Page No. :
Date : / /

Assignment-2 (Unit-2)

1 Explain Formatted Input in detail.

→

* Formatted Input : scanf()

- When we accept the data using ~~scan~~ scanf() function, it is formatted statement.

- It refers to an input data that has been arranged in a particular format.

* Syntax :

- scanf("control string", Arg1, Arg2, Arg3 ----Argn);

→ where

* control string is

→ format specifier or control string. It contains field specification which direct the interpretation of input data.

→ control string specify the field format in which the data is to be entered.

→ Generally, control string consist the conversion character % and data type character.

* Arg1, Arg2, Arg3---- Argn specifies the address location of the data.

Page No. :
Date : / /

* Inputting Integer values
* Inputting float values
* Inputting character values
* Inputting String values.

** Inputting Integer values

- To input Integer values %d format specifier is used.

* Inputting Float values.

- To input Float values %f format specifier is used.

* Inputting character values.

- To input character values %c format specifier is used.

* Inputting String values

- to input string values %s format specifier is used.

Page No. :
Date : / /

2 Explain Formatted output in details.

→

* Formatted output: Printf()

* Formatted output or Printf()

* Printf("control string", Arg1, Arg2, Arg3, ... Argn

→ when

→ control string consist the one of the ~~flow~~ following:

⇒ Characters that will be Printed on the screen as they appear.

⇒ Format specification that defines the output format for display its item.

⇒ Escape sequence characters like \n, \t, \b etc.

→ Arg1, Arg2, Arg3 .... Argn are the variables or constant which we want to print.

* Displaying Integer values

- To Display Integer values %d format specifier is used.

* Displaying Float value

- To Displar Float values %f format specifier is used.

* Displaying character values

- To Displar character values %c format specifier is used.

* Displaying string values

- To Displar string values %s format specifier is used.

3 Explain unformated input in detail.

→

* Unformatted input:

- getc()
- getchar()
- gets()
- getch()

* getch()

- In reads a single character from a given input stream and returns the corresponding integer value (typically ASCII value of read character) on success.

* Syntax:

⇒ int getc(FILE * stream);

* getchar()

- This function is used to take input of a single character.

* Syntax:

⇒ var name = getchar();

* gets()

- This function is used to take input of a string:

* Syntax:

⇒ gets(String variable);

* getch()

- getch is used to hold the output screen and wait until user gives any type of input (i.e. until user press any key) so that they can read the character and due to this we able to see the output on the screen.

* Syntax:

- puts (String variable);

* putchar()

- This function is used to take print a single character.

* Syntax:

- putchar(char variable);

5 What is operator? Explain Each operator in detail with proper example.

→

* operator: operators are ~~symbols~~ symbols that indicates the type of operations 'c' has to perform on data or value of variables.

* 'c' has a wide range of operators to perform variable ous operations.

1 Arithmetic operators (+, -, *, /, %, ++, --)

2 Logical operators (&&, ||, !)

3 Relation operators (>, <, ==, >=, <=, !=)

4 Bit-wise operators (&, |, ^, <<, >>)

5 Assignment operators (=, +=, -=, *=, /=, %=)

6 Ternary operator and use of sizeof() function.

1 Arithmetic operator

⇒ An arithmetic operator performs mathematical operations such as addition, subtraction, multiplication, de division etc on numerical values (constants and variables).

operator	meaning of operator.
+	addition or unary plus
-	subtraction or unary minus
*	multiplication
/	division
%	remainder after division (modulo division)

* Example

```
#include<stdio.h>
#include<conio.h>

Void main()

{
```

```
int a=9, b=4, c;

c=a+b;

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

printf("a-b=%d\n", c);

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

c=a-b
printf("a-b=%d\n", c);

c=a*b
printf("a*b=%d\n", c);

c=a/b;
printf("a/b=%d\n", c);

c=a%b;
printf("a modulo b=%d\n, c);

getch();

}
```

output

- a+b=13

- a-b=5
- a*b=3b
- a/b=2
- a modulo b=1

* Unary operator

operator	meaning of operator
++	Increment
--	Decrement

→ Increment operators are used to increase the value of the variable by one and decrement operators are used to decrease the value of the variable by one in c programs.

