Programming Fundamental

Week 14 – **Review**

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.

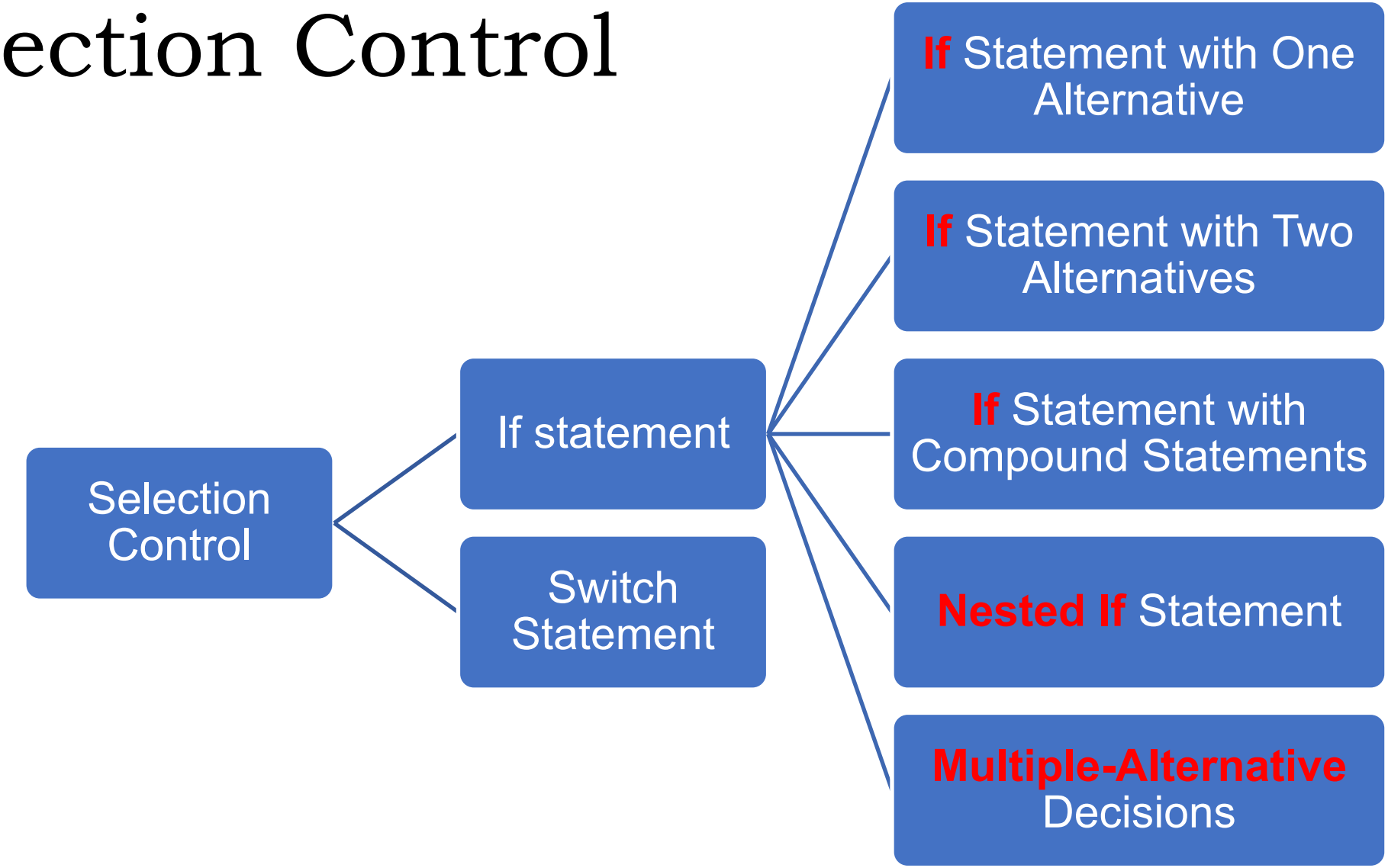
Weekly Learning Outcomes for Subjects (Sub-CPMK):

1. **Sub-CPMK 0212:** Students are able to explain the basic concepts of C language programming (C2).
2. **Sub-CPMK 0213:** Students are able to explain the concepts of operations and operators, as well as input and output, in the C programming language (C2).
3. **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).

