MAKALAH FUNGSI DAN STRUKTUR DATA

BAHASA C



OLEH:

DIKI CANDRA

NIM 2022903430010

D4 TEKNOLOGI REKAYASA KOMPUTER JARINGAN

POLITEKNIK NEGERI LHOKSEUMAWE

2022/2023

DAFTAR ISI

[BAB I 3](#_Toc116495990)

[PENDAHULUAN 3](#_Toc116495991)

[1.1  Latar Belakang 4](#_Toc116495992)

[1.2  Rumusan Masalah 4](#_Toc116495993)

[1.3  Tujuan 5](#_Toc116495994)

[1.4  Manfaat 5](#_Toc116495995)

[BAB II 6](#_Toc116495996)

[PEMBAHASAN 6](#_Toc116495997)

[A. POINTER 6](#_Toc116495998)

[1. Konsep Dasar Pointer 6](#_Toc116495999)

[2. Mendeklarasikan Variabel Pointer 6](#_Toc116496000)

[3. Menggatur Pointer Agar Menunjuk Ke Variabel Lain. 8](#_Toc116496001)

[4. Mengakses Isi Suatu Variabel Melalui Pointer 9](#_Toc116496002)

[5. Mengakses Dan Mengubah Isi Suatu Variabel Pointer. 10](#_Toc116496003)

[6. Pointer Dan Array 12](#_Toc116496004)

[7. Array dari Pointer 15](#_Toc116496005)

[9. Pointer Menunjuk Pointer 19](#_Toc116496006)

[10. pointer Dalam Fungsi 21](#_Toc116496007)

[B. STRUKTUR 29](#_Toc116496008)

[1. Mendefinisikan & Mendeklarasikan Struktur 36](#_Toc116496009)

[2. Mengakses Elemen Struktur Data 37](#_Toc116496010)

[3. Menginisialisasi Struktur 39](#_Toc116496011)

[4. Array Dan Struktur 41](#_Toc116496012)

[5. Struktur Dan Fungsi 43](#_Toc116496013)

[6. Struktur Dan Pointer (pointer ke struktur) 47](#_Toc116496014)

[BAB III 51](#_Toc116496015)

[PENUTUP 51](#_Toc116496016)

# BAB I

# PENDAHULUAN

## 1.1  Latar Belakang

Pemograman dalam struktur data ada beberapa macam. Salah satunya adalah pemograman C. Dalam pemograman ini biasanya menggunakan variable Array, Struktur dan Linked List

Makalah ini membahas tentang 3 variabel tersebut dimana ketiga variable mempunyai ciri dan umum yang berbeda sesuai dengan tipe file yang di gunakan pembaca. Seperti array yang menggunakan satu dimensi dan dua dimensi serta 3 dimensi dimana sangat berbeda dengan struktur yang menggunakan tingkatan prosedur.

Pemograman ini merupakan pemograman yang berbeda dari pemograman lainnya misalnya VB, Delphi atau Pascal namun perbedaan juga tidak begitu signifikan pada pemograman pascal.

## 1.2  Rumusan Masalah

Pada makalah ini kami merumuskan beberapa hal :

POINTER

* Konsep dasar pointer.
* Mendeklarasikan variabel pointer.
* Mangatur pointer agar menunjuk ke variabel lain.
* Manggakses isi suatu variabel melalui pointer.
* Menggakses dan mengubah isi suatu variabel pointer.
* Pointer dan array.
* Pointer dan string.
* Array dari pointer.
* Pointer menunjuk pointer.
* Pointer dalam fungsi.
* Ponter dalam parameter fungsi.
* Pointer sebagai keluaran fungsi (*return value*).

STRUKTUR

* Mendefinisiakan & mendeklarasikan struktur.
* Menggakses elemen struktur
* Menginisialisasi struktur.
* Array dan struktur
* Struktur dan fungsi.
* Melewatkan elemen struktur ke dalam fungsi
* Melewatkan struktur kedalam fungsi.

## 1.3  Tujuan

a. Untuk memenuhi tugas dari dosen pada pada kuliah Struktur Data

b. Bisa menggunakan/menjalankan program C terutama pada Array, Struktur dan Linked List.

## 1.4  Manfaat

a. Supaya kita mengerti atau mengetahui apa yang di maksud dengan Array dan beberapa contoh programnya, sehinnga kita dapat menjalankan programnya dengan baik dan benar.

b. Supaya kita mengerti atau mengetahui apa yang di maksud dengan Struktur dan beberapa contoh programnya, sehinnga kita dapat menjalankan programnya dengan baik dan benar.

c. Supaya kita mengerti atau mengetahui apa yang di maksud dengan Linked List dan beberapa contoh programnya, sehinnga kita dapat menjalankan programnya dengan baik dan benar.

# BAB II

# PEMBAHASAN

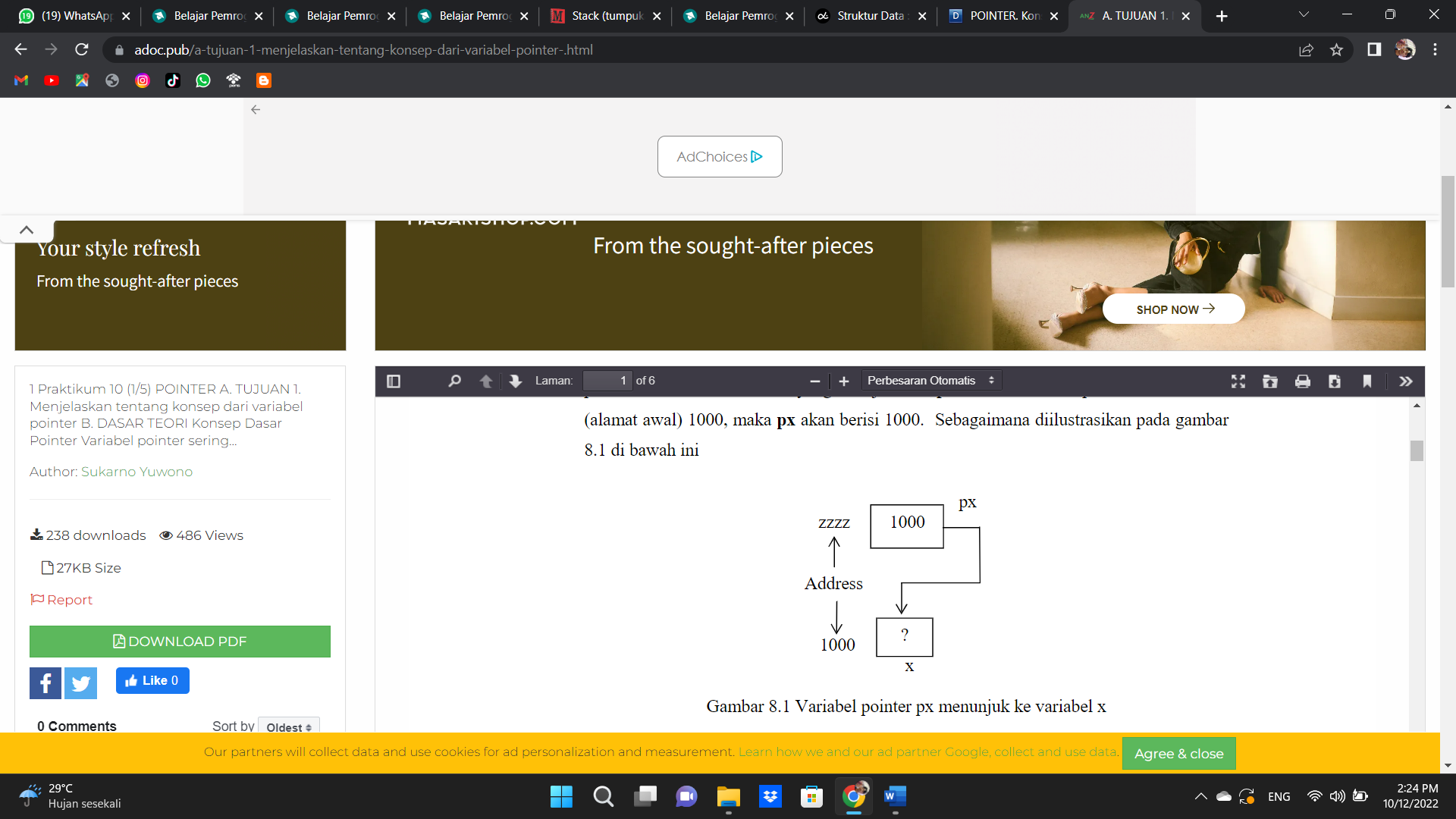
1. STRUKTUR DATA

## A. POINTER

Pointer adalah variabel yang menunjuk ke suatu variabel lain dengan menyimpan alamat memory variable tersebut. Setiap variabel yang kita buat pada program akan memiliki alamat memori. Alamat memori berfungsi untuk menentukan lokasi penyimpanan data pada memori (RAM). Kadang alamat memori ini disebut reference atau referensi. Untuk melihat alamat memori yang digunakan pada variabel, ernama pakai ernam & (emphasis).

### 1. Konsep Dasar Pointer

Variabel pointer sering dikatakan sebagai variabel yang menunjuk ke obyek lain. Pada kenyataan yang sebenarnya, variabel pointer berisi alamat dari suatu obyek lain (yaitu obyek yang dikatakan ditunjuk oleh pointer). Sebagai contoh, px adalah variabel pointer dan x adalah variabel yang ditunjuk oleh px. Kalau x berada pada alamat memori (alamat awal) 1000, maka px akan berisi 1000. Sebagaimana diilustrasikan pada gambar 8.1 dibawah ini



### 2. Mendeklarasikan Variabel Pointer

Pointer adalah suatu variabel yang menunjuk ke alamat memory variabel yang lainnya.Suatu pointer bukan berisi dengan suatu nilai data seperti halnya pada variabel biasa, variabel pointer berisi dengan suatu alamat.Untuk mendeklarasikan variabel pointer kita menggunakan tanda asterik / bintang (\*) didepan variabel yang di deklarasikan pada tipe data tertentu.Tanda ini juga dapat dipakai untuk mengakses nilai dari variabel yang telah ditunjuk. Untuk mendapatkan alamat dari variabel pointer kita menggunakan tanda “&”.

Deklarasi variabel pointer seperti halnya deklarasi variabel lainnya hanya ditambahkan tanda \* pada depan nama variabel.

Int \*b

Untuk mendapatkan alamat memori pointer (address of) maka perintah yang digunakan adalah menambahkan tanda “&” didepan variabel

&b

Untuk mendapatkan Isi atau nilai dari variabel pointer maka perintah yang digunakan cukup nama variabelnya saja, atau pada pointer menunjukkan pada

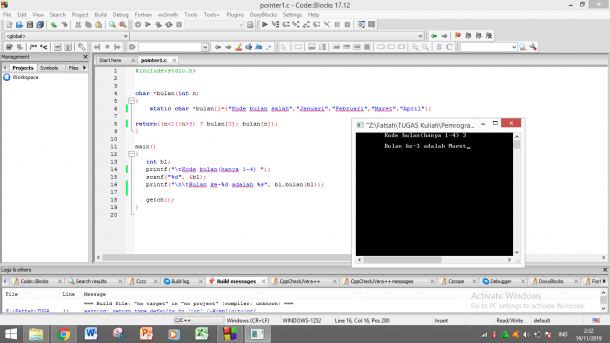
\*e = &b

Yang artinya e menunjukkan isi dari alamat yang disimpan pada b.

Pada sisi lain, pointer memiliki hubungan dengan array. Array sendiri sebenarnya juga merupakan terjemahan dari alamat tiap indeks variabel array. Apabila suatu indeks i pada c[i], kemudian diberikan perintah sebagai berikut:

b = &c[i]

maka b berisi alamat dari array c[i].

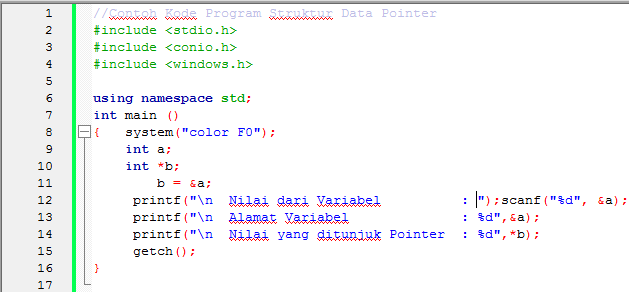


Pada contoh fungsi diatas, nilai int n memberikan pengaruh untuk menyatakan pada indeks berapa dari array yang akan ditunjuk dan disimpan alamatnya. Misalkan n bernilai 4, maka pointer menyimpan alamat pada array indeks 4, bukan pada alamat variabel array secara menyeluruh.

