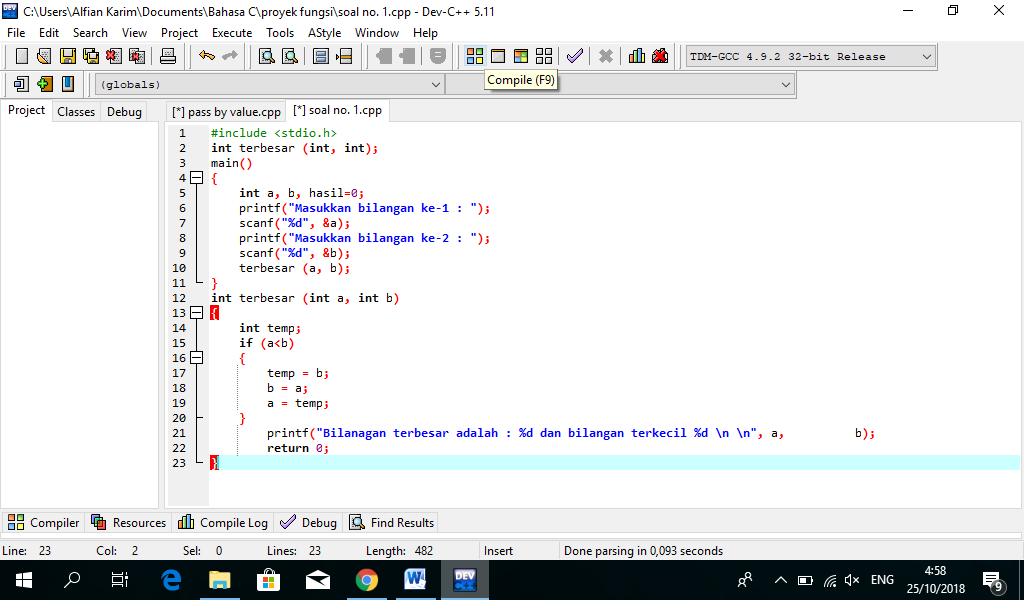
a = temp;