Outline

1. Selection Control
2. Repetition Control
3. Functions
4. Pointers
5. Array Access

Selection Control



If Statement

If Statement with One Alternative	If Statement with Two Alternatives	If Statement with Compound Statements	Nested If Statement	Multiple-Alternative Decisions

if(condition)
statement$_T$;
	if(condition)
	statement$_T$;
	else
	statement$_F$;
		if(condition)
		{
		statement$_{T1}$;
		statement$_{T2}$;
		...
		statement$_{Tn}$;
		}
		else
		{
		statement$_{F1}$;
		statement$_{F2}$;
		...
		statement$_{Fn}$;
		}
			if(condition$_1$)
			{
			if(condition$_2$)
			statement$_1$;
			else
			statement$_2$;
			}
			else
			{
			if(condition$_3$)
			statement$_3$;
			else
			statement$_4$;
			}
				if(condition$_1$)
				statement$_1$;
				else if(condition$_2$)
				statement$_2$;
				.
				.
				.
				else if(condition$_n$)
				statement$_n$;
				else
				statement$_e$;

Switch Statement

```
switch(expression)
{
        case label_set1:   statement1;
                           break;
        case label_set2:   statement2;
                           break;
        ...
        case label_setn:   statementn;
                           break;
        [default: statementd;]
}
```

F:\switch.exe

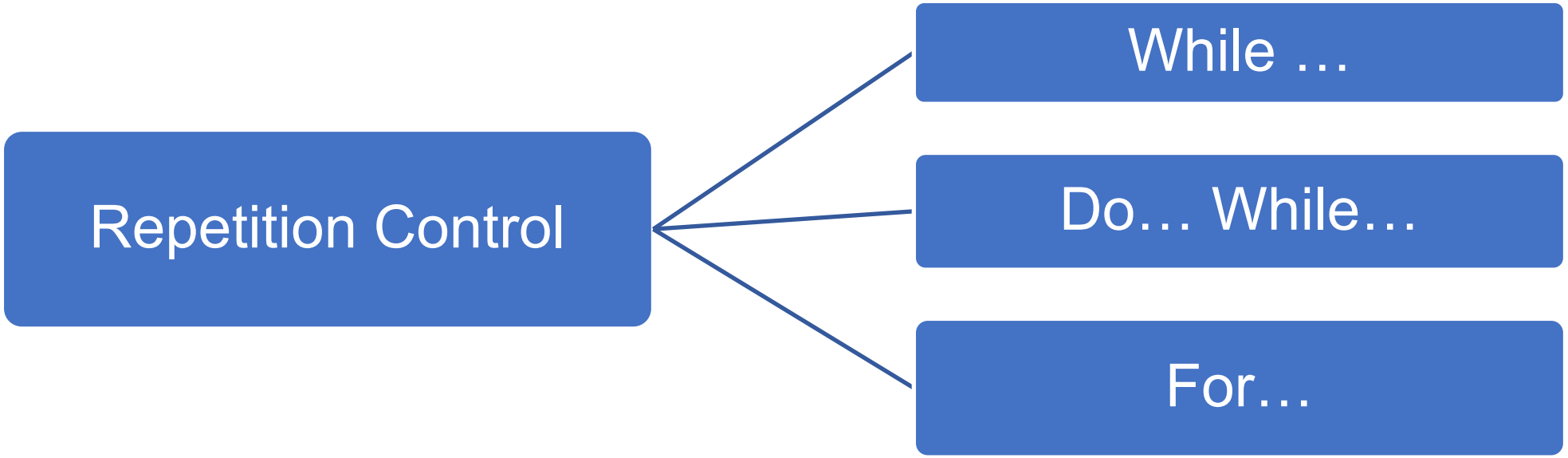
```
Ship: C
Cruiser

Process returned 0 (0x0)
execution time : 1.343 s
Press any key to continue.
```