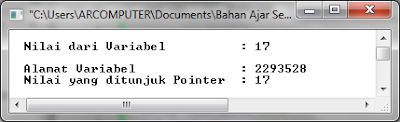
Hasil yang ditampilkan pada akhir program merupakan sebuah isi dari variabel yang alamatnya ditunjuk. Sekilas hasil keluaran ialah seperti sekedar mencetak isi dari array indeks ke-3. Tetapi perhatikan, nama fungsi memiliki kesamaan dengan nama variabel didalamnya, tetapi pointer menangkap bahwa mereka pada alamat yang berbeda, sehingga tidak terjadi eror yang fatal pada program.

Berikut beberapa contoh pengaplikasian daripada pointer

Contoh Program:



Hasilnya:



Pointer memungkinkan suatu program melaksanakan kerja lebih efisien dan praktis. Untuk suatu data yang hendak diproses langsung dituju dengan pointer, maka program akan lebih tertuju pada data yang dimaksud dan mungkin dapat mengabaikan data yang tidak tercantum untuk proses.

### 3. Menggatur Pointer Agar Menunjuk Ke Variabel Lain.

Agar suatu pointer menunjuk ke variabel lain, mula-mula pointer harus diisi dengan alamat dari variabel yang akan ditunjuk. Untuk menyatakan alamat dari suatu variabel, operator & (operator alamat, bersifat unary) bisa dipergunakan, dengan menempatkannya di depan nama variabel. Sebagai contoh, bila x dideklarasikan sebagai variabel bertipe int, maka

&x

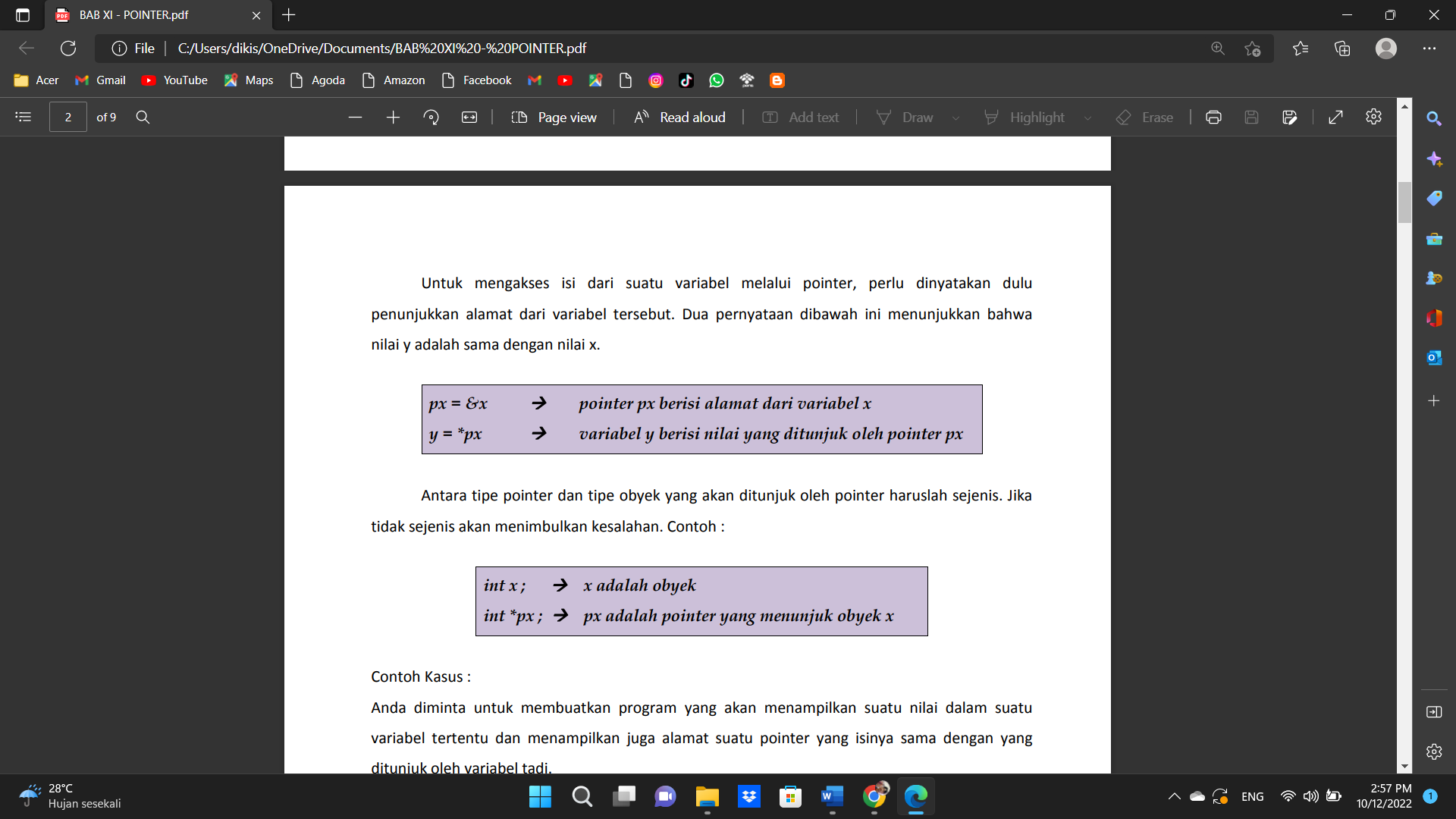
berarti “alamat dari variabel x”. Adapun contoh pemberian alamat x ke suatu variabel pointer px (yang dideklarasikan sebagai pointer yang menunjuk ke data bertipe int) yaitu : Pernyataan di atas berarti bahwa px diberi nilai berupa alamat dari variabel x. Setelah pernyataan tersebut dieksekusi barulah dapat dikatakan bahwa px menunjuk ke variabel x.

px = &x;

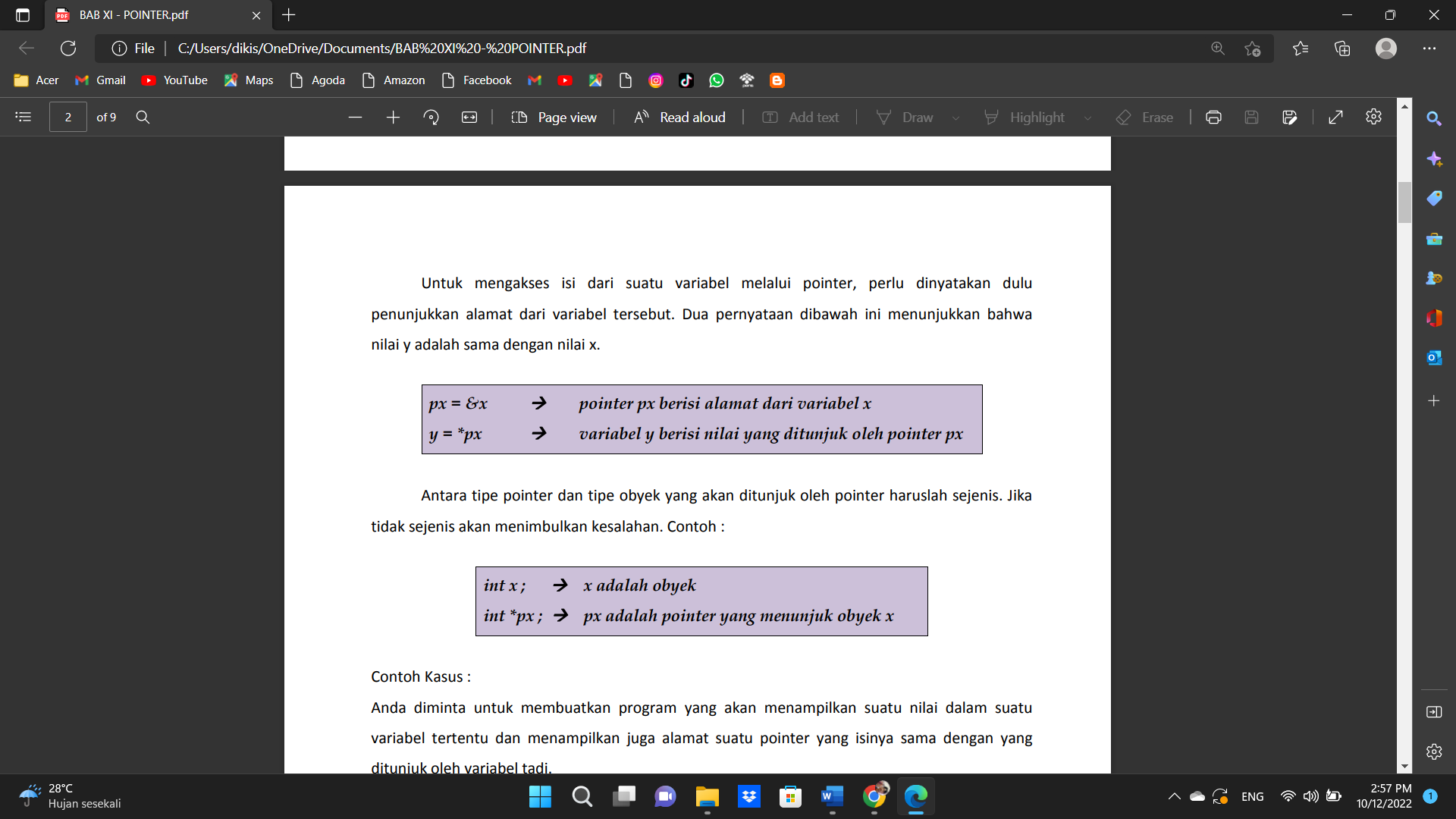
Pernyataan di atas berarti bahwa px diberi nilai berupa alamat dari variabel x. Setelah pernyataan tersebut dieksekusi barulah dapat dikatakan bahwa px menunjuk ke variabel x.

### 4. Mengakses Isi Suatu Variabel Melalui Pointer

Untuk mengakses isi dari suatu variabel melalui pointer, perlu dinyatakan dulu penunjukkan alamat dari variabel tersebut. Dua pernyataan dibawah ini menunjukkan bahwa nilai y adalah sama dengan nilai x.



Antara tipe pointer dan tipe obyek yang akan ditunjuk oleh pointer haruslah sejenis. Jika tidak sejenis akan menimbulkan kesalahan. Contoh :



Contoh Kasus :

Anda diminta untuk membuatkan program yang akan menampilkan suatu nilai dalam suatu variabel tertentu dan menampilkan juga alamat suatu pointer yang isinya sama dengan yang ditunjuk oleh variabel tadi. Penyelesaian :

Algoritma yang digunakan untuk memecahkan masalah diatas adalah sebagai berikut :

1. Tentukan variabel yang menampung suatu nilai tertentu, misalkan nilai ini bertipe integer dan nama variabel nya adalah z. (misalkan int z)

