

# ALGORITHMS & DATA STRUCTURES

## 02 – Pointer & Array 1

# REVIEW

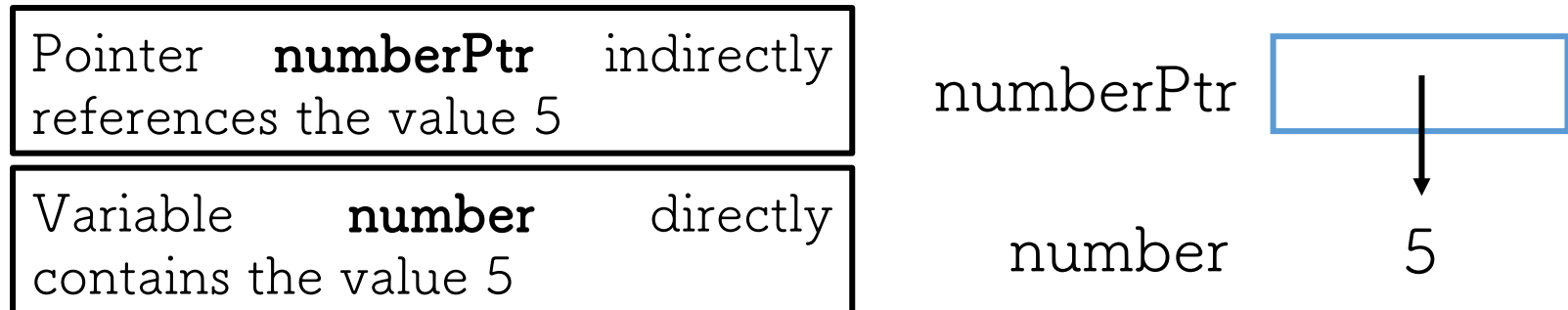
- Control Structures
- Functions
- Structures
- Unions
- Enumerations

# OUTLINE

- Pointer Initialization
- Pointer Operations
- Array Declaration
- Array Initialization
- Array Access

# POINTER VARIABLE DEFINITIONS & INITIALIZATION

- Pointers are variables whose values are **memory addresses**
- A pointer contains an address of a variable that contains a specific value
- A variable name directly references a value, but a pointer indirectly references a value



# POINTER VARIABLE DEFINITIONS & INITIALIZATION

- Pointers must be defined before they can be used

```
data_type *pointer_name;
```

- Example

```
int *iPtr;
```

- What is the data type of **iPtr** ?

```
int *iPtr;
```

- What is the data type of **\*iPtr** ?

```
int *iPtr;
```

# POINTER VARIABLE DEFINITIONS & INITIALIZATION

- Pointers should be initialized either when they are defined or in an assignment statement
- A pointer may be initialized to NULL or an address
- A pointer with the value NULL points to nothing

# POINTER OPERATORS

```
int number = 5;  
int *numberPtr;  
  
numberPtr = &number;  
  
printf("%d", *numberPtr);
```

---

numberPtr

0xFF08

number

5

---

- The address operator (&) returns the **address of its operand** (variable)
- The indirection operator / dereferencing operator (\*) returns the **value of the object** to which its operand (pointer) points

# EXAMPLES

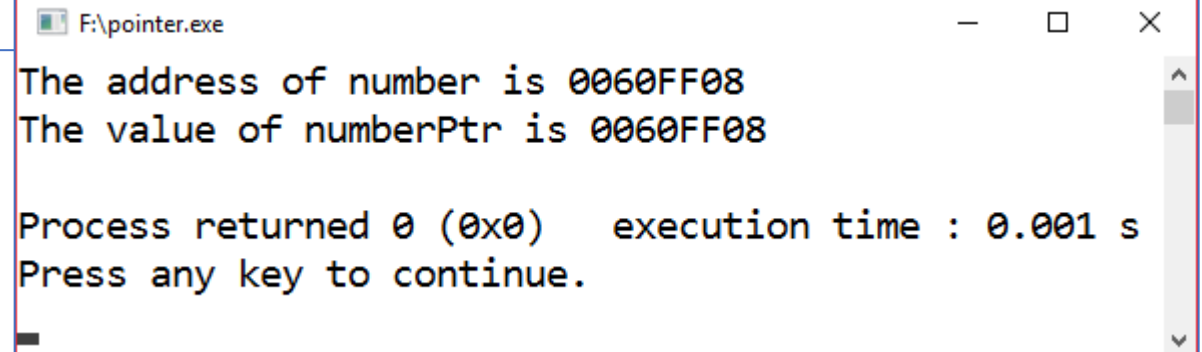
```
#include <stdio.h>

int main()
{
    int number = 8;
    int *numberPtr;

    numberPtr = &number;

    printf("The address of number is %p\n",&number);
    printf("The value of numberPtr is %p\n",numberPtr);

    return 0;
}
```



```
F:\pointer.exe
The address of number is 0060FF08
The value of numberPtr is 0060FF08

Process returned 0 (0x0)    execution time : 0.001 s
Press any key to continue.
```



