21-Day C Programming Challenge

Learning Topics (Days 4–5)

Beginner-Friendly Student Notes with Examples

By the end of Days 4 and 5, you will learn how to:

- Accept input from the user.
- Display output in different formats.
- Convert one data type into another.
- Use comparison, logical, and bitwise operators.
- Understand operator precedence.
- Write simple decision-making programs.

Day 4 – Input, Output & Type Conversion

Learning Objectives

After completing this lesson, you will be able to:

- Use scanf() to take input.
- Use printf() to display output.
- Understand format specifiers.
- Perform type conversion.
- Write interactive programs.

---

1. Input and Output in C

A program becomes useful when it can:

- Accept data from the user (**Input**)
- Display results (**Output**)

Example:

Enter Age → 20

Program → Age is 20

2. Output Function - printf()

printf() displays information on the screen.

Syntax

```
printf("Message");
```

Example

```
#include<stdio.h>

int main()
{
   printf("Welcome to C Programming");

   return 0;
}
```

Output

```
Welcome to C Programming
```

Printing Variables

```
#include<stdio.h>

int main()
{
   int age = 20;

   printf("Age = %d", age);

   return 0;
}
```

Output

```
Age = 20
```

---

3. Input Function - scanf()

scanf() reads data from the keyboard.

Syntax

```
scanf("format",&variable);
```

Notice the & before the variable name.

Example

```
#include<stdio.h>

int main()
{
   int age;

   printf("Enter Age: ");

   scanf("%d",&age);

   printf("Age = %d",age);

   return 0;
}
```

Output

```
Enter Age: 21

Age = 21
```

Why do we use '&'?

Example

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

& means **address of** the variable.

scanf() stores the entered value directly into the variable's memory location.

Think of it like giving the program the "house address" where the value should be stored.

4. Format Specifiers

Data Type	Format Specifier
int	%d
float	%f
char	%c
double	%lf
string	%s

Example

```
int age;
float salary;
char grade;

scanf("%d",&age);
scanf("%f",&salary);
scanf("%c",&grade);
```

Program 1 – Read an Integer

```
#include<stdio.h>

int main()
{
   int number;

   printf("Enter Number: ");

   scanf("%d",&number);

   printf("You entered %d",number);

   return 0;
}
```

Program 2 – Read a Float

```
#include<stdio.h>

int main()
{
   float marks;

   printf("Enter Marks: ");

   scanf("%f",&marks);

   printf("Marks = %.2f",marks);

   return 0;
}
```

Output

```
Enter Marks: 95.50

Marks = 95.50
```

%.2f prints only two digits after the decimal point.

Program 3 – Add Two Numbers

```
#include<stdio.h>

int main()
{
   int a,b,sum;

   printf("Enter Two Numbers: ");

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

   sum = a + b;

   printf("Sum = %d",sum);

   return 0;
}
```

Output

```
Enter Two Numbers: 10 20

Sum = 30
```

Program 4 – Average of Three Numbers

Formula

```
Average = (a+b+c)/3
```

Program

```
#include<stdio.h>

int main()
{
   float a,b,c,average;

   printf("Enter Three Numbers: ");

   scanf("%f%f%f",&a,&b,&c);

   average = (a+b+c)/3;

   printf("Average = %.2f",average);

   return 0;
}
```

5. Type Conversion

Sometimes one data type changes into another.

Example

```
int a = 10;

float b = a;
```

Output

```
10.000000
```

The integer becomes a float automatically.

Implicit Type Conversion

Done automatically by the compiler.

```
int a = 10;

float b = a;

printf("%f",b);
```

Output

```
10.000000
```

Explicit Type Conversion (Casting)

Done by the programmer.

Example

```
float average;

average = (float)(10+20)/3;
```

Output

```
10.00
```

Without casting

```
average=(10+20)/3;
```

Output : 10

Because integer division removes the decimal part.

Challenge Program

Find Simple Interest

Formula

$SI = (P \times R \times T)/100$

Program

```
#include<stdio.h>

int main()
{
   float p,r,t,si;

   printf("Enter Principal, Rate and Time: ");

   scanf("%f%f%f",&p,&r,&t);

   si = (p*r*t)/100;

   printf("Simple Interest = %.2f",si);

   return 0;
}
```

Practice Questions (Day 4)

Write programs to:

1. Read your age.
2. Read two integers and find the sum.
3. Read three numbers and find the average.
4. Calculate the area of a circle.
5. Calculate simple interest.
6. Convert kilometers into meters.
7. Convert minutes into hours and minutes.

