Fundamental Programming

Week 09 – **Operations, operators, input and output**

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.

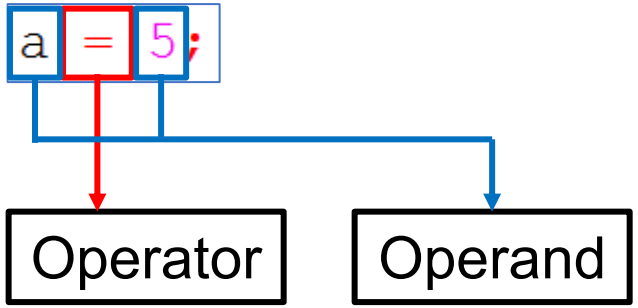
Weekly Learning Outcomes for Subjects (Sub-CPMK):

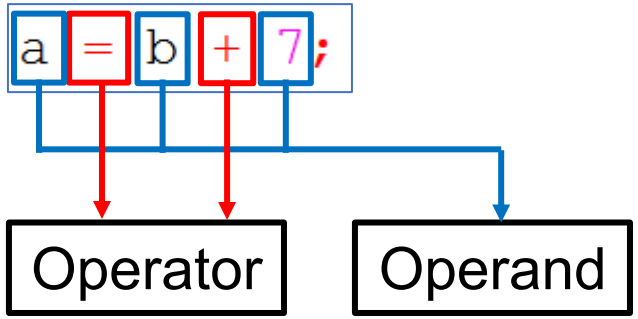
1. **Sub-CPMK 0213:** Students are able to explain the concepts of operations and operators, as well as input and output, in the C programming language (C2).

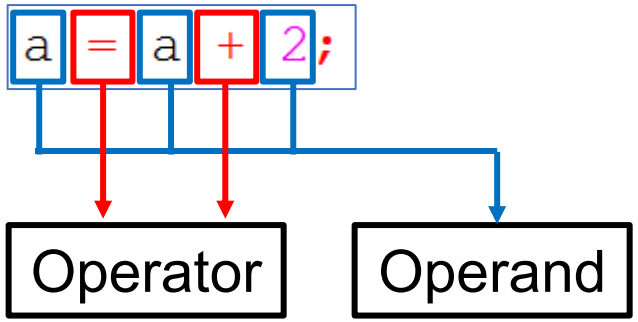
Outline

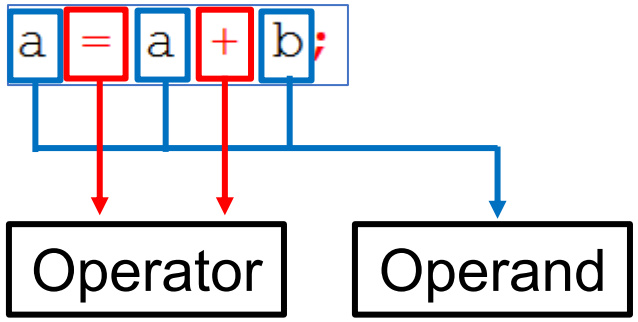
1. Assigment operator
2. Identifiers and Keywords
3. Operators and Operations
4. Memory Concepts
5. Function Prototype
6. Formatted & Unformatted Input
7. Formatted & Unformatted Output

Operations


a = 5;
Operator
Operand


a = b + 7;
Operator
Operand


a = a + 2;
Operator
Operand


a = a + b;
Operator
Operand

Operators

- **Unary operators**: operators that work on a single operand

```
a++; --b;
```

- **Binary operators**: operators that connect two operands

```
x = a+b; y = a*b;
```

- **Ternary operator**: operator that has three operands

```
x = a>b ? a-b : b-a;
```

Simple Assignment Operator

- Assign x the value of y

```
x = y;
```

- Associativity: **right to left**

```
x ← = y;
```

- Left operand **must be** a variable & right operand is an expression

```
variable_name = expression;
```

- Example

```
a = 5;
a = b + 7;
a = a + 2;
a = a + b;
```

Arithmetic Operators

Operator	Meaning	Example	Explanation
+	Addition	x + y	The sum of x and y
-	Subtraction	x - y	The difference of x and y
*	Multiplication	x * y	The product of x and y
/	Division	x / y	The quotient of x by y
%	Modulo Operation	x % y	The remainder of x divided by y