# EXAMPLES

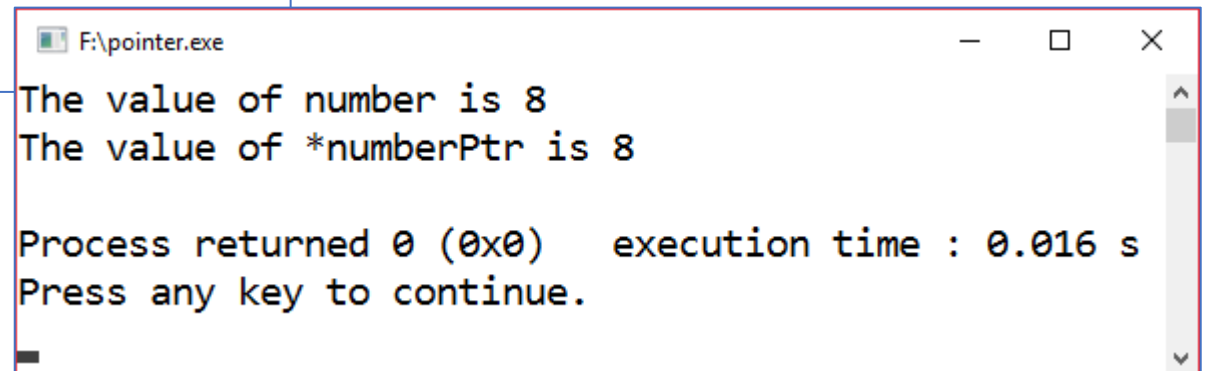
```
#include <stdio.h>

int main()
{
    int number = 8;
    int *numberPtr;

    numberPtr = &number;

    printf("The value of number is %d\n", number);
    printf("The value of *numberPtr is %d\n", *numberPtr);

    return 0;
}
```



```
F:\pointer.exe
The value of number is 8
The value of *numberPtr is 8

Process returned 0 (0x0)   execution time : 0.016 s
Press any key to continue.
```

# EXAMPLES

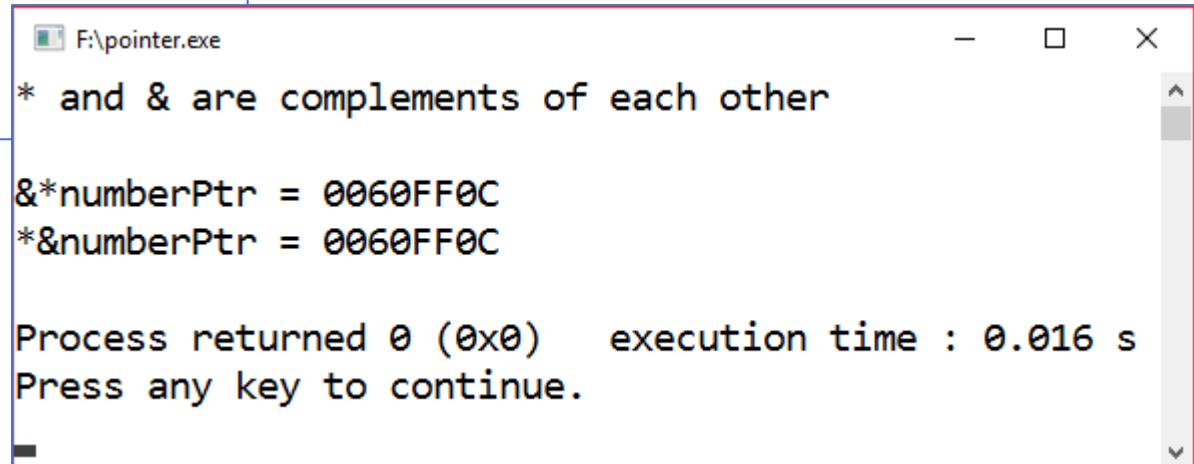
```
#include <stdio.h>

int main()
{
    int number = 8;
    int *numberPtr;

    numberPtr = &number;

    printf("* and & are complements of each other\n\n");
    printf("&*numberPtr = %p\n", &*numberPtr);
    printf("*&numberPtr = %p\n", *&numberPtr);

    return 0;
}
```



```
F:\pointer.exe
* and & are complements of each other

&*numberPtr = 0060FF0C
*&numberPtr = 0060FF0C

Process returned 0 (0x0)   execution time : 0.016 s
Press any key to continue.
```

