Programming Fundamental

Week 13 –
Pointer and Array in the C Programming Language

Dr. Maria Irmina Prasetiyowati, S.Kom,. M.T.
Alethea Suryadibrata, S.Kom., M.Eng.
Putri Sanggabuana Setiawan, S.Kom, M.T.I.
Januar Wahjudi, S.Kom., M.Sc.
Drs Slamet Aji Pamungkas, M.Eng
Kursehi Falgenti S.Kom., M.Kom.

Weekly Learning Outcomes for Subjects (Sub-CPMK):

Sub-CPMK 0614: Students are able to create simple programs with elements of selection control, repetition control, functions, or procedures, as well as implementing arrays and pointers in the C programming language (C6).

Review

1. Introduction
2. Standard Library
3. Programmer-Defined Functions
4. Recursion

Outline

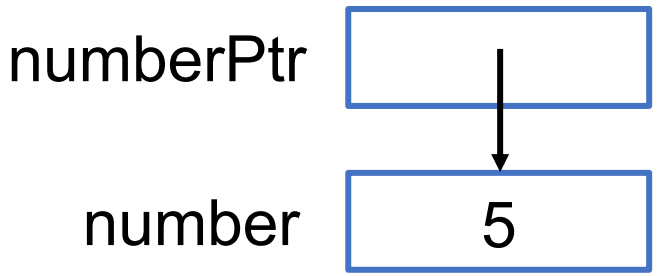
1. Pointer
2. Pointer Operators
3. Array Declaration
4. Array Access

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 **numberPtr** indirectly references the value 5

Variable **number** directly contains the value 5



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	0xFF04
number	5	0xFF08

- 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

```c
#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

```c
#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

```c
#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

```c
#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

```
F:\callbypointer.exe

5! = 120

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

```c
#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;
}
```

Array Definition

- 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 (0th) element**
- The position number contained within square brackets ([ ]) is more formally called a **subscript** or **index** → must be an integer or integer expression

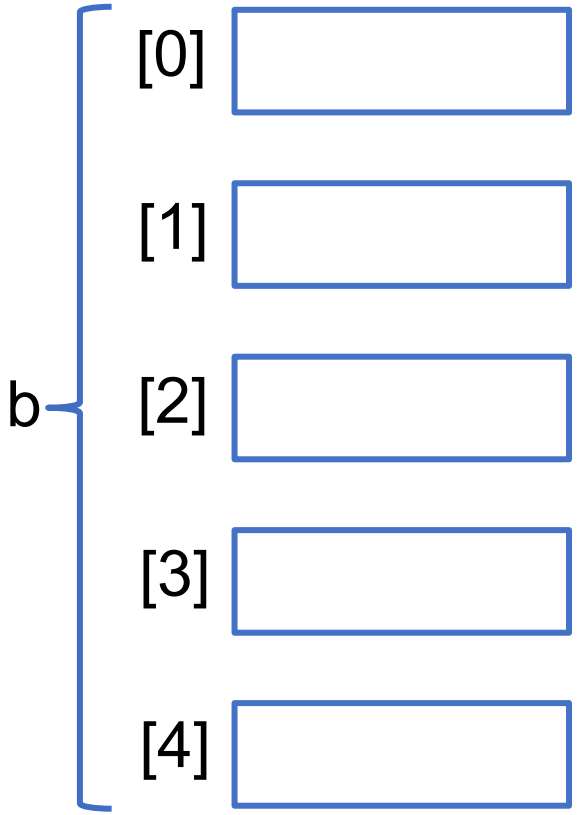
Array Declaration

- Syntax

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

- Example

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



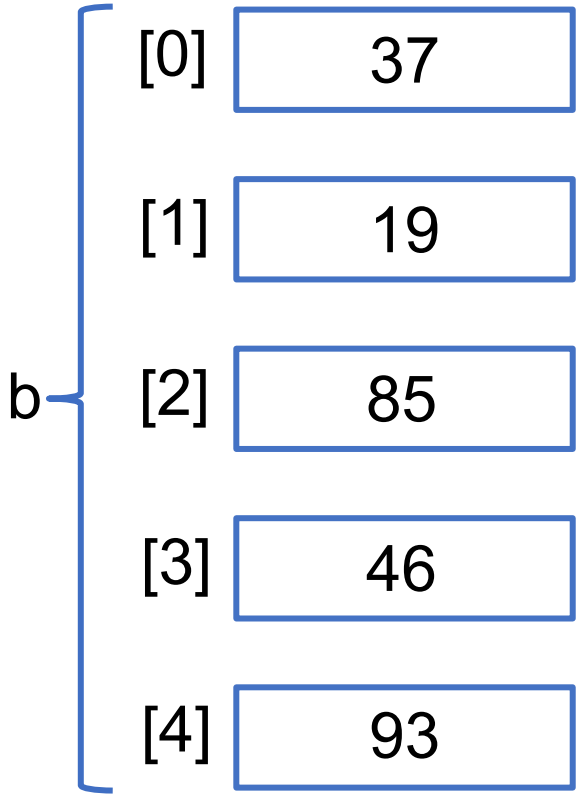
Array Initialization

- 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

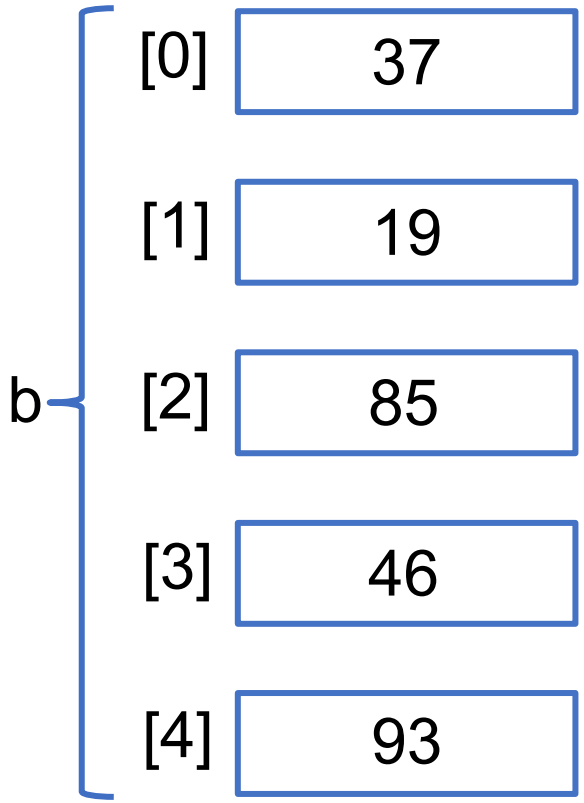
```
array_name[index]
```

- Example
  - How to print 46 ?

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

  - How to print 37 ?

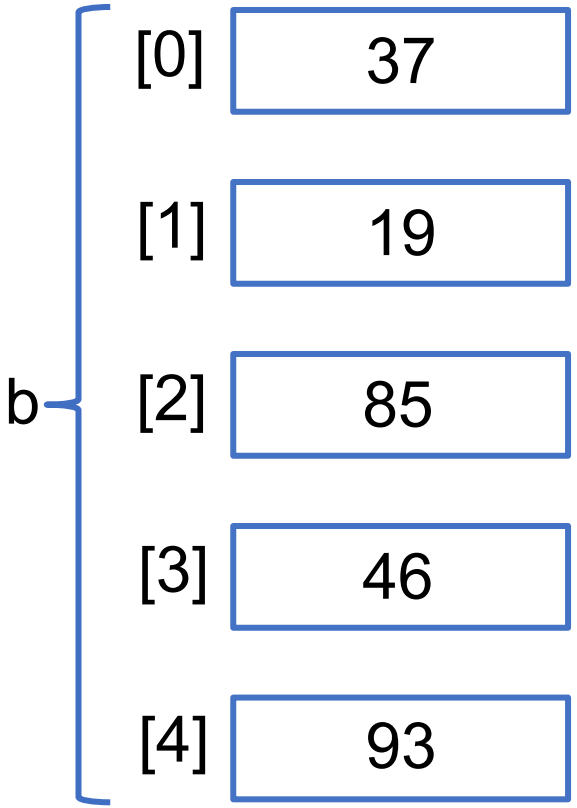
```
printf("%d",b[0]);
```



Array Access

- Using for loops for sequential access

