# FLUORESCENT SECONDARY SCHOOL

Tokha-08, Kathmandu


## A Project Report of Computer SCI:
## 'C' Programming

Submitted By: Niranjan Kattel
Class: XI
Faculty: Science

Date of Performance:
2079/12/25

Submitted to: (.....)
Er. Gopal Pd. Shah
(Department of Computer)

Date of Submission:
2080/01/04

# Acknowledgment

I would like to express my special thanks of gratitude to my computer teacher **Mr. Gopal Prashad Shah** for their able guidance and support in competing mind project. I came to know so many new things. I am really thankful to them. Thank you.

**Niranjan Kattel**

**Fluorescent Secondary School**

# Certificate


This is to certify that this lab assignment prepared by **Niranjan Kattel** in partial fulfillment of the requirement for the program of grade 11, computer science has been evaluated. In your opinion, it is satisfactory in the scope and quality as a lab assessment for the required program.

(……)
Internal Examiner

(……)
External Examiner

# Programming Concepts and Logics

## Table of Contents


**Programming Language** ..... **1**

Types of programming Language ..... 1

**Language Translators** ..... **3**

Compiler ..... 3

Interpreter ..... 3

**Introduction to C language** ..... **4**

Features ..... 4

Advantages ..... 4

Applications ..... 5

**Basic Structure of C program** ..... **5**

**Variables in C** ..... **7**

**Constants in C** ..... **8**

**Data Types in C** ..... **9**

**Printf and scanf in C** ..... **10**

**Format Specifiers** ..... **12**

**Escape Sequence** ..... **12**

**Operators in C** ..... **13**

**Control Statements in C** ..... **16**

**Looping Control Statements** ..... **22**

**Arrays in C** ..... **27**

# Programming Language

The language which is used to develop program in computer is called programming language. For e.g.: C, C++, Java, etc.

## Types of Programming languages

1. **Low level language**

   The low-level language is a machine- oriented language so it is closer to what machine can understand.

   a. **Machine level language (1GL)**

      It is the first programming language used to make programming computer. It is the language of CPU as it consists of series of zeros and ones. It is a machine dependent programming language. Program written for one type of processor cannot be run in other type of processor.

   b. **Assembly level (2GL)**

      In Assembly level language, instead of writing the instruction in the series of zeros and ones, we can use symbolic instructions like add, sub, etc. Along with the decimal number in order to perform different types of tasks and calculations. It is not directly understood by the CPU, so it needs conversion and conversion is done by a similar.

2. **High level language**

   a. **Third generation language**

      High level language code is written in English like structure and mathematical notation. It is easier to write debug and understand the program in high level language. It needs conversion and the conversion is done by translating software like compiler or interpreter.

**b. Fourth generation language**

It was developed after a high-level language, so it is one step ahead from high level language. It is a result-oriented programming language, and it contains database query language. It is also translated to machine codes by compiler or interpreter.

**c. Fifth generation language**

Fifth generation language are still in development. It is more intelligent and user friendly. The special feature of 5th generation language is that it can be expressed as human understandable language.

## Language Translators

### 1) Compiler

It is a software that converts source program written in high level language into missing level language at once. The compiler reads the complete program at 1st and if it is free, then it converts the source program into object program at once. It converts source program into object program faster than interpreter.

### 2) Interpreter

It is also software that translates source program code written in high level language into mission level language, line by line. Unlike compiler, the interpreter reads each line of source program and if it is bug free, then it converts program source code into machine language.

# Introduction to C language

C is a general-purpose, procedural computer programming language. It is the base for all the high-level programming languages and is also known as middle level programming language. It was developed by Denish Ritchie in 1972. Initially, it was designed to implement UNIX OS.

## Features:

- It is a middle-level language so it has the combined form of both capabilities of assembly and high-level language.
- C language is portable as programs that are written in C language can run and compile on any system with either none or small changes.
- From system programming to photo editing software, the C programming language is used in various applications.
- It is a diversified language with a rich set of built-in operators which are used in complex or simplified C programs.
- It has high speed, efficiency and flexibility.