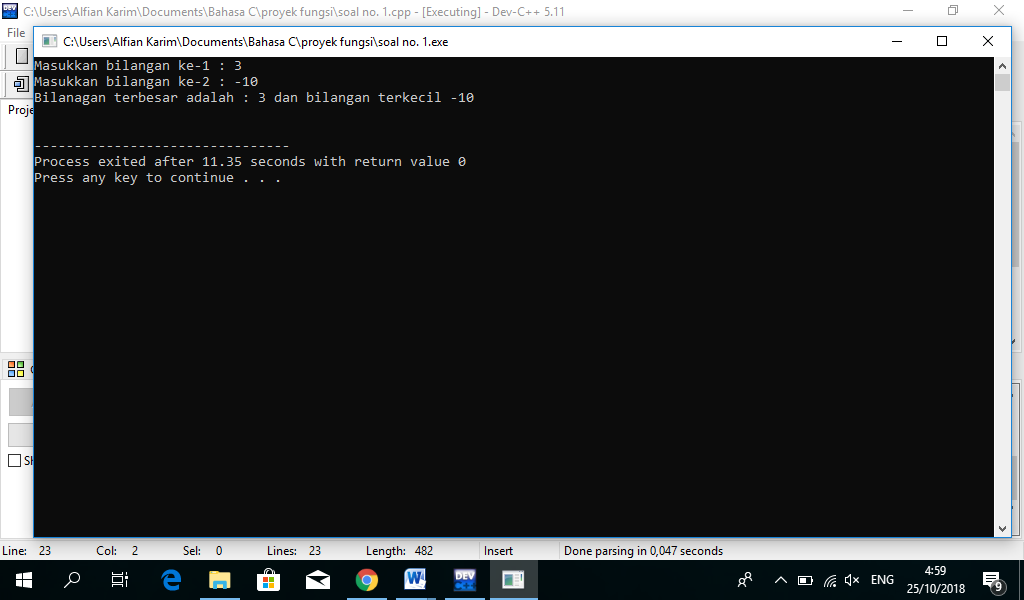
}

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

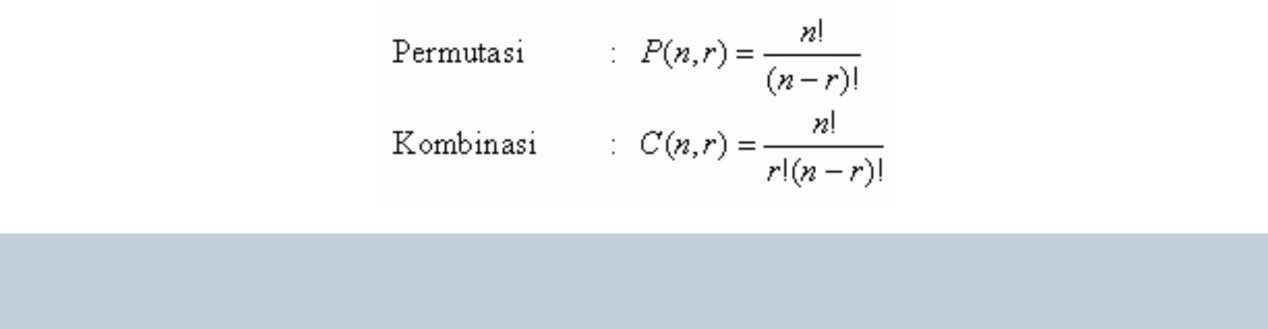
return 0;

}

­



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



/\*MENCARI PERMUTASI DAN KOMBINASI\*/

#include <stdio.h>

long int faktorial(int);

int permutasi(int, int);

int kombinasi(int, int);

main (){ int n, r;

printf("MENCARI PERMUTASI DAN KOMBINASI\n");

printf("RUMUS Permutasi : P (n, r) = n! / (n-r)!\n");

printf("RUMUS Kombinasi : C(n, r)=n!/r!(n-r)!\n");

printf("\nMasukkan nilai n = ");

scanf("%d", &n);

printf("\nMasukkan nilai r = ");

scanf("%d", &r);

printf("\n%d data dipermutasikan %d = %d\n", n, r, permutasi(n,r));

printf("\n%d data dikombinasikan %d = %d\n\n", n, r, kombinasi(n,r));

}

long int faktorial(int x)

{ int fak, i;

fak = 1;

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

fak = fak \* i;

return(fak);}

int permutasi(int n, int r){

int hasil, temp;

temp = n - r;

hasil = faktorial(n) / faktorial(temp);

return(hasil);}

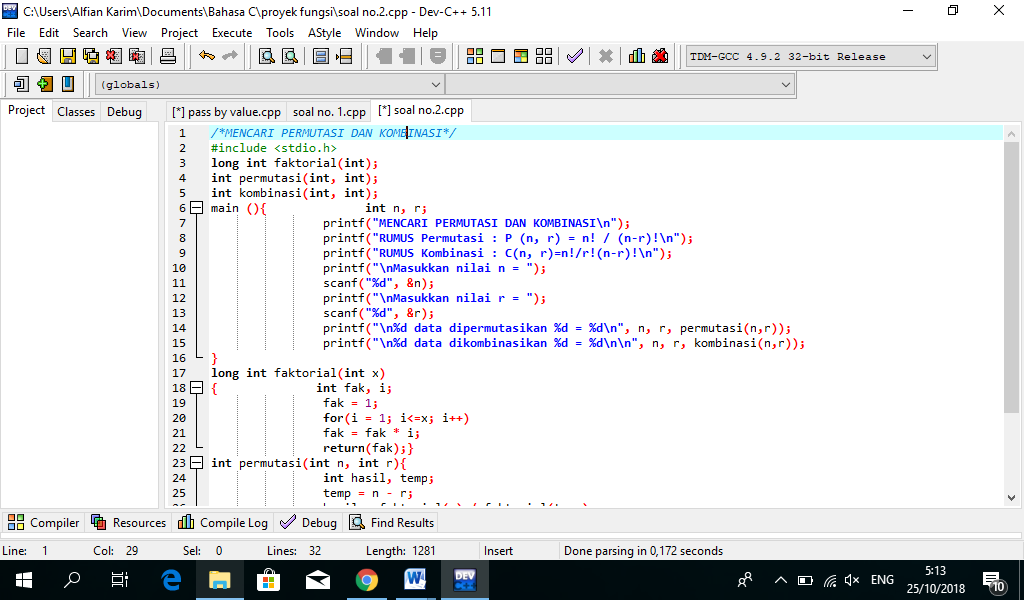
int kombinasi(int n, int r)