Arithmetic Operators

```c
#include <stdio.h>

int main()
{
    int x = 12, y = 4, z;

    z = x + y;
    printf("x + y = %d\n",z);
    printf("x - y = %d\n",x-y);
    printf("x * y = %d\n",x*y);
    printf("x / y = %d\n",x/y);
    printf("x %% y = %d\n",x%(y+1));

    return 0;
}
```

F:\operators.exe

```
x + y = 16
x - y = 8
x * y = 48
x / y = 3
x % y = 2

Process returned 0 (0x0)
execution time : 0.012 s
Press any key to continue.
```

Increment & Decrement Operators

Operator	Meaning	Example	Explanation
++	**Increment** Increases the value of x by one (a = a + 1)	a++ ++a	Use the current value of a in the expression in which a resides, then increment a by 1 Increment a by 1, then use the new value of a in the expression in which a resides
- -	**Decrement** Decreases the value of x by one (a = a – 1)	a - - - - a	Use the current value of a in the expression in which a resides, then decrement a by 1 Decrement a by 1, then use the new value of a in the expression in which a resides

Increment & Decrement Operators

```c
#include <stdio.h>

int main()
{
    int x = 5;

    printf("%d\n",x++);
    printf("%d\n",x);

    return 0;
}
```

F:\operators.exe

```
5
6

Process returned 0 (0x0)
execution time : 0.013 s
Press any key to continue.
```

```c
#include <stdio.h>

int main()
{
    int x = 5;

    printf("%d\n",++x);
    printf("%d\n",x);

    return 0;
}
```

F:\operators.exe

```
6
6

Process returned 0 (0x0)
execution time : 0.008 s
Press any key to continue.
```

Comparative Operators

- Equality Operators

Operator	Meaning	Example	Result (1 = true, 0 = false)
==	Equal to	x == y	1 if x is equal to y, otherwise 0
!=	Not equal to	x != y	1 if x is not equal to y, otherwise 0

- Relational Operators

Operator	Meaning	Example	Result (1 = true, 0 = false)
<	Less than	x < y	1 if x is less than y, otherwise 0
<=	Less than or equal to	x <= y	1 if x is less than or equal to y, otherwise 0
>	Greater than	x > y	1 if x is greater than y, otherwise 0
>=	Greater than or equal to	x >= y	1 if x is greater than or equal to y, otherwise 0

Comparative operators

```c
#include <stdio.h>

int main()
{
    printf("%d\n",5 > 8);
    printf("%d\n",5 <= 8);
    printf("%d\n",7 != 7);
    printf("%d\n",7 == 7);

    return 0;
}
```

F:\operators.exe

```
0
1
0
1

Process returned 0 (0x0)
execution time : 0.015 s
Press any key to continue.
```

Logical Operators

```c
#include <stdio.h>

int main()
{
    printf("%d\n",!0);
    printf("%d\n",!1);
    printf("%d\n",0 && 1);
    printf("%d\n",1 && 1);
    printf("%d\n",0 || 1);
    printf("%d\n",1 || 1);

    return 0;
}
```

```
F:\operators.exe
1
0
0
1
1
1

Process returned 0 (0x0)
execution time : 0.018 s
Press any key to continue.
```

Operator	Meaning	Example	Result (1 = true, 0 = false)
&&	Logical AND	**x && y**	1 if each of the operands x and y is not equal to zero, otherwise 0
||	Logical OR	**x || y**	0 if each of x and y is equal to zero, otherwise 1
!	Logical NOT	**!x**	1 if x is equal to zero, otherwise 0

Bitwise Operators

- Boolean Bitwise Operators

Operator	Meaning	Example	Result (for each bit position) (1 = set, 0 = cleared)
&	Bitwise AND	**x & y**	1, if 1 in both x and y 0, if 0 in x or y, or both
|	Bitwise OR	**x | y**	1, if 1 in x or y, or both 0, if 0 in both x and y
^	Bitwise exclusive OR	**x ^ y**	1, if 1 either in x or in y, but not in both 0, if either value in both x and y
~	Bitwise NOT (one's complement)	**~x**	1, if 0 in x 0, if 1 in x

- Shift Operators

Operator	Meaning	Example	Result
<<	Shift left	**x << y**	Each bit value in x is moved y positions to the left
>>	Shift right	**x >> y**	Each bit value in x is moved y positions to the right

Bitwise Operators

```c
#include <stdio.h>

int main()
{
    printf("%d\n",3 & 5);
    printf("%d\n",3 | 5);
    printf("%d\n",3 ^ 5);
    printf("%d\n",~3);
    printf("%d\n",3 << 2);
    printf("%d\n",24 >> 2);

    return 0;
}
```