2. Tentukan variabel pointer yang nantinya isinya akan menunjuk pada variabel pada ernama no.1. (misalkan variabel \*p)

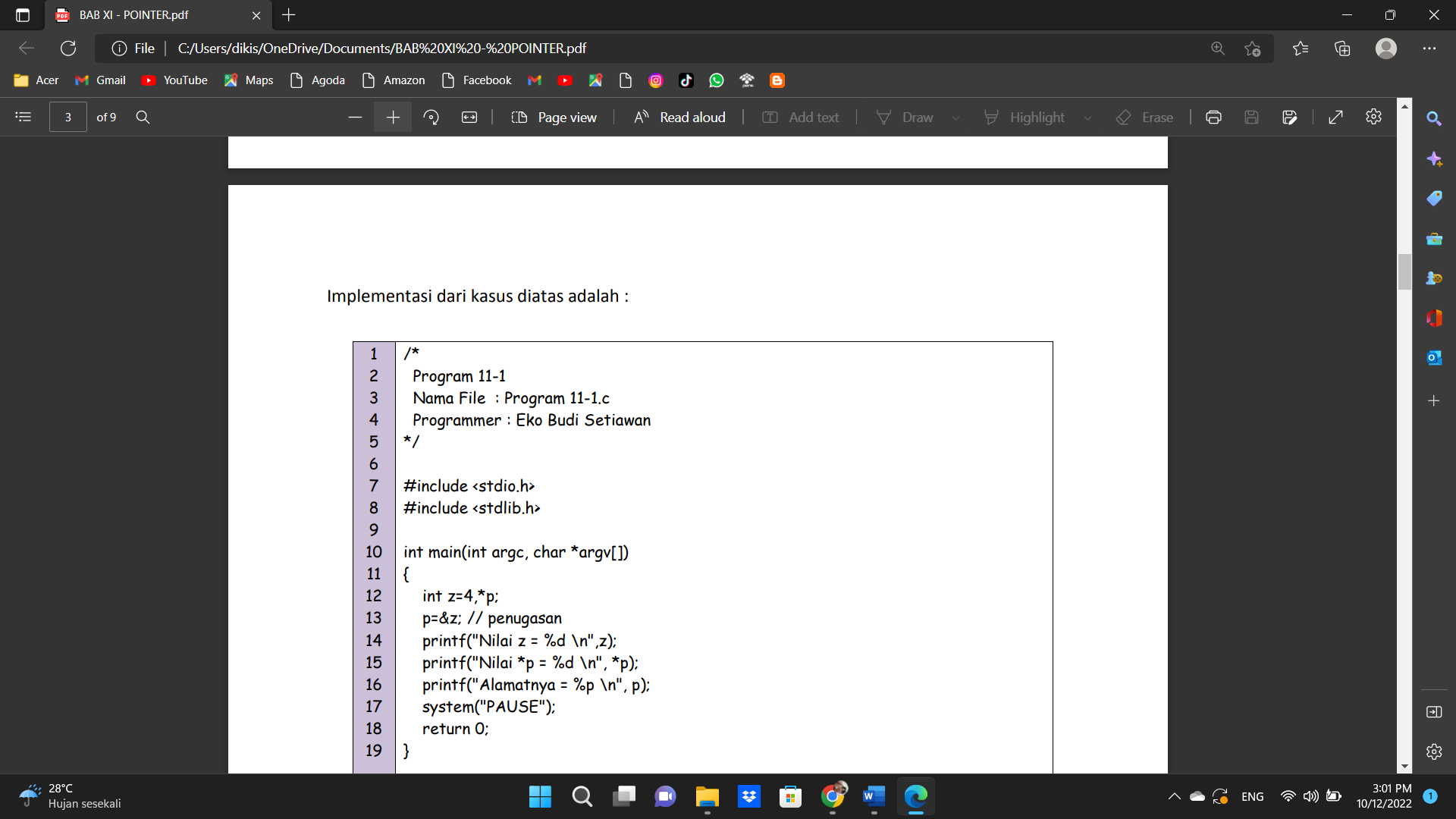
3. Tentukan variabel lain yang isinya menunjuk ke variabel pertama dalam ernama no.1. (misalkan variabel p dan prosesnya adalah p = &z)

4. Masukan suatu nilai yang disimpan pada variabel z

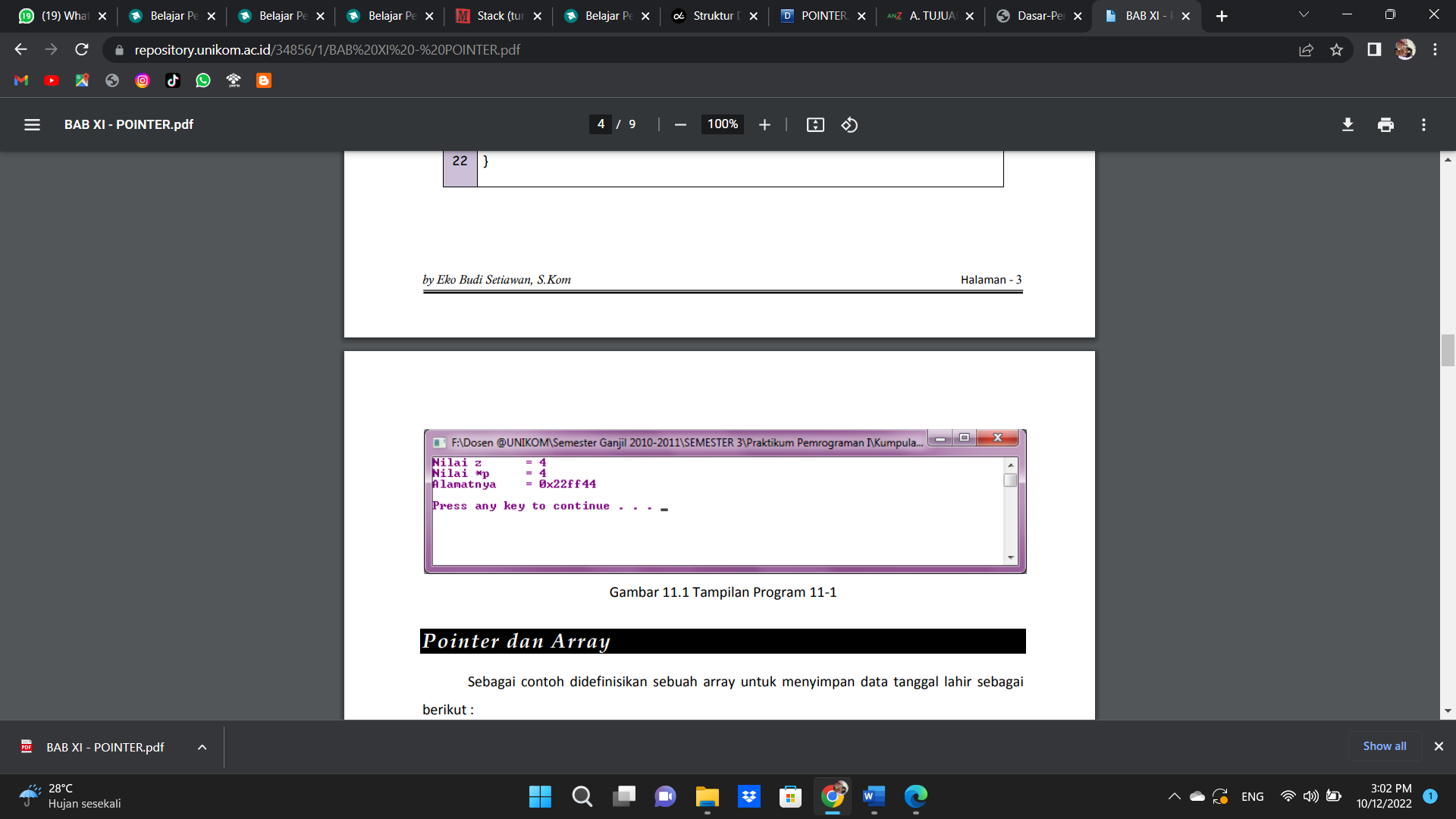
5. Tampilkan nilai tersebut

6. Tampilkan memori yang tepat untuk menyimpan variabel pointer yang sesuai dengan

Implementasi dari kasus di atas:



Ouput:



### 5. Mengakses Dan Mengubah Isi Suatu Variabel Pointer.

Contoh berikut memberikan gambaran tentang pengubahan isi suatu variabel secara tak langsung (yaitu melalui pointer). Mula-mula pd dideklarasikan sebagai pointer yang menunjuk ke suatu data bertipe float dan d sebagai variabel bertipe float. Selanjutnya

d = 54.5;

digunakan untuk mengisikan nilai 54,5 secara langsung ke variabel d. Adapun

pd = &d;

digunakan untuk memberikan alamat dari d ke pd. Dengan demikian pd menunjuk kevariabel d. Sedangkan pernyataan berikutnya

\*pd = \*pd + 10; (atau: \*pd += 10; )

merupakan instruksi untuk mengubah nilai variabel d secara tak langsung. Perintah di atas berarti “jumlahkan yang ditunjuk pd dengan 10 kemudian berikan ke yang ditunjuk oleh pd”, atau ernama dengan pernyataan

d = d + 10;

Akan tetapi, seandainya tidak ada instruksi

pd = &d;

maka pernyataan

\*pd = \*pd + 10;

tidaklah sama dengan

d = d + 10;

Contoh Program:

#include <stdio.h>

main()

{

float d = 54.5f, \*pd;

printf(“Isi d mula-mula = %g\n”, d);

pd = &d;

\*pd += 10;

printf(“Isi d sekarang = %g\n”, d);

}

Contoh eksekusi :

Isi d mula-mula = 54.5

Isi d sekarang = 64.5

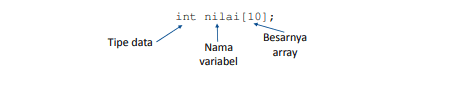
### 6. Pointer Dan Array

a. Array

Array atau larik adalah variabel yang terdiri dari sekumpulan data terstruktur yang mempunyai tipe data yang sama. Setiap data disimpan di alamat memori yang berbeda-beda yang disebut dengan elemen array. Setiap elemen mempunyai nilai index sesuai urutannya. Melalui index inilah kita dapat mengakses data-data tersebut. Umumnya index dari elemen array dimulai dari 0.

Array juga bersifat statis, dimana alokasi memori dilakukan saat deklarasi source code.

Contoh deklarasi array:



* Dalam contoh diatas terdapat 10 variabel dengan tipe yang sama yaitu integer. Untuk mengakses array harus ditentukan index yang diakses.

Contoh:

#include <stdio.h>

void main() {

int bilangan[10], i;

for(i=0;i<10;i++){

bilangan[i] = 2\*i;

printf(“%d %d\n”, i, bilangan[i]);

}

}

/\*

Output:

7 14

8 16

9 18

\*/

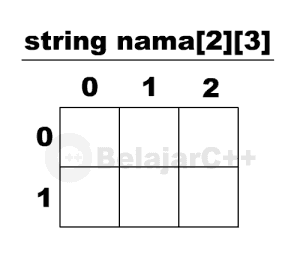
* Contoh diatas kita mengakses elemen array berdasarkan looping(perulangan) kemudian menginisiasikan valuenya dan ernama mengakses arraynya untuk dicetak kedalam output.

Terdapat juga jenis array yang lainnya yaitu:

* Array 2 Dimensi

Deklarasi array 2 dimensi: string nama[2][3] Array dua dimensi memungkinkan kita untuk menggunakan array di dalam array, setiap array memiliki array di dalamnya.

Array dua dimensi bisa digambarkan seperti sebuah erna yang mempunyai baaris dan kolom array pertama adalah sebuah baris dan array kedua adalah sebuah kolom berikut ilustrasinya:



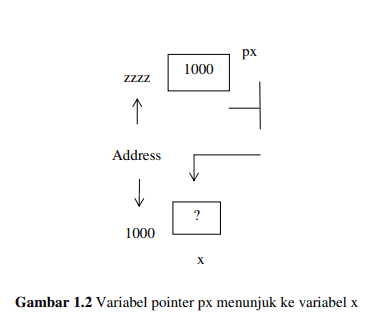
* Array Multidimensi

Array multidimensi merupakan array yang mempunyai ukuran lebih dari dua. Jika kita sudah mengerti mengenai konsep dari array, kita akan dengan mudah untuk memodifikasi array, menambahkan beberapa subscript untuk menjadikanya array multidimensi. Secara garis besar, pembuatan, penggunaan, pengaksesan array multi dimensi memiliki konsep yang sama seperti apa yang telah penulis jelaskan di atas pada array dua dimensi.

b. Pointer

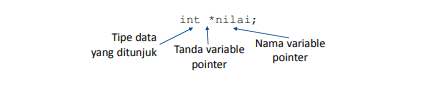
Pointer adalah variabel yang menunjuk ke lokasi alamat memori tertentu. Variabel pointer sering dikatakan sebagai variabel yang menunjuk ke obyek lain. Pada kenyataan yang sebenarnya, variabel pointer berisi alamat dari suatu obyek lain (yaitu obyek yang dikatakan ditunjuk oleh pointer). Sebagai contoh px adalah variabel pointer dan x adalah variabel yang ditunjuk oleh px. Kalau x berada pada alamat memori (alamat awal) 0x1000, maka px akan berisi 1000.

Lebih jelasnya mari kita lihat ilustrasi berikut:



Variabel pointer juga bersifat dinamis (kebalikan dari array), dimana alokasi memori dilakukan pada saat program dijalankan.

Contoh deklarasi pointer:



* Variabel nilai menunjuk ke variable lain yang bertipe integer

Berikut ini adalah contoh cara mengakses pointer:

#include <stdio.h>

void main() {

int a, \*b;

a = 5;

b = &a;

printf(“%d %d\n”, a, \*b);

}

/\*

Output:

5 5

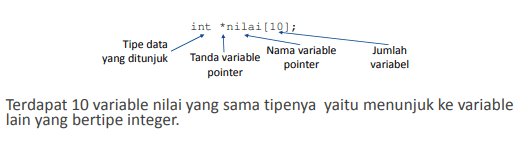
\*/

* variabel b menampung alamat memori variabel a

### 7. Array dari Pointer

Array of pointer adalah sekumpulan array dengan tipe data pointer.

Contoh deklarasi array of pointer:



#include <stdio.h>

void main() {

int \*bil[3], a, b, c, i;

a = 5;

b = 10;

c = 15;

bil[0] = &a;

bil[1] = &b;

bil[2] = &c;

for(i=0;i<3;i++)

printf(“%d\n”, \*bil[i]);

}

/\*

Output:

5

10

15

\*/

* kita liat pada source code diatas setiap elemen arraynya menampung alamat memori dari variabel yang didefinisikan sebelumnya.
* kemudian kita tampilkan value dari array bil dengan method print maka valuenya akan berisi value dari alamat memori variabel yang direferensikan.