* Syntax:

Increment operator:

⇒ ++ var_name; (or) var_name ++;

Decrement operator:

⇒ -- var_name; (or) var_name --;

Page No.:
Date: / /

* Operator and Description

1 Pre-increment operator (++i)
→ value of i is incremented before assigning it to the variable i

2 Post increment operator (i++)
→ value of i is incremented after assigning it to the variable i

3 Pre-decrement operator (--i)
→ value of i is decremented before assigning it to the variable i

4 Post-decrement operator (i--)
→ value of i is decremented after Assigning it to the variable i

* Example

```
#include <stdio.h>
#include <conio.h>

void main()
{

int a=5;

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

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

getch();

}
```

* Output

a=5
a++ = 5
++a = 7
a-- = 7
--a = 5
a=5

2 Logical operator

⇒ These operators are used to perform logical operations on the given expressions.

Operator	Description
&&	called logical AND operator. If both the operands are non-zero, then the condition becomes true.
||	called logical OR operator. If any of the two operands is non-zero, then the condition becomes true.

!	called Logical NOT operator. It is used to reverse the logical state of its operand. If a condition is true, then logical NOT operator will make it false.

* Example

```
#include <stdio.h>
#include <conio.h>

void main():

{

int a=5, b=5, c=10, result;

result=(a==b) && (c>b);
printf("%d\n", result);

result=(a==b) && (c<b);
printf("%d\n", result);

result=(a==b) || (c<b);
printf("%d\n", result);

result=(a!=b) || (c<b);
printf("%d\n", result);

result=!(a!=b);
printf("%d\n", result);
```

```
result =!(a==b)
printf("%d\n", result);

getch();
}
```

3 Relational or comparision operator

⇒ A relational operator checks the relationship between two operands.

⇒ If the relation is true then the result of the relational expression is 1, if the relation is false ~~is f~~ then the result of the relational expression is 0.

operator	Description
==	Checks if the values of two operands are equal or not. If yes, then the condition becomes true.
!=	Checks if the values of two operands are not equal, then the condition becomes true.
>	Checks if the values of left operand is greather than the value of right operand. If yes, then the condition becomes true.
<	Checks if the values of left operand is less than the value of right operand. If yes, then the condition becomes true.

>=	checks if the value of left operand is greater than or equal to the value of right operand. If yes, then the condition becomes true.
<=	checks if the value of left operand is less than or equal to the value of right operand. If yes then the condition becomes true.

* Example

```
#include<stdio.h>
#include<conio.h>

void main()

{

int x=12, y=13;

Printf("x=%d\n", x);
Printf("y=%d\n", y);
Printf("x>y: %d\n", x>y);
Print
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);
Printf("x!=y : %d\n", x!=y);
```

```
getch();
}}
```

output

```
x=12
y=13

x>y : 0
x>=y : 0
x<y : 1
x<=y : 1
x==y : 0
x!=y : 1
```

4 Bitwise operators

→

operators	meaning of operators
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
<<	Shift left
>>	Shift right

* Bitwise &

⇒ The bitwise AND operator combines the bits of two numbers. It returns a new number whose bits are set to 1 only if the bits were equal to 1 in both input numbers.

Example

```
#include<stdio.h>
#include<conio.h>

void main()
{

int a=97, b=5;
clrscr();

Printf("%d", (a&b));

getch();
}
```

* Bitwise |

⇒ The bitwise or operator (|) compares the bits of two numbers. The operator returns a new number whose bits are set to 1 if the bits are equal to 1 in either input number.

Page No. :
Date : / /

example

```
#include <stdio.h>
#include <conio.h>

void main()
{

int a=7, b=5
clrscr();
Printf("%d", (a|b));

getch();
}
```

* Bitwise ^

⇒ The bitwise XOR operator, or "exclusive OR operator", compares the bits of two numbers. The operator returns a new number whose bits are set to 1 where the input bits are different and are set to 0 where the input bits are the same;

* Example

```
# include <stdio.h>
# include <conio.h>

void main()

{
```

```
int a=7, b=5;
clrscr();
printf("%d", (a^b));

getch();
}
```

* Bitwise <<

⇒ << Takes two numbers, left shifts the bits of the first operand, the second operand decides the number of places to shift

