CHAPTER 4

CONDITIONAL CONTROL STRUCTURE

SHORT AND LONG QUESTIONS

Q1. Define a control statement.

Ans: Control Statement:

A control statement is an instruction which determines the sequence of execution of other statements. In other words, it controls the flow of execution of program statements.

Q2. Define a conditional statement.

Ans: Conditional Statement:

A conditional statement is an instruction in a programming language that contains a condition. When a conditional statement is executed, first the condition is evaluated and then based on the result (true or false), a particular statement or a set of statements is executed. Conditional statements of C language are **if, if-else-else-if** and **switch** statements.

Q3. What is the purpose and structure of if statement? Explain with the help of examples.

Ans: Structure of If Statement:

The If statement has the following general form / Syntax.

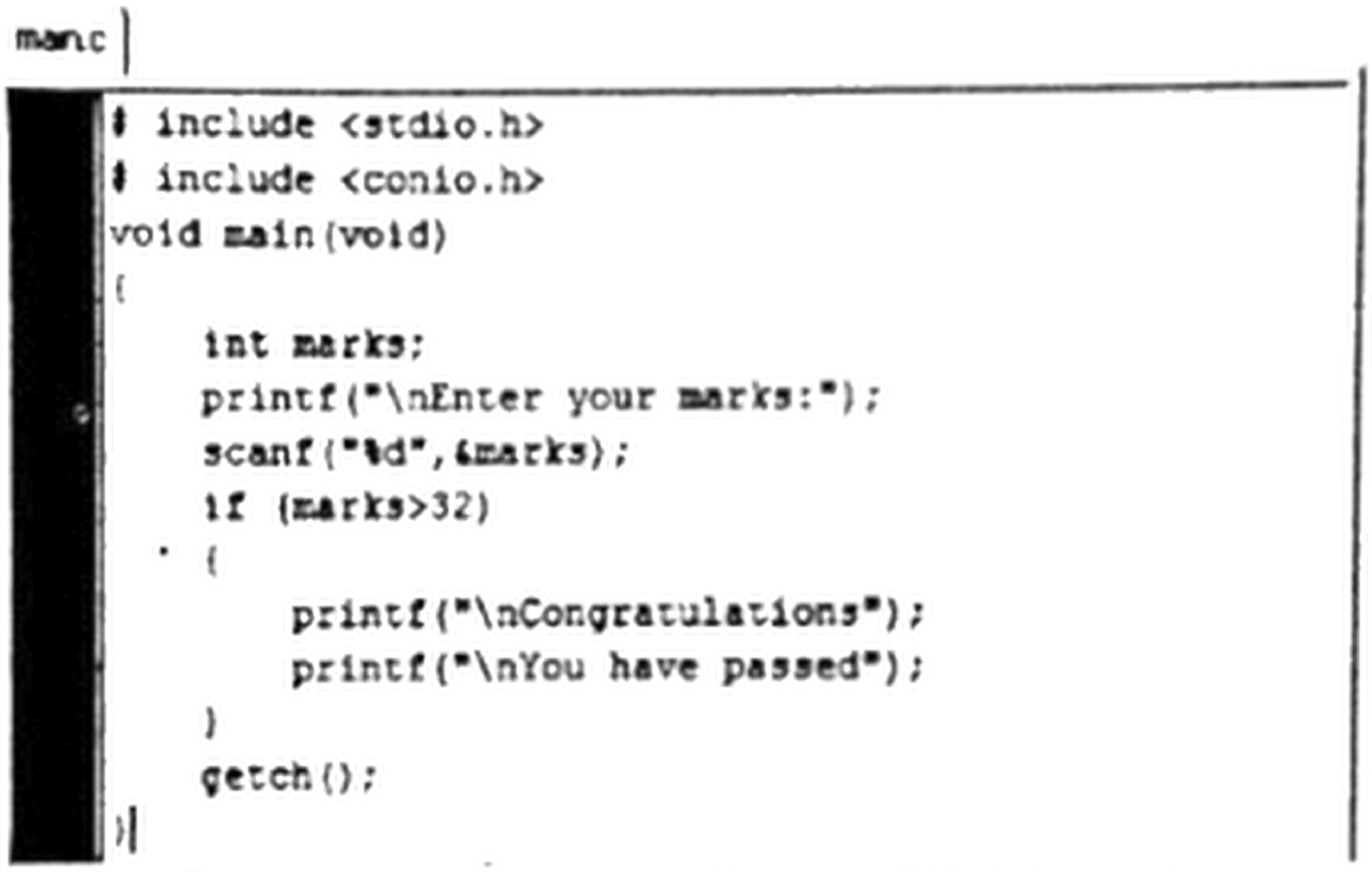
```
If (condition)
{
   Block of statements
}
```

When this statement is executed, the condition is evaluated. If the condition is true then the block of statements within the braces will be executed. If the condition is false then the block of statements within the braces will be skipped and the control will be transferred to the next statement if any exists.

If there is only one statement to be executed if the condition is true then braces are not required.

