**PRAKTIKUM 15**

**Fungsi : Passing Parameter by Value & by Reverence**

1. **DASAR TEORI**

Pemanggilan dengan nilai merupakan cara yang dipakai untuk seluruh fungsi buatan yang telah dibahas pada praktikum sebelumnya. Pada pemanggilan dengan nilai, nilai dari parameter aktual akan disalin ke parameter formal. Dengan cara ini nilai parameter aktual tidak bisa dirubah sekalipun nilai parameter formal berubah. Untuk lebih jelasnya lihat pada fungsi **tukar()** pada contoh berikut ini.

/\* File program : tukar1.c

Untuk melihat pengaruh pemanggilan nilai pada fungsi untuk penukaran dua data \*/

#include <stdio.h>

void tukar (int, int);

main()

{

int a = 88, b = 77;

printf("Nilai sebelum pemanggilan fungsi\n");

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

tukar(a,b);

printf("\nNilai setelah pemanggilan fungsi\n");

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

}

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);

}

Tampak bahwa sekeluarnya dari pemanggilan fungsi **tukar()**, variabel **a** dan **b** (yang dilewatkan ke fungsi **tukar()** tidak berubah, walaupun pada fungsi **tukar()** telah terjadi penukaran antara parameter **x** dan **y** . Mengapa hal ini bisa terjadi ? Sebab **x** hanyalah salinan dari **a** dan **y** adalah salinan dari **b**. Pada saat pemanggilan fungsi, maka :

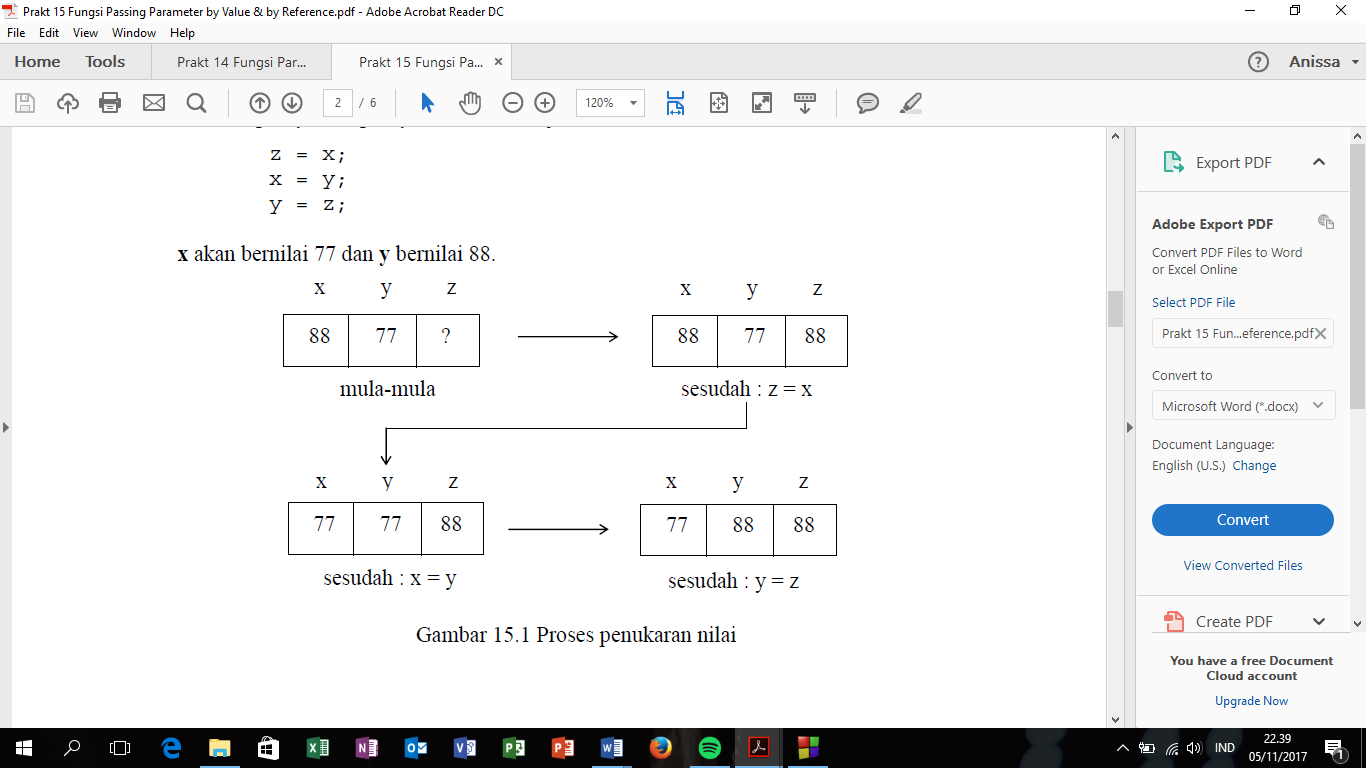
* **x** bernilai 88 (nilai **a**)
* **y** bernilai 77 (nilai **b**)