Hubungan antara pointer dan array pada C sangatlah erat. Sebab sesungguhnya array secara internal akan diterjemahkan dalam bentuk pointer.

Misalnya dideklarasikan di dalam suatu fungsi

static int tgl\_lahir[3] = { 01, 09, 64 }; dan int \*ptgl;

Kemudian diberikan instruksi

ptgl = &tgl\_lahir[0];

maka ptgl akan berisi alamat dari elemen array tgl\_lahir yang berindeks nol. Instruksi di atas bisa juga ditulis menjadi

ptgl = tgl\_lahir;

sebab nama array tanpa tanda kurung menyatakan alamat awal dari array. Sesudah penugasan seperti di atas,

\*ptgl dengan sendirinya menyatakan elemen pertama (berindeks sama dengan nol) dari array tgl\_lahir.

8. Pointer Dan String.

Karena string pada dasarnya juga adalah array dengan dimensi satu, maka hubungan antara pointer dan string juga sangat erat. Sebagai contoh diberikan kode program berikut ini.

#include

void main(){

char \*pointer\_kota = “PURWODADI”;

printf(“String yang ditunjuk oleh pointer\_kota: %s\n”, pointer\_kota);

}

Pada baris kode yang dicetak tebal, akan memberikan perintah kepada compiler untuk mengalokasikan variabel pointer\_kota sebagai variabel pointer yang menunjuk objek bertipe data string dengan nilai konstanta “PURWODADI” dalam memory ernama. Kemudian pointer pointer\_kota akan menunjuk pada alamat string “PURWODADI”. Bentuk lain dari kode program yang dicetak tebal di atas bisa juga dituis dengan char kota[] = “PURWODADI”. Namun kedua penulisan ini memiliki perbedaan, yaitu pointer\_kota adalah pointer yang menunjuk alamat string, jika terdapat string lain maka dapat dengan mudah diarahkan ke string lain tersebut. Sedangkan kota[] adalah string yang menyatakan array dengan nilai konstan (tidak dapat diubah) “PURWODADI”. Perhatikan contoh progam di bawah ini.

#include

#include

void main(){

char \*kota1= “GROBOGAN”;

char \*kota2= “SEMARANG”;

char \*kotax;

printf(“Data Awal: \n”);

printf(“\tKota1: %s\n”, kota1);

printf(“\tKota2: %s\n”, kota2);

kotax = kota1;

kota1 = kota2;

kota2 = kotax;

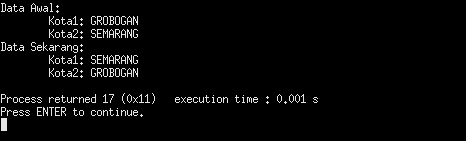
printf(“Data Sekarang: \n”);

printf(“\tKota1: %s\n”, kota1);

printf(“\tKota2: %s\n”, kota2);

}

Pada baris kode yang dicetak tebal di atas menunjukkan erna terjadi pertukaran data. Mula-mula kota1 berisi data string “GROBOGAN” dan kota2 berisi data string “SEMARANG”. Kemudian ditukar dimana kotax = kota1, kota1 = kota2 dan kota2 = kotax sehingga setelah dilakukan pertukaran data akan menghasilkan output seperti gambar berikut.



### 9. Pointer Menunjuk Pointer

Suatu pointer juga bisa menunjuk ke pointer lain. Sebagai contoh akan diberikan deklarasi variabel seperti berikut.

Int x = 123;

int \*pointer1;

int \*\*pointer2;

Keterangan:

variabel x merupakan variabel dengan tipe data int.

variabel pointer1 merupakan variabel pointer yang menunjuk variabel dengan tipe data int, dalam kasus ini menunjuk ke variabel x. Agar pointer1 menunjuk ke variabel x, diberikan perintah pointer1 = &x;

variabel pointer2 merupakan variabel pointer yang menunjuk ke variabel pointer lain dengan tipe data int, dalam hal ini menunjuk ke variabel pointer1. Agar pointer2 menunjuk ke variabel ponter1, maka diberikan perintah pointer2 = &pointer1;

Untuk lebih jelasnya, silakan simak kode program berikut ini.

#include

void main(){

int x = 123;

int \*pointer1;

int \*\*pointer2;

pointer1 = &x;

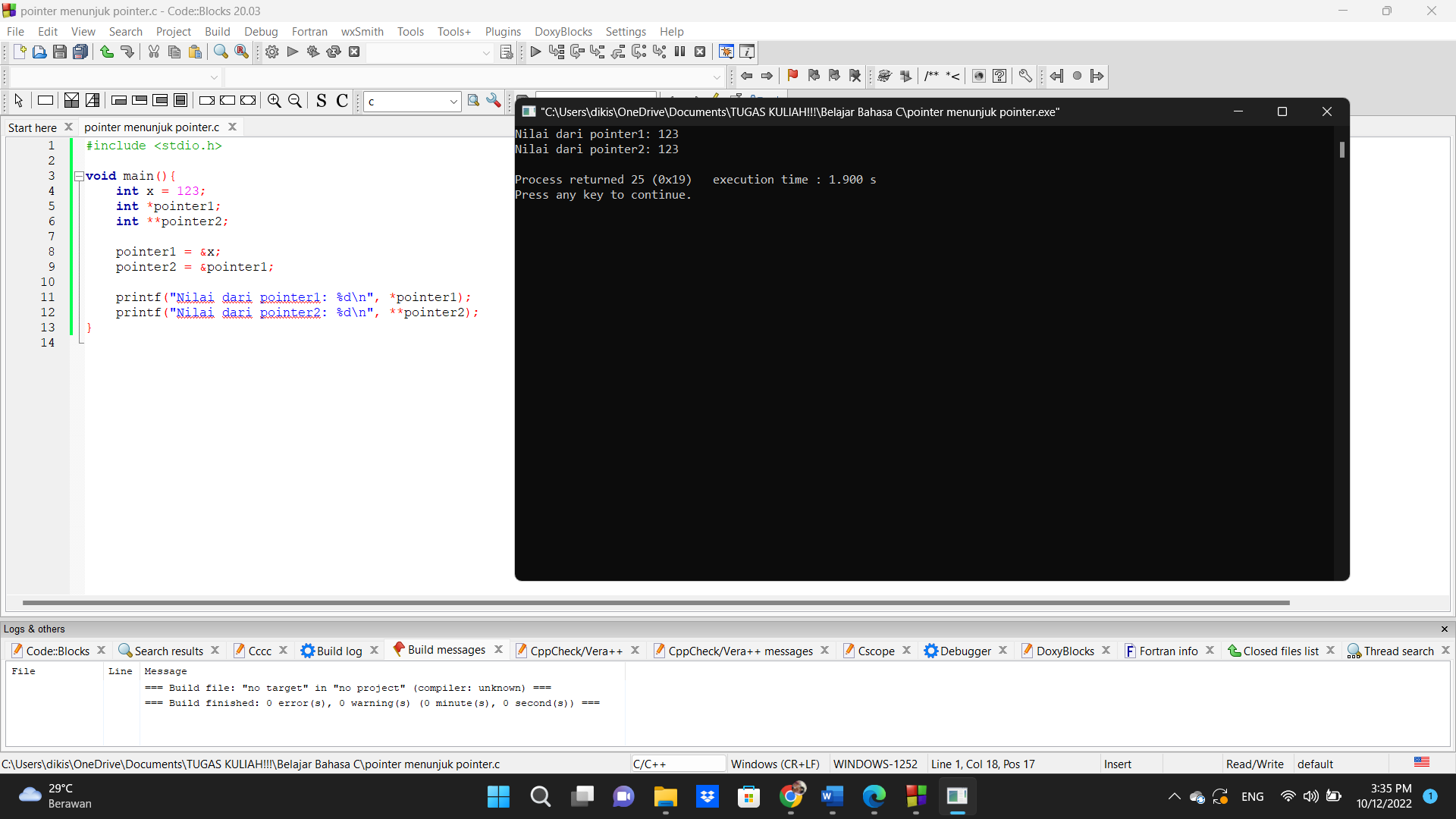
pointer2 = &pointer1;

printf(“Nilai dari pointer1: %d\n”, \*pointer1);

printf(“Nilai dari pointer2: %d\n”, \*\*pointer2);

}

Outputnya:



### 10. Pointer Dalam Fungsi

Suatu pointer adalah suatu variable yang menyimpan alamat dari suatu variable yang ditunjuk oleh pointer yang bersangkutan. Karena data yang ditunjuk pointer bervariasi, maka tipe dari pointer harus disesuaikan dengan data yang ditunjuk. Untuk memperkirakan ukuran data data yang ditunjuk, sehingga alokasi memori terjaga dengan baik. Pointer berguna untuk mentransfer data yang mempunyai kapasitas besar, melalui suatu fungsi. Pointer sangat erat kaitannya dengan array, sehingga variable pointer dapat menggantikan fungsi dari variable array.

#### A. Melewatkan elemen Struktur Kedalam Fungsi

Pointer sebagai parameter fungsi ini diterapakan sebagai parameter yaitu jika diinginkan agar nilai suatu variabel internal dapat diubah oleh fungsi yang dipanggil.

Sebagai contoh

void naikan\_nilai (int \*x, int \*y)

{

\*x = \*x + 2;

\*y = \*y + 2;

}

fungsi diatas dimaksudkan agar erna dipanggil, variabel yang berkenan dengan parameter ernam dapat diubah nilainya, masing-masing dinaikan sebesar 2.

Contoh pemanggilan:

naikkan\_nilai(&a,&b);

pointer sebagai keluaran fungsi, suatu fungsi dapat dibuat agar keluarannya berupa pointer. Misalya, suatu fungsi menghasilkan keluaran berupa pointer yang menunjukan keluaran berupa pointer yang menunjukan ke string nama\_bulan, seperti pada contoh berikut:

char \*nama\_bulan(int n)

{

static char \*bulan[]=

{“kode bulan salah”, “januari”, “februari”, “maret”, “erna”, “mei”, “juni”, “juli”, “agustus”, “ernama”, “oktober”, “ernama”, “desember”

};

retrun ( (n<1 | | n>12)? Bulan[0] : bulan{n});

}

nah setelah kita tau nih pointer dari segi teori maka mari kita coba dengan percobaan dibawah ini:

Membuat program untuk menyimpan bilangan pecahan acak sebanyak 12 bilangan, yang disimpan dalam suatu array. Urutkan bilangan-bilangan tersebut dan ditampilkan sebelum dan sesudah diurutkan menggunakan index array dan pointer.

Jawab :

#include<stdio.h>

#include<string.h>

main()

{

static int a,b[16],c,d;

int \*p\_A;

menu:

printf(“\n Angka sebelum diurutkan …\n”);

for(a=1;a<13;a++)

{

printf(“Angka ke %d = “,a);

scanf(“%d”,&b[a]);

}

printf(“\n Angka setelah diurutkan …\n”);

for(a=1;a<13;a++)

{

for(d=a+1;d<13;d++)

{

if(\*(b+a)>\*(b+d))

{

c=\*(b+d);

\*(b+d)=\*(b+a);

\*(b+a)=c;

}

}

printf(“Angka ke %d = %d\n”,a,\*(b+a));

printf(” index pointer = “);

