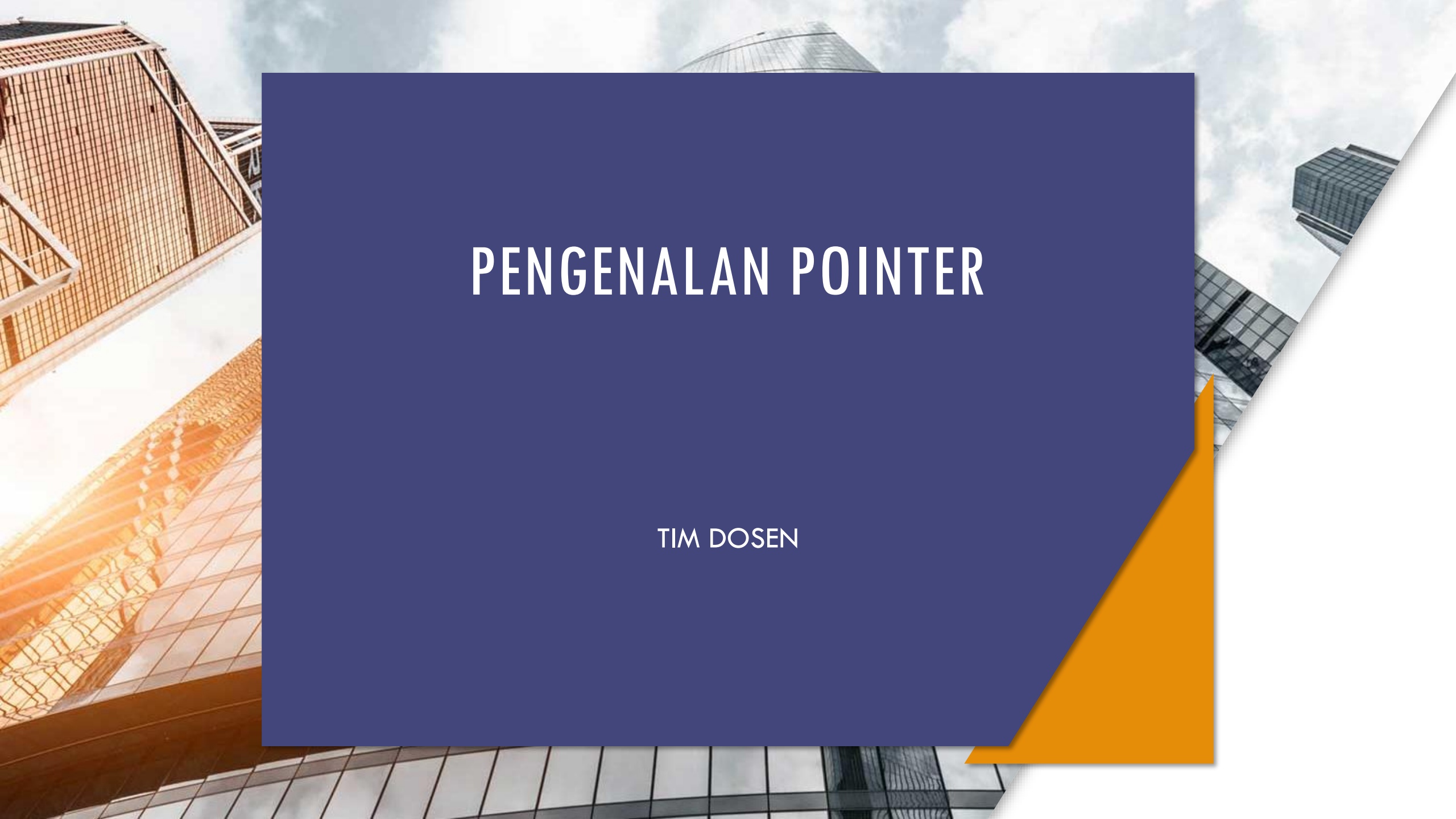


PENGENALAN POINTER

TIM DOSEN

OUTLINE

GARIS BESAR PEMBAHASAN

a. Alamat dan Pointer
b. Deklarasi dan Mengisi Variabel Pointer
c. Mengakses Nilai yang ditunjuk Pointer
d. Pointer Void
e. Pointer dan Array
f. Pointer dan String
g. Pointer dan Fungsi
   - Pointer sebagai Argument Fungsi
   - Pointer sebagai Keluaran Fungsi
   - String sebagai Argumen