Use of If Statement (Examples):

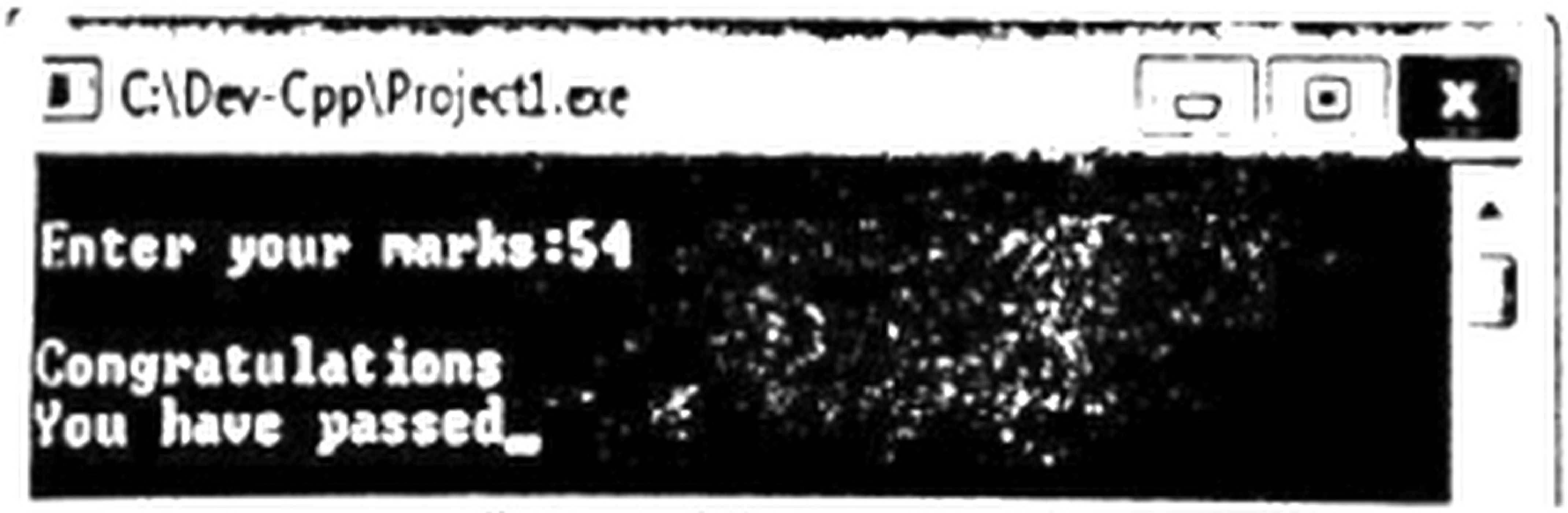
Program1: The program in Fig, demonstrates the use of If statement.



```
# include <stdio.h>
# include <conio.h>
void main(void)
{
     int marks;
     printf("\nEnter your marks:");
     scanf("%d",&marks);
     if (marks>32)
     {
          printf("\nCongratulations");
          printf("\nYou have passed");
     }
     getch();
}
```

Program to demonstrate the use of if statement

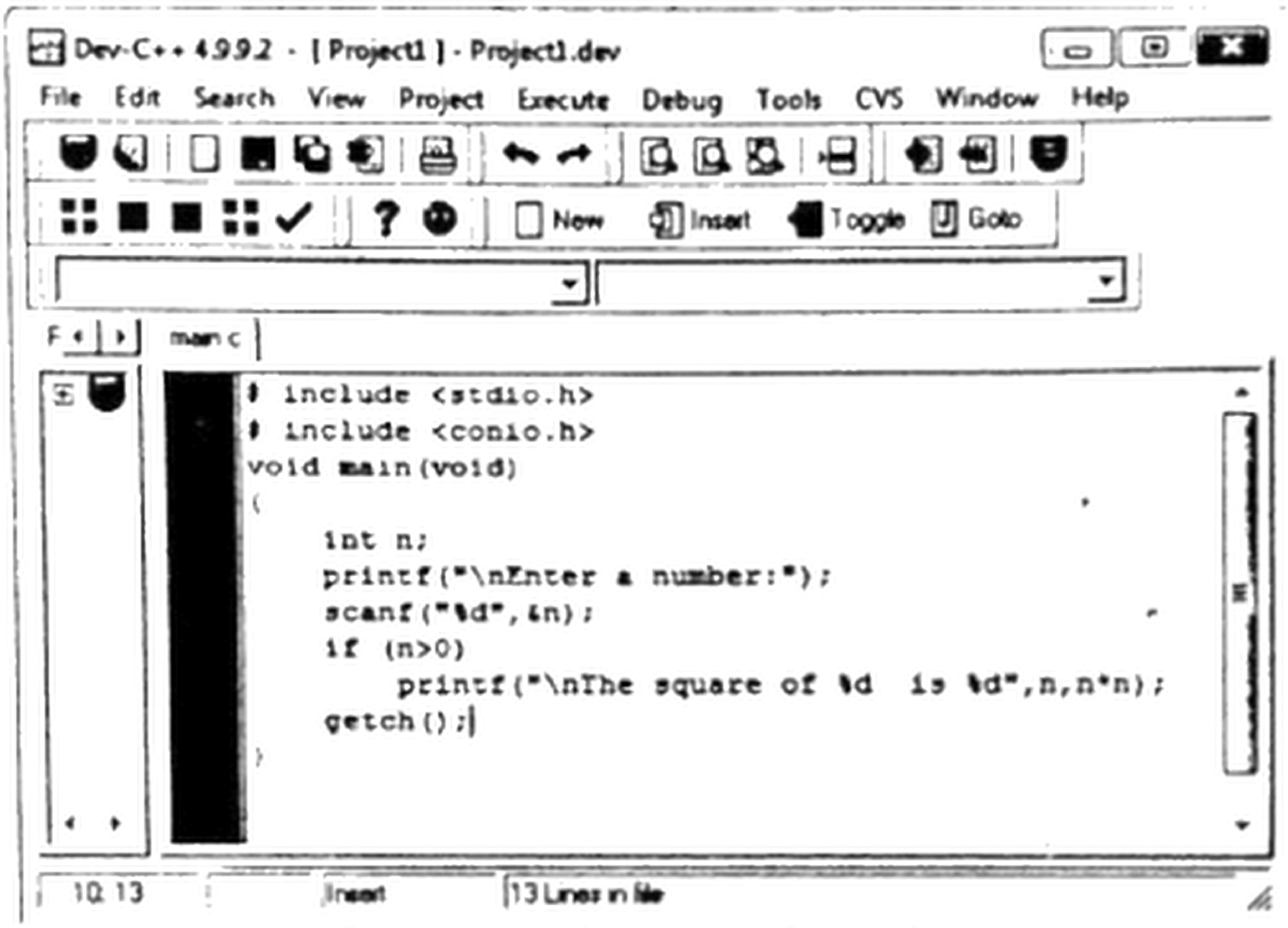
- When if statement of this program is executed the condition inside the brackets is evaluated.
- If the condition is true that is, the marks entered are greater than 32, then the two statements following the keyword if are executed.
- If the condition is false, that is the marks entered are less than 33, then the following two statements will be skipped and the program will terminate and there will be no output
- The output of the program is shown in Fig. if the marks entered are more than 32.



```
Enter your marks:54

Congratulations
You have passed_
```