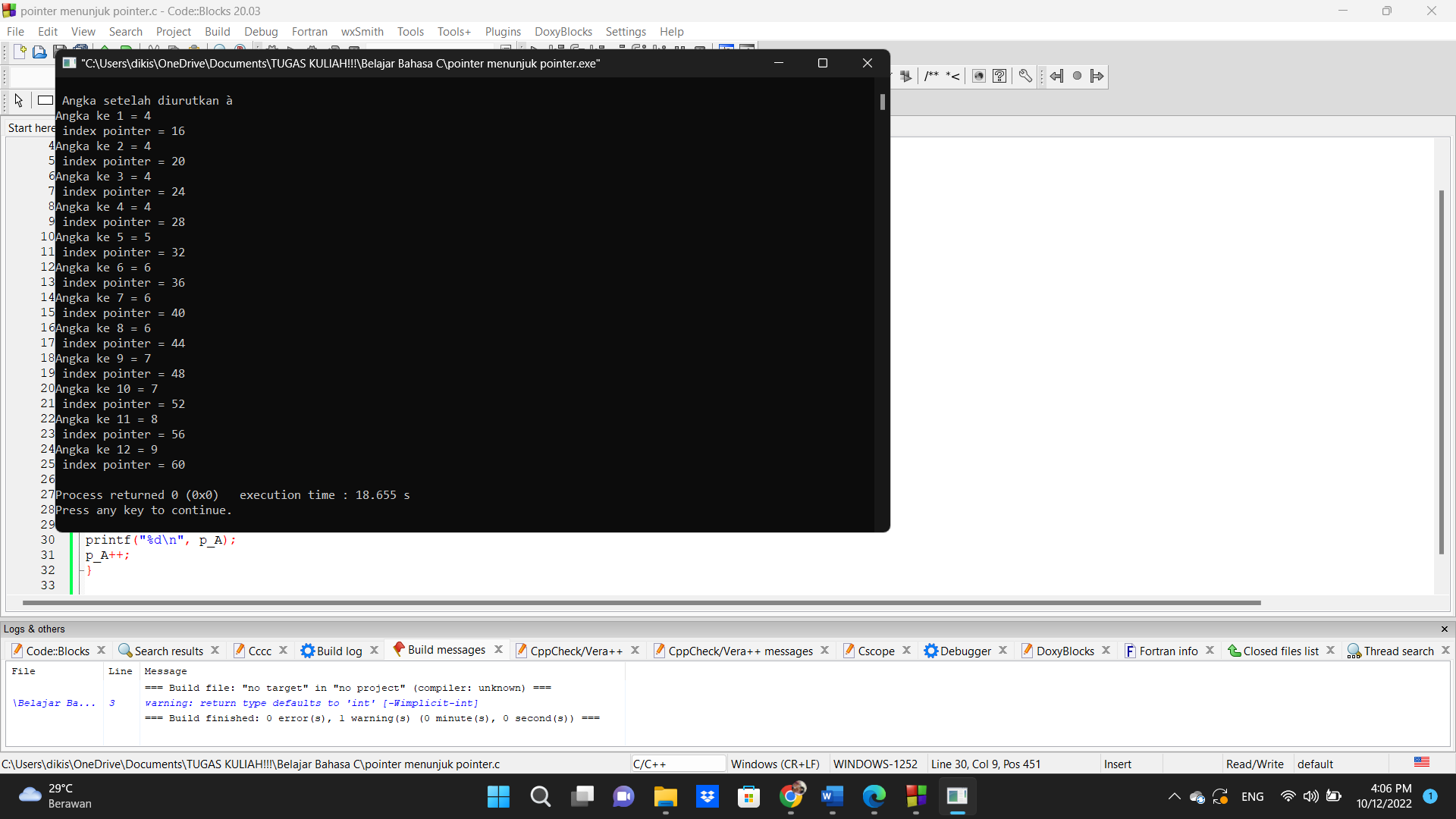
printf(“%d\n”, p\_A);

p\_A++;

}

}

Lalu, menekan ikon ‘Build and run’ untuk menjalankan program tersebut. Jika penulisan struktur program tersebut sudah benar maka program akan berjalan seperti pada gambar dibawah.



Membuat program yang menyimpan data karakter ‘X’ pada suatu array yang berdimensi 12 x 12, kemudian tampilan menggunakan index pointer.

Jawab :

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

main()

{

static int a[12][12],b,c,\*d,e,f,g;

char h[8],\*i;

menu1:

g=1;

for(e=0;e<12;e++)

{

for(f=0;f<12;f++)

a[e][f]=48;

}

menu2:

system(“cls”);

d=&a;

printf(“\n\r”);

for(e=1;e<13;e++)

{

for(f=1;f<13;f++)

printf(“%2c”,\*d++);

puts(“”);

}

if(g>1)

{

printf(“\n Ketik ‘lanjut’ untuk melanjutkan.”);

printf(“\n Ketik ‘ulang’ untuk mengulang dari awal.”);

printf(“\n Ketik ‘exit’ untuk keluar.\n “);

gets(h);

gets(h);

i=strlwr(h);

if(strcmp(i,”lanjut”)==0)

{

g=1;

goto menu2;

}

else if(strcmp(i,”ulang”)==0)

goto menu1;

else if(strcmp(i,”exit”)==0);

else goto menu2;

}

else

{

g++;

printf(“\n Masukkan koordinat matrix dengan jarak 1 – 12 …\n “);

scanf(“%d%d”,&c,&b);

-b;

-c;

a[b][c]=120;

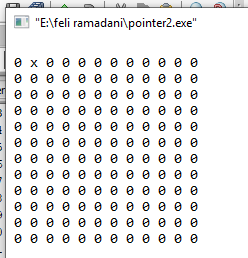
goto menu2;

}

}

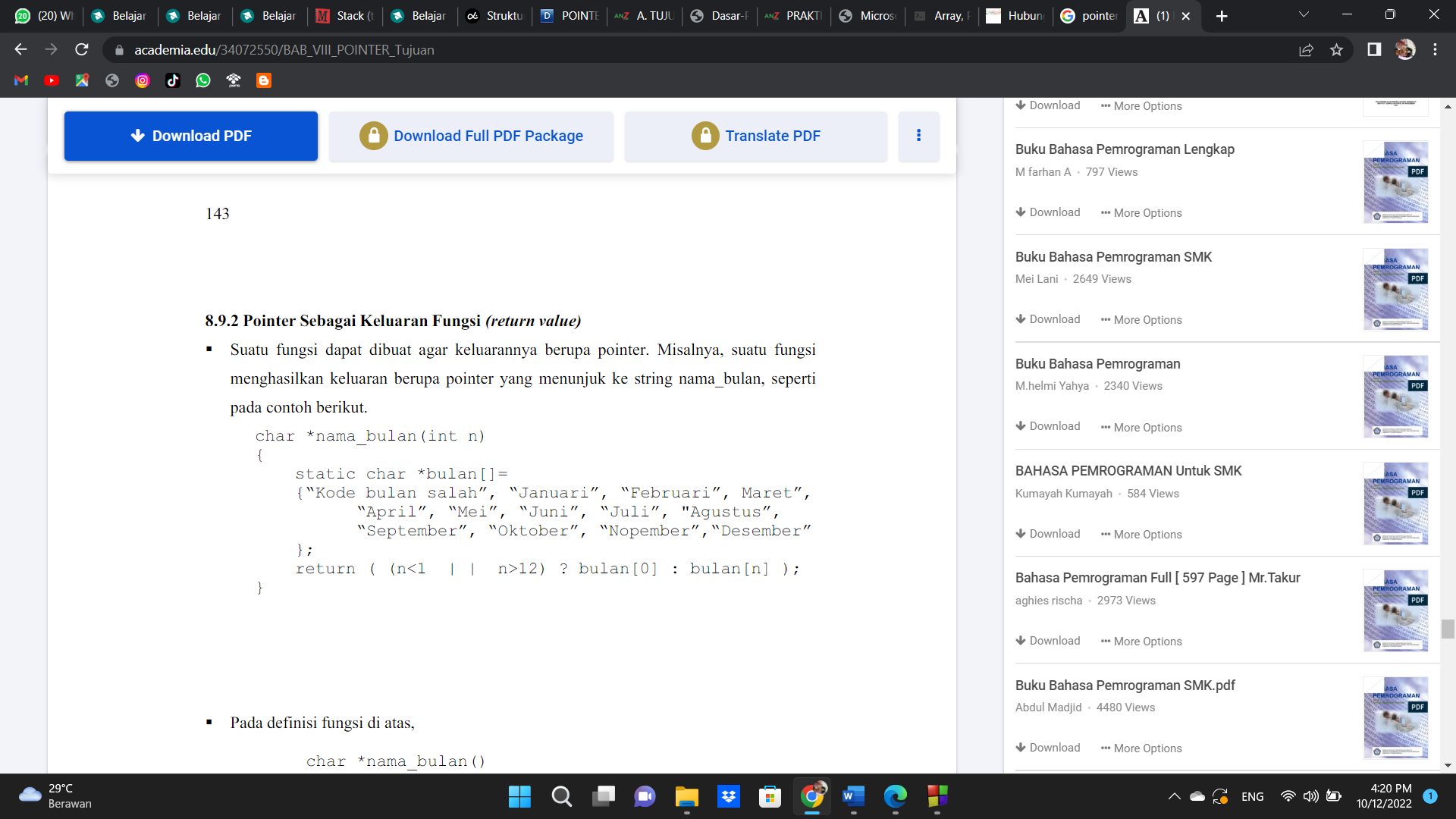
}

Lalu, menekan ikon ‘Build and run’ untuk menjalankan program tersebut. Jika penulisan struktur program tersebut sudah benar maka program akan berjalan seperti pada gambar dibawah.



#### b. Pointer Sebagai Keluaran Fungsi

Suatu fungsi dapat dibuat agar keluarannya berupa pointer. Misalnya, suatu fungsimenghasilkan keluaran berupa pointer yang menunjuk ke string nama\_bulan, seperti pada contoh berikut.



Pada definisi fungsi di atas,

char \*nama\_bulan()

menyatakan bahwa keluaran fungsi nama\_bulan() berupa pointer yang menunjuk keobyek char (atau string).

Dalam fungsi nama\_bulan(), mula-mula array Bernama bulan dideklarasikan dansekaligus diinisialisasi agar menunjuk sejumlah string yang menyatakan nama bulan.Di bagian akhir fungsi, pernyataan

return ( (n<1 || n>12) ? bulan[0] : bulan[n] );

menyatakan bahwa hasil fungsi berupa pointer yang menunjuk ke

* string “Kode bulan salah” (bulan[0]) jika masukan fungsi n<1 atau n>12
* bulan[n] untuk n yang terletak antara 1 sampai dengan 12.

Kesimpulan

• Tipe variabel pointer adalah tipe variabel yang berisi alamat dari variabel yangsebenarnya.

• Tipe variabel pointer harus sama dengan tipe varibel yang ditunjuk.

• Hubungan antara pointer dan array pada C sangatlah erat, sebab sesungguhnya arraysecara internal akan diterjemahkan dalam bentuk pointer

• Varibel pointer bisa berupa string, array atau tipe variabel yang lainnya.

• Suatu pointer bisa saja menunjuk ke pointer lain(pointer to pointer)

• Variabel pointer bisa digunakan sebagai parameter dalam sebuah fungsi,sebagaimana juga bisa dijadikan sebagai nilai balik (return value) dari sebuahfungsi.

## B. STRUKTUR

Structure atau struct adalah kumpulan dari beberapa variabel dengan beragam tipe data yang dibungkus dalam satu varabel.

Struct juga dikenal dengan records dalam ernam pemrograman lain seperti Pascal.

Mengapa kita membutuhkan struct?

Sekarang coba pikirkan…

Misalnya kita ingin menyimpan data mahasiswa. Kita bisa saja melakukannya seperti ini:

char name[] = “Dian”;

char address[] = “Mataram”;

int age = 22;

Lalu bagaimana erna ada lebih dari satu mahasiswa?

Mungkin bisa saja kita buat seperti ini:

char name[] = “Dian”;

char address[] = “Mataram”;

int age = 22;

char name2[] = “Bambang”;

char address2[] = “Surabaya”;

int age2 = 23;

char name3[] = “Bimo”;

char address3[] = “Jakarta”;

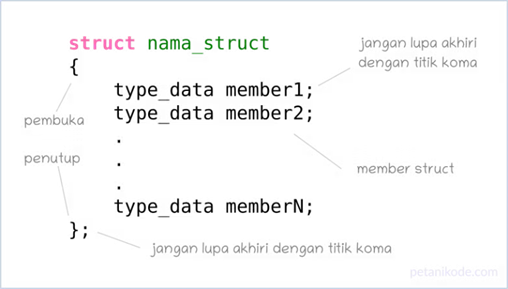
int age3 = 23;

Ugh! Terlihat kurang bagus.

Biar tidak membuat banyak variabel seperti ini, maka ernama gunakan struct.

Cara Membuat Struct

Struct dapat kita buat dengan kata kunci struct kemudian diikuti dengan nama struct dan isinya.



Contoh:

struct Mahasiswa

{

char \*name;

char \*address;

int age;

};

Catatan: kita menggunakan pointer \* untuk data string

Kita sudah tahu cara membuat struct, lalu bagaimana cara menggunakannya?

Cara Menggunakan Struct

Agar struct dapat digunakan, kita harus membuat variabel untuknya.

Contoh:

Buatlah program baru dengan nama contoh\_struct.c kemudian isi dengan kode berikut.