## Advantages:

- It is easy to understand
- Presence of many Libraries
- Easy to write
- Low cost and portability
- Fast execution and Compilation speed
- Easy debugging
- Procedure Oriented Language
- Execution of algorithms and data structures
- Dynamic memory allocation

## Applications:

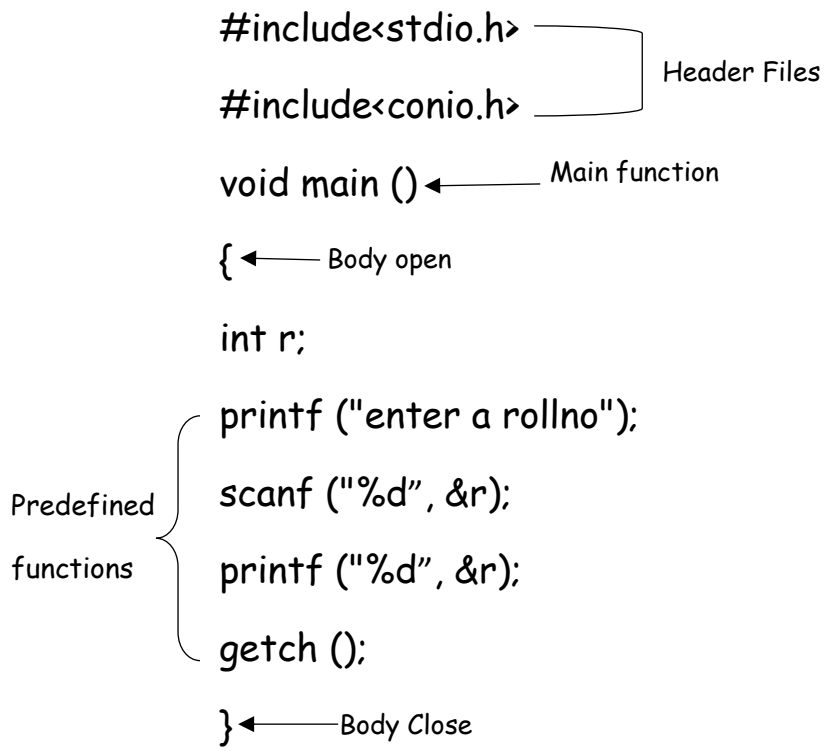
C programming language can be used:

- To design the system software like operating system and Compiler.
- To develop application software like database and spread sheets.
- For develop Graphical related application like computer and mobile games.
- To evaluate any kind of mathematical equation.
- To design GUI Applications such as Adobe Photoshop.

## C tokens

C Tokens are the smallest element of a C program, which are meaningful to the compiler. Keywords, Identifiers, Constants, Strings and operators are some of the tokens in C language

## Basic Structure of a C Program



1. **Stdio**
    - It stands for Standard Input Output.
    - It is a collection of pre-defined functions.
    - It is also called library of "C".
2. **Conio**
    - It stands for Console input output.
    - It is used to show output on console window.
3. **Void**
    - It indicates that no value is being returned by the function.
4. **Main**
    - This function is also called the entry point of any program.
    - The execution of any program starts from the main function
5. **clrscr**
    - It stands for clear screen.
    - It is a predefined function defined under conio.h header file, which is used to clear the output screen.
6. **printf**
    - It is a predefined function which is used to print information or data on the output screen.
    - It is defined in the stdio.h header file.
7. **getch**
    - It is a predefined function which is used to hold the output screen.
    - It is defined in the conio.h header file.

# Variables in C language

A variable is a named location in the memory of computer which can hold a value at a time. It is called variable because it's value can be changed during the program execution. It always contains a last value stored to it and is declared with data type.