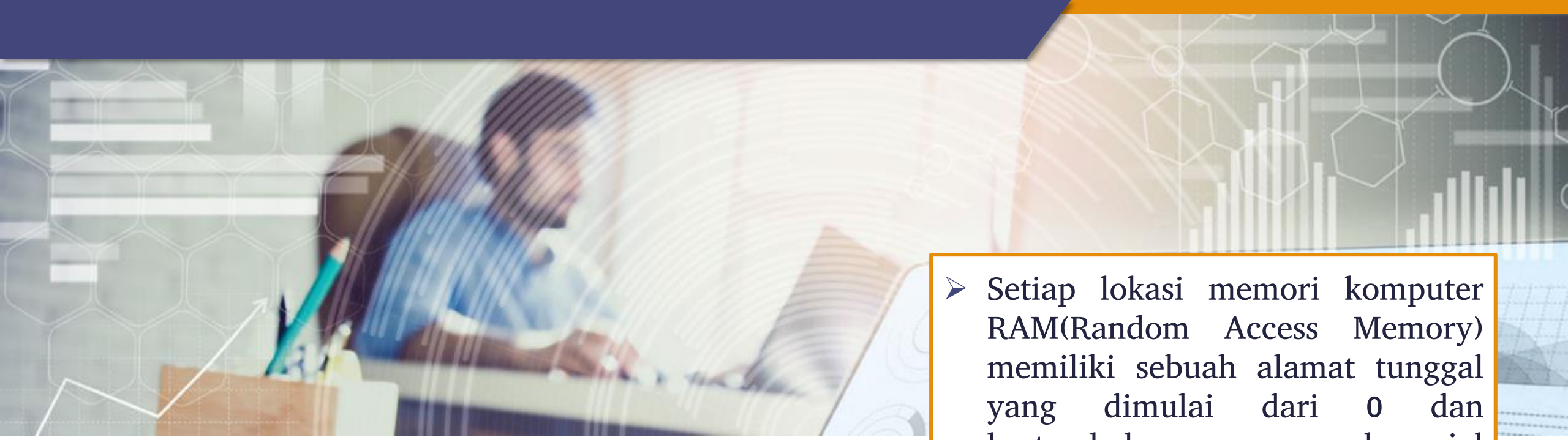
ALAMAT DAN POINTER

PENGENALAN POINTER

- **Pointer berisi alamat mesin.**
- Pointer menunjuk kepada nama yang diacunya.
- Sehingga informasi yang disimpan pada nama dapat diakses.
- Pointer dapat menunjuk kepada pointer *(pointer to pointer).*

- *Pointer* memungkinkan **alokasi secara dinamis**.
- Memori baru dialokasi berdasarkan kontrol pemrograman.
- Jika tidak dibutuhkan lagi, ruang memori yang dialokasi dapat di-dealokasi /dikembalikan ke mesin.
- Pemrogram harus berhati-hati dan cermat dalam melakukan alokasi/dealokasi, Jika tidak maka suatu **saat memori tidak cukup** atau *address* mengacu ke suatu yang tidak terdefinisi dan dapat **menyebabkan computer 'Hang'**

- **Pointer** diartikan sebagai penunjuk alamat suatu lokasi memori di mana nilai data tersebut tersimpan.
- Pointer bernilai dari `0 – 65535`, jika Anda memiliki RAM sebesar `64K`.
- **Lokasi memori** tersebut diwakili oleh sebuah variabel yang mempunyai nama, atau juga lokasi bebas dalam memori.
- Lokasi memori tersebut memiliki **pointer**
- Penunjuk suatu variable
- variabel pointer berisi suatu nilai yang menyatakan alamat suatu lokasi.



PEMANFAATAN POINTER

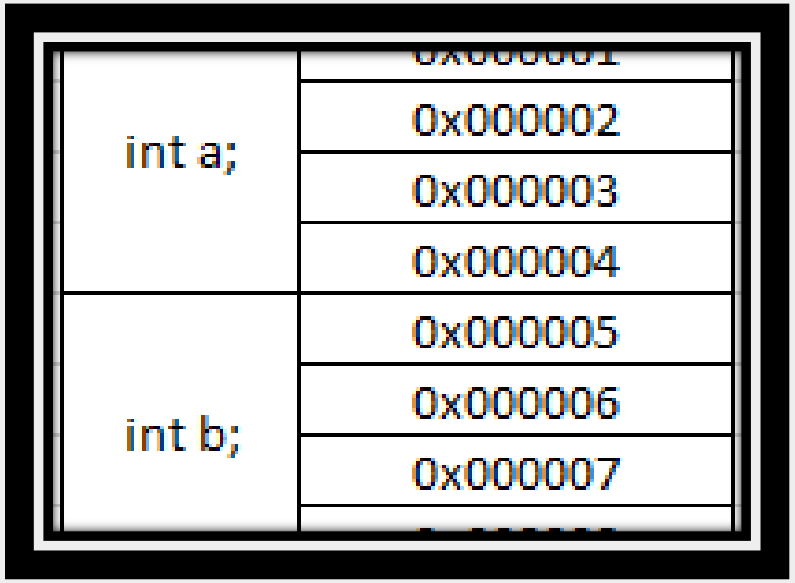
- Setiap lokasi memori komputer RAM(Random Access Memory) memiliki sebuah alamat tunggal yang dimulai dari 0 dan bertambah secara sekuensial hingga maksimum memori yang tersedia.
- Sebuah komputer menyimpan informasi pada memorinya.
- Memori komputer terbagi dalam sejumlah lokasi yang disebut sebagai ***storage cells***.