Output of the program 1

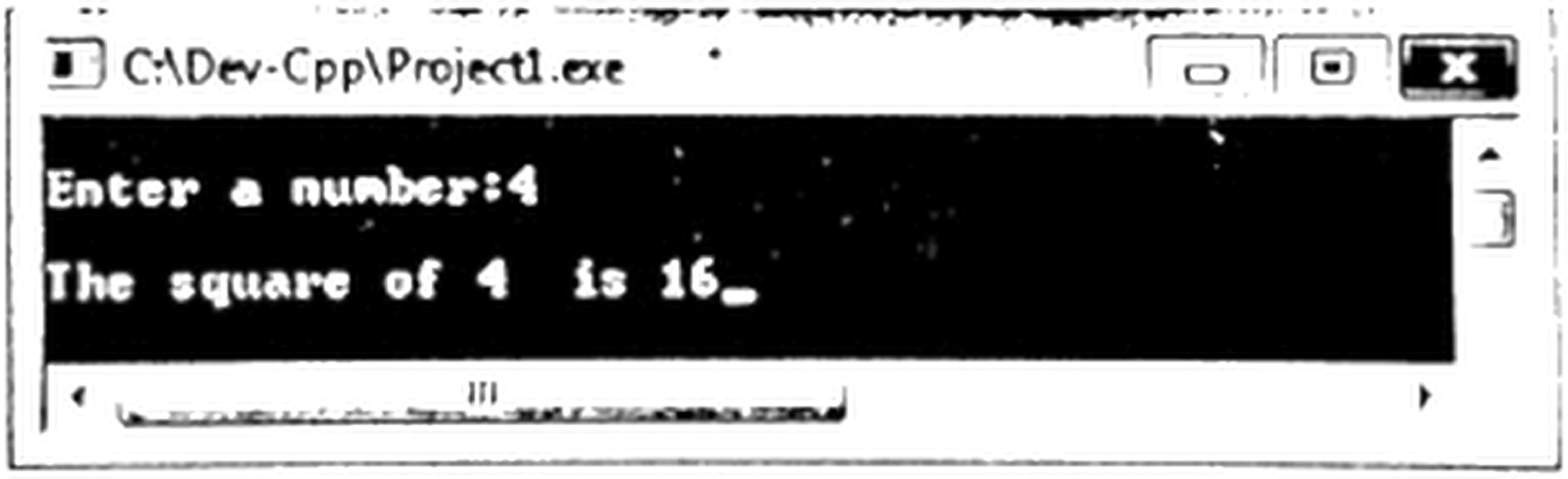
Program 2: The program in Fig, prints square of a number.


Dev-C++ 4.9.9.2 - [Project1] - Project1.dev
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
main.c
10: 13 Insert 13 Lines in file

```
# include <stdio.h>
# include <conio.h>
void main(void)
{
    int n;
    printf("\nEnter a number:");
    scanf("%d",&n);
    if (n>0)
        printf("\nThe square of %d  is %d",n,n*n);
    getch();
}
```

Program to print square of a number

- When the above program is executed, it will prompt the user to enter a number.
- If the user enters a number greater than zero, it will print the square of the number as shown in Fig.



```
Enter a number:4
The square of 4  is 16_
```

Output of the program 2

Q4. What is the purpose and structure of if-else statement? Explain with the help of examples.

Ans: Structure of If-Else Statement:

The **if-else** statement is used in situation where some code is to be executed if a condition is true and some other code is to be executed if the condition is false.