⇒ or in other words left shifting an integer "x" with an integer "y" (x<<y)

* Example

```
#include <stdio.h>
#include <conio.h>

void main()
{

int a=7
clrscr();
printf("%d", (a<<1));

getch
}
```

Page No. :
Date : / /

* Bitwise >>

⇒ >> takes two numbers, right shifts the bits of the first operand the second operand decides the number of places to shift

⇒ or in other words right Shifting an integer "x" with an integer "y" (x>>y)

* Example

```
#include <stdio.h>
#include <conio.h>

void main()
{

int a=7;
clrscr();
printf("%d", a(a>>1));

getch();
}
```

5 Assignment operator.

⇒

operator	Description
=	simple assignment operator. Assigns values from right side ~~will~~ operands to left side operand.

`+=`	Add AND Assignment operator it adds the right operand to the left operand and assign the result to the left operand.
`-=`	subtract AND assignment operator. It subtracts the right operand from the left operand and assigns the result to the left operand.
`*=`	multiply AND assignment operator. It multiplies the right operand with the left operand and assigns the result to the left operand.
`/=`	Divide AND Assignment operator. It divides the left operand with the right operand and assigns the result to the left operand
`%=`	modulus AND assignment operator. It takes modulus using two operands and assigns the result to the left operand.

* example

```
#include <stdio.h>
#include <conio.h>

void main()
{
```

```
int a=5, c;

c=a;
Printf("c = %d \n", c);
Printf

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

c-=a;
Printf("c= %d\n", c);

c*=a
Printf("c= %d\n", c);

c/=a;
Printf("c= %d\n", c);

c%=a;
Printf("c= %d\n", c);

getch();
}
```

* Output

```
c=5
c = 10
c = 5
c = 25
c = 5
c = 0
```

Page No.:
Date: / /

6 Conditional or Ternary or operator

⇒ conditional operators return one value if condition is true and returns another value is condition is false.

* Syntax:

→ Expr1 ? Expr2 : Expr3

- Expr1 will be condition, it will executes ~~fo~~ first

- If Expr1 is true then Expr2 will be executed

- If expr1 is false then Expr3 will be executed.

* example

```
#include <stdio.h>
#include<conio.h>

void main()
{

int a=5, b=7, c;
c=a<b ? a:b;

printf("%d", c);
getch();
}
```

* sizeof() operator

⇒ It returns the size of a variable. It can be applied to any date type, float type, pointer type variables.

⇒ When sizeof() is used with the data types, it simply returns the amount of memory allocated to that data type.

⇒ The output can be different on different machines like a 32 bit system can show different output while a 64-bit system can show different of same data types.

* Example

```
#include <stdio.h>
#include <conio.h>

int main() {
int a = 16;

printf("size of variable a: %d\n", sizeof(a));
printf("size of int data type: %d\n", sizeof(int));
printf("size of char data type: %d\n", sizeof(char));
printf("size of float data type: %d\n", sizeof(float));
printf("size of double data type: %d\n", sizeof(double));

return 0;
}
```

output

size of variable a: 2
size of Int : 2
size of Char: 1
size of float: 4
size of double: 8

b what is expression? explain each expression in detail with proper example.

⇒

Expressions:

- An expression is a combination of variables, constants and operators.

- Every expression results in some value of a certain type that can be assigned to a variable.

1 Arithmetic Expression

2 Boolean OR logical Expression

1 Arithmetic Expression

⇒ An arithmetic expression contains only Arithmetic operators and operands.

⇒ An Arithmetic expression is a combination of variables, constants and arithmetic operators (+, -, *, /, %)

Page No. :
Date : / /

* example

5+7
8/2
a+a
a-b

2 Boolean OR Logical Expression
⇒ A Boolean expression is a logical statement that is either TRUE or False.

⇒ An Boolean expression is a combination of variables, constants and logical operators (&&, ||, !).

* example

- a<b && a<c
- b>a || b>c
- ! (a>b)

Algebraic expression	c expression
a×b-c	a * b-c
(m+n) (x+y)	(m+n)*(x+y)
(ab/c)	a*b/c
$3x^2+2x+1$	3*x*x+2*x+1

(x/y) +c

x/y+c

2 π r

2*3.14*r

7 Explain string functions in detail with example

⇒ String function

- strlen
- strcmp
- strcpy
- strcat
- strrev

1 strlen()

⇒ strlen() function calculates the length of string.

x Syntax:

- strlen(string_name);

⇒ here string_name is the string which we want length

⇒ Function strlen() returns the value of type integer.