- Setiap lokasi memiliki address (alamat).
- Karena pointer menunjuk ke alamat memori, kompiler dapat secara langsung mencari data yang diinginkan pada lokasi memori di mana data tersimpan.
- Alamat lokasi penyimpanan data diketahui oleh kompiler pada saat variabel dideklarasi.
- Dengan menggunakan alamat tunggal ini, komputer dapat dengan mudah menelusuri memori komputer.

- Apabila kompiler akan menyimpan hasil perhitungan di dalam memori, kompiler mencari alamat lokasi variabel tempat hasil perhitungan itu disimpan, dan menyimpannya pada alamat tersebut.
- Kadangkala dalam program yang besar, penghematan memori wajib untuk dilakukan.
- Mekanisme *copy* dan *paste* nilai variabel satu kedalam variabel lain, akan sangat memboroskan memori.
- Dengan mekanisme pointer, suatu variabel dalam suatu fungsi dapat diakses oleh fungsi yang lain.

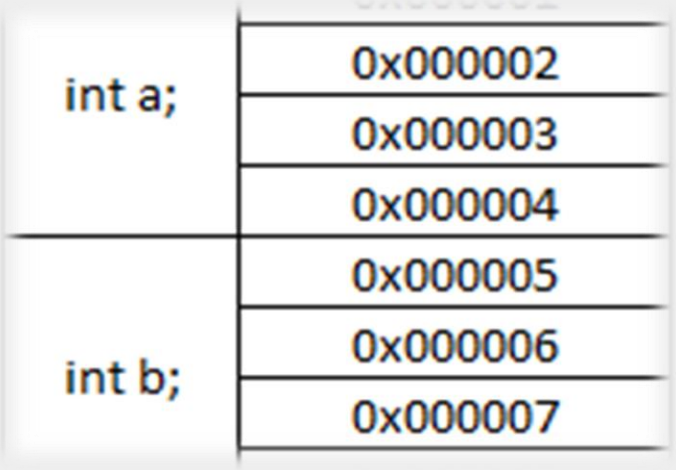
ALAMAT MEMORI

- Misal variabel x dan terletak di memori 0x000002.
- Kemudian didefinisikan nilai 100 ke dalam variabel x, maka processor harus membawa nilai 100 tersebut ke dalam variabel x yang terletak di alamat memori 0x000002.
- Setiap variabel ternyata memiliki ukuran byte yang berbeda-beda dalam memori.
- Sebagai contoh suatu variabel bertipe int memiliki ukuran 4 byte dalam memori. Maka variabel tersebut akan menempati 4 kapling lokasi dalam memori, misalkan 0x000001, 0x000002, 0x000003, dan 0x000004.



LANJUT

- Jika terdapat dua buah variabel bertipe int yang bersebelahan, maka alamat variabel pertama terletak di 0x000001 dan variabel kedua terletak dialamat 0x000005.
- Memori menggunakan bilangan heksadesimal yang ditandai dengan awalan ‘0x’, sehingga jika suatu variabel menempati blok ke sepuluh dalam memori, maka alamatnya adalah 0x00000a



OPERATOR POINTER

MACAM-MACAM OPERATOR POINTER

1. Operator Dereference (&)

- **Operator Dereference** sering juga disebut sebagai **Operator alamat**
- Berfungsi mendeklarasikan suatu variabel di dalam penggantian memori
- Mendapatkan/melihat alamat memori yang dimiliki oleh variabel tersebut.

2. Operator Reference (*)

- Operator Reference disebut juga value pointed by yang berfungsi mengakses secara langsung nilai yang terdapat di dalam variabel yang berpointer , hal tersebut dilakukan dengan menambahkan (*) .
- Mendapatkan isi/nilai dari sebuah memori berdasarkan alamat memori.

LANJUT

1. Operator Dereference (&)

```
#include <iostream>

using namespace std;

 int main(){

int a = 5;

cout<<"Alamat Variabel a adalah :"<<&a<<endl;

cout<<"Nilai Variabel a adalah :"<<a<<endl;

return 0;

}
```

```
Alamat Variabel a adalah :0x28ff1c
Nilai Variabel a adalah :5

Process returned 0 (0x0)   execution time : 0.028 s
Press any key to continue.
```

2. Operator Reference (*)

- Pointer (*) Lebih detail slide berikut

KONSEP UTAMA POINTER

(1) Pointer constants

- Alamat memori yang sebagai ***pointer constants*** tidak bisa diganti namun dapat digunakan untuk menyimpan sebuah nilai.