F:\operators.exe

```
1
7
6
-4
12
6

Process returned 0 (0x0)
execution time : 0.013 s
Press any key to continue.
```

Compound Assignment Operators

Operator	Example	Meaning
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
&=	x &= y	x = x & y
^=	x ^= y	x = x ^ y
|=	x |= y	x = x | y
<<=	x <<= y	x = x << y
>>=	x >>= y	x = x >> y

```c
#include <stdio.h>

int main()
{
    int x = 5;

    x += 3;
    printf("%d\n",x);
    x *= 2;
    printf("%d\n",x);

    return 0;
}
```

F:\operators.exe

```
8
16

Process returned 0 (0x0)
execution time : 0.015 s
Press any key to continue.
```

Conditional operator

- The conditional operator is sometimes called the ternary or trinary operator because it is the only one that has three operands

```
Condition ? Expression 1 : Expression 2
```

- The operation first evaluates the condition
- If the result is not equal to 0 (the condition is true), then only expression 1 is evaluated and the entire operation yields the value of expression 1
- If the condition does yield 0 (the condition is false), then only expression 2 is evaluated and the entire operation yields the value of expression 2

```
fail = (score >= 55) ? ‘N’ : ‘Y’;
```

Conditional Operator

```c
#include <stdio.h>

int main()
{
    int x = 5, y = 8, distance;

    distance = x > y ? x - y : y - x;
    printf("Distance = %d\n",distance);

    return 0;
}
```

F:\operators.exe

```
Distance = 3

Process returned 0 (0x0)
execution time : 0.022 s
Press any key to continue.
```

Operator Precedence & Associativity

- The **precedence** of the operators determines which part of the expression is treated as the operand of each operator
- Example

  `a - b * c` is equivalent to `a - (b * c)`

- If two operators in an expression have the same precedence, then their **associativity** determines whether they are grouped with operands in order from left to right, or from right to left
- Example

Expression	Associativity	Effective Grouping
`a / b % c`	Left to right	`(a / b) % c`
`a = b = c`	Right to left	`a = (b = c)`

Operator precedence & associativity

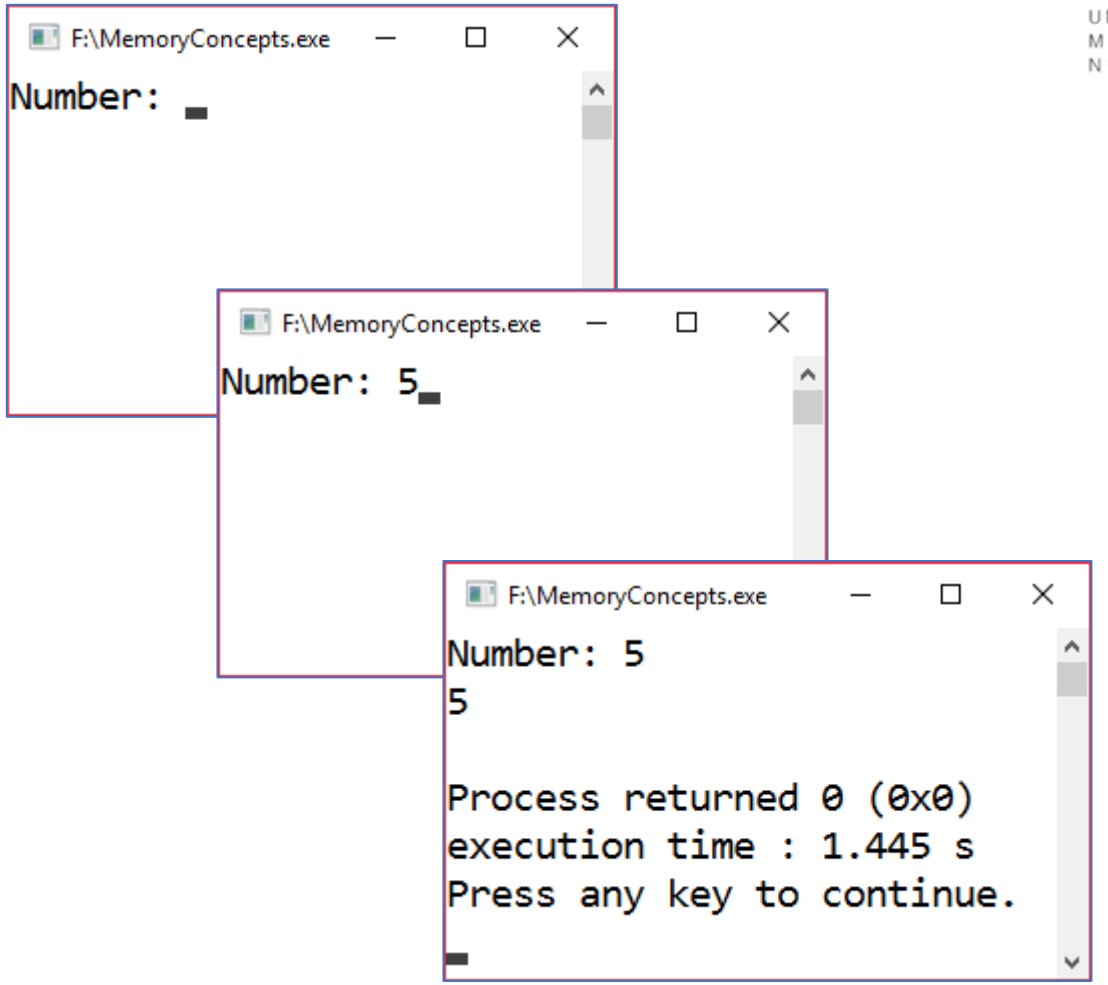
Precedence	Operators	Associativity
1	Postfix operators: ++ --	Left to right
2	Unary operators: ++ -- ! ~	Right to left
3	Multiplicative operators: * / %	Left to right
4	Additive operators: + -	Left to right
5	Shift operators: << >>	Left to right
6	Relational operators: < <= > >=	Left to right
7	Equality operators: == !=	Left to right
8	Bitwise AND: &	Left to right
9	Bitwise exclusive OR: ^	Left to right
10	Bitwise OR: |	Left to right
11	Logical AND: &&	Left to right
12	Logical OR: ||	Left to right
13	Conditional operator: ? :	Right to left
14	Assignment operators: = += -= *= /= %= &= ^= |= <<= >>=	Right to left