F:\switch.exe

```
Ship: F
Unknown

Process returned 0 (0x0)
execution time : 1.282 s
Press any key to continue.
```

```
switch(ship)
{
    case 'B': printf("Battleship\n");
              break;
    case 'C': printf("Cruiser\n");
              break;
    case 'D': printf("Destroyer\n");
              break;
    default : printf("Unknown\n");
}
```

Repetition Control



While Statement

While Statement	Do while Statement	For Statement

`while(loop_repetition_condition)`
`{`
`statement;`
`}`
	`do`
	`{`
	`statement;`
	`}while(loop_repetition_condition);`
		`for(initialization_expression;`
		`loop_repetition_condition`
		`;update_expression)`
		`{`
		`statement;`
		`}`

Break Statement

The break statement, when executed in a **while**, **do…while**, **for**, or **switch** statement, causes an **immediate exit** from the statement

```c
#include <stdio.h>

int main()
{
    int iSum = 0, iNumber;

    while(1)
    {
        scanf("%d",&iNumber);
        if(iNumber < 0) break;
        iSum += iNumber;
    }
    printf("Sum = %d\n",iSum);

    return 0;
}
```

F:\break.exe

```
5
7
8
0
3
-1
Sum = 23

Process returned 0 (0x0)
execution time : 6.759 s
Press any key to continue.
```

Continue Statement

The **continue** statement, when executed in a **while**, **do…while**, or **for** statement, **skips** the remaining statements in the body of that control statement and **performs the next iteration** of the loop

```c
#include <stdio.h>

int main()
{
    int iCounter, iLimit;

    printf("Limit: ");
    scanf("%d",&iLimit);

    iCounter = 0;
    while(iCounter < iLimit)
    {
        iCounter++;
        if(iCounter % 3 != 0) continue;
        printf("%d\n",iCounter);
    }

    return 0;
}
```

```
F:\continue.exe
Limit: 20
3
6
9
12
15
18

Process returned 0 (0x0)
execution time : 1.132 s
Press any key to continue.
```

Nested Loops

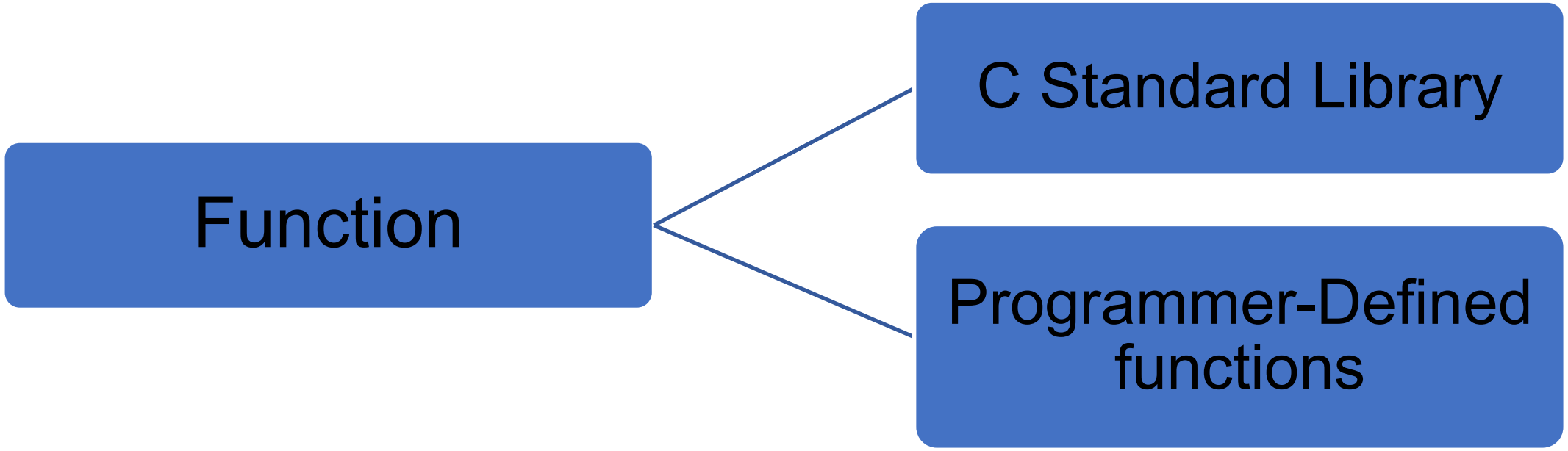
- Nested loops consist of an outer loop with one or more inner loops
- Each time the outer loop is repeated, the inner loops are reentered, their loop control expressions are reevaluated, and all required iterations are performed

```c
#include <stdio.h>

int main()
{
    int iRow, iColumn;

    for(iRow = 0;iRow < 5;iRow++)
    {
        for(iColumn = 0;iColumn < 8;iColumn++)
        {
            printf("*");
        }
        printf("\n");
    }

    return 0;
}
```