The **if-else** statement has the following **general form/ Syntax.**

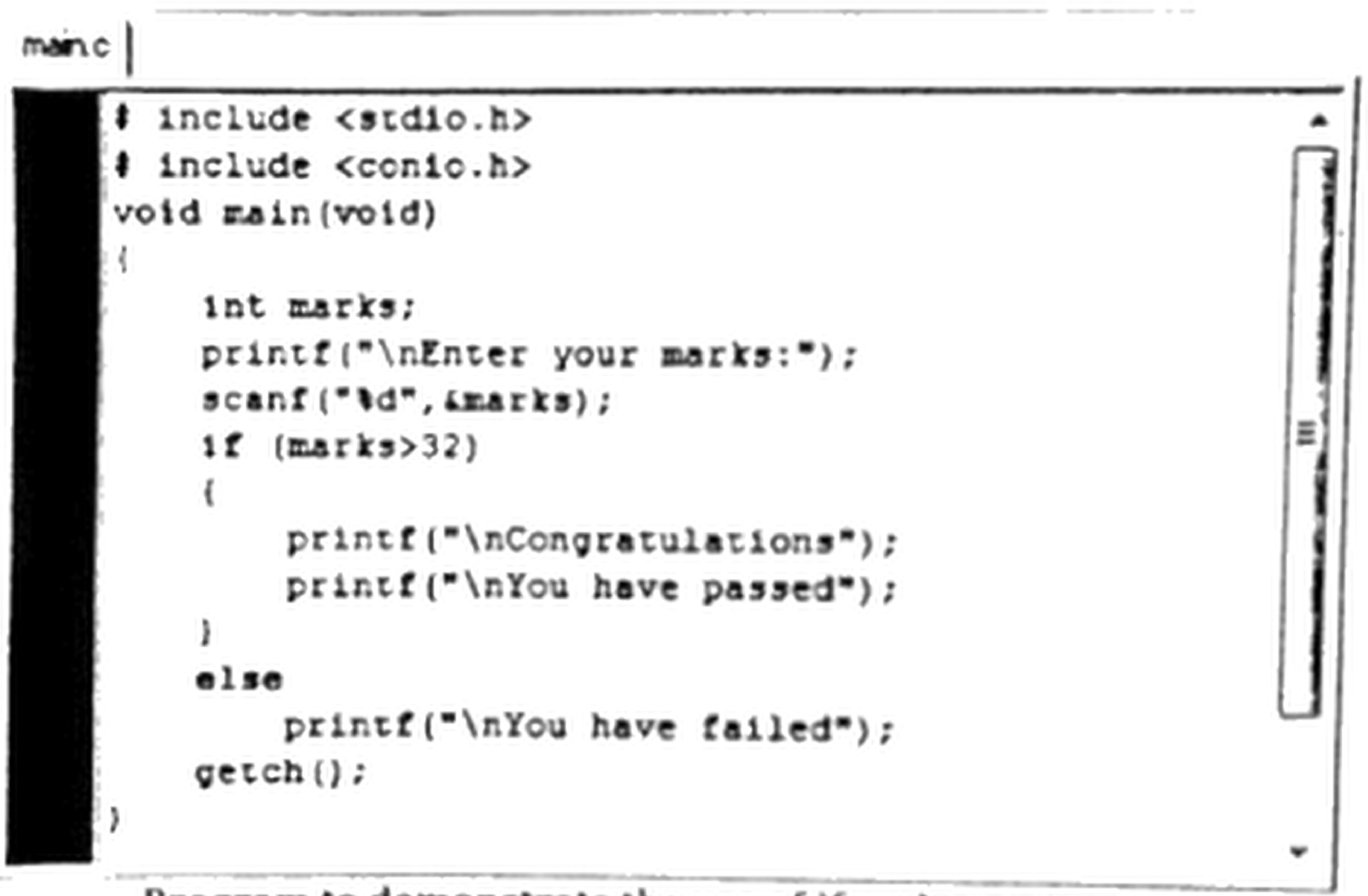
```
If (condition)
{
    Block of statements
}
Else
{
    Block of statements
}
```

- When if-else statement is executed, the condition is evaluated.
- If the condition is true then the block of statements following if will be executed and the block of statements following else will be skipped.

- If the condition is false then the block of statements following if will be skipped and the block of statements following else will be executed.
- If a single statement is to be executed after if or else then braces are not required.

Use of If-Else Statement (Examples):

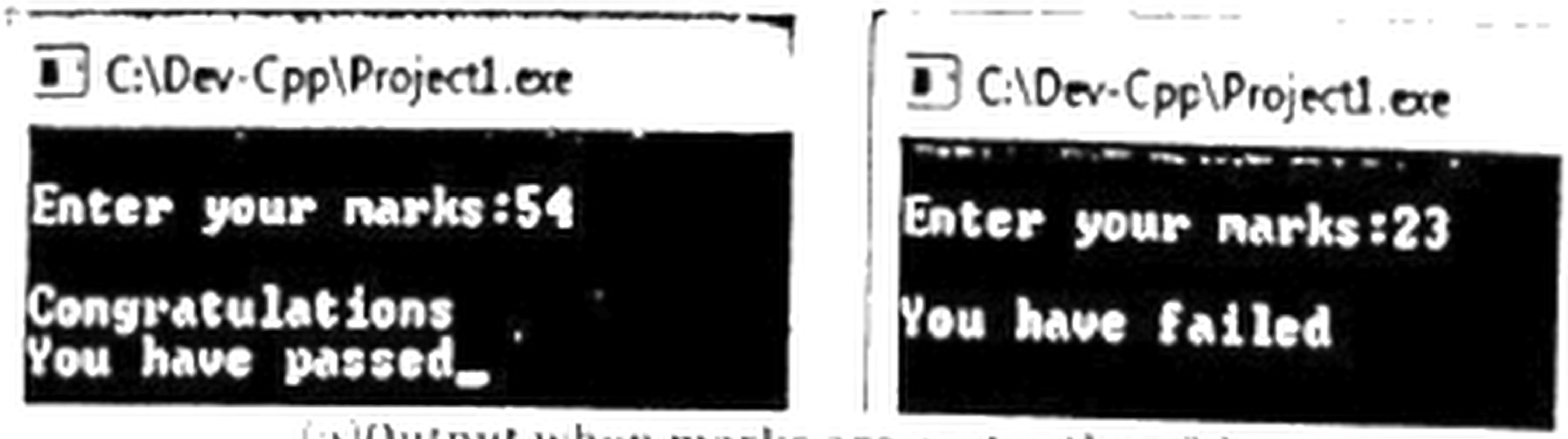
Program 1: The program in Fig reads marks and prints the message whether the student passed or failed.



```
# include <stdio.h>
# include <conio.h>
void main(void)
{
    int marks;
    printf("\nEnter your marks:");
    scanf("%d",&marks);
    if (marks>32)
    {
        printf("\nCongratulations");
        printf("\nYou have passed");
    }
    else
        printf("\nYou have failed");
    getch();
}
```