```
int i;

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



Practice 1

- What does the following code print?

```c
#include <stdio.h>

void f(int *x, int *y)
{
    *x *= 2;
    *y *= 3;
}

int main()
{
    int a = 25, b = 30;

    f(&a,&b);
    printf("%d %d",a,b);

    return 0;
}
```

Practice 2

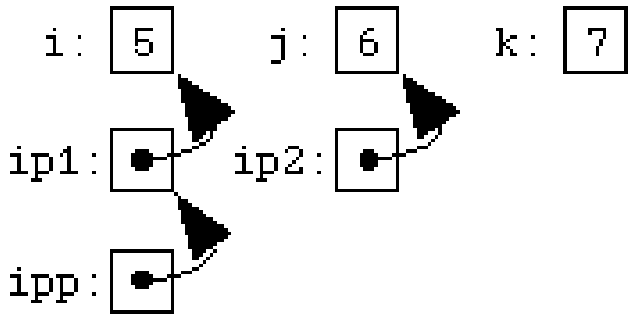
- What does the following code print?

```c
#include <stdio.h>

void swap(int *x, int *y)
{
    int *z;

    z = x;
    x = y;
    y = z;
}

int main()
{
    int a = 25, b = 30;

    swap(&a,&b);
    printf("%d %d",a,b);

    return 0;
}
```

Practice 3

- What does the following code print?

```
int i = 5, j = 6; k = 7;
int *ip1 = &i, *ip2 = &j;
int **ipp = &ip1;
```




```c
#include <stdio.h>

void f(int **x, int **y)
{
    **y *= **x;
}

int main()
{
    int a = 25, b = 30;
    int *aPtr = &a, *bPtr = &b;

    f(&aPtr,&bPtr);
    printf("%d %d",a,b);

    return 0;
}
```

Practice 4

- Write a function
  void sigma(int *n)
  that calculates 1 + 2 + … + n.

```
#include <stdio.h>

void sigma(int *n)
{
    ?
}

int main()
{
    int a = 10;

    sigma(&a);
    printf("Result = %d",a);

    return 0;
}
```

Practice 5

- Write a C program to **display the index of the smallest and the largest numbers in an array** x of **10 integers.** Array x need to be assign values first to each element.

REFERENCES

- Hanly, Jeri R. and Koffman, Elliot B., 2013, Problem Solving and Program Design in C, Seventh Edition, Pearson Education, Inc.
- Deitel, Paul and Deitel, Harvey, 2016, C How to Program, Eighth Edition, Pearson Education, Inc.

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**.

INFORMATIKA
UMN
UNIVERSITAS MULTIMEDIA NUSANTARA

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.