Day 4 Summary

You learned:

- printf()
- scanf()
- Format specifiers
- User input
- Type conversion
- Casting
- Average calculation

Day 5 – Operators

Learning Objectives

After this lesson, you will be able to:

- Compare numbers.
- Make logical decisions.
- Use bitwise operators.
- Understand operator precedence.

1. Relational Operators

Relational operators compare two values.

The result is either:

- 1 (True)
- 0 (False)

Operator	Meaning
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
==	Equal to
!=	Not equal

Example

```
#include<stdio.h>

int main()
{
   int a=20,b=10;

   printf("%d\n",a>b);
   printf("%d\n",a<b);
   printf("%d\n",a==b);

   return 0;
}
```

Output

```
1
0
0
```

Program – Compare Two Numbers

```
#include<stdio.h>

int main()
{
   int a,b;

   printf("Enter Two Numbers: ");

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

   if(a>b)
      printf("%d is Greater",a);

   return 0;
}
```

2. Logical Operators

Logical operators combine conditions.

Operator	Meaning
&&	AND
||	OR
!	NOT

Logical AND (&&)

Both conditions must be true.

```
if(age>=18 && age<=60)
```

Logical OR (||)

At least one condition must be true.

```
if(marks>=35 || sports=="Yes")
```

Logical NOT (!)

Reverses the result.

```
if(!(a>b))
```

Example

```
#include<stdio.h>

int main()
{
   int age=20;

   if(age>=18 && age<=60)
      printf("Eligible");

   return 0;
}
```

Output

Eligible

---

3. Bitwise Operators (Important)

Bitwise operators work on the binary form of numbers.

Example

5 = 0101

3 = 0011

Bitwise AND (&)

0101

0011

0001

Result

1

Program

```
printf("%d",5&3);
```

Output

1

Bitwise OR (|)

0101

0011

0111

Result

7

Program

```
printf("%d",5|3);
```

Bitwise XOR (^)

0101

0011

0110

Output

6

Program

```
printf("%d",5^3);
```

Bitwise NOT (~)

Example

```
printf("%d",~5);
```

Output

-6

~ flips all the bits of the number.

4. Operator Precedence

When an expression has multiple operators, C follows precedence rules.

Example

```
int x;

x = 5 + 3 * 2;
```

Output

```
11
```

Multiplication is done before addition.

Equivalent to:

$5 + (3 \times 2)$

Using Parentheses

```
x = (5+3)*2;
```

Output

```
16
```

Parentheses have the highest priority and make expressions easier to understand.

Common Operator Precedence (High to Low)

Priority	Operators
Highest	()
Unary	++, --, !
Multiplication	*, /, %
Addition	+, -
Relational	<, >, <=, >=
Equality	==, !=
Logical AND	&&
Logical OR	`
Assignment	=

Challenge Program

Check Even or Odd Using Bitwise Operator

Idea:

If the last bit of a number is 0, it is even. If it is 1, it is odd.

Program

```
#include<stdio.h>

int main()
{
   int number;

   printf("Enter Number: ");

   scanf("%d",&number);

   if(number & 1)
      printf("Odd Number");
   else
      printf("Even Number");

   return 0;
}
```

Example

Input

15

Output

Odd Number

Input

20

Output

Even Number

Practice Questions (Day 5)

Write programs to:

1. Compare two numbers and print the larger one.
2. Check whether a number is positive or negative.
3. Check whether two numbers are equal.
4. Check whether a person is eligible to vote (age ≥ 18).
5. Check if a number lies between 1 and 100 using &&.
6. Print the results of 5 & 3, 5 | 3, and 5 ^ 3.
7. Check whether a number is even or odd using number & 1.
8. Evaluate and explain the output of:
    - 5 + 3 * 2
    - (5 + 3) * 2
    - 20 / 5 + 3
    - 20 / (5 + 3)

Days 4–5 Quick Revision

Concept	Key Point
printf()	Displays output on the screen.
scanf()	Reads input from the keyboard using the variable's address (&).
Format specifiers	%d, %f, %c, %lf, %s match different data types.
Type conversion	Converts data from one type to another, either automatically or with casting.
Relational operators	Compare values and return 1 (true) or 0 (false).
Logical operators	Combine or negate conditions using &&, `
Bitwise operators	Perform operations on binary representations of integers.
Operator precedence	Determines the order in which parts of an expression are evaluated; use parentheses to make intent clear.

Study Tip: Type every program yourself instead of copying and pasting. Then modify the input values and operators to observe how the output changes. This hands-on practice is the fastest way to build confidence in C programming.