Program to demonstrate the use of if – else statement

- In this program, if the marks entered are above 32 then the statements following if are executed and the message shown in Fig (a) will be printed.
- If the marks entered are below 33 then the statement following else are executed and the message shown in Fig (b) will be printed



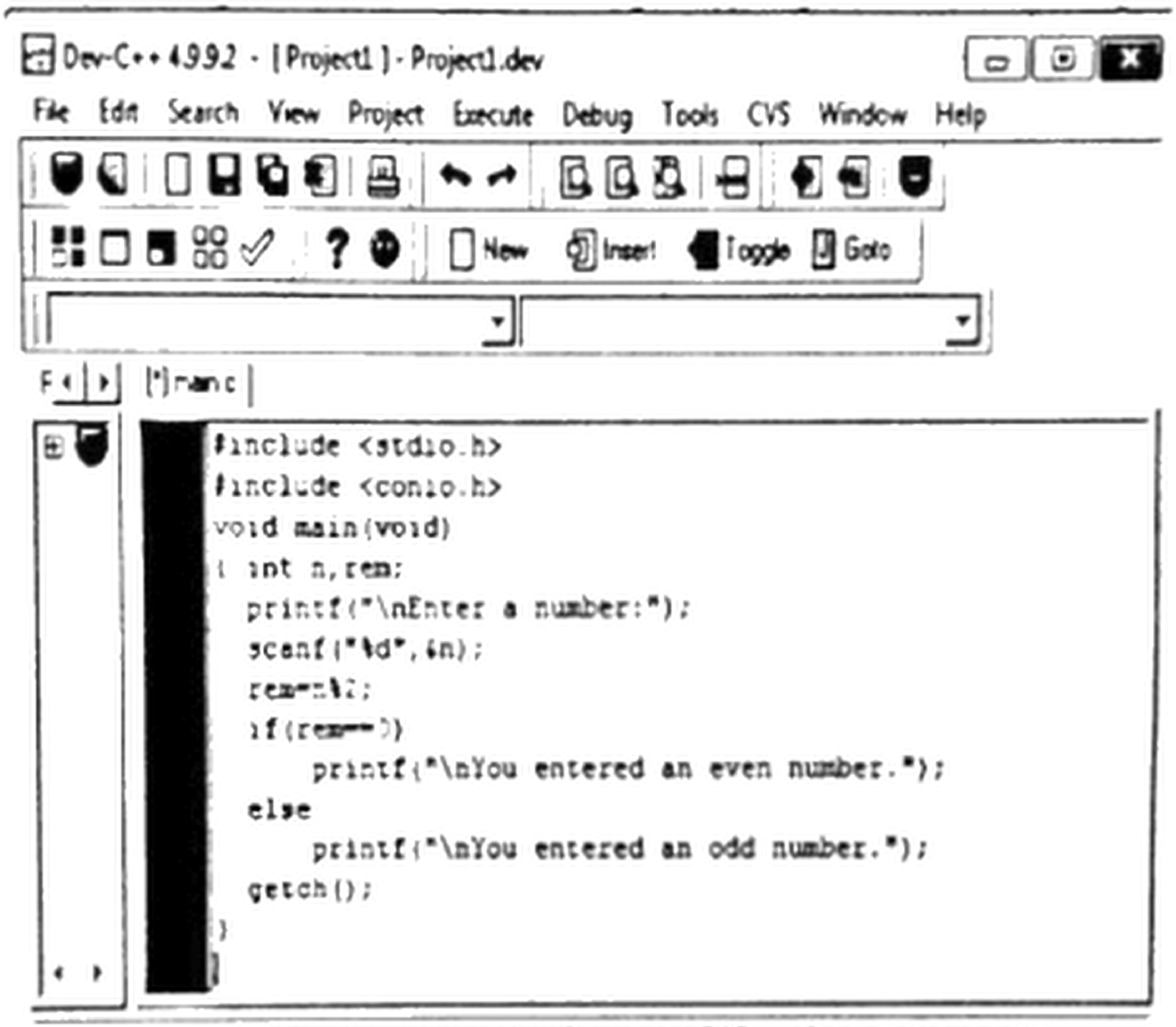
```
Enter your marks:54

Congratulations
You have passed_
```

```
Enter your marks:23

You have failed
```

(a)Output when marks are grater than 32
(b)Output when marks are less than33

- Also note that there is a single statement to be executed if the condition is false. Therefore, braces are not required after else.