Sesudah pernyataan-pernyataan berikut dijalankan, maka :

z = x;

x = y;

y = z;

**x** akan bernilai 77 dan **y** bernilai 88.

Gambar tersebut menjelaskan bahwa **a** dan **b** tidak berubah. Yang berubah hanyalah parameter **x** dan **y**.

Pemanggilan dengan referensi *(call by reference)* merupakan upaya untuk melewatkan alamat dari suatu variabel ke dalam fungsi. Cara ini dapat dipakai untuk mengubah isi suatu variabel di luar fungsi dengan pelaksanaan pengubahan dilakukan di dalam fungsi. Sebagai contoh perhatikan program **tukar2.c** yang merupakan modifikasi dari **tukar1.c**. Perubahan yang pertama terletak dalam definisi fungsi, yang kini berupa

void tukar(int \*px, int \*py)

{

int z;

z = \*px;

\*px = \*py;

\*py = z;

printf("\nNilai di akhir fungsi tukar()\n");

printf("x = %d y = %d\n", \*px, \*py);

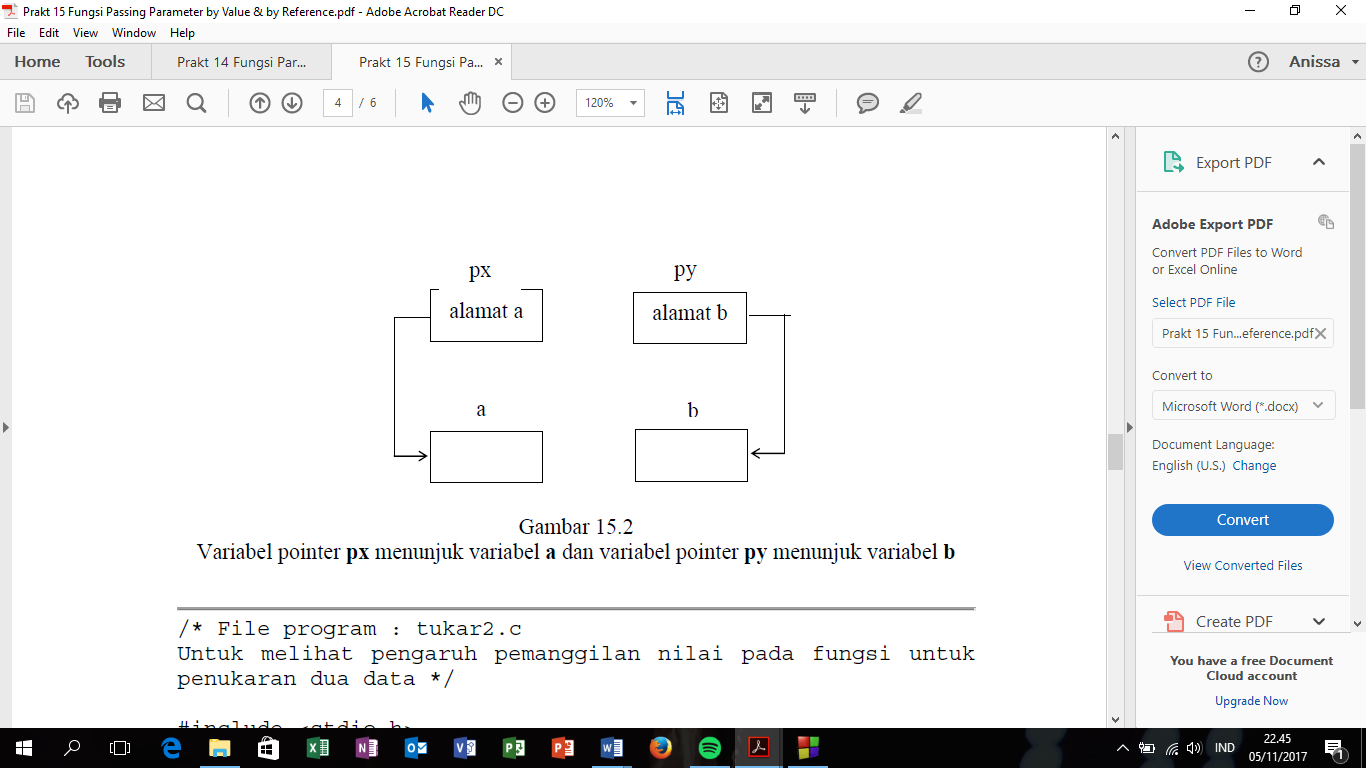
}

Adapun perubahan dalam parameter aktualnya menjadi :

tukar(&a,&b); //alamat a dan alamat b

Dalam deklarasi parameter

int \*px, int \*py

menyatakan bahwa **px** dan **py** adalah suatu variabel pointer. Yang dimaksudkan sebagai variabel pointer adalah suatu variabel yang menunjuk ke variabel lain. Lebih jelasnya, variabel pointer berisi alamat dari variabel lain. Adapun pada pemanggilan fungsi, **&a** dan **&b** masing-masing berarti "alamat a" dan "alamat b". Dengan pemanggilan seperti ini, hubungan antara variabel pointer **px** dan **py** dengan variabel **a** dan **b** adalah seperti ditunjukkan pada gambar di bawah ini Dalam hal ini, **px** dikatakan menunjuk variabel **a** dan **py** menunjuk variabel **b**. Ilustrasi penempatan di memory untuk masing-masing variabel

Variabel pointer **px** menunjuk variabel **a** dan variabel pointer **py** menunjuk variabel **b**