## Variable Declaration: Syntax

- Data-type variable_name; (E.g.: int a;)
- Data-type variable_name = value to variable;
  E.g.: float pi = 3.1415, r;

NOTE: All declaration systems must end with a semicolon. (;)

## Variable initialization

The process of assigning a value to a variable is known as variable initialization. A variable that is not initialized does not have a defined value, hence it cannot be used until it is assigned such a value.

E.g.: int rollno = 17;  float marks = 78.5;  char grade = 'A';

## Rules to Declare variables in C

- A variable can have alphabets, digits, and underscore.
- A variable name can't start with a digit.
- No space is allowed within the variable name.
- A variable name must not be any reserved word or keyword, e.g., int, goto, etc.

## Types of variables

i. Local variable: declared within a function.
ii. Global variable: declared outside the scope of any function.
iii. Static variable: declared using static keyword.

# Constants in C language

A constant is a fixed value which cannot be changed during the program execution. For example, 10 and 5 are constants. The fixed values are also called literals.

## Types of Constants

1. **Numeric**
   - Integer: The constant that contain natural numbers (no decimal point) is called integer constant. E.g.:56,7896,863890
   - Float: The constant that contain fractional number (having decimal point) is called floating constant. E.g.: 4.5, 578.976,68394.8.

2. **Character**
   - Character: The constant that contains a single character is called character constant. It is enclosed in a single quotation mark. E.g.: 'A','b',etc.
   - String: The constant that contains alpha-numeric variables inside the double quotation marks is called string constant. E.g.: "welcome", "home", "grin", etc.

## Syntax to declare a constant

Const data-type variable-name = value-to-variable;

E.g.:

integer constant: const int a = 7;

floating point constant: const float pi = 3.1415;

# Data types in C

Each variable in C has an associated data type. Each data type requires different amounts of memory and has some specific operations which can be performed over it. Different data types also have different ranges up to which they can store numbers. These ranges may vary from compiler to compiler. In C programming there are two types of data type.

They are:

1. **Primary/ Basic data types**

   - char: The most basic data type in C. It stores a single character and requires a single byte of memory in almost all compilers.

   - int: As the name suggests, an int variable is used to store an integer.

   - float: It is used to store decimal numbers (numbers with floating point value) with single precision.

   - double: It is used to store decimal numbers (numbers with floating point value) with double precision.

**2.Secondary/Derived data types**

The data type which is constructed by merging some features of Primary data type is called secondary a type. These can hold similar and dissimilar data on it. It includes array, pointer, structure, union, etc.

# printf and scanf in C language

They pre- defined functions in the library of c in header file stdio.h. They are commonly used for input and output operation.

### printf function

It is a predefined function used to print information or data on the output screen. It's case sensitive so must be written in lowercase.

Syntax: printf (format_specifier, variable_name);

### scanf function

It is a predefined function used to take user input at the time of execution of program. It's case sensitive so must be written in lowercase.

Syntax: scanf (format_specifier, address of variable);

# printf and scanf with integer

```c
//printf and scanf with integer
#include<stdio.h>
#include<conio.h>
int main ()
{
    int rollno;
    printf("Enter your rollno\n");
    scanf("%d",&rollno);
    printf("roll no = %d",rollno);
}
```

Output:

```
Enter your rollno
17
roll no = 17
--------------------------------
```

# Printf and scanf with float

```c
//printf and scanf with float
#include<stdio.h>
#include<conio.h>
void main()
{
    float marks;
    printf(" enter your marks\n");
    scanf("%f",&marks);
    printf("Your mark = %f",marks);
    getch();
}
```

Output:

```
 enter your marks
48.5
Your mark = 48.500000
```

## Printf and scanf with string

```c
//printf and scanf with string
#include <stdio.h>
#include <conio.h>
int main()
{
    char name[20];
    printf("enter your name:\n");
    scanf("%s",&name);
    printf("Your name is %s",name);
    return(0);
}
```

output:

```
enter your name:
Niranjan
Your name is Niranjan
--------------------------------
```