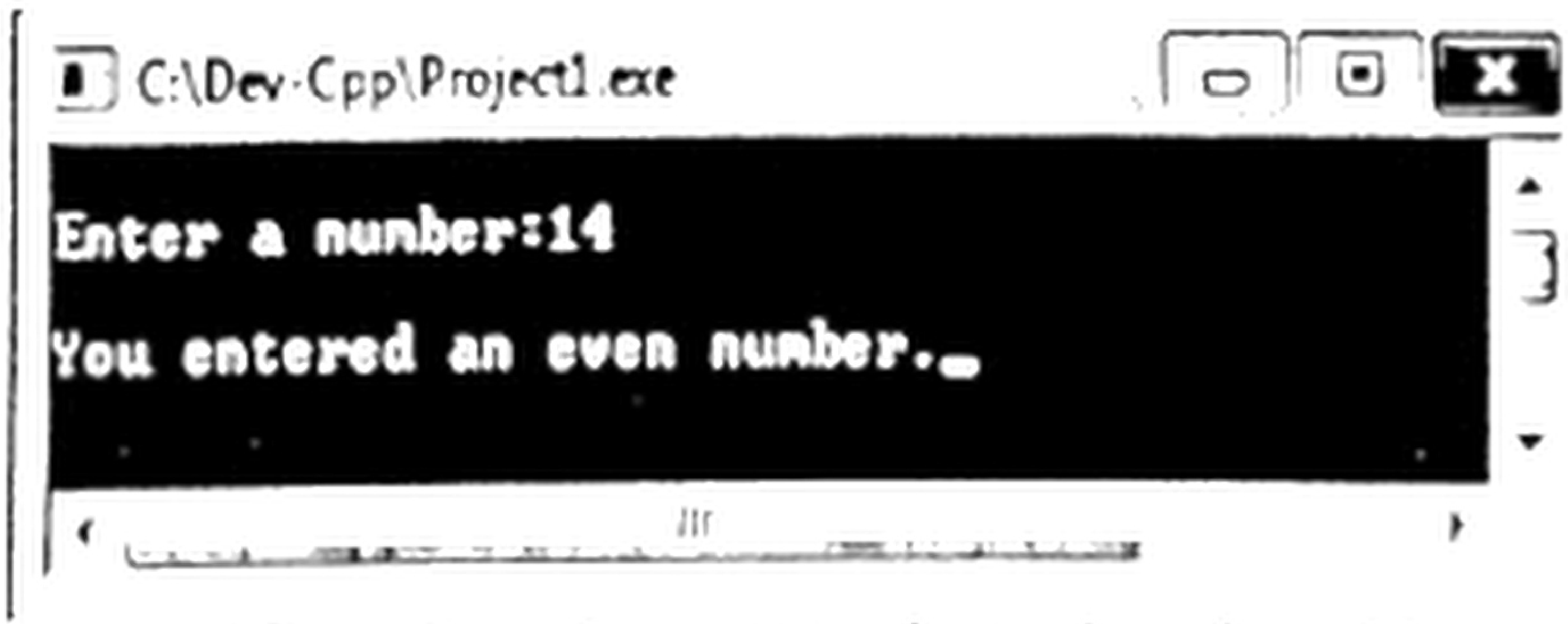
Program 2: The program in Fig, reads a number from the keyboard and prints the message whether it is an even or an odd number.


Dev-C++ 4.9.9.2 - [Project1] - Project1.dev
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
[*] main.c

```
#include <stdio.h>
#include <conio.h>
void main(void)
{ int n,rem;
  printf("\nEnter a number:");
  scanf("%d",&n);
  rem=n%2;
  if(rem==0)
      printf("\nYou entered an even number.");
  else
      printf("\nYou entered an odd number.");
  getch();
}
```

Program to demonstrate the use of if – else statement

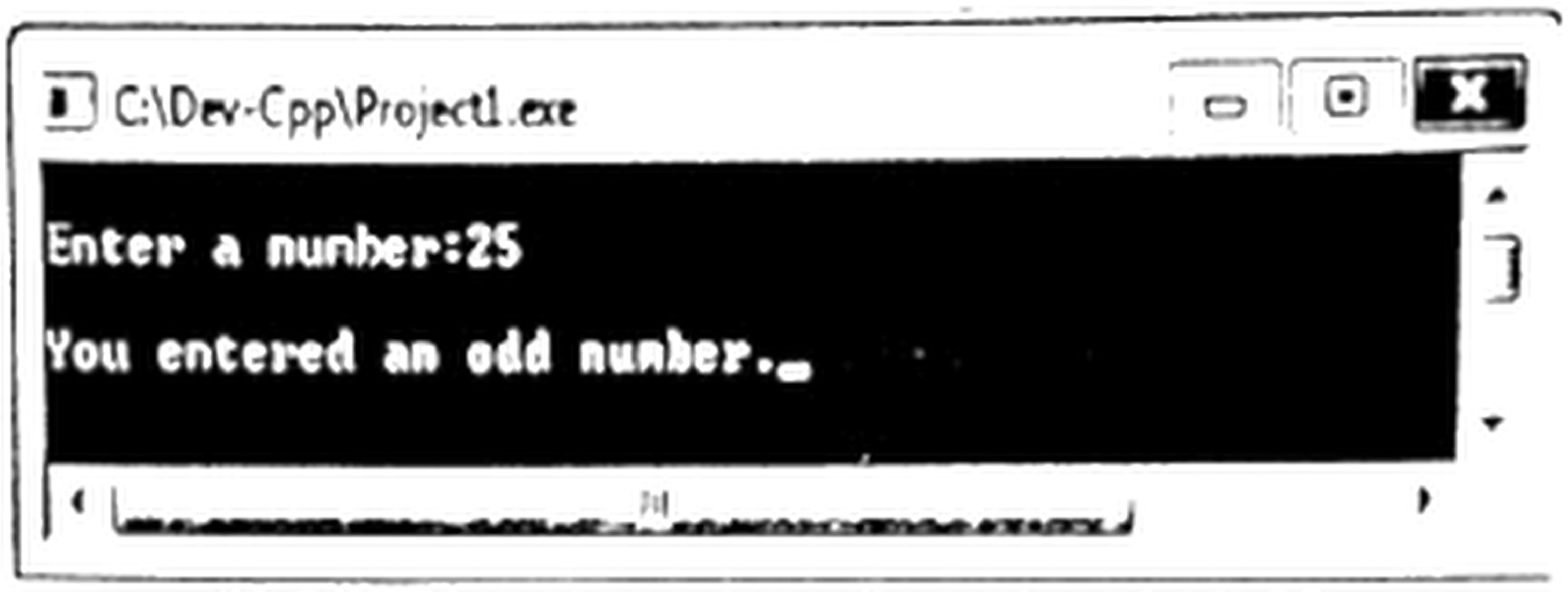
- When the program is executed it prompts the user to enter the value of n.
- The remainder operator % is used to divide the number by 2 and store the remainder in variable rem.
- The value of rem is checked. If it is equal to 0 then the number n, is an even number as shown is Fig (a)



```
Enter a number:14
You entered an even number._
```