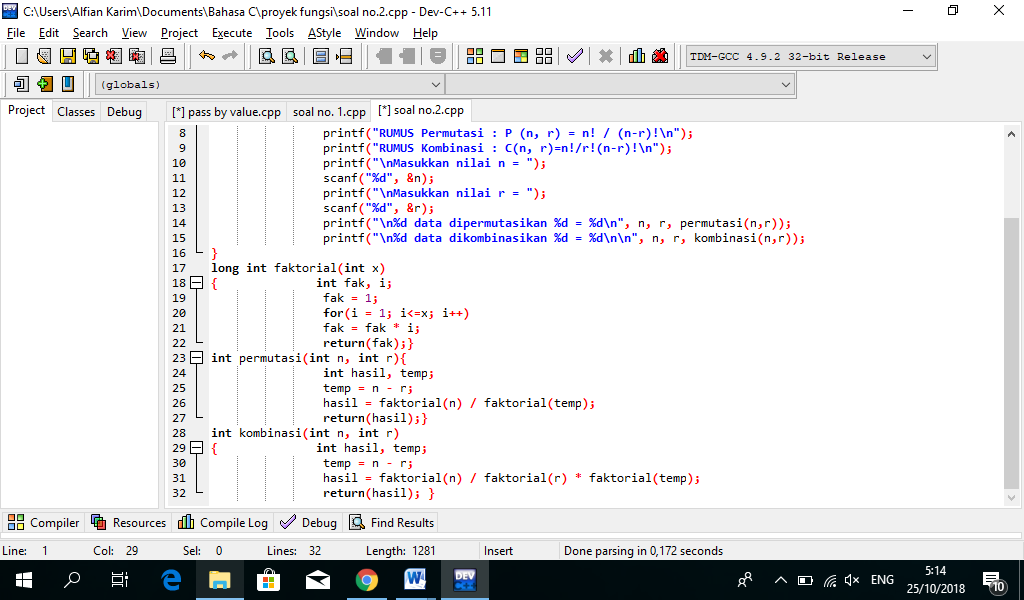
{ int hasil, temp;

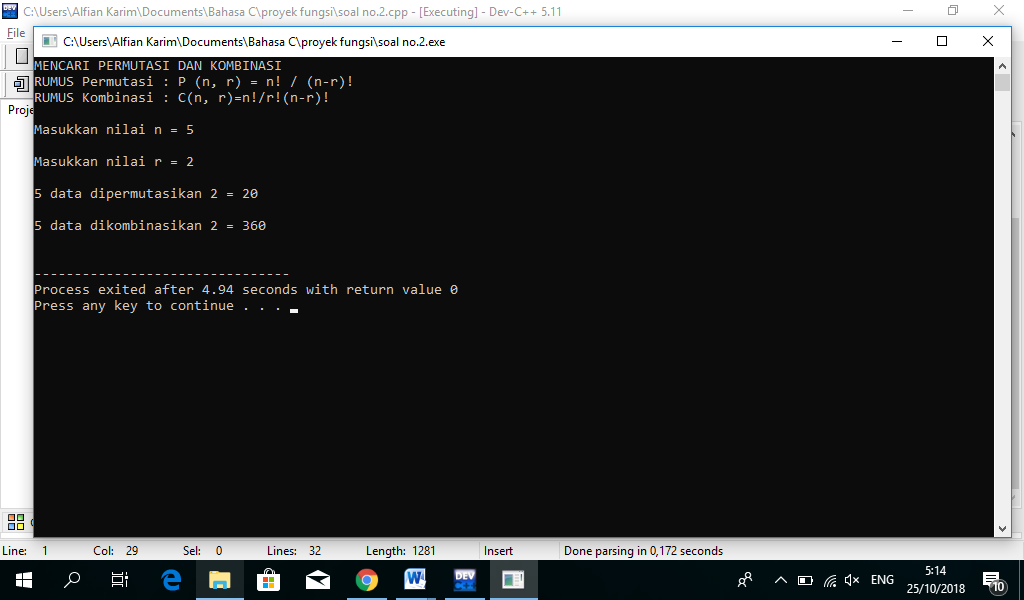
temp = n - r;

hasil = faktorial(n) / faktorial(r) \* faktorial(temp);

return(hasil); }







1. #include <stdio.h>

int OddEventTest (int);

main()

{

int a,hasil;

a=5;

hasil=OddEventTest (a);

printf("a=%d\t hasil=%d",a,hasil);