(2) Pointer Values

- Nilai pada suatu alamat memori yang menunjukkan alamat dari lokasi memori

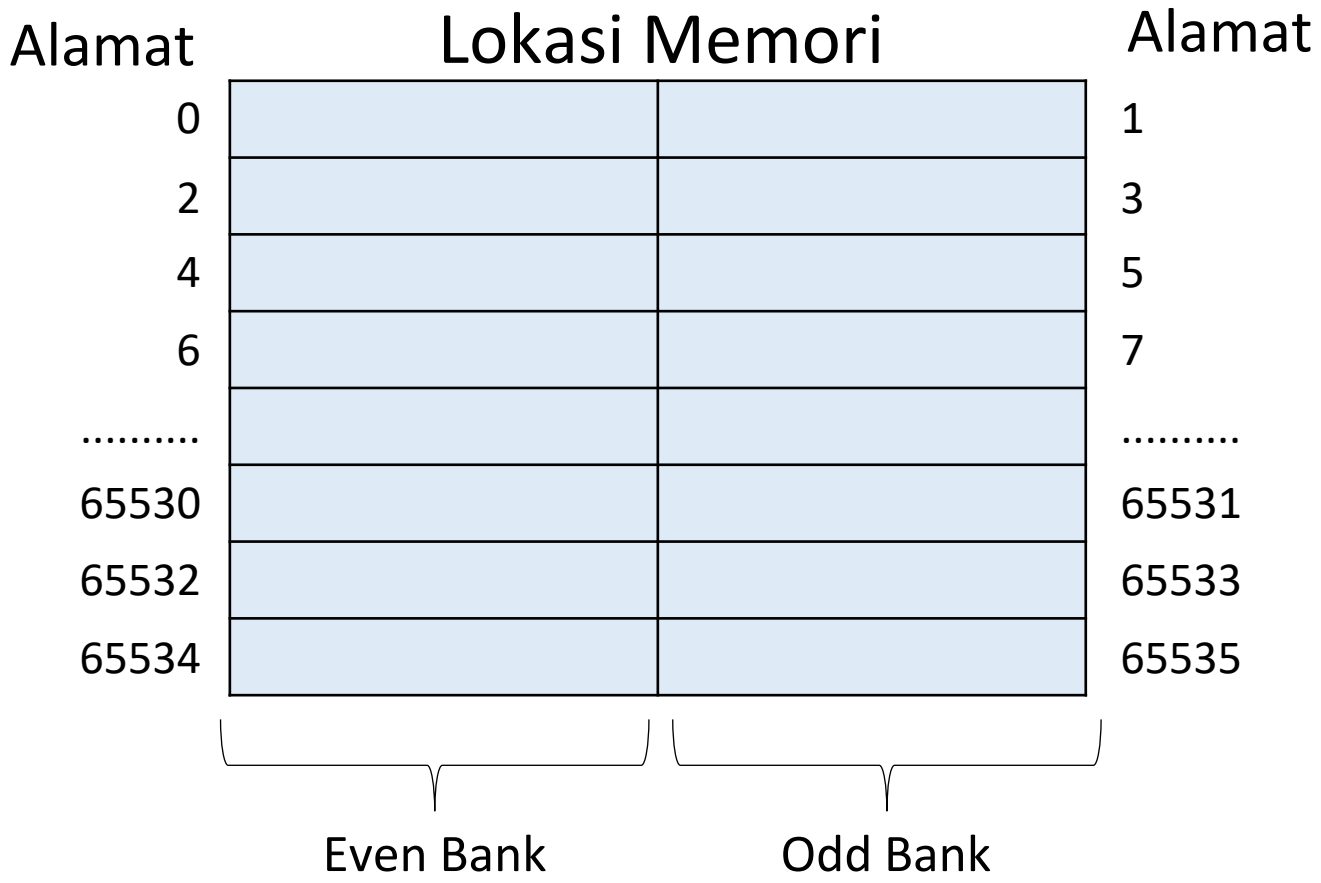
(3) Pointer Variables

- Variabel menyimpan alamat memori adalah pointer variable
- Variabel yang menyimpan pointer value dari sebuat variable lain