/\* File program : tukar2.c

Untuk melihat pengaruh pemanggilan nilai pada fungsi untukpenukaran dua data \*/

#include <stdio.h>

void tukar (int \*px, int \*py); //prototype fungsi

main()

{

int a= 88,b = 77;

printf("Nilai sebelum pemanggilan fungsi\n");

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

tukar(&a,&b); //alamat a dan alamat b

printf("\nNilai setelah pemanggilan fungsi\n");

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

}

void tukar(int \*px, int \*py)

{

int z;

z = \*px;

\*px = \*py;

\*py = z;

printf("\nNilai di akhir fungsi tukar()\n");

printf("x = %d y = %d\n", \*px, \*py);

}

Setelah **px** menunjuk **a** dan **py** menunjuk **b**, proses penukaran isi **a** dan **b** dilakukan dengan cara sebagai berikut :

z = \*px; /\* 1 \*/

\*px = \*py; /\* 2 \*/

\*py = z; /\* 3 \*/

Pertama variabel **z** diisi dengan nilai yang ditunjuk oleh **px**. Kedua, yang ditunjuk oleh **px** diisi dengan yang ditunjuk oleh **py** (berarti **a** diisi dengan **b**). Ketiga, yang ditunjuk oleh **py** diberi nilai **z**. Dengan melalui tiga pernyataan di atas, nilai **a** dab **b** dapat diubah dalam fungsi.

1. **PERCOBAAN**
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.
3. Listing

#include <stdio.h>

#include <stdlib.h>

int pembanding(int,int);

int main()

{

int bil1, bil2, max;

puts("PROGRAM MENENTUKAN BILANGAN TERBESAR\n");

printf("Masukkan 2 buah bilangan pisahkan dengan spasi : ");

scanf("%d %d", &bil1, &bil2);

max = pembanding(bil1,bil2);

printf("Bilangan terbesar antara %d dan %d adalah %d", bil1, bil2, max);

return 0;

}

int pembanding(int x, int y)

{

int maks;

if(x<y)

maks = y;