}

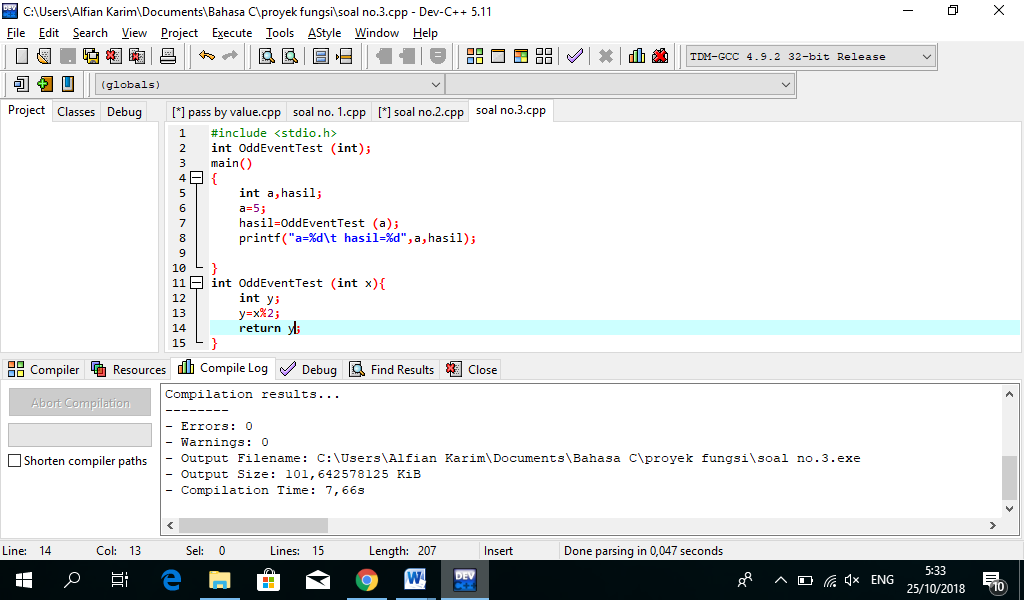
int OddEventTest (int x){

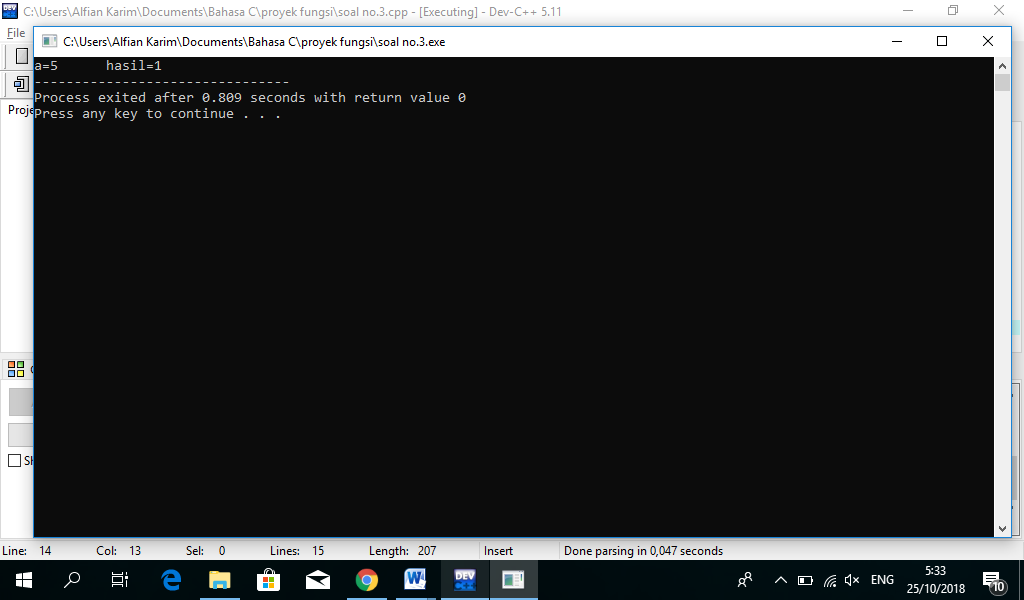
int y;

y=x%2;

return y;

}





1. #include<stdio.h>

void fungsi\_a(void);

void fungsi\_b(void);

int x=20;

main(){

x+=2;

fungsi\_a();

fungsi\_a();