## Printf and scanf with char

```c
//printf and scanf with char
#include<stdio.h>
#include<conio.h>
int main()
{
    char grade;
    printf("Enter your grade\n");
    scanf("%c",&grade);
    printf("Grade=%c",grade);
}
```

output:

```
Enter your grade
A
Grade=A
--------------------------------
```

| Format Specifier | Use case |
|---|---|
| %d, %i | Signed integer |
| %u | Unsigned integer |
| %o | Octal integer |
| %x | Hexadecimal number, lowercase |
| %X | Hexadecimal number, Uppercase |
| %G, %g | Floating point number |
| %c | Character |
| %s | String |
| %[^\n] | String until enter character |
| %f | Scans floating point number |
| %lf | Scans double floating point no |
| %Lf | Scans double floating point no |
| %e | Float no for sci. exponent no |

| Escape Sequence | Meaning |
|---|---|
| \' | Prints ' in output |
| \" | Prints " in output |
| \\ | Print \ in output |
| \0 | Terminates string |
| \a | Audible beep sound |
| \b | Brings cursor 1 step back |
| \n | New line in output |
| \r | Enter key pressed |
| \t | Creates tab or 8 spaces |

# Operators in C

It is a special symbol which is used to perform logical or mathematical operation on one or more operands. Operands are the values or variables declared within a program on which the operation is to be performed.

**Types of operators**

- Arithmetic Operators (+, -, *, /, %)
- Relational Operators (==, !=, >, <, >=, <=)
- Logical Operators (&&, !)
- Assignment Operators (=, +=, -=, *=, /=, %=)
- Bitwise Operators (&, <<, >>)
- Increment/Decrement Operators (++, --)
- Conditional Operators (?)
- Special Operator ( &, *, sizeof())

# Keyword

A special word in programming language whose meaning has been already defined to the compiler. A keyword performs a specific task in computer program.

Some keywords are:

| | | | |
|---|---|---|---|
| char | float | double | union |
| structure | for | const | void |
| int | goto | return | if |
| else | if | do | while |

## Some Practice questions

1.

```c
//write a c program to input three numbers and display their sum
#include<stdio.h>
#include<conio.h>
void main()
{
    int a,b,c,d;
    printf("enter 3no. a,band c\n");
    scanf("%d %d %d", &a,&b,&c);
    d=a+b+c;
    printf("Sum=%d",d);
    getch();
}
```

```
enter 3no. a,band c
12
14
45
Sum=71
```

2.

```c
//write a c program to input a no. and find the square of it
#include<stdio.h>
#include<conio.h>
void main()
{
    int a,b;
    printf("enter a no\n");
    scanf("%d",&a);
    b=a*a;
    printf("square= %d",b);
    getch();
}
```

```
enter a no
1458
square= 2125764
--------------------
```

3.

```c
//write a c program to input principal amount, time and
//rate and find the simple interest
#include<stdio.h>
#include<conio.h>
void main()
{
    int p,t,r,s;
    printf("enter principal,time and rate");
    scanf("%d %d %d",&p,&t,&r);
    s=(p*t*r)/100;
    printf("Simple interest=%d",s);
    getch();
}
```

```
enter principal,time and rate 123456
12
7
Simple interest=103703
```

5.

```c
//write a c program to input length in km and convert it into m
#include<stdio.h>
#include<conio.h>
void main()
{
    float km,m;
    printf("Enter a length in kilometer\n");
    scanf("%f",&km);
    m=km*1000;
    printf("length in meter=%f",m);
    getch();
}
```

```
Enter a length in kilometer
1234
length in meter=1234000.000000
--------------------------------
```

6.

```c
//write a c program to divide an integer by another
//integer and find the quotient and remainder
#include<stdio.h>
#include<conio.h>
void main()
{
    int a,b,quot,rem;
    printf("enter two numbers\n");
    scanf("%d %d",&a,&b);
    quot = a/b;
    rem = a%b;
    printf("Quotient=%d",quot);
    printf("\nRemainder=%d",rem);
    getch();
}
```

```
enter two numbers
327
47
Quotient=6
Remainder=45
----------------------
```

# Control Statement in C

Control statement determines the flow of control in a program and enables us to specify the order in which the various instructions in a program are to be executed by computer. There are some control statements in C language which are as follows.