Operator Precedence & Associativity

- The increment and decrement operators (++ and --) have a higher precedence when used as postfix operators (as in x++) than the same tokens when used as prefix operators (as in ++x)

Memory Concepts

```c
#include <stdio.h>

int main()
{
    int number;

    printf("Number: ");
    scanf("%d",&number);
    printf("%d\n",number);

    return 0;
}
```



Memory Concepts

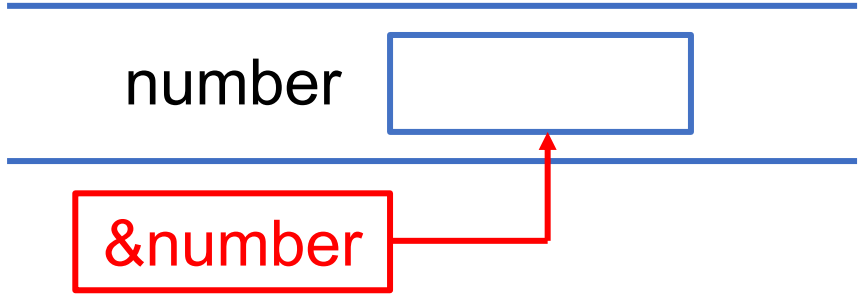
```c
#include <stdio.h>

int main()
{
    int number;

    printf("Number: ");
    scanf("%d",&number);
    printf("%d\n",number);

    return 0;
}
```

```c
scanf("%d",&number);
```

- scanf is used to obtain a value from the user
- The scanf function reads from the standard input, which is usually the keyboard
- The first argument (format control string) indicates the type of data that should be input by the user
- The %d conversion specifier indicates that the data should be an integer

Memory Concepts

```
#include <stdio.h>

int main()
{
    int number;

    printf("Number: ");
    scanf("%d",&number);
    printf("%d\n",number);

    return 0;
}
```




```
scanf("%d",&number);
```

- The second argument of scanf begins with an ampersand (&) [address operator] followed by the variable name

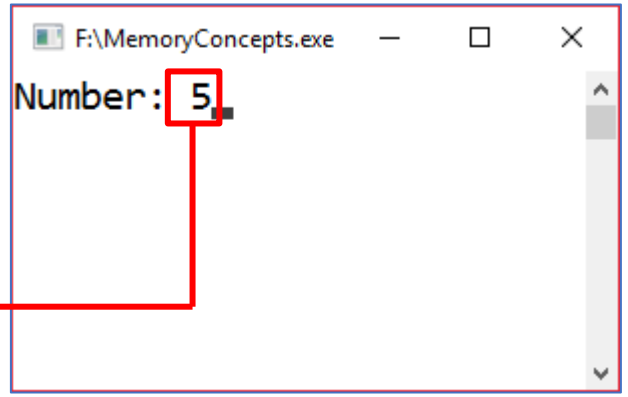
- The ampersand, when combined with the variable name, tells scanf the location / address in memory at which the variable *number* is stored