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

}

void fungsi\_a(void){

//int x;

int static x = 5;

x++;

printf("nilai x dalam fungsi\_a()=%d\n",x);

fungsi\_b();

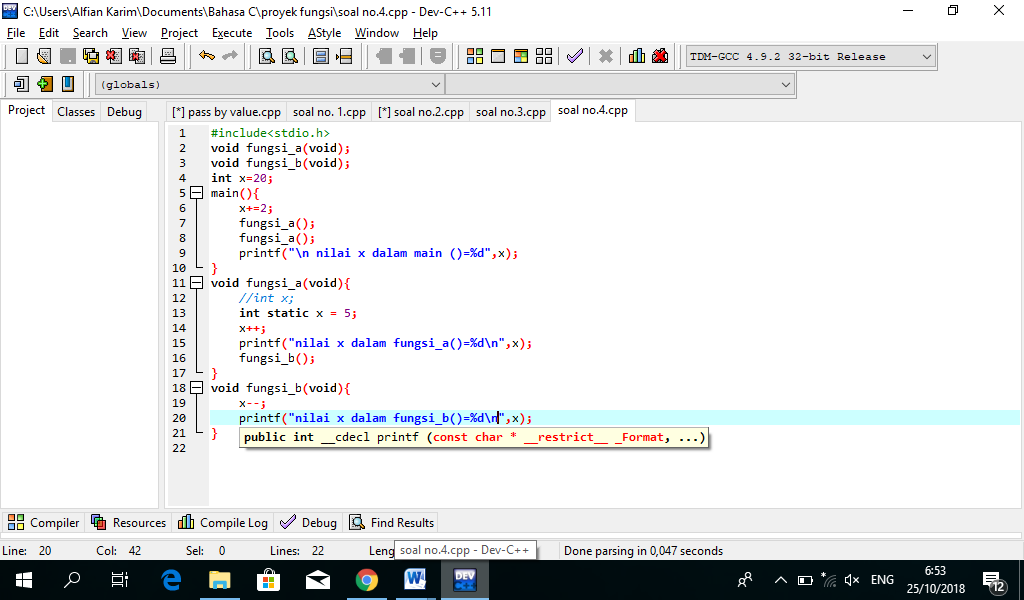
}

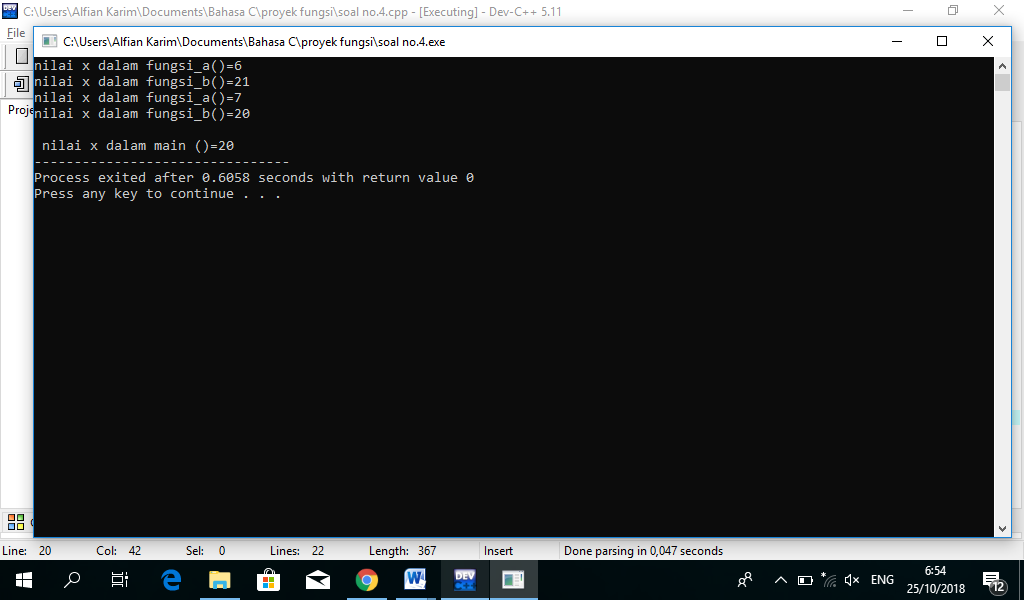
void fungsi\_b(void){

x--;

printf("nilai x dalam fungsi\_b()=%d\n",x);