#include <stdio.h>

// membuat struct

struct Mahasiswa {

char \*name;

char \*address;

int age;

};

void main(){

// menggunakan struct

struct Mahasiswa mhs1, mhs2;

// mengisi nilai ke struct

mhs1.name = “Dian”;

mhs1.address = “Mataram”;

mhs1.age = 22;

mhs2.name = “Bambang”;

mhs2.address = “Surabaya”;

mhs2.age = 23;

// mencetak isi struct

printf(“## Mahasiswa 1 ##\n”);

printf(“Nama: %s\n”, mhs1.name);

printf(“Alamat: %s\n”, mhs1.address);

printf(“Umur: %d\n”, mhs1.age);

printf(“## Mahasiswa 2 ##\n”);

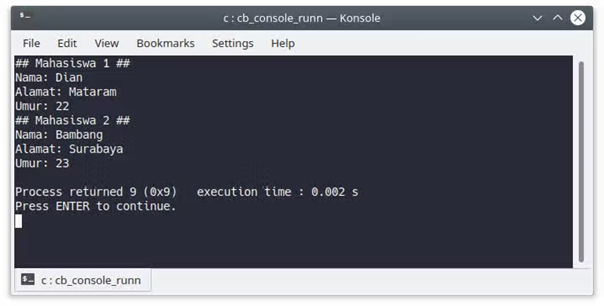
printf(“Nama: %s\n”, mhs2.name);

printf(“Alamat: %s\n”, mhs2.address);

printf(“Umur: %d\n”, mhs2.age);

}

Hasilnya:



Menggunakan typedef pada Struct

Kata kunci typedef adalah kata kunci untuk mendefinisikan tipe data baru.

Kita bisa menggunakan kata kunci ini di depan struct untuk menyatakannya sebagai tipe data baru.

Contoh:

Tanpa typedef kita akan menggunakan struct seperti ini.

// membuat struct

struct Distance{

int feet;

float inch;

};

void main() {

// menggunakan struct

struct Distance d1, d2;

}

Nah, erna menggunakan typedef akan menjadi seperti ini:

// membuat struct dengan typedef

typedef struct Distance{

int feet;

float inch;

} distances;

void main() {

// menggunakan struct

distances dist1, dist2, sum;

}

Struct Bersarang

Struct dapat dibuat bersarang (nested). Bersarang artinya ada struct di dalam struct.

Contoh:

struct complex

{

int imag;

float real;

};

struct number

{

struct complex comp;

int integers;

} num1, num2;

Lalu cara menggunanya akan seperti ini:

num1.integer = 12;

num1.comp.real = 44.12;

num2.comp.imag = 11;

Passing Struct ke dalam Fungsi

Struct dapat kita buat sebagai parameter untuk fungsi.

Contoh:

#include <stdio.h>

struct student

{

char name[50];

int age;

};

void main() {

struct student s1;

printf(“Enter name: “);

scanf(“%[^\n]%\*c”, s1.name);

printf(“Enter age: “);

scanf(“%d”, &s1.age);

display(s1); // passing structure as an argument

}

// membuat fungsi dengan struct sebagai parameter

void display(struct student s) {

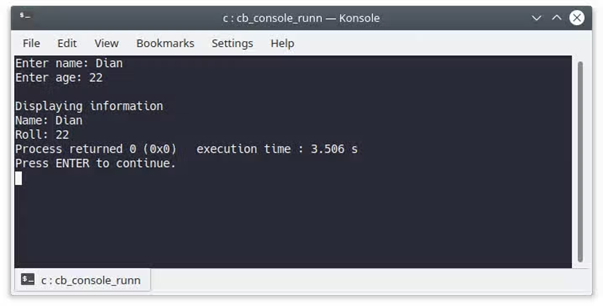
printf(“\nDisplaying information\n”);

printf(“Name: %s”, s.name);

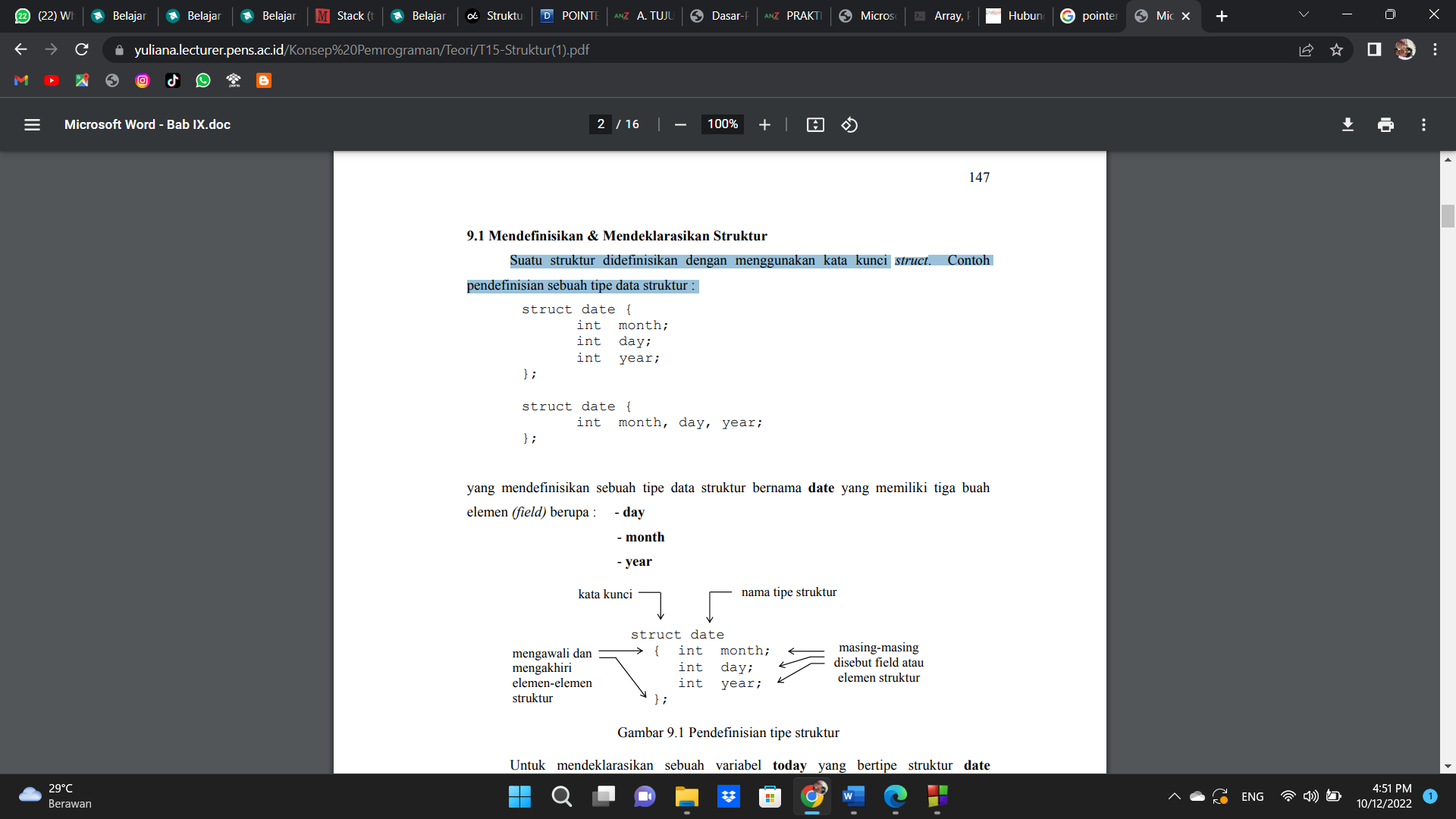
printf(“\nRoll: %d”, s.age);

}

Hasilnya:



### 1. Mendefinisikan & Mendeklarasikan Struktur

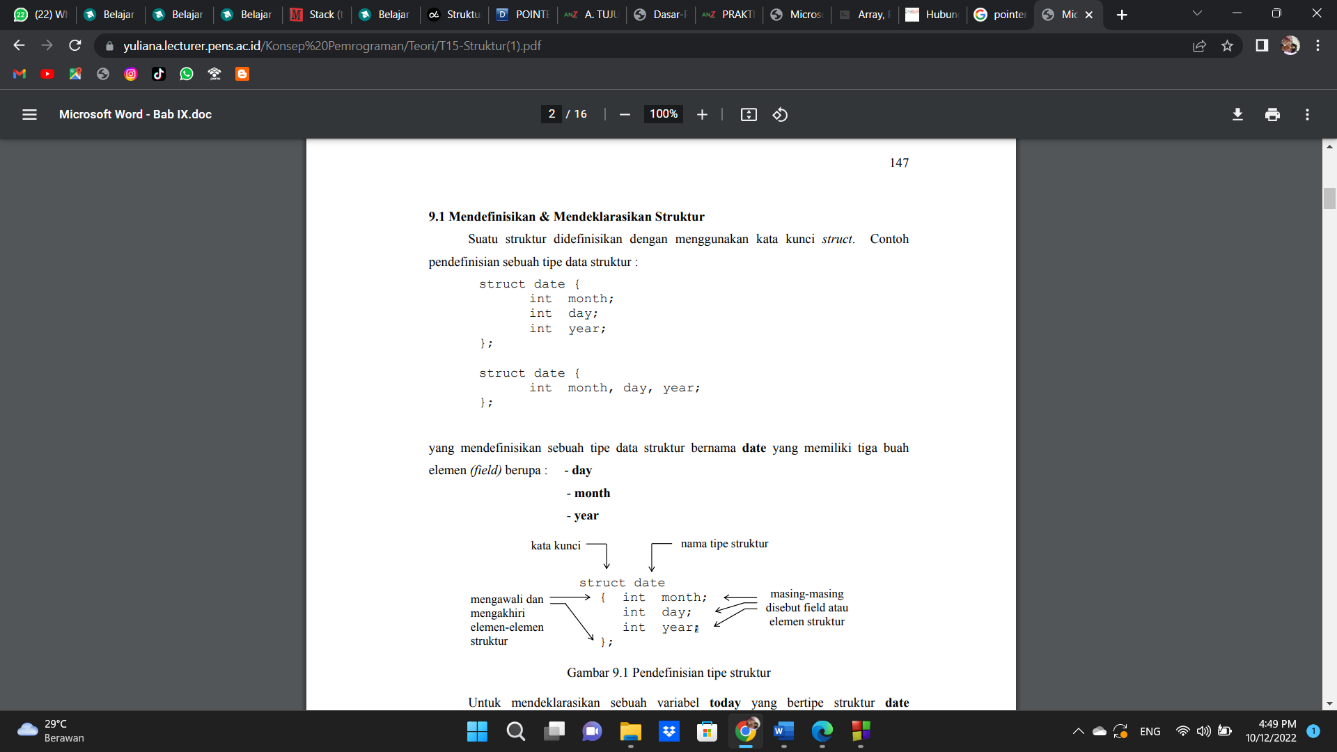
Suatu struktur didefinisikan dengan menggunakan kata kunci struct. Contoh pendefinisian sebuah tipe data struktur :

yang mendefinisikan sebuah tipe data struktur ernama date yang memiliki tiga buah

elemen (field) berupa - day

- month

- year



Pernyataan di atas menyatakan bahwa variabel today bertipe struktur date.

Dalam mendefinisikan sebuah struktur, elemen yang terkandung di dalamnya bisa juga berupa sebuah struktur, contoh :

struct date {

int month, day, year;

};

struct student {

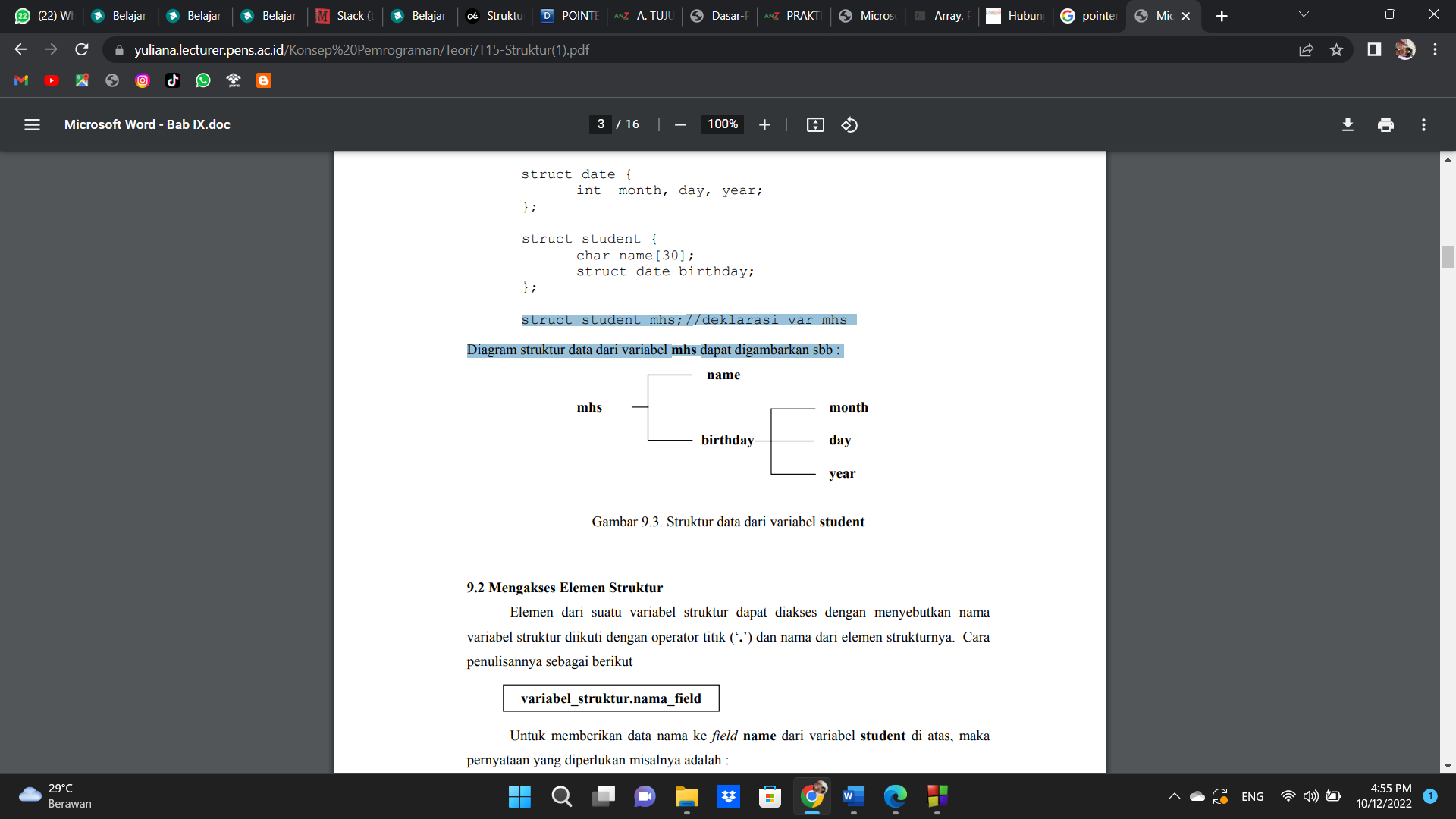
char name[30];

struct date birthday;