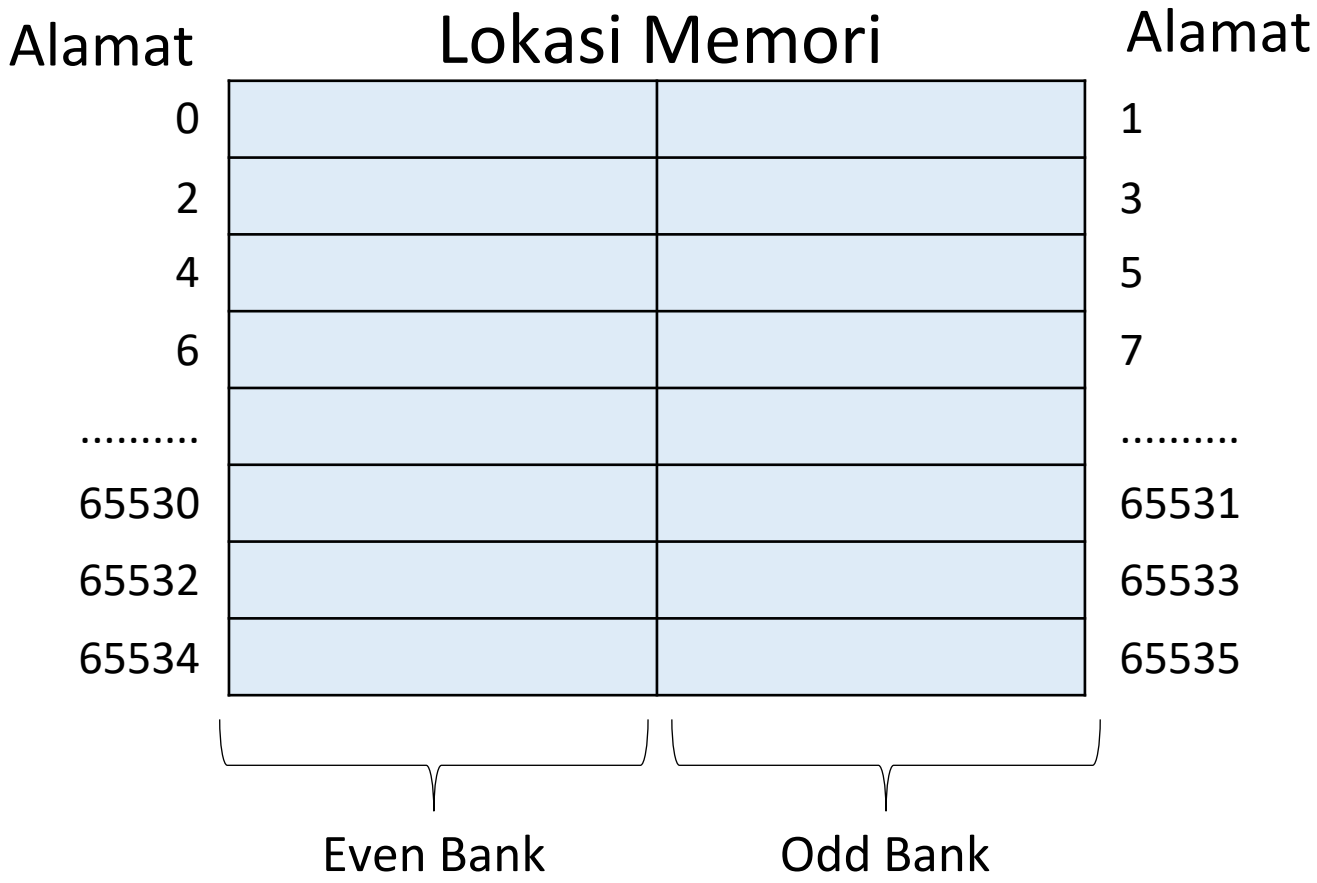
POINTER CONSTANTS - 1

- Asumsikan bahwa sebuah komputer memiliki ukuran memori 64kB. Di mana 1 kB = 1024 Byte.
- Sehingga jumlah lokasi memori adalah 1024 * 64 = 65536 lokasi.
- Secara *logic*, keseluruhan 65536 lokasi pada memori komputer memiliki urutan yang sekuensial dari 0 – 65535. Namun secara fisik, lokasi memori tersebut dikelompokkan dalam penyimpanan genap (***even bank***) dan penyimpanan ganjil (**odd bank**).