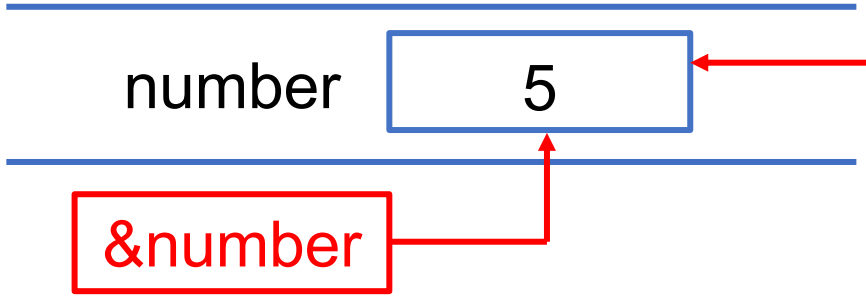
Memory concepts

```c
#include <stdio.h>

int main()
{
    int number;

    printf("Number: ");
    scanf("%d",&number);
    printf("%d\n",number);

    return 0;
}
```

```c
scanf("%d",&number);
```

- The value typed by the user is placed into a memory location to which the name *number* has been assigned



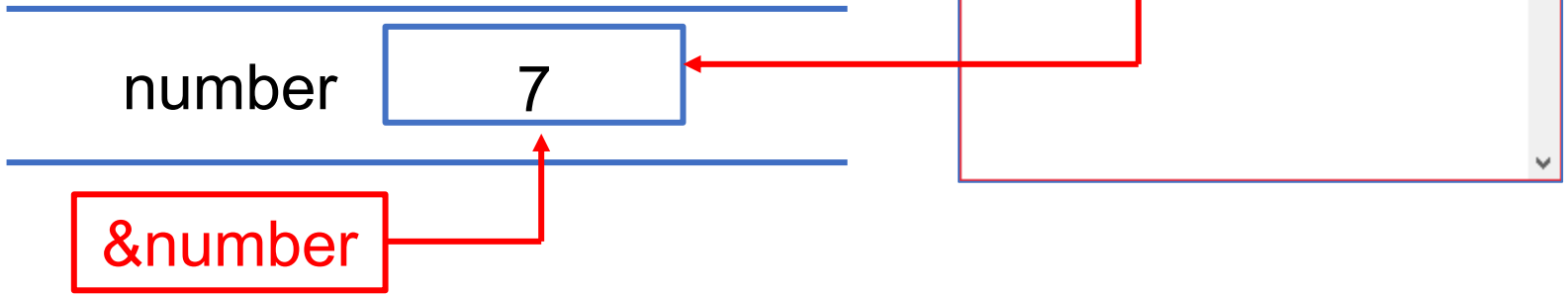


Memory concepts

```c
#include <stdio.h>

int main()
{
    int number;

    printf("Number: ");
    scanf("%d",&number);
    printf("%d\n",number);

    return 0;
}
```

```c
scanf("%d",&number);
```

- Whenever a value is placed in a memory location, the value replaces the previous value in that location



Standard Input / Output

- **scanf** and **printf** functions input data from the standard input stream and output data to the standard output stream

- Other functions that use the standard input and standard output

gets getchar getche getch	puts putchar putch

- Include the header **<stdio.h>** in programs that call these functions

Function Prototype

```
return_value_type function_name(parameter_list);
```

- The **function_name** is any valid identifier
- The **return_value_type** is the data type of the result returned to the caller
  - The **return_value_type** ***void*** indicates that a function does not return a value
- The **parameter_list** is a comma-separated list that specifies the parameters received by the function when it's called
  - If a function does not receive any values, **parameter_list** is void
  - A type must be listed explicitly for each parameter

Function Prototype

```
return_value_type function_name(parameter_list);
```

- A function prototype tells the compiler
  - the type of data returned by the function
  - the number of parameters the function expects to receive
  - the types of the parameters
  - the order in which these parameters are expected
- Example
  - int rand (void);
  - int abs (int x);
  - double pow (double x, double y);
  - void clrscr (void);