};

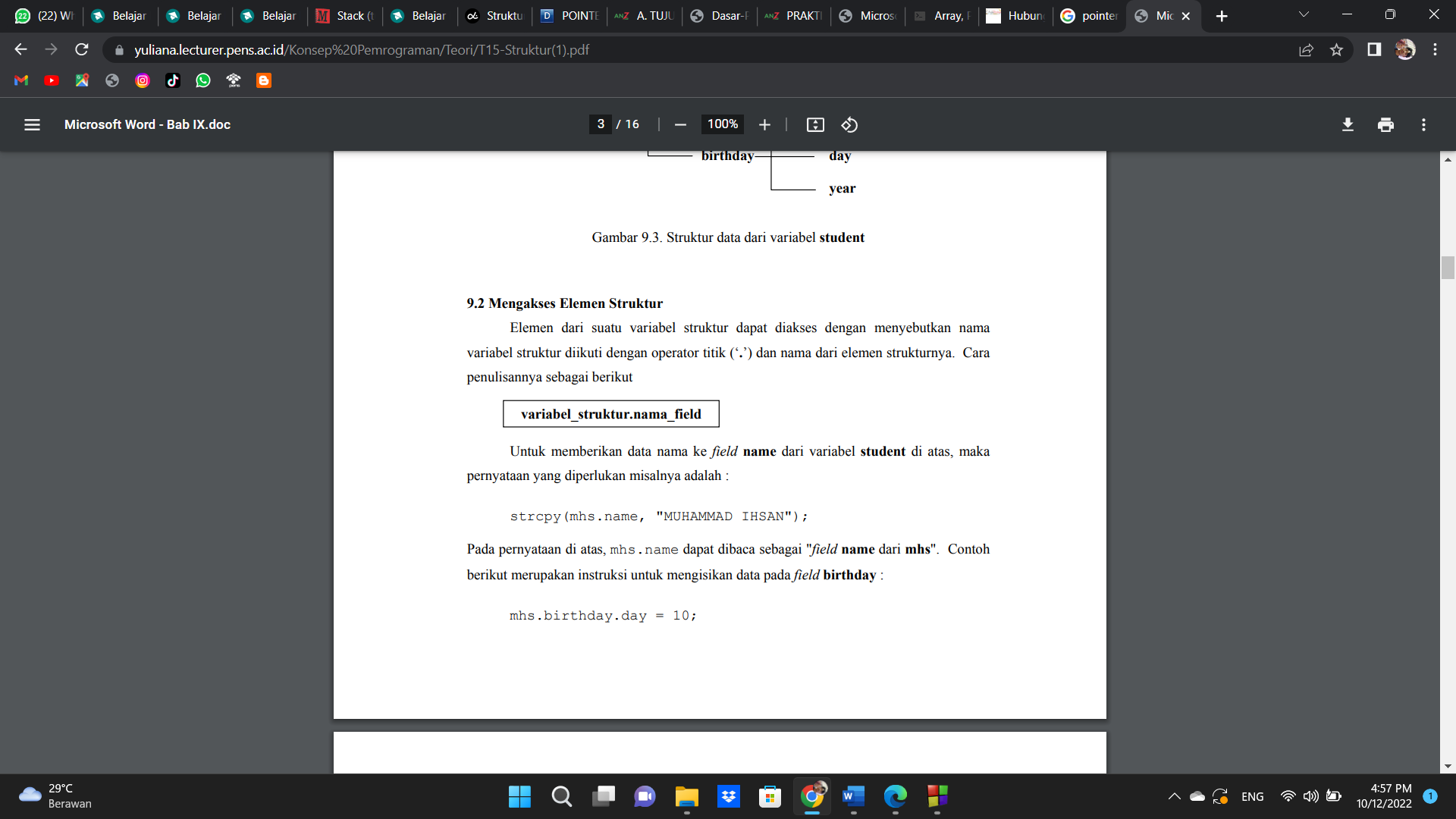
struct student mhs; //deklarasi var mhs

Diagram struktur data dari variabel mhs dapat digambarkan sbb :



### 2. Mengakses Elemen Struktur Data

Elemen dari suatu variabel struktur dapat diakses dengan menyebutkan nama variabel struktur diikuti dengan operator titik (‘.’) dan nama dari elemen strukturnya. Cara penulisannya sebagai berikut:



Untuk memberikan data nama ke field name dari variabel student di atas, maka

pernyataan yang diperlukan misalnya adalah :

strcpy(mhs.name, “DIKI CANDRA”);

Pada pernyataan di atas, mhs.name dapat dibaca sebagai “field name dari mhs”. Contoh berikut merupakan instruksi untuk mengisikan data pada field birthday

mhs.birthday.day = 10;

Sedangkan untuk mendapatkan isi suatu field dari variabel struktur, contohnya :

* tgl = mhs.birthday.day;
* puts(mhs.name);

Contoh pertama merupakan instruksi untuk memberikan isi dari field day ke variabel tgl.

Sedangkan contoh kedua merupakan instruksi untuk menampilkan isi dari field name.

Program berikut merupakan contoh yang melibatkan variabel struktur. Mulamula field dari struktur diisi dengan suatu data, kemudian isinya ditampilkan.

/\* File program : student1.c

Mengisi field dr variabel struktur kemudian menampilkannya \*/

#include <stdio.h>

#include <string.h>

struct date { /\* definisi global dari tipe date \*/

int month;

int day;

int year;

};

struct student{ /\* definisi global dari tipe student \*/

char name[30];

struct date birthday;

};

/\* deklarasi global dari variabel mhs\*/

struct student mhs;

main()

{

/\* memberikan nilai kepada field dari struktur mhs \*/

strcpy(mhs.name, “DIKI CANDRA”);

mhs.birthday.month = 5;

mhs.birthday.day = 28;

mhs.birthday.year = 2004;

/\* menampilkan isi semua field dari struktur mhs \*/

printf(“Name : %s\n”, mhs.name);

printf(“Birthday : %d-%d-%d\n”,mhs.birthday.month,

mhs.birthday.day, mhs.birthday.year);

}

Contoh eksekusi :

Name : DIKI CANDRA

Birthday : 5-28-2004

### 3. Menginisialisasi Struktur

Sebuah struktur juga bisa diinisialisasi pada saat dideklarasikan. Hal ini serupa dengan inisialisasi array, yaitu elemen-elemennya dituliskan di dalam sepasang kurung kurawal (‘{ }‘) dengan masing-masing dipisahkan dengan koma. Deklarasi struktur didahului dengan kata kunci static, contoh

static struct ernam bintang =

{“Sagitarius”, 22, 11, 21, 12};

Selengkapnya perhatikan contoh program di bawah ini.

Menentukan ernam berdasarkan data tanggal lahir masukan \*/

#include <stdio.h>

main()

{

struct ernam {

char nama[11];

int tgl\_awal;

int bln\_awal;

int tgl\_akhir;

int bln\_akhir;

};

static struct ernam bintang =

{“Sagitarius”, 22, 11, 21, 12};

int tgl\_lhr, bln\_lhr, thn\_lhr;

printf(“Masukkan tgl lahir Anda (XX-XX-XXXX): “);

scanf(“%d-%d-%d”,&tgl\_lhr, &bln\_lhr, &thn\_lhr);

if((tgl\_lhr >= bintang.tgl\_awal && bln\_lhr ==

bintang.bln\_awal) || (tgl\_lhr <= bintang.tgl\_akhir &&

bln\_lhr == bintang.bln\_akhir))

printf(“Bintang Anda adalah %s\n”, bintang.nama);

else

printf(“Bintang Anda bukan %s\n”, bintang.nama);

}

Contoh eksekusi :

Masukkan tgl lahir Anda (XX-XX-XXXX): 23–11-1972

Bintang Anda adalah Sagitarius

### 4. Array Dan Struktur

Elemen-elemen dari suatu array juga dapat berbentuk sebuah struktur. Misalnya array yang dipakai untuk menyimpan sejumlah data siswa (struct student). Array struktur berdimensi satu ini membentuk suatu erna, dengan barisnya menunjukkan elemen dari array-nya dan kolomnya menunjukkan elemen dari struktur. Dalam hal ini maka deklarasi yang dibutuhkan adalah sebagai berikut :

#define MAKS 20

.

.

.

struct date { /\* definisi dari tipe date \*/

int month;

int day;

int year;

};

struct student { /\* definisi dari tipe student \*/

char name[30];

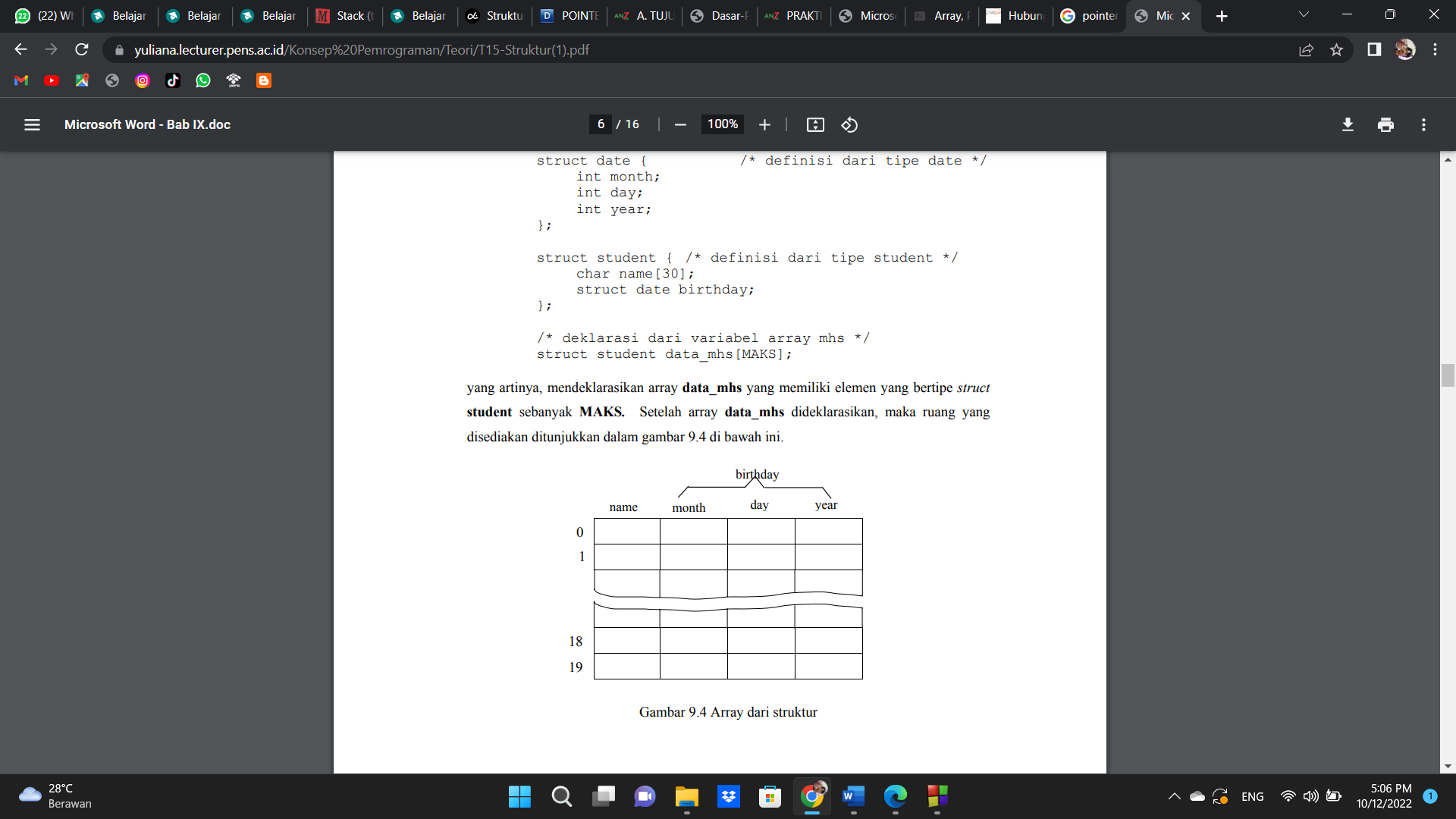
struct date birthday;