POINTER CONSTANTS

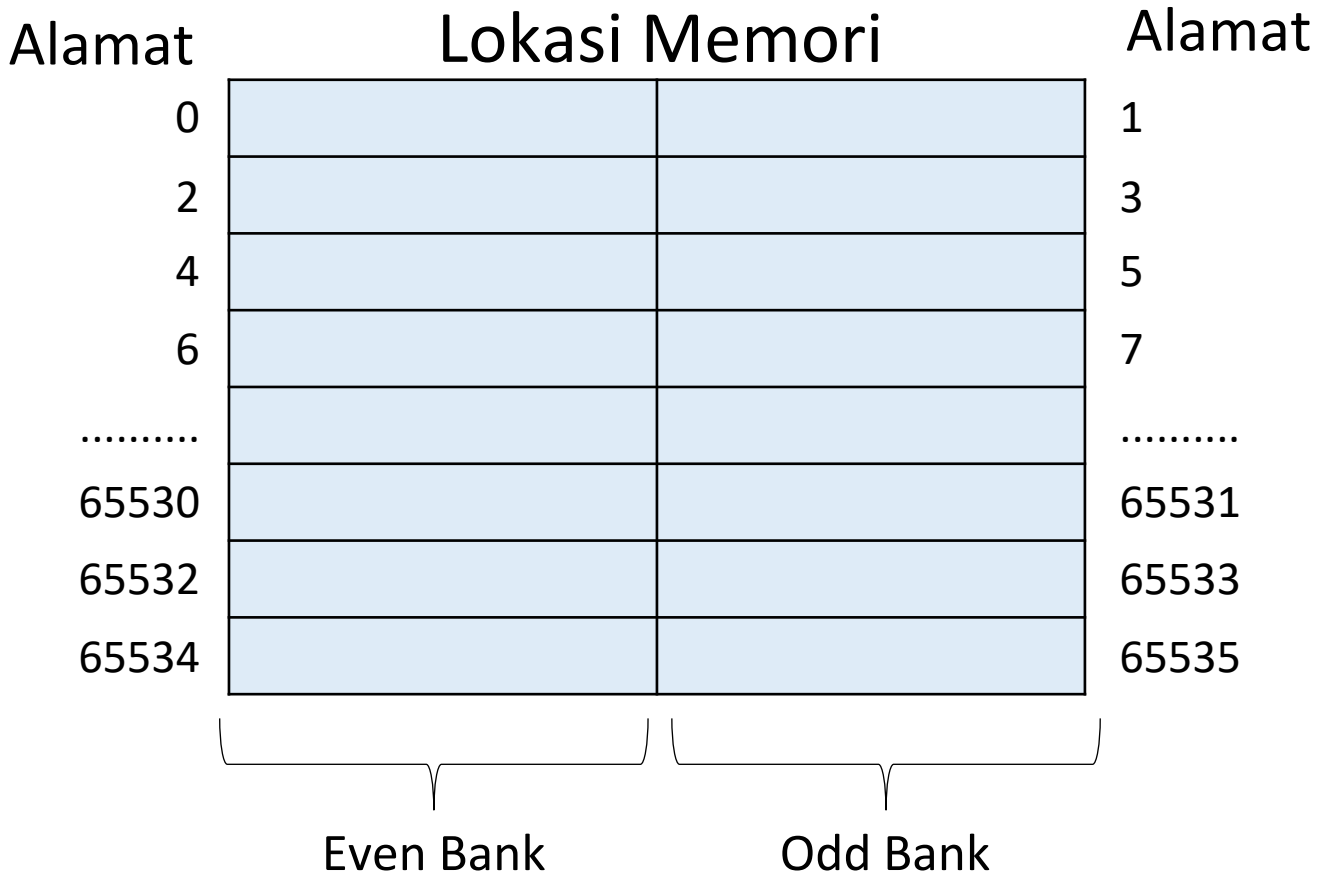


POINTER CONSTANTS



- **Even bank** merupakan sejumlah lokasi memori yang memiliki alamat genap
- **Odd bank** merupakan sejumlah lokasi memori yang memiliki alamat memori ganjil

POINTER CONSTANTS



Alamat memori inilah yang disebut sebagai ***pointer constants***. Alamat tersebut tidak bisa diganti namun dapat digunakan untuk menyimpan sebuah nilai.

POINTER VALUES - 2

- Misalkan dalam bahasa c++, kita deklarasikan seperti di bawah:
  - **int i =100, j=200, k=300;**
- Deklarasi tersebut memberitahukan kepada compiler untuk melakukan aktfitas di bawah ini:
  1. Pesan tempat untuk tiga buah integer pada memori.
  2. Asosiasikan variabel i, j, dan k pada lokasi memori.
  3. Simpan nilai 100 pada lokasi i, nilai 200 pada lokasi j, dan nilai 300 pada lokasi k.

POINTER VALUES

- Misalkan dalam bahasa c++, kita deklarasikan seperti di bawah:

  int i =100, j=200, k=300;

- Deklarasi tersebut memberitahukan kepada compiler untuk melakukan aktfitas di bawah ini:
  1. Pesan tempat untuk tiga buah integer pada memori.
  2. Asosiasikan variabel i, j, dan k pada lokasi memori.
  3. Simpan nilai 100 pada lokasi i, nilai 200 pada lokasi j, dan nilai 300 pada lokasi k.

	Alamat	Lokasi Memori		
	0			1
	2			3
	4			5
	6			7
	.....			.....
i	65530	100		65531
j	65532	200		65533
k	65534	300		65535

POINTER VALUES

- Misalkan dalam bahasa c++, kita deklarasikan seperti di bawah:

int i =100, j=200, k=300;

- Deklarasi tersebut memberitahukan kepada compiler untuk melakukan aktfitas di bawah ini:
  1. Pesan tempat untuk tiga buah integer pada memori.
  2. Asosiasikan variabel i, j, dan k pada lokasi memori.
  3. Simpan nilai 100 pada lokasi i, nilai 200 pada lokasi j, dan nilai 300 pada lokasi k.

Compiler telah memilih alamat 65530 untuk variabel i, alamat 65532 untuk variabel j, dan alamat 65534 untuk variabel k. Alamat yang diberikan untuk variabel i, j, dan k inilah yang disebut sebagai **pointer values**.