# EXAMPLES

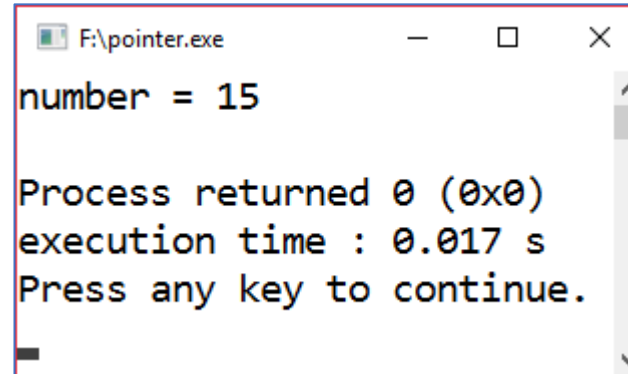
```
#include <stdio.h>

int main()
{
    int number = 8;
    int *numberPtr;

    numberPtr = &number;

    *numberPtr = number + 7;
    printf("number = %d\n", number);

    return 0;
}
```



```
F:\pointer.exe
number = 15

Process returned 0 (0x0)
execution time : 0.017 s
Press any key to continue.
```

# CALL BY POINTER

```
#include <stdio.h>

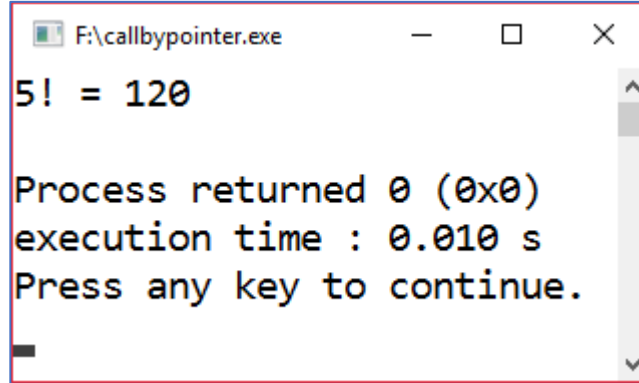
void factorial(int *n)
{
    int i;

    for(i = *n - 1; i > 1; i--)
    {
        *n *= i;
    }
}

int main()
{
    int number = 5;

    factorial(&number);
    printf("5! = %d\n", number);

    return 0;
}
```



F:\callbypointer.exe

5! = 120

Process returned 0 (0x0)  
execution time : 0.010 s  
Press any key to continue.

# ARRAY

- Array: a group of memory locations → **same name & same type**
- To refer to a particular location or element in the array, we specify the **name** of the array and the **position number** of the particular element in the array
- The first element in every array is the **zeroth (0<sup>th</sup>) element**
- The position number contained within square brackets is more formally called a **subscript** or **index** → must be an integer or integer expression