}



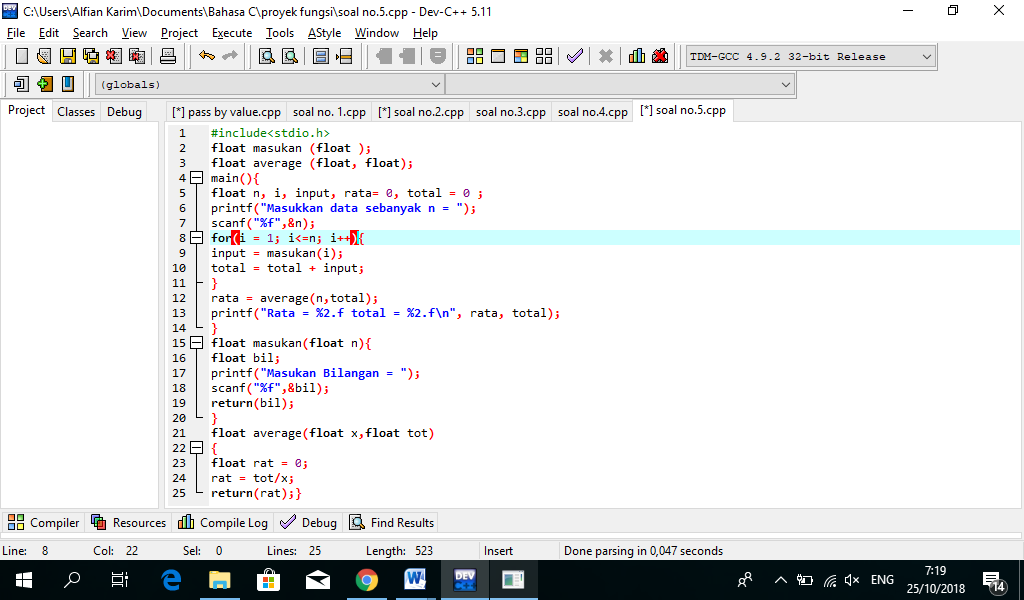


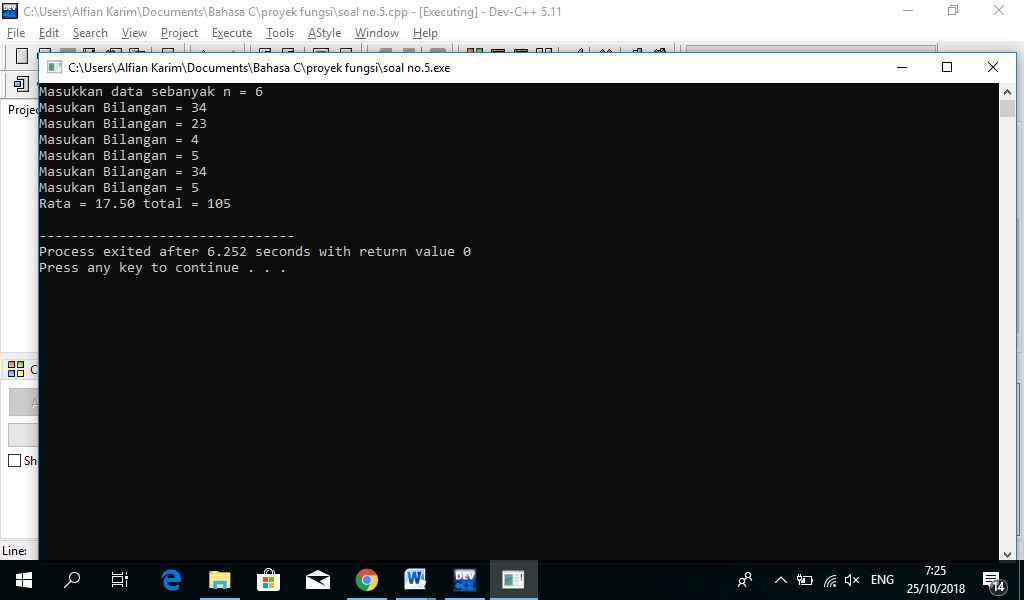
1. Definisikanlah fungsi main(), masukan() dan average(), sebagai berikut :

– 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.