**If statement**

In this control statement, the body executes only if the given condition is true otherwise it does not execute.

Syntax:

if (expression)

{

Statement;

}

Examples:

```c
//to input two integers and show which is greater
#include<stdio.h>
#include<conio.h>
void main ()
{
    int x,y;
    printf (" Input two integers as x and y\n");
    scanf("%d %d",&x,&y);
    if (x==y)
    printf (" x is equal to y");
    if (x>y)
    printf("x is greater than y");
    printf("y is greater than x");
    getch();
}
```

```
 Input two integers as x and y
45
47
y is greater than x
--------------------------------
```

```c
//to input a no. and disply it its
//exactly divisible by 5 but not by 11
#include<stdio.h>
#include<conio.h>
void main ()
{
    int n;
    printf ("Enter a number\n");
    scanf ("%d",&n);
    if (n%5==0 && n%11!=0)
    {
        printf ("%d is divisible by 5 but not by 11",n);
    }
    getch();
}
```

```
Enter a number
45
45 is divisible by 5 but not by 11
--------------------------------
```

**If……. Else Statement**

In this control statement, if part of the program executes if the condition is true and if the condition is false, else part of the program executes. Syntax:

If (expression)

Statement1;

Else

State ment2;

```c
//input a no. and check if it's odd or even
#include<stdio.h>
#include<conio.h>
void main ()
{
    int n;
    printf("enter a number\n");
    scanf("%d",&n);
    if (n%2==0)
    printf("%d is even",n);
    else
    printf("%d is odd",n);
    getch();
}
```

```
enter a number
1234568092
1234568092 is even
----------------------
```

```c
//to input a mark and check if it's pass or fail
#include<stdio.h>
#include<conio.h>
void main ()
{
    int mark;
    printf("enter marks of a subject\n");
    scanf("%d",&mark);
    if (mark<35)
    printf("FAIL");
    else
    printf("PASS");
    getch();
}
```

```
enter marks of a subject
57
PASS
--------------------------
```

**If else ladder**

In this case, the program executes the condition that becomes true first from the top. If all conditions are false then else part executes. It's a part of conditional statement that executes only one statement at a time.

Syntax:

If (expression1)

Statement1;

Else if (expression2)

Statement 2;

Else if (expression3)

Statement 3;

Else

Statement 4;

```c
//to identify the type of entered character
#include<stdio.h>
#include<conio.h>
void main ()
{
    char c;
    printf ("Input a character\n");
    scanf("%c",&c);
    if (c>= 'a'&& c <='z')
    printf ("lowercase letter");
    else if (c>='0'&& c <='9')
    printf ("A digit");
    else if (c>= 'A'&& c <='Z')
    printf ("Uppercase letter");
    else if (c== ' ')
    printf (" Blank Space");
    else
    printf("an unknown character");
    getch();
}
```

```
Input a character
4
A digit
------------------------
```

```
Input a character
$
an unknown character
```

```c
// to find middle no in three integers
#include <stdio.h>
#include <stdlib.h>
void main()
{
    int a,b,c;
    printf("Enter three numbers\n");
    scanf("%d %d %d",&a,&b,&c);
    if(b>a && a>c || c>a && a>b)
    printf("\n%d is a middle number",a);

    if(a>b && b>c || c>b && b>a)
    printf("\n%d is a middle number",b);
    if(a>c && c>b || b>c && c>a)
    printf("\n%d is a middle number",c);
    getch();
}

```

```
Enter three numbers
16
15
17

16 is a middle number
------------------------------
```

## Nested if

Nested means one inside another so one if inside another if is called nested if.