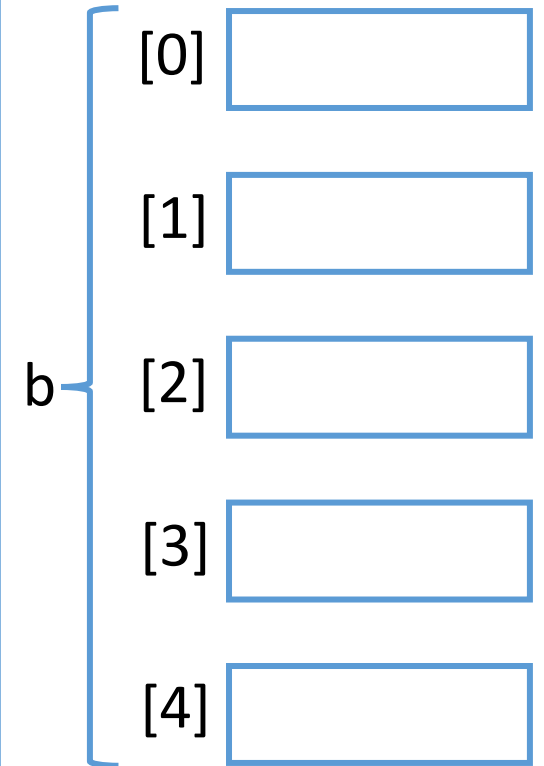
# ARRAY DECLARATION

- Syntax

```
element_data_type array_name[size];
```

- Example

```
int a[100];  
int b[5];
```



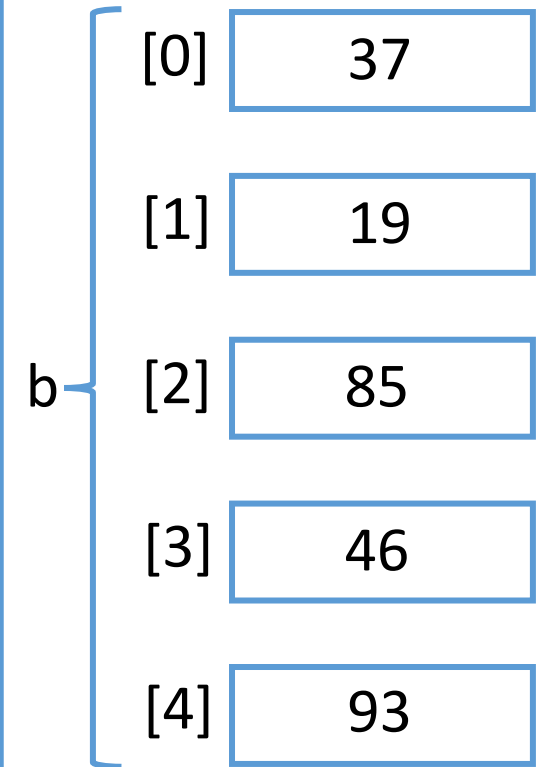
# ARRAY INITIALIZATION

- Initialization using for statements

```
int a[100], i;  
  
for(i = 0; i < 100; i++)  
{  
    a[i] = 0;  
}
```

- Initialization using an initializer list

```
int b[5] = {37, 19, 85, 46, 93};
```



# ARRAY INITIALIZATION

- If there are fewer initializers than elements in the array, the remaining elements are initialized to zero

```
int a[100] = {0};
```

- This explicitly initializes the first element to zero and initializes the remaining 99 elements to zero
- If the array size is omitted from a definition with an initializer list, the number of elements in the array will be the number of elements in the initializer list

```
int b[] = {37,19,85,46,93};
```

- This would create a five-element array



# ARRAY ACCESS

- Syntax

```
array_name[index]
```

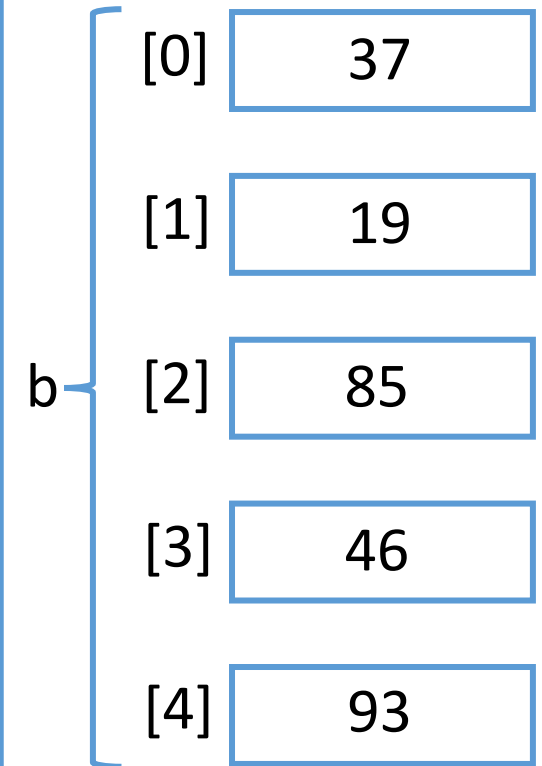
- Example

- How to print 46 ?

```
printf("%d", b[3]);
```

- How to print 37 ?