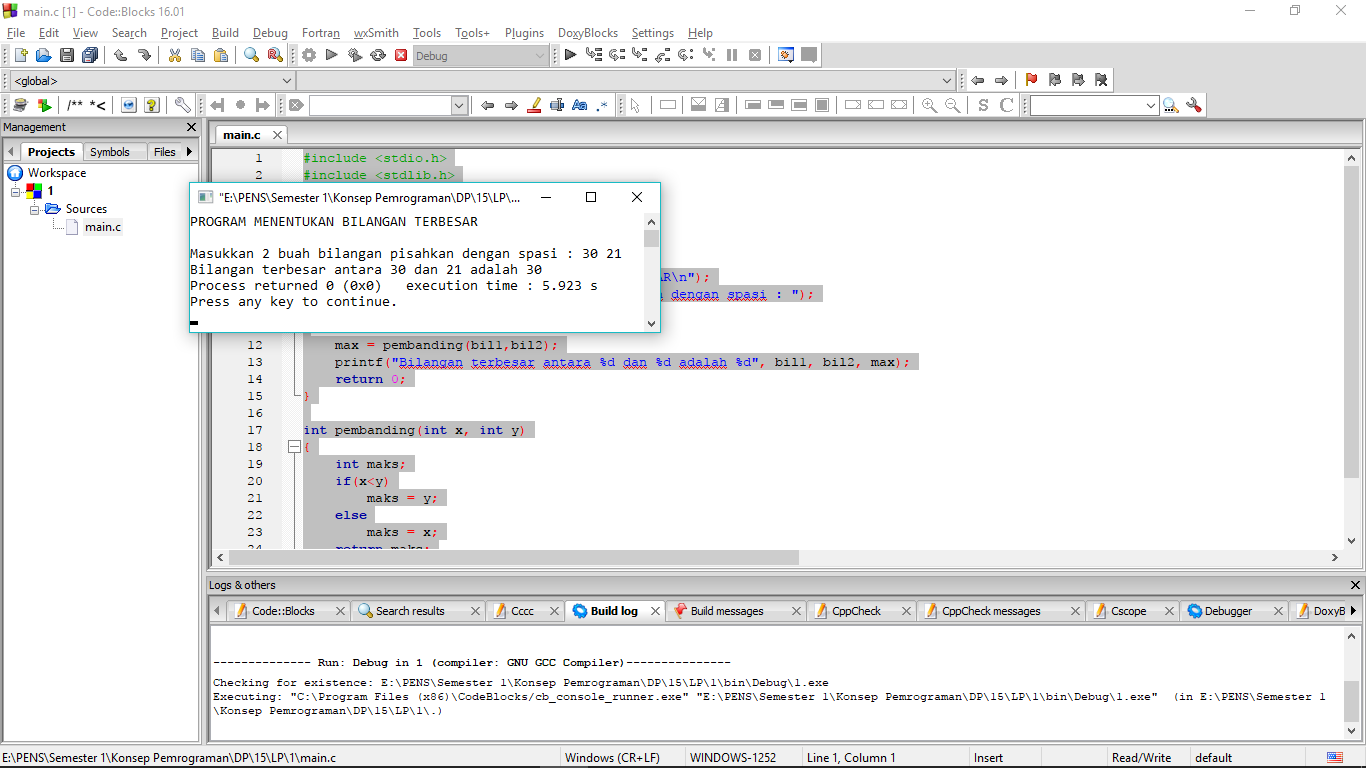
else

maks = x;

return maks;

}

1. Output



1. Buatlah suatu fungsi **permutasi()** dan **kombinasi()** untuk menghitung permutasi dan kombinasi dari suatu pasangan bilangan

Contoh Input : n = 3, r = 2

Output : Permutasi = 6

Kombinasi = 3

1. Listing

#include <stdio.h>

#include <stdlib.h>

int kombinasi(int,int);

int permutasi(int,int);

int faktorial(int);

int main()

{

int n,r, komb, perm;

puts("PROGRAM MENENTUKAN KOMBINASI DAN PERMUTASI\n");

printf("Masukkan 2 bilangan pisahkan dengan spasi : ");

scanf("%d %d", &n, &r);

komb = kombinasi(n,r);

perm = permutasi(n,r);

printf("Permutasi = %d", perm);

printf("\nKombinasi = %d", komb);

return 0;