Calling Functions

- Without return value
  - clrscr();

- With return value
  - random_value = rand();
  - absolute_value = abs(value);
  - result = pow(2,5);

Formatted Output

- Precise output formatting is accomplished with printf

```
printf(format_control_string, other_arguments);
```

- **format_control_string** describes the output format
- **other_arguments** (which are optional) correspond to each conversion specification in format control string
- Each conversion specification begins with **a percent sign (%)** and ends with **a conversion specifier**
- There can be many conversion specifications in one format control string

Formatted Output

- Conversion Specifications

```
%[flags][field_width][.precision][length_modifier]specifier
```

- The parts of this syntax that are indicated in **square brackets** are all **optional**, but any of them that you include must be placed in the order shown here
- Any conversion specification can include a field width
- The precision **does not** apply to all conversion types

Formatted Output

- Conversion Specifiers

Conversion Specifier	Description
d	Signed decimal integer
i	Signed decimal integer
o	Unsigned octal integer
u	Unsigned decimal integer
x or X	**Unsigned hexadecimal integer** X causes the digits 0 – 9 and the letters A – F to be displayed x causes the digits 0 – 9 and a – f to be displayed
e or E	**Signed floating-point number in exponential notation** E and e print exactly one digit to the left of the decimal point E prints uppercase E preceding the exponent e prints lowercase e preceding the exponent
f	Signed floating-point number in fixed-point notation
c	Character
s	String

Formatted Output

- Flags

Flag	Description
- (minus sign)	Left justify the output within the specified field
+ (plus sign)	Display a plus sign preceding positive values and a minus sign preceding negative values
space	Print a space before a positive value not printed with the + flag
0 (zero)	Pad a field with leading zeros
#	Prefix 0 to the output value when used with the octal conversion specifier o Prefix 0x or 0X to the output value when used with the hexadecimal conversion specifiers x or X Force a decimal point for a floating-point number printed with e, E, or f that does not contain a fractional part (Normally the decimal point is printed only if a digit follows it)

Formatted Output

- Length Modifiers

Length Modifier	Description
h	Indicates that a short integer is displayed
l	Indicates that a long integer is displayed
L	Indicates that a long double floating-point value is displayed

Formatted Output

```c
#include <stdio.h>

int main()
{
    int a = 95;

    printf("a = %d     [integer]\n",a);
    printf("a = %o    [octal]\n",a);
    printf("a = %X     [hexadecimal (%%X)]\n",a);
    printf("a = %#x   [hexadecimal (%%#x)]\n",a);

    return 0;
}
```

F:\output.exe

```
a = 95     [integer]
a = 137    [octal]
a = 5F     [hexadecimal (%X)]
a = 0x5f   [hexadecimal (%#x)]

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

Formatted Output

```c
#include <stdio.h>

int main()
{
    int a = 95;

    printf("a = %d \n",a);
    printf("a = %5d \n",a);
    printf("a = %-5d%d \n",a,a);
    printf("a = %05d \n",a);

    return 0;
}
```

F:\output.exe

```
a = 95
a =    95
a = 95   95
a = 00095

Process returned 0 (0x0)
execution time : 0.031 s
Press any key to continue.
```

Formatted Output

```c
#include <stdio.h>

int main()
{
    int a = 95;

    printf("a = %d\n",a);
    printf("a = % d\n",a);
    printf("a = %+d\n",a);

    return 0;
}
```

F:\output.exe

```
a = 95
a =  95
a = +95

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

Formatted Output

```c
#include <stdio.h>

int main()
{
    float pi = 3.1415926;

    printf("pi = %f\n",pi);
    printf("pi = %.3f\n",pi);
    printf("pi = %07.2f\n",pi);

    return 0;
}
```

F:\output.exe

```
pi = 3.141593
pi = 3.142
pi = 0003.14

Process returned 0 (0x0)
execution time : 0.003 s
Press any key to continue.
```

Formatted Output

```c
#include <stdio.h>

int main()
{
    printf("ASCII 65 = %c\n",65);
    printf("ASCII 90 = %c\n",90);
    printf("ASCII %d = a\n",'a');
    printf("ASCII %d = z\n",'z');

    return 0;
}
```

F:\output.exe

```
ASCII 65 = A
ASCII 90 = Z
ASCII 97 = a
ASCII 122 = z

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

unFormatted Output

- **putchar**, **putch**, and **puts**