Page No. :
Date : | |

* Example

```
#include<stdio.h>
#include<conio.h>
#include<string.h>

void main()
{

char a[50] = "VNSGU";
clrscr();
printf("%d", strlen(a));
printf("\n%d", strlen("20BCA"));

getch();
}
```

2 strcmp

⇒ strcmp() compares two string and returns value 0, if the two strings are equal

⇒ function strcmp() takes two arguments i.e, name of two string to compare

X Syntax:

- strcmp(string1, string2);

Page No. :
Date : / /

* Example

```
#include <stdio.h>
#include <conio.h>
#include<string.h>

Void main()
{

Char a[50] ="VNSGCU";
Char b[50]="VNSGCU";
clrscr();
Printf("%d", strcmp(a,b));
getch();
}
```

3 strcpy()

⇒ strcpy() copies the content of one string to the content of another string.

X Syntax:

- strcpy (destination, source);

⇒ Here, source and destination are both the name of the string. This statement copies the content of string source to the content of string destination.

Page No. :
Date : | |

* ~~exnple~~ example

```
#include <stdio.h>
#include<conio.h>
#include<string.h>

void main()
{

char a[50]="VNSCTU";
char b[50]=" ";
clrscr();
strcpy(b,a);
printf("%s",b);
}
getch();
}
```

4 strcat()

⇒ strcat concatenates (joins) two strings.

⇒ It takes two arguments i.e, two strings and resultant string is stored in the first string specified in the argument

X Syntax:

```
strcat(first_string,second_string);
```

Page No.:
Date: / /

* example

```
#include<stdio.h>
#include<conio.h>
#include<string.h>

void main();
{

char a[50]="VNSGCU";
char b[50]="SURAT";
clrscr();
strcmp(a,b);
printf("%s",a);

getch();
}
```

5 strrev()

⇒ The strrev() function is used to reverse the given string.

X syntax:

- strrev(string);

* example

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
```

```
void main()
{

Char a[50]="VNSGTU";
clrscr();
Printf("%s",strrev(a));

getch();
}
```

8 Explain maths functions in details with example.

⇒ mathematical functions

- abs()
- floor()
- round()
- ceil()
- sqrt()
- exp()
- log()
- sin
- cos()
- tan()
- pow
- trunc()

1 abs

⇒ abs() function in c returns the absolute value of an integer.

⇒ The absolute value of a number is always positive. only integer values are supported in c.

✗ Syntax:

```
int abs (int n);
```

* example

```
# include <stdio.h>
#include <conio.h>
# include <math.h>

void main()
{

int a=77;
clrscr();
Printf("%d", abs (a));
Printf("\n%d", abs (88));

getch();
}
```

2 floor()

⇒ floor function in c returns the nearest integer value which is less than or equal to the floating point argument passed to this function.

Page No.:
Date: / /

x Syntax:

```
double floor (double x);
```

* example

```
#include<stdio.h>
#include<conio.h>
#include<math.h>

void main()
{

float a=7.77;
clrscr();
printf("%f",floor(a));
getch();
}
```

3 round()

⇒ round() function in C returns the nearest integer value of the float/double/long double argument passed to this function.

⇒ If decimal value is from ".1 to .5", it returns integer value less than the argument. If decimal value is from ".6 to .9", it returns the integer value greater than the argument.

* Syntax:

```
double round (double a);
```

* Example

```
#include<stdio.h>
#include<conio.h>
#include<math.h>

void main()
{

float a=7.77;
clrscr();
printf("%f",round(a));
getch();
}
```

4 ceil()

⇒ ceil() function in C returns nearest integer value which is greater than or equal to the argument passed to this function.

* Syntax:

```
double ceil( double x);
```

Page No. :
Date : / /

* example

```
#include <stdio.h>
#include<conio.h>
#include<math.h>

void main()
{

float a=7.77;
clrscr();
printf("%.0f", ceil(a));
getch();
}
```

5 sqrt()

⇒ sqrt() function in c is used to find the square root of the given number.

X syntax

```
double sqrt(double x);
```

* example

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
```

```
void main
{

clrscr();
printf("%f", sqrt(64));
getch();
}
```

6 exp()

⇒ The exp() function computers the exponential raised to the given argument.