}

int faktorial(int bil)

{

int i, total=1;

for(i=bil; i>=1; i--)

total = total \*i;

return total;

}

int kombinasi(int x, int y)

{

int hasilkomb;

hasilkomb = faktorial(x) / (faktorial(y)\*faktorial(x-y));

return hasilkomb;

}

int permutasi(int a, int b)

{

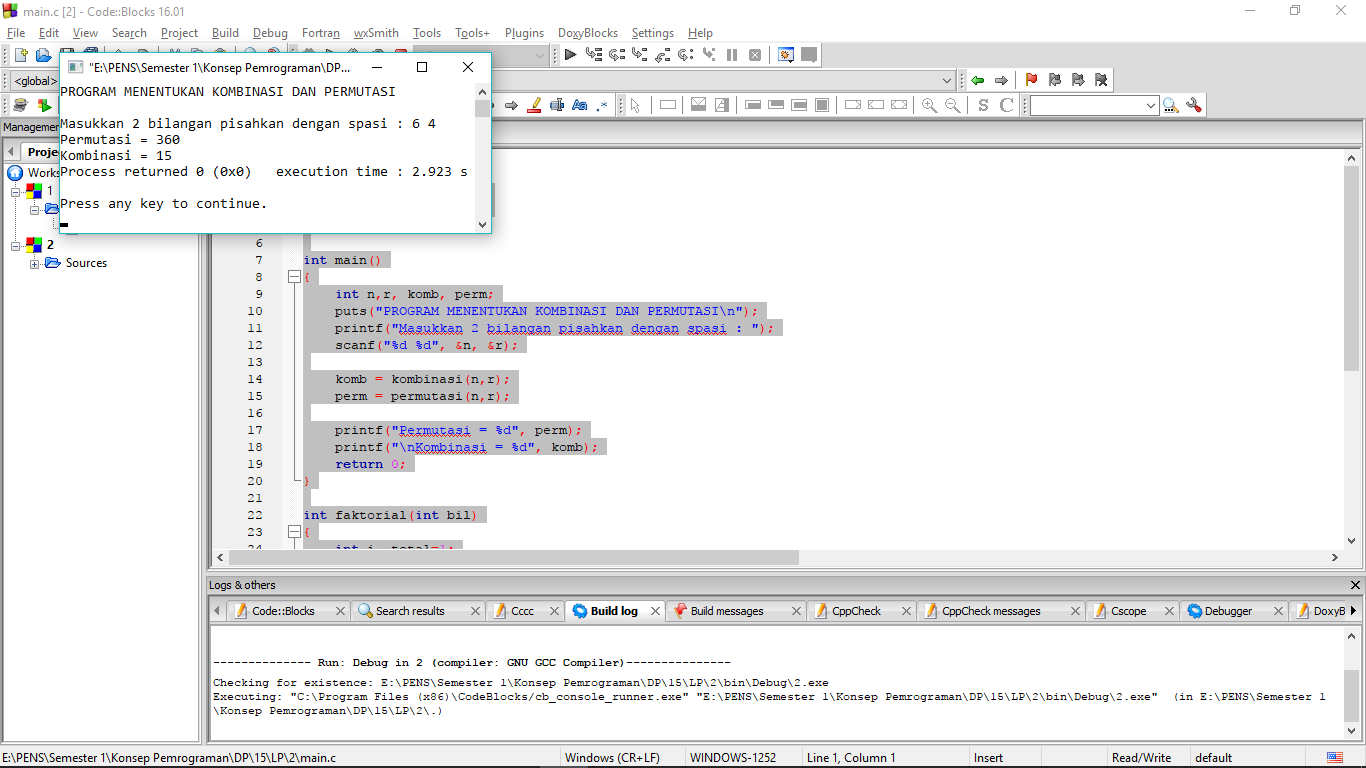
int hasilperm;

hasilperm = faktorial(a) / faktorial(a-b);