F:\nestedloops.exe

```
********
********
********
********
********

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

Functions


Function
C Standard Library
Programmer-Defined functions

C Standard Library

Standard Library Header	Explanation
<stdio.h>	Standard input / output library functions
<math.h>	Math library functions
<string.h>	String-processing functions
<time.h>	Time and date manipulation functions
<stdlib.h>	Conversions of numbers to text and text to numbers, memory allocation, random numbers, and other utility functions
<ctype.h>	Functions that test characters for certain properties Functions that can be used to convert lowercase letters to uppercase letters and vice versa

Programmer-Defined functions

- Functions definition

```c
#include <stdio.h>

int maximum(int a, int b, int c);

int maximum(int a, int b, int c)
{
    int max = a;

    if(b > max) max = b;
    if(c > max) max = c;

    return max;
}

int main()
{
    int maxNumber;

    maxNumber = maximum(37,19,62);
    printf("%d\n",maxNumber);

    return 0;
}
```

function prototype

function definition

```c
return_value_type function_name(parameter_list)
{
        statement;
}
```

F:\function.exe

```
62

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


14

Functions

Calling Functions

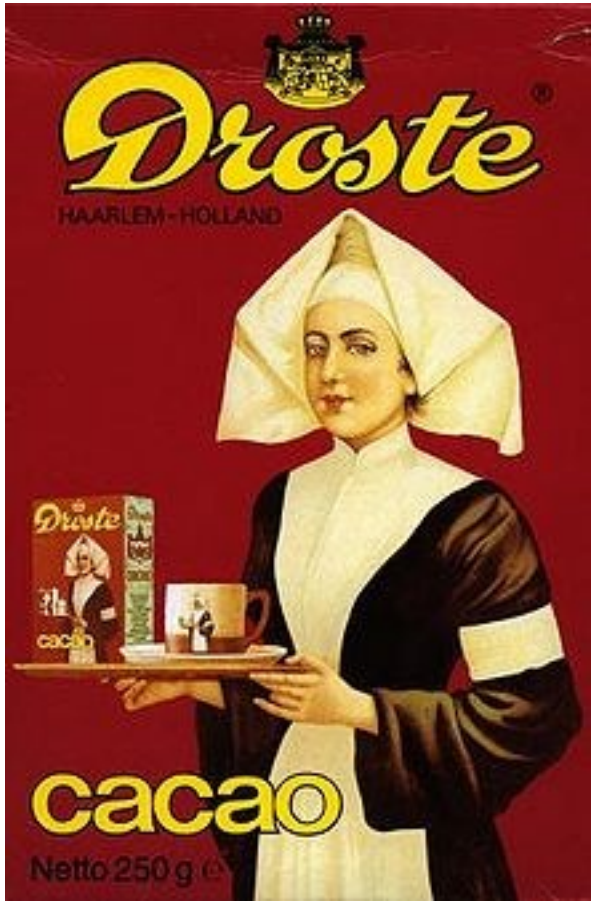
Call-by-Value

- When arguments are passed by value, a copy of the argument's value is made and passed to the called functions
- Changes to the copy do not affect an original variable's value in the caller