```
int putchar(int c);
int putch(int c);
int puts(const char *s);
```

```c
#include <stdio.h>

int main()
{
    putchar(65);
    putch('B');
    puts("CDE");

    return 0;
}
```

F:\output.exe

```
ABCDE

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

Formatted Input

- Precise input formatting can be accomplished with **scanf**

```
scanf(format_control_string, other_arguments);
```

- **format_control_string** describes the formats of the input
- **other_arguments** are **pointers to variables** in which the input will be stored

Formatted Input

- Conversion Specifications

```
%[*][field_width][length_modifier]specifier
```

- * is an assignment suppression character
- The assignment suppression character enables scanf to **read any type of data** from the input and discard it without assigning it to a variable

Formatted Input

- Conversion Specifiers

Conversion Specifier	Description
d	Signed decimal integer
i	Signed decimal, octal, or hexadecimal integer
o	Octal integer
u	Unsigned decimal integer
x or X	Hexadecimal integer
e, E, or f	Floating-point value
c	Character
s	String

Formatted Input

- Length Modifiers

Length Modifier	Description
h	Indicates that a short integer is to be input
l (with integer)	Indicates that a long integer is to be input
l (with floating-point numbers)	Indicates that a double value is to be input
L	Indicates that a long double value is to be input

Formatted Input

```c
#include <stdio.h>

int main()
{
    int a;
    char b;

    scanf("%d",&a);
    scanf("%c",&b);
    printf("a = %d\nb = %c\n",a,b);

    return 0;
}
```

F:\input.exe

```
5G
a = 5
b = G

Process returned 0 (0x0)
execution time : 1.578 s
Press any key to continue.
```

Formatted Input

```c
#include <stdio.h>

int main()
{
    int day, month, year;

    printf("Date: ");
    scanf("%d-%d-%d",&day,&month,&year);
    printf("Day = %d\n",day);
    printf("Month = %d\n",month);
    printf("Year = %d\n",year);

    return 0;
}
```

F:\input.exe

```
Date: 25-12-2015
Day = 25
Month = 12
Year = 2015

Process returned 0 (0x0)
execution time : 3.125 s
Press any key to continue.
```

Formatted Input

```c
#include <stdio.h>

int main()
{
    int day, month, year;

    printf("Date: ");
    scanf("%d%*c%d%*c%d",&day,&month,&year);
    printf("Day = %d\n",day);
    printf("Month = %d\n",month);
    printf("Year = %d\n",year);

    return 0;
}
```

F:\input.exe

```
Date: 25-12-2015
Day = 25
Month = 12
Year = 2015

Process returned 0 (0x0)
execution time : 2.997 s
Press any key to continue.
```

F:\input.exe

```
Date: 25/12/2015
Day = 25
Month = 12
Year = 2015

Process returned 0 (0x0)
execution time : 3.320 s
Press any key to continue.
```

Formatted Input

```c
#include <stdio.h>

int main()
{
    int a;
    char b;

    printf("Hexadecimal: ");scanf("%x",&a);
    fflush(stdin);
    printf("Octal = %o\n",a);
    printf("Integer = %d\n\n",a);

    printf("Character: ");scanf("%c",&b);
    printf("ASCII = %d\n",b);

    return 0;
}
```

F:\input.exe

```
Hexadecimal: BC
Octal = 274
Integer = 188

Character: G
ASCII = 71

Process returned 0 (0x0)
execution time : 2.565 s
Press any key to continue.
```

Formatted Input

```c
#include <stdio.h>

int main()
{
    char name[30];

    printf("Name: ");
    scanf("%s",&name);
    printf("Hello, %s\n",name);

    return 0;
}
```

```
Name:Mahasiswa Prodi Informatika
Hello, Mahasiswa

Process returned 0 (0x0)   execution time : 14.619 s
Press any key to continue.
```

```c
#include <stdio.h>

int main()
{
    char name[30];

    printf("Name: ");
    scanf("%[^\n]",&name);
    printf("Hello, %s\n",name);

    return 0;
}
```

```
Name:Mahasiswa Prodi Informatika
Hello, Mahasiswa Prodi Informatika

Process returned 0 (0x0)   execution time : 8.853 s
Press any key to continue.
```