Alamat		Lokasi Memori		
	0			1
	2			3
	4			5
	6			7
	.....			.....
i	65530	100		65531
j	65532	200		65533
k	65534	300		65535

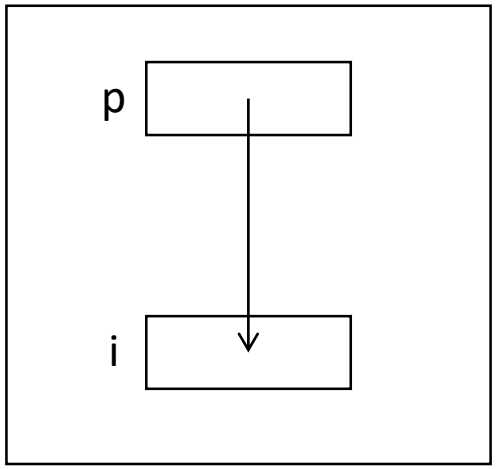
POINTER VARIABLES - 3

- Jika **i** merupakan variabel yang memiliki nilai 100 (int i=100) dan alamat **i** disimpan pada variabel **p**, maka dapat digambarkan sebagai berikut:

Alamat		Lokasi Memori	
	0		
	2		
	4		
	6		
	.....		
p	50000	**65534**	
	.....		
i	65534	**100**	

Representasi Fisik

Variabel **p** inilah yang disebut sebagai ***pointer variable*** (variabel pointer)



Representasi Logis

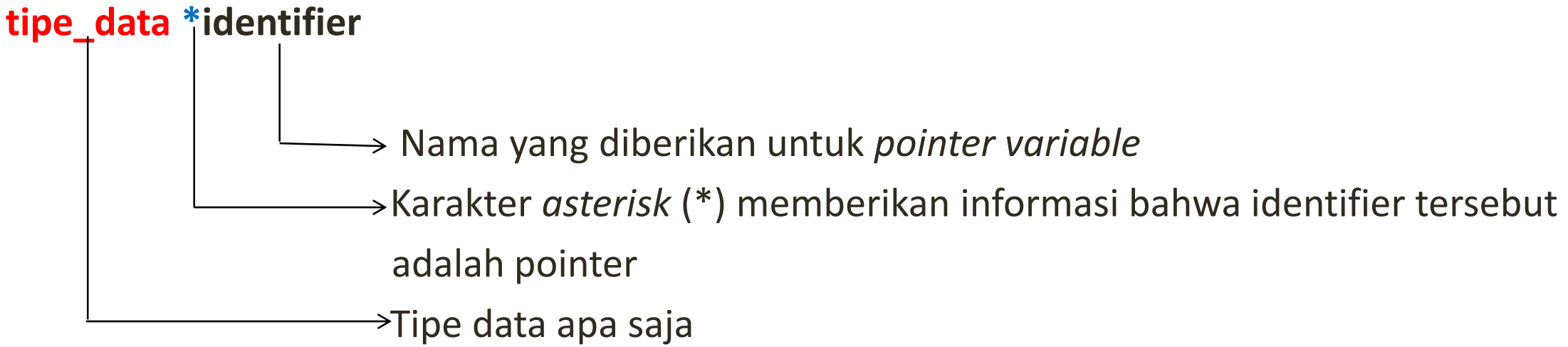
DEKLARASI DAN MENGISI VARIABEL POINTER

VARIABLE POINTER (*)

- Reference (*) merupakan suatu operator yang berfungsi menyatakan suatu variabel adalah variabel pointer.
- Sama halnya dengan operator deference, peletakan simbol operator reference diletakan diawal variabel.
- Operator reference ini akan membuat suatu variabel pointer untuk menampung alamat

DEKLARASI POINTER

- Deklarasi pointer pada bahasa pemrograman c++:



- Contoh deklarasi:

  int *a, b, c; □

  a merupakan ***pointer variable*** yang dapat menyimpan alamat memori dari variabel yang bertipe data integer, sedangkan ***b*** dan ***c*** merupakan variabel integer biasa.

CONTOH DEKLARASI ARRAY

```
int * i;                    // pointer ke integer
float *t;                   // pointer ke float
char *cc;                   // pointer ke karakter
FILE *fileku;               // Hadle sebuah file
char **argv;          // pointer ke pointer ke karakter
int (*tabel ) [13];         // pointer ke array dengan 13 elemen integer
int *tabel [13] ;           // pointer ke array dengan 13 elemen integer
                               yang bertipe pointer ke integer
void* p;                    // pointer ke objek yang tidak diketahui tipenya
```

MENGAKSES VARIABEL YANG DITUNJUK POINTER

- Contoh kode program untuk penggunaan pointer dalam bahasa c++:

```
int a;
a = 5;
int *p;
p = &a;
cout << "*p = " << *p << endl;
```

Output:
*p = 5

- Pointer p meyimpan nilai berupa alamat dari variabel a, sehingga *p dapat merepresentasikan nilai dari a.
- Karakter * di atas disebut sebagai ***indirectional operator*** atau ***dereferencing operator***.

DANGLING POINTERS

- Variabel Pointer yang tidak menyimpan alamat memori yang valid disebut ***dangling pointer***.
- Contoh:

```
int a = 500;

int *p;

/*p merupakan dangling pointer*/
```

- Sebuah program seharusnya tidak memiliki dangling pointer.