Syntax:

```
If (expression)
{
If (expression)
  {
    Statement;
  }
}
```

Example:

```c
//to show the use of nested loop.
#include<stdio.h>
#include<conio.h>
void main ()
{
    int x = 10;
    if (x>5)
    {
        if (x<15)
        {
            printf ("x is greater than 5 and less than 15");
        }
    }
}
```

```
x is greater than 5 and less than 15
--------------------------------
```

## Switch-case Statement

The switch case statement is a special decision maker that tests whether an expression matches one of the numbers of signed integer constant or character or character constant values, with expression given inside the braces accordingly.

```c
// to display the day in accordance with no. e
#include<stdio.h>
#include<conio.h>
void main ()
{
    int option;
    printf("enter a number of day");
    scanf("%d",&option);
    switch (option)
    {
        case 1:
            printf("Sunday");
        break;
        case 2:
            printf("Monday");
        break;
        case 3:
            printf("Tuesday");
        break;
        case 4:
            printf("Wednesday");
        break;
        case 5:
            printf("Thusday");
        break;
        case 6:
            printf("Friday");
        break;
        case 7:
            printf("Saturday");
        break;
        default:
            printf("Invalid option");
        break;
    }
    getch();
}
```

```
enter a number of day3
Tuesday
--------------------------
```

```
enter a number of day7
Saturday
--------------------------
```

```
enter a number of day47
Invalid option
--------------------------
```

# Looping Control Statements

A loop is a way of repeating a statement number of times. Loops are used to repeat a block of codes. The main advantage of loop is that it makes the program very simple and short to produce a significantly greater result simply by the repetition. The control statements to loop certain block of statements in known as Looping control statements. C provides three looping control structures:

- For loop
- While loop
- Do while loop

## For loop

The for loop is one of the broadly used loop in programming language. The execution of for loop until the condition is true. The for loop is an entry control loop because it checks the condition at entry point.

Syntax:

For (variable initialization; condition; variable increment/decrement)

{

Block of codes

}

```c
//to print numbers form 1-10
#include<stdio.h>
#include<conio.h>
 void main()
{
    int i;
    for (i=1;i<=10;i++)
    {
        printf("  %d\n",i);
    }
}
```

```
1
2
3
4
5
6
7
8
9
10

-----------
```

```c
//sum of n natural numbers
#include<stdio.h>
#include<conio.h>
 void main()
 {
    int n, a,b;
    b=0;
    printf("enter a number");
    scanf("%d",&n);
    for (a=1;a<=n; a++)
    {
        b += a;
     }
     printf(" Sum=%d",b);
     getch();
 }
```

```
enter a number19
 Sum=190
--------------------------------
```

**While loop**

The while loop is also an entry control loop. The wile loop must initialize variable before loop begins. The termination condition consists inside while parentheses.

Syntax:

While (termination condition)

{

Block of codes

}

## Examples

```c
//to print numbers form 1-10
#include<stdio.h>
#include<conio.h>
 void main()
{
    int i=1;
    while (i<=10)
    {
        printf("  %d\n",i);
    i++;
    }
}
```

```
1
2
3
4
5
6
7
8
9
10

-------
```

```c
//sum of n natural numbers
#include<stdio.h>
#include<conio.h>
 void main()
 {
    int n, a,b;
    a=1;
    b=0;
    printf("enter a number");
    scanf("%d",&n);
    while (a<=n)
    {
        b += a;
        a++;
     }
     printf(" Sum=%d",b);
     getch();
 }
```

```
enter a number17
 Sum=153
------------------------
```

**Do while loop**

This loop is an exit control loop. This loop runs at least once even though the termination statement is set false. This loop test the condition at exit point hence it is called exit control loop.