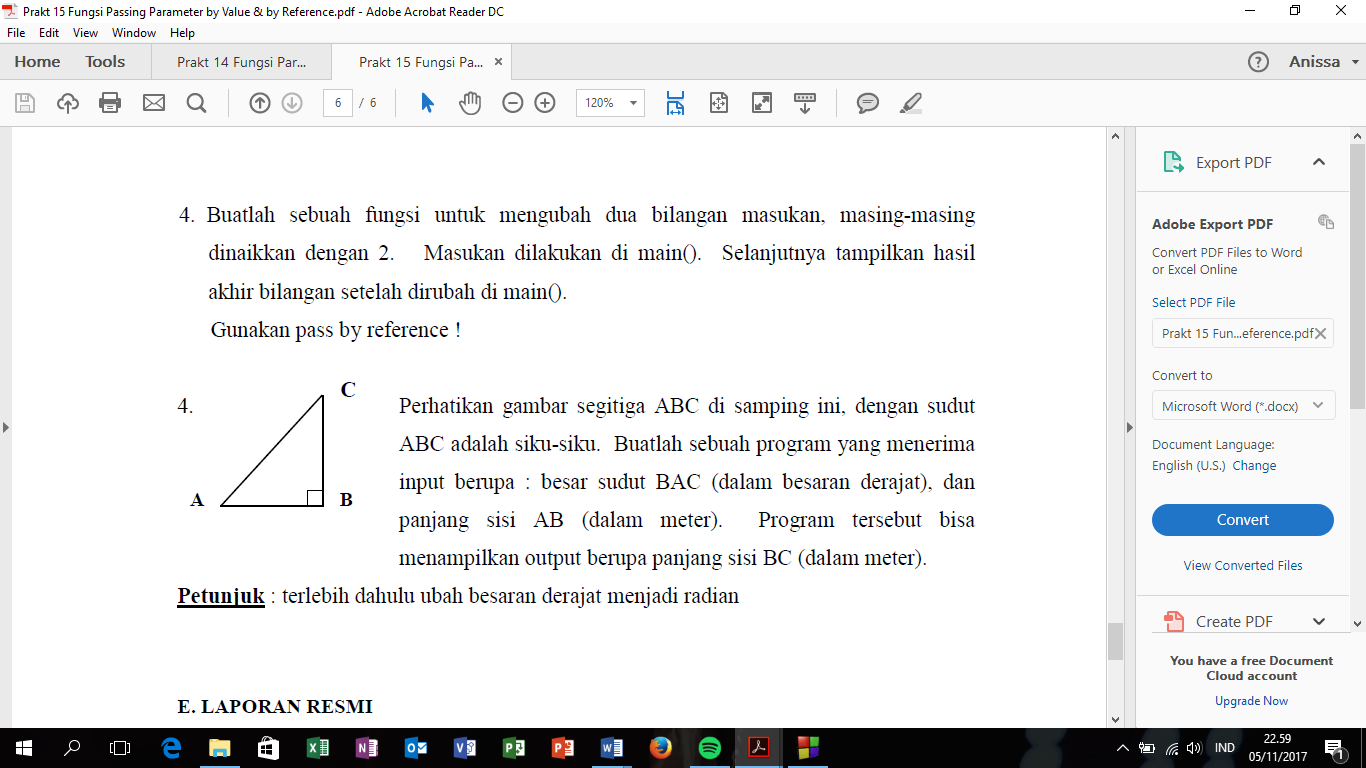
return hasilperm;

}

1. Output



1. Perhatikan gambar segitiga ABC di samping ini, dengan sudut ABC adalah siku-siku. Buatlah sebuah program yang menerima input berupa : besar sudut BAC (dalam besaran derajat), dan panjang sisi AB (dalam meter). Program tersebut bisa menampilkan output berupa panjang sisi BC (dalam meter). **Petunjuk** : terlebih dahulu ubah besaran derajat menjadi radian



1. Listing

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#define PHI 3.14159f

float findingBC(float,float);

int main()

{

float sudut, ab, bc;

puts("PROGRAM MENENTUKAN PANJANG BC SEGITIGA SIKU-SIKU\n");

printf("Masukkan besaran sudut BAC (derajat) : ");

scanf("%f", &sudut);

printf("Masukkan panjang ab (meter) : ");

scanf("%f", &ab);

bc = findingBC(sudut,ab);

printf("Jadi panjang sisi bc : %f", bc);

return 0;