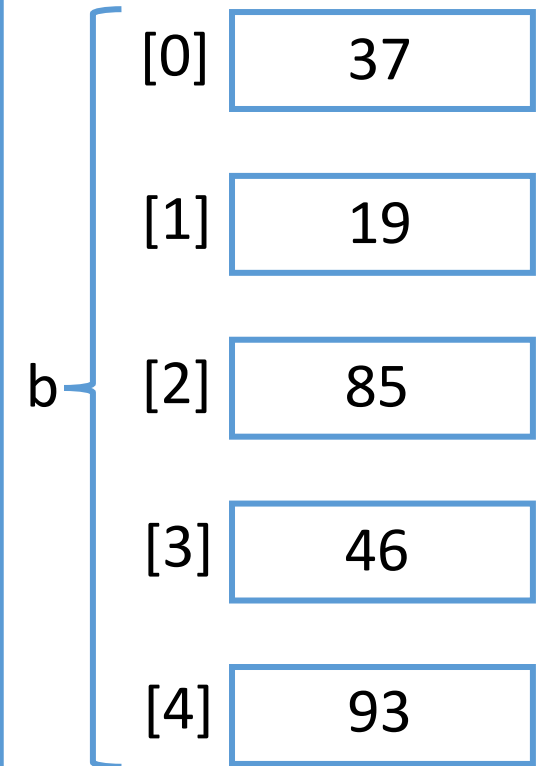
```
printf("%d", b[0]);
```



# ARRAY ACCESS

- Using for loops for sequential access

```
int i;  
  
for(i = 0; i < 5; i++)  
{  
    printf("%d\n", b[i]);  
}
```





**UMN**  
UNIVERSITAS  
MULTIMEDIA  
NUSANTARA

# PRACTICE

# PRACTICE 0

- Write a for loop that sums the odd values from the 10-element array `n`. For example, the sum for this array would be  $45 + 97 + 29 + 7 + 21 = 199$ .

<code>n[0]</code>	<code>n[1]</code>	<code>n[2]</code>	<code>n[3]</code>	<code>n[4]</code>	<code>n[5]</code>	<code>n[6]</code>	<code>n[7]</code>	<code>n[8]</code>	<code>n[9]</code>
45	97	29	42	50	7	12	62	21	10

# PRACTICE 1

- Write a for loop that sums the even-numbered elements from array n. For example, the sum for this array would be  $45 + 29 + 50 + 12 + 21 = 157$ .

n[0]	n[1]	n[2]	n[3]	n[4]	n[5]	n[6]	n[7]	n[8]	n[9]
45	97	29	42	50	7	12	62	21	10

# PRACTICE 2

- Write a program to store an input list of five integers in an array, then display each data value and what percentage each value is of the total of all five values. The screen dialogue should appear as follows:

48	24.00
62	31.00
37	18.50
3	1.50
50	25.00

`0 <= input <= 999`

# NEXT WEEK'S OUTLINE

- Passing Array to Functions
- Pointers & Arrays
- Two-Dimensional Array
- Multidimensional Array
- String

# REFERENCES

- Weiss, M.A. 2014. *Data Structures and Algorithm Analysis in C++*, 4<sup>th</sup> Edition, International Edition. Great Britain: Pearson Education Limited. (wajib)
- Barnett, G. and Tongo, L.D. 2008. *Data Structures and Algorithms: Annotated Reference with Examples*. DotNetSlackers (<http://dotnetslackers.com/>)
- Shaffer, C.A. 2011. *A Practical Introduction to Data Structures and Algorithm Analysis (Java Version)*, edition 3.1. <http://people.cs.vt.edu/~shaffer/Book/>
- Kernighan, B.W. and Ritchie, D.M. *The C Programming Language*, 2<sup>nd</sup> Edition. New Jersey: AT&T Bell Laboratories.
- Other online or offline references



# Visi

Menjadi Program Studi Strata Satu Informatika **unggulan** yang menghasilkan lulusan **berwawasan internasional** yang **kompeten** di bidang Ilmu Komputer (*Computer Science*), **berjiwa wirausaha** dan **berbudi pekerti luhur**.



# Misi

1. Menyelenggarakan pembelajaran dengan teknologi dan kurikulum terbaik serta didukung tenaga pengajar profesional.
2. Melaksanakan kegiatan penelitian di bidang Informatika untuk memajukan ilmu dan teknologi Informatika.
3. Melaksanakan kegiatan pengabdian kepada masyarakat berbasis ilmu dan teknologi Informatika dalam rangka mengamalkan ilmu dan teknologi Informatika.

# THANK YOU

“Don’t Study Hard, Study Smart.”

-Amy Lucas-