1. Definisikanlah fungsi-fungsi sebagai berikut :

– Fungsi f\_to\_i() untuk mengubah ukuran dari satuan kaki *(feet)* ke inci

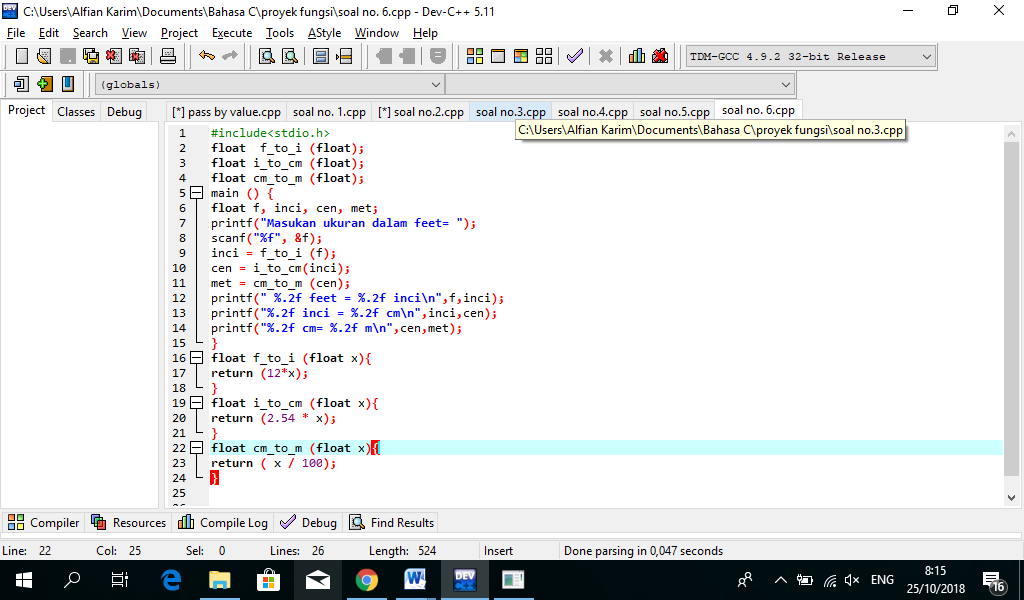
– Fungsi i\_to\_cm() untuk mengubah ukuran dari satuan inci ke centimeter

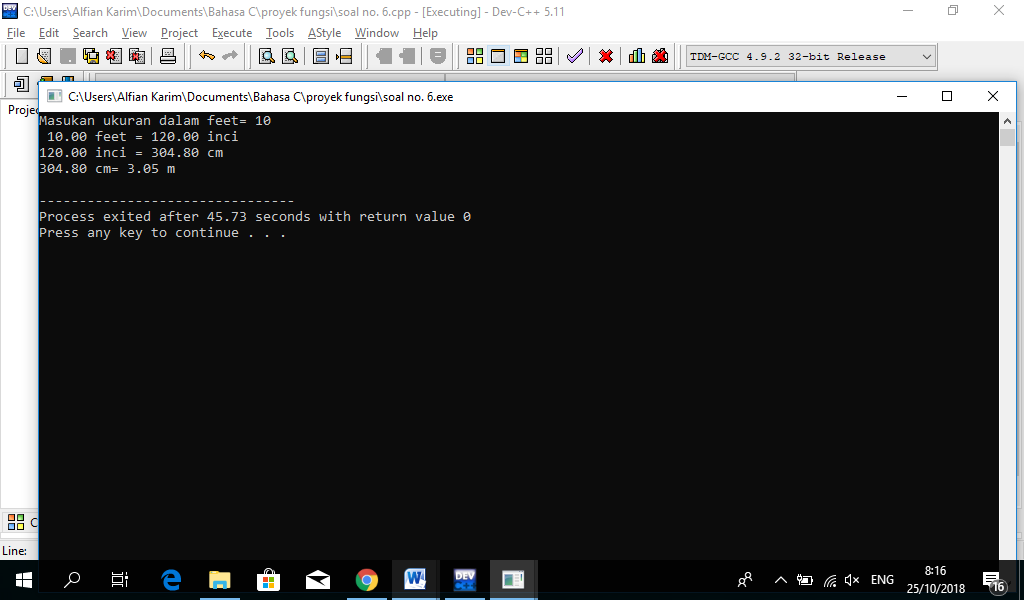
– Fungsi c\_to\_m()untuk mengubah ukuran dari satuan centimeter ke meter

– 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 meter





Kesimpulan :