Syntax

Do

{

Block of statements

}

While (condition);

## Differences Between While loop and do while loop

| While Loop | Do while loop |
|---|---|
| In this, controlling condition appears at the start of the loop. | In this, the controlling condition at the end of the loop. |
| The loop does not occur if the condition at the fist loop is false. | The loop occurs at least once even if the condition is false at the first loop. |
| It is known as entry-controlled loop. | It is known as exit-controlled loop. |
| Semi colon is not used at the end of loop. | Semi colon is used at the end of loop. |
| Syntax:<br>While (termination condition)<br>{<br>Block of codes<br>} | Syntax:<br>Do<br>{<br>Block of statements<br>}<br>While (condition); |

Example of do while loop

```c
//to print numbers form 1-10
#include<stdio.h>
#include<conio.h>
 void main()
{
    int i=1;
    do
    {
        printf("  %d\n",i);
    i++;
    }
    while (i<=10);
}
```

```
1
2
3
4
5
6
7
8
9
10

--------
```

```c
//sum of n natural numbers
#include<stdio.h>
#include<conio.h>
 void main()
 {
    int n, a,b;
    a=1;
    b=0;
    printf("enter a number");
    scanf("%d",&n);
    do
    {
        b += a;
        a++;
     }
     while (a<=n);
     printf(" Sum=%d",b);
     getch();
 }
```

```
enter a number87
 Sum=3828
----------------------
```

## Arrays in "C"

It is a collection of data od same type. Thy are used to store group of data simultaneously. They are also called collection of homogenous type of data types. The main advantage of array is that it makes very easy to handle large number of similar values in a program. They also make a program very short and easy to develop.

Syntax:

Int array name[size of array];

Types of arrays

1. One-dimensional array
   If the values of array are assigned in one row is called One-dimensional array.

```c
/*Read 5 integer and store it in array
and display it*/
#include <stdio.h>
#include <conio.h>
int main()
{
    int i,a[10];
    printf("Enter 5 numbers\n");
    for (i=0; i<5; i++)
    {
        scanf("%d",&a[i]);
    }
    printf("The no are:\n");
    for(i=0; i<5; i++)
    {
    printf("%d\n",a[i]);
    }
    return 0;
}
```

```
Enter 5 numbers
1
2
3
4
5
The no are:
1
2
3
4
5
```

2. Two- dimensional array

   Two dimensional arrays are capable of storing data in rows and columns.

```c
#include <stdio.h>
#include <conio.h>
int main() {
  int r, c, a[100][100], b[100][100], sum[100][100], i, j;
  printf("Enter the number of rows (between 1 and 100): ");
  scanf("%d", &r);
  printf("Enter the number of columns (between 1 and 100): ");
  scanf("%d", &c);

  printf("\nEnter elements of 1st matrix:\n");
  for (i = 0; i < r; ++i)
    for (j = 0; j < c; ++j) {
      printf("Enter element a%d%d: ", i + 1, j + 1);
      scanf("%d", &a[i][j]);
    }

  printf("Enter elements of 2nd matrix:\n");
  for (i = 0; i < r; ++i)
    for (j = 0; j < c; ++j) {
      printf("Enter element b%d%d: ", i + 1, j + 1);
      scanf("%d", &b[i][j]);
    }

  for (i = 0; i < r; ++i)
    for (j = 0; j < c; ++j) {
      sum[i][j] = a[i][j] + b[i][j];
    }
  printf("\nSum of two matrices: \n");
  for (i = 0; i < r; ++i)
    for (j = 0; j < c; ++j) {
      printf("%d   ", sum[i][j]);
      if (j == c - 1) {
        printf("\n\n");
      }
    }

  return 0;
}
```

```
Enter the number of rows (between 1 and 100): 2
Enter the number of columns (between 1 and 100): 3

Enter elements of 1st matrix:
Enter element a11: 1
Enter element a12: 2
Enter element a13: 3
Enter element a21: 4
Enter element a22: 5
Enter element a23: 6
Enter elements of 2nd matrix:
Enter element b11: 6
Enter element b12: 5
Enter element b13: 4
Enter element b21: 3
Enter element b22: 2
Enter element b23: 1

Sum of two matrices:
7   7   7

7   7   7
```