}

float findingBC(float x, float y)

{

float rad, hasilbc;

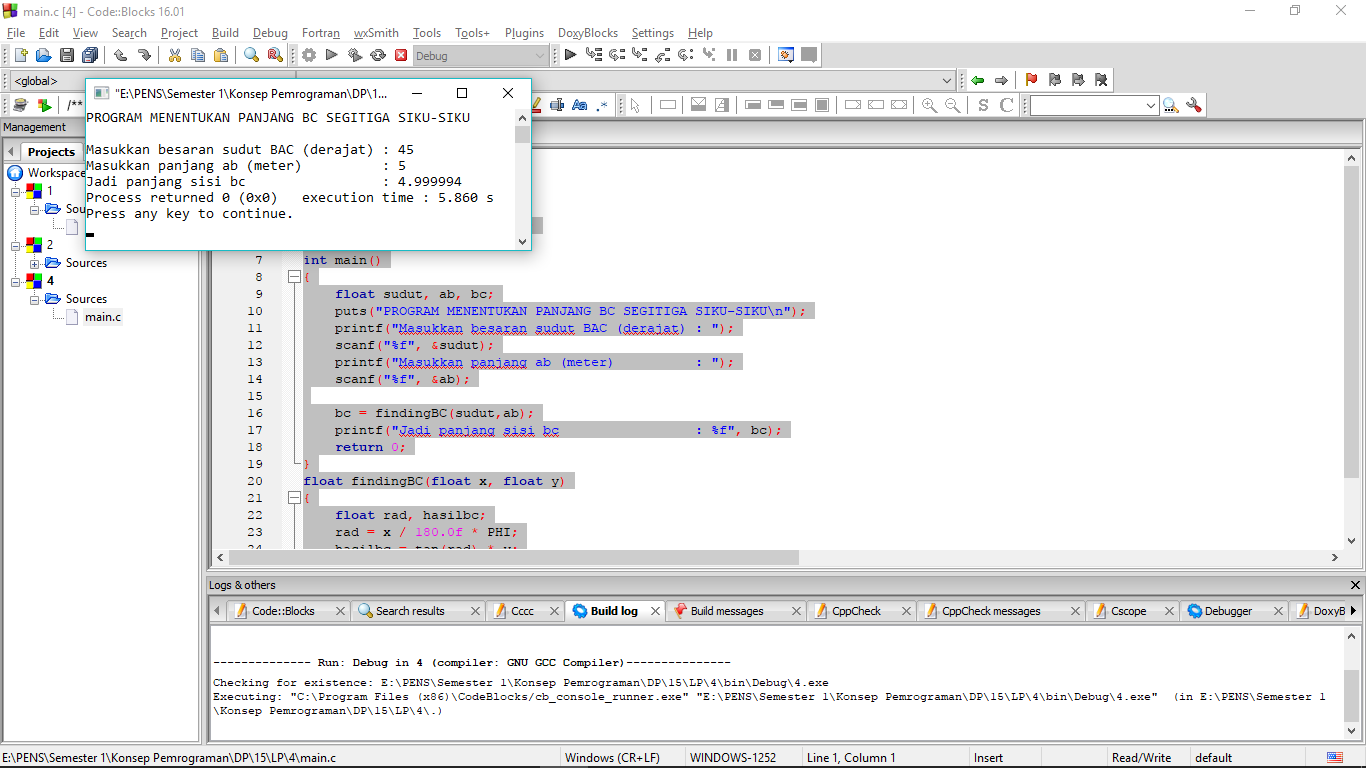
rad = x / 180.0f \* PHI;

hasilbc = tan(rad) \* y;

return hasilbc;

}

1. Output



1. **SOAL TAMBAHAN**
2. Jelaskan perbedaan antara *pass by value* dengan *pass by reference* ! Berikan contoh function call dan definisi fungsinya!

|  |  |  |
| --- | --- | --- |
| Perbedaan | Pemanggilan fungsi | Definisi fungsi |
| *Pass by value* | kalikan(x, y); | int kalikan(int a, int b)  {  pernyataan;  } |
| *Pass by reference* | kalikan(x, y); | int kalikan(int \*a, int \*b)  {  pernyataan;  } |