NULL POINTERS

- ***Null pointer*** didefinisikan sebagai pointer khusus yang tidak menunjuk ke alamat memori manapun.
- Contoh:

  int *p = NULL;
- Berdasarkan kode program di atas, variabel pointer p tidak menunjuk lokasi memori manapun.
- Sebelum menggunakan varibel pointer, sebaiknya dilakukan pengecekan terlebih dahulu apakah variabel pointer tersebut menunjuk ke **NULL** atau ke alamat yang valid.

```
if ( p == NULL )
{
        /* code to perform necessary actions */
}
else
{
        /* code to access data using pointer */
}
```

CONTOH PENJUMLAHAN MENGGUNAKAN POINTER

```
include <stdio.h>


int A, B, *pA, *pB, hasil;
A = 13;
B = 7;


pA = &A;
pB = &B;


hasil = *pA + *pB;


cout << "Hasil = " << hasil ;
```

Output:
Hasil = 20

KOMPATIBILITAS POINTER

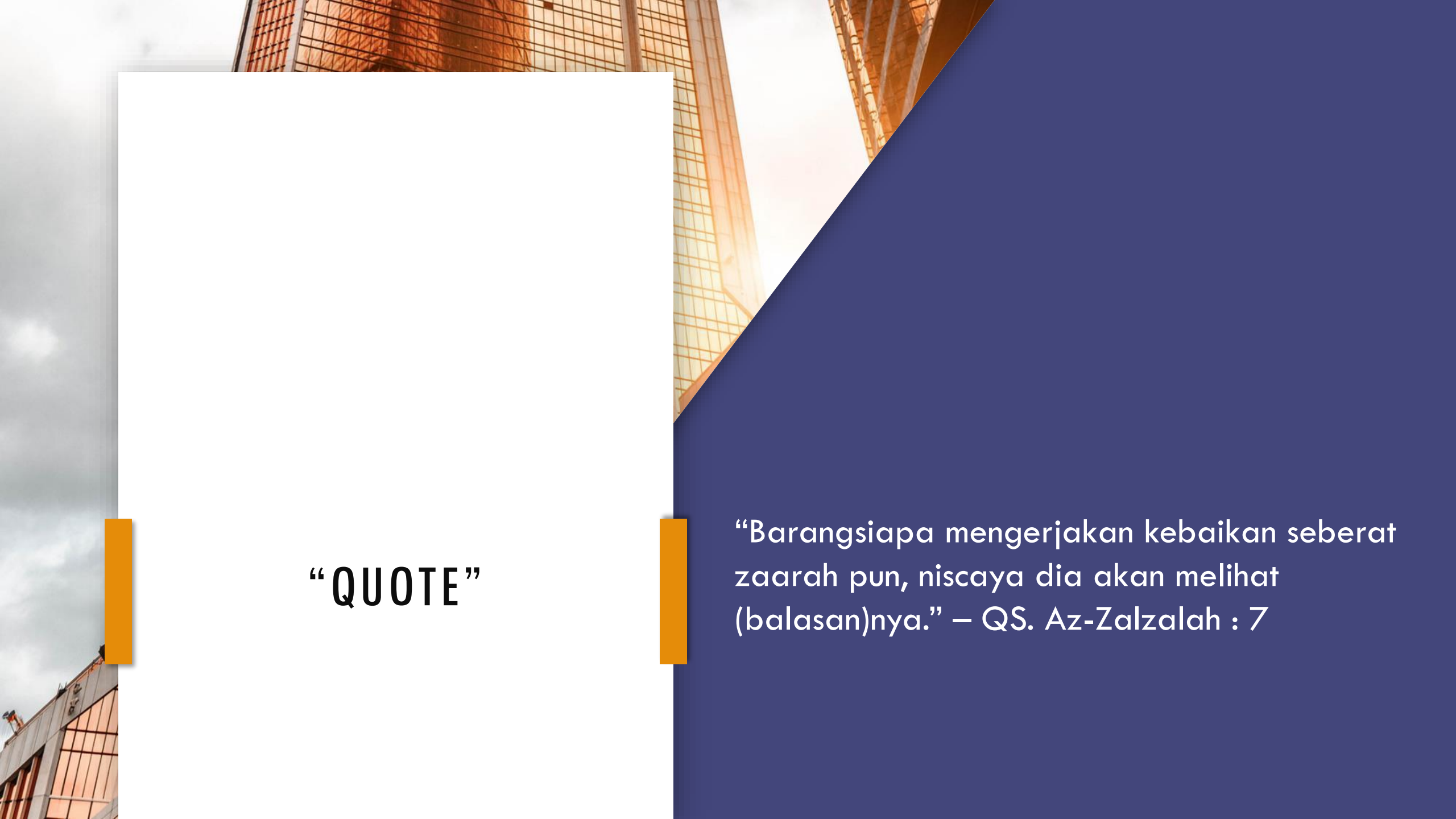
- Variabel pointer tidak boleh menunjuk/menyimpan alamat variabel yang memiliki tipe data berbeda dengan tipe datanya.

- Contoh program yang benar:

```
int a=10;
int *p;
p = &a;
```

- Contoh program yang salah:

```
int a=10;
float *p;
p = &a;
```



“QUOTE”

“Barangsiapa mengerjakan kebaikan seberat zaarah pun, niscaya dia akan melihat (balasan)nya.” – QS. Az-Zalzalah : 7