Call-by-Reference

- When an argument is passed by reference, the caller allows the called **function to modify the original variable's value**

Recursion



- A **recursive function** is a function that calls itself
  - base case
  - recursion step

Pointers

- Pointers are variables whose **values are memory addresses**
- 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
- Pointers **must be defined** before they can be used

```
data_type *pointer_name;
```

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

```
int *iPtr;
```

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

```
int *iPtr;
```

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

- 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

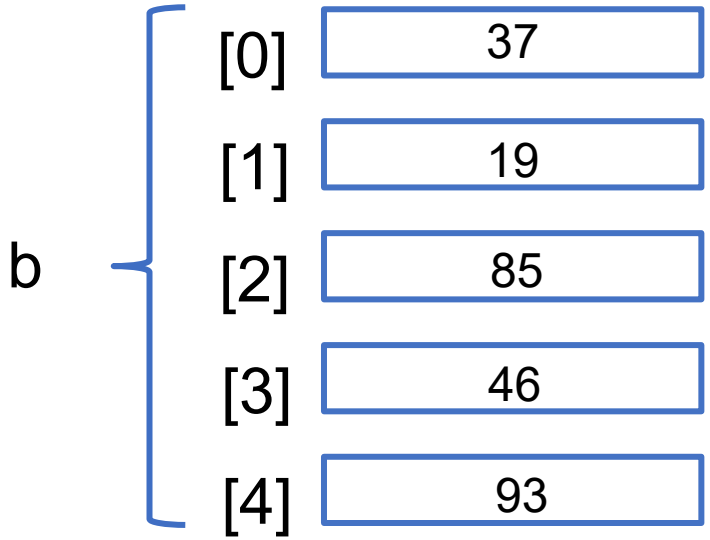
```
element_data_type array_name[size];
```

- Example: int b[5]
- Array Initialization
- Initialization using for statements

```
int b[5], i;

for(i = 0;i < 5;i++)
{
      a[i] = 0;
}
```

Initialization using an **initializer list**

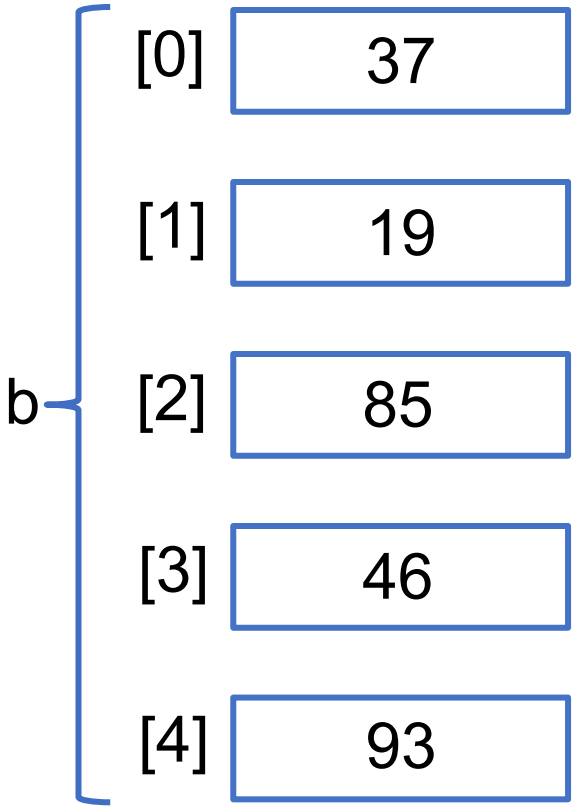
```
int b[5] = {37,19,85,46,93};
```



Array Access

- Using for loops for sequential access

```
int i;

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



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.