× Syntax

- double exp(double arg);

* example

```
#include<stdio.h>
#include<conio.h>
#include<math.h>

void main
{

clrscr();
printf("%f", exp(2));
getch();
}
```

Page No. :
Date : / /

7 log()

=> log() function is used to calculates natural logarithm.

X Syntax:

```
double log(double arg);
```

* Example

```
#include <stdio.h>
#include<conio.h>
#include <math.h>

void main()
{

clrscr();
Printf("%f", log(20));
getch();
}
```

8 sin()

=> sin() function in c are used to calculate sine value.

X Syntax

```
double sin(double arg);
```

Page No. :
Date : / /

* example

```
#include<stdio.h>
#include<conio.h>
#include<math.h>

Void main()
{

clrscr();
Printf("%.f", sin(10));
getch();
}
```

9 cos()

⇒ cos() function in C are used to calculate ~~con~~ cosine values.

X syntax:

```
double cos(double arg);
```

* example

```
#include<stdio.h>
#include<conio.h>
#include<math.h>

void main()
{
```

```
clrscr();
Printf("%.f", cos(20));
getch();
}
```

10 tan()

⇒ tan() function in c are used to calculate tangent values.

x Syntax

```
double tan(double arg);
```

* Example

```
#include<stdio.h>
#include<conio.h>
#include<math.h>

Void main()
{
clrscr();
Printf("%.f", tan(20));
getch();
}
```

11 Pow()

⇒ Pow() function in C is used to find the Power of the given number.

```
clrscr();
printf("%f", cos(20));
getch();
}
```

10 tan()

⇒ tan() function in C are used to calculate tangent values.

x Syntax

```
double tan(double arg);
```

* Example

```
#include<stdio.h>
#include<conio.h>
#include<math.h>

Void main()
{
clrscr();
printf("%f", tan(20));
getch();
}
```

11 Pow()

⇒ Pow() function in C is used to find the power of the given number.

Page No.:
Date: / /

x Syntax

double pow(double base, double exponent);

* example

```
#include <stdio.h>
#include <~~math.h~~> <conio.h>
#include <math.h>

void main()
{
clrscr();
printf("%f", pow(5,2));
getch();
}
```

12 trunc()

⇒ Truncates a double value after the decimal point and gives the integer part as the result. The return value and the arguments are of the type double

x syntax:

double trunc(double x);

Page No. :
Date : / /

* example

```
#include<stdio.h>
#include<conio.h>
#include<math.h>

void main()

{
clrscr();
Printf("%f", trunc(5.55));
getch();
}
```

9 explain #include and #define directives.

⇒ ~~in~~

~~include Syntax~~

~~#include<file>~~

~~#include"file"~~

* stdio.h

⇒ The header file stdio.h stands for standard Input output. It has the information related to input/output functions.

⇒ Printf(), Scanf(), getc(), putc(), etc are in stdio.h

* conio.h

⇒ The header file conio.h stands for console Input output

⇒ cprscr(), getch, getche(), etc are in conio.h

X Use of #include

⇒ The #include preprocessor directive is used to paste code of given file into current file. It is used include system-defined and user-defined header files. If included file is not found, compiler renders errors.

X Include syntax

#include <file>

#include "file"

* #define

⇒ The #define directive allows the definition of macros within your source code.

⇒ macro definitions are not variables and cannot be changed by your program code like variables.

⇒ You generally use this syntax when creating constants that represent numbers, strings or expressions.

* Syntax

- #define CNAME value

⇒ CNAME: The name of the constant. most C programmers define their constant name in uppercase, but it is not a requirement of the C Language.

⇒ value: The value of the constant.

10 explain type specifiers in detail.

→

Data type	Printf specifier	Scanf specifier
long double	%lf	%lf
double	%f	%lf
fioat	%f	%f
unsigned long int	%lu	%lu
long int	%ld	%ld
unsigned int	%u	%u
int	%d	%d
short	%hd	%hd
char	%c	%c