Formatted Input

```c
#include <stdio.h>

int main()
{
    char name[30];

    printf("Name: ");
    scanf("%8[^\n]",&name);
    printf("Hello, %s\n",name);

    return 0;
}
```

```
Name:Mahasiswa Prodi Informatika
Hello,Mahasiswa P

Process returned 0 (0x0)   execution time : 18.876 s
Press any key to continue.
```

unFormatted Input

- **getchar**, **getche**, **getch**, and **gets**

```
int getchar(void);
int getche(void);
int getch(void);
char* gets(char *s);
```

```c
#include <stdio.h>

int main()
{
    char name[30];

    printf("Name: ");
    gets(name);
    printf("Hello, %s\n",name);

    return 0;
}
```

```
Name:Mahasiswa Prodi Informatika
Hello,Mahasiswa Prodi Informatika

Process returned 0 (0x0)   execution time : 12.344 s
Press any key to continue.
```

unFormatted input

```
#include <stdio.h>

int main()
{
    char a,b,c;

    printf("a: "); a = getchar(); printf("a = %c\n",a);
    printf("b: "); b = getche(); printf("b = %c\n",b);
    printf("c: "); c = getch(); printf("c = %c\n",c);

    return 0;
}
```

F:\input.exe

```
a: P
a = P
b: Qb = Q
c: c = R

Process returned 0 (0x0)
execution time : 3.108 s
Press any key to continue.
```

Practice 1

- What is the value of x after the following code is executed?

```
int x;
x = 5 * 8 – 7 + 2 * 3 - 3 + 4 / 2 + 2;
```

Practice 2

- What is the value of x and y after the following code is executed?

```
int x = 14, y = 5;
y = x++ % 3;
```

Practice 3

- What is the value of x and y after the following code is executed?

```
int x = 3, y = 5;
y *= 12 / ++x;
```

Practice 4

- What is the value of x, y, and z after the following code is executed?

```
int x = 10, y = 20, z = 30;
x *= y += ++z;
```

Practice 5

- What is the value of x and y after the following code is executed?

```
int x = 10, y;
y = x++ + ++x;
```

Practice 6

- Write a program that inputs two different integers from the keyboard, then prints the sum and the average of these numbers. The screen dialogue should appear as follows:

```
Number 1: 5
Number 2: 8

Sum = 13
Average = 6.5
```

```
Number 1: 4
Number 2: 6

Sum = 10
Average = 5
```

```
Number 1: 9
Number 2: 3

Sum = 12
Average = 6
```

Practice 7

- Write a program that reads in the radius of a circle and prints the circle's diameter, circumference, and area. Use the constant value 3.14159 for π. The screen dialogue should appearas follows:

```
Radius: 10
Diameter = 20
Circumference = 62.831800
Area = 314.159000
```

Practice 8

- What does the following code print?

```
float f = 387.469;
printf("%08.1f",f);
```

Practice 9

- What does the following code print?

```
int a = 153, b = 27;
printf("%5d%4d",a,b);
```

Practice 10

- What is the value of y after the following code is executed?

```
int x = 8, y;
y = ~x;
```

Practice 11

- What is the value of p, q, and r after the following code is executed?

```
int x = 8, y = 6, p, q, r;
p = x & y;
q = x | y;
r = x ^ y;
```

Practice 12

- What is the value of x, y, and z after the following code is executed?

```
int x = 8, y = 12, z;
z = x << y / 4;
y >>= x / 4;
```

Practice 13

- What does the following code print?

```
int score = 48;
score >= 55 ? printf("Pass\n") : printf("Fail\n");
```

Practice 14

- What does the following code print?

```
int a = 422, c = 53;
float b = 93.244;
printf("%7.2f%-5d%d",b,a,c);
```

Practice 15

- What does the following code print?

```
int a,b,c;
scanf("%d %o %x",&a,&b,&c);
printf("%3d%3d%d",a,b,c);
```

- Input

8 25 32

Practice 16

- What does the following code print?

```
int a,b,c;
scanf("%d %d %d",&a,&b,&c);
printf("%d %#o %#x",a,b,c);
```

- Input

15 25 35

Practice 17

- What does the following code print?

```
int a = 88;
putchar(a++);
printf(“%3d”,++a);
```

Practice 18

- What does the following code print?

```
char word[10];
scanf("%[aiueo]",&word);
printf("%s",word);
```

- Input

autumn

Practice 19

Write a program that inputs student's name and ID from the keyboard, then stores the data in variable **studentName** and **studentID**, respectively. Print the values of each variable. The screen dialogue should appear as follows:

```
Misael Azarya#00000010430

ID: 00000010430
Name: Misael Azarya
```

```
Eric Darson#00000010821

ID: 00000010821
Name: Eric Darson
```

NEXT WEEK'S OUTLINE

1. IF…
2. Nested IF
3. Switch Case

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.