};

/\* deklarasi dari variabel array mhs \*/

struct student data\_mhs[MAKS];

yang artinya, mendeklarasikan array data\_mhs yang memiliki elemen yang bertipe struct student sebanyak MAKS. Setelah array data\_mhs dideklarasikan, maka ruang yang disediakan ditunjukkan dalam gambar 9.4 di bawah ini



Elemen-elemen dari array stuktur tersebut bisa diakses dengan cara sebagai berikut :

for (i=0; i<MAKS; i++)

{

printf(“Name : “);

fgets(data\_mhs[i].name,sizeof data\_mhs[i].name,stdin);

printf(“Birthday (mm-dd-yyyy): “);

scanf(“%d-%d-%d”, &data\_mhs[i].birthday.month,

&data\_mhs[i].birthday.day,

&data\_mhs[i].birthday.year);

printf(“\n”);

/\* hapus sisa data dalam penampung keyboard \*/

fflush(stdin);

};

### 5. Struktur Dan Fungsi

Melewatkan sebuah struktur untuk menjadi parameter sebuah fungsi dapat dilakukan sama dengan pengiriman parameter berupa variabel biasa. Fungsi yang mendapat kiriman parameter tersebut juga bisa mengirimkan hasil baliknya yang juga berupa sebuah struktur (pass by reference).

#### a. Melewatkan Elemen Struktur Kedalam Fungsi

Melewatkan parameter berupa elemen struktur dapat dilakukan sebagaimana pengiriman parameter berupa variabel biasa, dapat dilakukan baik secara nilai (pass by value) maupun secara acuan (pass by reference).

Melewatkan elemen struktur sbg parameter fungsi scr nilai \*/

#include <stdio.h>

void cetak\_tanggal(int, int, int);

main()

{

struct date { /\* definisi erna dari tipe date \*/

int month;

int day;

int year;

} today;

printf(“Enter the current date (mm-dd-yyyy): “);

scanf(“%d-%d-%d”, &today.month, &today.day, &today.year);

cetak\_tanggal(today.month, today.day, today.year);

}

void cetak\_tanggal(int mm, int dd, int yy)

{

static char \*nama\_bulan[] = {

“Wrong month”, “January”, “February”, “March”,

“April”, “May”, “June”, “July”, “August”,

“September”, “October”, “November”, “December”

};

printf(“Todays date is %s %d, %d\n\n”,

nama\_bulan[mm],dd,yy);

}

Contoh eksekusi :

Enter the current date (mm-dd-yyyy): 5-29-2001

Todays date is May 29, 2001

Tampak bahwa elemen dari struktur dilewatkan ke fungsi memakai bentuk pengaksesan elemen struktur, berupa :

cetak\_tanggal(today.month, today.day, today.year);

Apabila nilai suatu elemen struktur diharapkan akan diubah oleh fungsi, maka yang dilewatkan haruslah berupa alamat dari elemen struktur (pass by reference). Untuk keperluan ini, operator alamat ditempatkan di depan nama variabel struktur (bukan di depan nama elemen struktur).

Melewatkan elemen struktur sbg parameter fungsi scr acuan \*/

#include <stdio.h>

void tukar\_xy(int \*, int \*);

main()

{

struct koordinat {

int x;

int y;

} posisi;

printf(“Masukkan koordinat posisi (x, y) : “);

scanf(“%d, %d”, &posisi.x, &posisi.y);

printf(“x, y semula = %d, %d\n”, posisi.x, posisi.y);

tukar\_xy(&posisi.x, &posisi.y);

printf(“x, y sekarang = %d, %d\n”, posisi.x, posisi.y);

}

void tukar\_xy(int \*a, int \*b)

{

int z;

z = \*a;

\*a = \*b;

\*b = z;

}

Contoh eksekusi :

Masukkan koordinat posisi (x, y) : 34, 21

x, y semula = 34, 21

x, y sekarang = 21, 34

#### b. Melewatkan Struktur Kedalam Fungsi

Pada program cetak1.c di atas misalnya, semua elemen dari struktur dikirimkan ke fungsi cetak\_tanggal(), dengan maksud nilai elemen dari struktur akan ditampilkan di erna. Untuk keadaan seperti ini, lebih baik erna parameter fungsi diubah menjadi bentuk struktur, sehingga parameter fungsi tidak lagi sebanyak tiga buah, melainkan hanya satu. Selengkapnya, perhatikan program di bawah ini.

Melewatkan struktur sebagai parameter fungsi \*/

#include <stdio.h>

struct date { /\* definisi global dari tipe date \*/

int month;

int day;

int year;

};

void cetak\_tanggal(struct date);

main()

{

struct date today;

printf(“Enter the current date (mm-dd-yyyy): “);

scanf(“%d-%d-%d”, &today.month, &today.day, &today.year);

cetak\_tanggal(today);

}

void cetak\_tanggal(struct date now)

{

static char \*nama\_bulan[] = {

“Wrong month”, “January”, “February”, “March”,

“April”, “May”, “June”, “July”, “August”,

“September”, “October”, “November”, “December”

};

printf(“Todays date is %s %d, %d\n\n”,

nama\_bulan[now.month], now.day, now.year);

}

Contoh eksekusi :

Enter the current date (mm-dd-yyyy): 5-29-2001

Todays date is May 29, 2001

### 6. Struktur Dan Pointer (pointer ke struktur)

Jika sebuah struktur mengandung banyak field dan diputuskan bahwa keseluruhan field-nya akan diubah oleh fungsi, maka cara yang efisien adalah dengan melewatkan (passing) alamat dari struktur. Dengan demikian pada pendefinisian fungsi, parameter formalnya berupa pointer yang menunjuk ke struktur.

Masalah pointer ke struktur dapat diterapkan dalam program posisi1.c. Argumen dari fungsi tukar\_xy() dapat disederhanakan menjadi satu ernama saja, yakni sebagai berikut :

void tukar\_xy(struct koordinat \*pos\_xy)

{

int z;

z = (\*pos\_xy).x;

(\*pos\_xy).x = (\*pos\_xy).y;

(\*pos\_xy).y = z;

}

Pada definisi fungsi di atas,

struct koordinat \*pos\_xy

menyatakan bahwa pos\_xy adalah pointer yang menunjuk ke obyek bertipe struktur koordinat. Adapun penulisan :

(\*pos\_xy).x

menyatakan : elemen ernama x yang ditunjuk oleh pointer pos\_xy Perlu diperhatikan bahwa penulisan tanda kurung seperti pada contoh (\*pos\_xy).x merupakan suatu keharusan. Sebab

\*pos\_xy.x

mempunyai makna yang berbeda dengan

(\*pos\_xy).x

Ungkapan \*pos\_xy.x mempunyai makna yaitu : “yang ditunjuk oleh pos\_xy.x “ (sebab operator titik mempunyai prioritas yang lebih tinggi daripada operator \*).

Fungsi parameternya berupa pointer yg menunjuk ke struktur \*/

#include <stdio.h>

struct koordinat

{ int x;

int y;

};

void tukar\_xy(struct koordinat \*);

main()

{

struct koordinat posisi;

printf(“Masukkan koordinat posisi (x, y) : “);

scanf(“%d, %d”, &posisi.x, &posisi.y);

printf(“x, y semula = %d, %d\n”, posisi.x, posisi.y);

tukar\_xy(&posisi);

printf(“x, y sekarang = %d, %d\n”, posisi.x, posisi.y);

}

void tukar\_xy(struct koordinat \*pos\_xy)

{

int z;

z = (\*pos\_xy).x;

(\*pos\_xy).x = (\*pos\_xy).y;

(\*pos\_xy).y = z;

}

Contoh eksekusi :

Masukkan koordinat posisi (x, y) : 34, 21

x, y semula = 34, 21

x, y sekarang = 21, 34

Bentuk semacam :

(\*pos\_xy).x

dapat ditulis dengan bentuk lain menjadi

pos\_xy->x

160

Dalam C operator -> (berupa tanda minus – diikuti dengan tanda lebih dari >) disebut sebagai operator panah. Dengan menggunakan operator panah, maka fungsi tukar\_xy() dalam program posisi2.c dapat ditulis menjadi

void tukar\_xy(struct koordinat \*pos\_xy)

{

int z;

z = pos\_xy->x;

pos\_xy->x = pos\_xy->y;

pos\_xy->y = z;

}

#### Kesimpulan :

• Struktur adalah pengelompokan variabel-variabel yang bernaung dalam satu nama yang sama, namun tipe datanya tidak harus sama.

• Variabel-variabel yang membentuk suatu struktur, selanjutnya disebut sebagai elemen dari struktur atau field.

• Suatu struktur didefinisikan dengan menggunakan kata kunci struct.

• Elemen dari suatu variabel struktur dapat diakses dengan menyebutkan nama variabel struktur diikuti dengan operator titik (‘.’) dan nama dari elemen strukturnya.

• Sebuah struktur juga bisa diinisialisasi pada saat dideklarasikan. Hal ini serupa dengan inisialisasi array, yaitu elemen-elemennya dituliskan di dalam sepasang kurung kurawal (‘{ }‘) dengan masing-masing dipisahkan dengan koma.

• Elemen-elemen dari suatu array juga dapat berbentuk sebuah struktur (array of struct).

• Melewatkan sebuah struktur untuk menjadi parameter sebuah fungsi dapat dilakukan sama dengan pengiriman parameter berupa variabel biasa. Fungsi yang mendapat kiriman parameter tersebut juga bisa mengirimkan hasil baliknya yang juga berupa sebuah struktur (pass by reference).

• Jika sebuah struktur mengandung banyak field dan diputuskan bahwa keseluruhan field-nya akan diubah oleh fungsi, maka cara yang efisien adalah dengan melewatkan (passing) alamat dari struktur. Dengan demikian pada pendefinisian fungsi, parameter formalnya berupa pointer yang menunjuk ke struktur (pointer to struct).

# 

# BAB III

# PENUTUP

Demikianlah Makalah ini saya buat dengan sebaik-baiknya. Terima kasih buat Orang tua yang selalu mendukung saya baik materil maupun moril,dan terima kasih untuk abang dan kakak assisten laboratorium paket applikasi, juga terima kasih buat kawan-kawan yang telah membantu dalam mengerjakan makalah ini.semoga makalah ini dapat berguna buat kita semua dan buat angkatan ke depannya kelak. lebih dan kurangnya saya mohon maaf, kepada abang dan kakak saya ucapkan terima kasih.

Wassalammu’alaikum Wr.Wb.

DAFTAR PUSTAKA

* <https://aglabridgemedia.com/abm/blog/hubungan-pointer-dengan-array-string-fungsi-dan-pointer-lain/3086419725>
* <https://adoc.pub/praktikum-11-pointer-1.html>
* <https://adoc.pub/a-tujuan-1-menjelaskan-tentang-konsep-dari-variabel-pointer-.html>
* <https://docplayer.info/46483060-Pointer-konsep-dasar.html>
* <https://www.belajarstatistik.com/blog/2021/12/04/pointer-dalam-bahasa-c/#:~:text=Seperti%20yang%20dijelaskan%20di%20awal,ada%20di%20suatu%20alamat%20memori>.
* <https://www.petanikode.com/c-pointer/>
* <https://www.markijar.com/2015/05/stack-tumpukan-dalam-bahasa-c.html#:~:text=Stack%20merupakan%20jenis%20linked%20list,terakhir%20akan%20diproses%20%2F%20dikeluarkan%20pertama>.
* <https://www.petanikode.com/c-struct/>
* <https://www.petanikode.com/c-array/>
* <https://www.petanikode.com/c-fungsi/>