(a) Execution of program when value of n is 14

- If the value of rem is not equal to 0. In other words, it is equal to 1 then the number n is an odd number as shown in Fig (b).



```
Enter a number:25
You entered an odd number._
```

(b) Execution of program when value of n is 25

Q5. What is the purpose and structure of if-else-if statement? Explain with the help of examples.

Ans: Structure of If-Else-If Statement:

The else-if statement is a type of conditional statement that combines more than two conditions. It allows the programmer to make a decision based on several conditions.

The **else-if** statement has the following **general form / Syntax.**

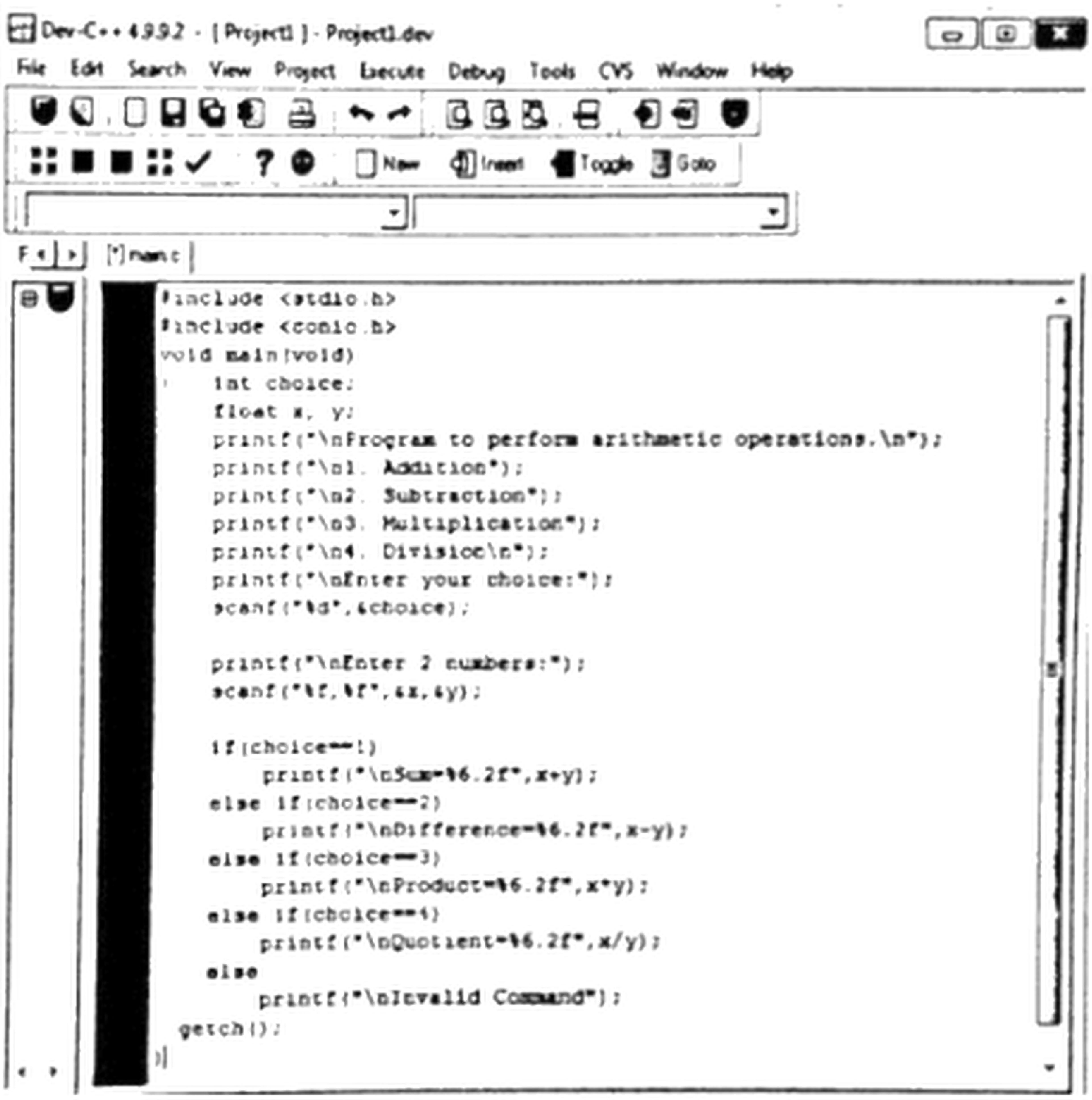
```
If (condition-1)
{
    Block of statements
}
Else if (condition -2)
{
     Block of statements
}
Else if (condition-3)
{
     Block of statements
}
          .
          .
          .
Else
{
    Block of statements to be executed
    When none of the conditions is true
```

}

- When this statement is executed, condition-1 is evaluated, if it is true then the block of statements following if is executed and if it is false, the next condition is evaluated
- If any condition is true then following block of statements is executed
- If none of the conditions is true then the block of statements following else is executed automatically.
- If a single statement is to be executed after if, else if or else, instead of a set of statements then the braces are not required

Use of IF-Else-If Statement (Examples):

Program 1: The program in Fig, performs an arithmetic operation based on the choice of user.


Dev-C++ 4.9.9.2 - [Project1] - Project1.dev
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
[*] main.c

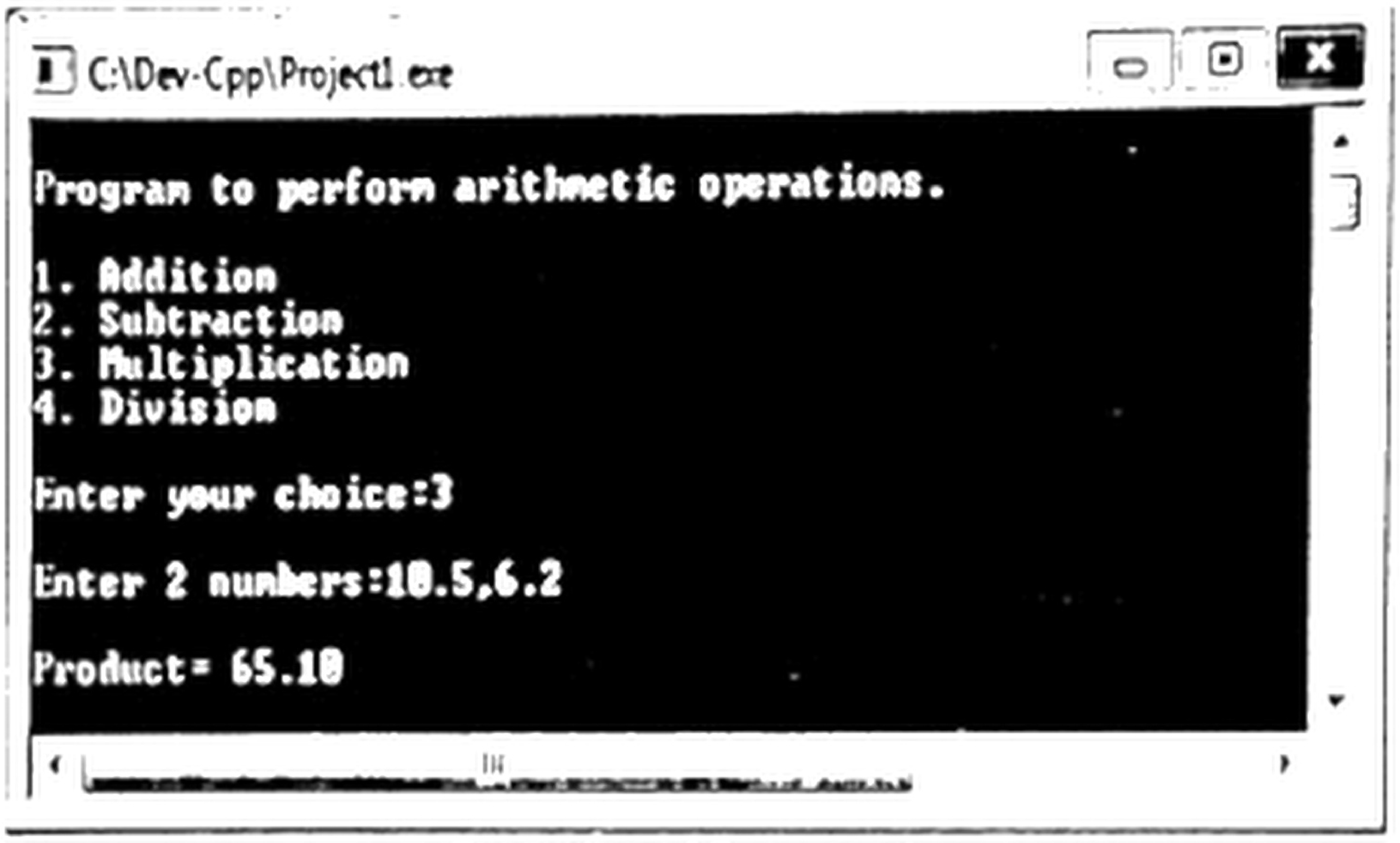
```
#include <stdio.h>
#include <conio.h>
void main(void)
{   int choice;
    float x, y;
    printf("\nProgram to perform arithmetic operations.\n");
    printf("\n1. Addition");
    printf("\n2. Subtraction");
    printf("\n3. Multiplication");
    printf("\n4. Division\n");
    printf("\nEnter your choice:");
    scanf("%d",&choice);

    printf("\nEnter 2 numbers:");
    scanf("%f,%f",&x,&y);

    if(choice==1)
        printf("\nSum=%6.2f",x+y);
    else if(choice==2)
        printf("\nDifference=%6.2f",x-y);
    else if(choice==3)
        printf("\nProduct=%6.2f",x*y);
    else if(choice==4)
        printf("\nQuotient=%6.2f",x/y);
    else
        printf("\nInvalid Command");
  getch();
}
```

- When this program is executed, it will display four choices and prompt the user to enter his choice. User's choice will be stored in the variable choice.
- After this, it will again prompt the user the enter two numbers which will be stored in variables x and y.
- Now, the if-else-if statement will be executed. First condition will be checked first, if it is true, that is, the value stored in the variable choice is 1 then sum of x and y will be evaluated and printed.
- If the first condition is false then the second condition will be checked and if it is true, then the difference (x-y) will be evaluated and printed and so on.

- If all the conditions are false then the statement following else will the executed and the message "Invalid Command" will be printed.

In this program, the format specifier %6.2f is used for printing floating-point value. Here, 6 is the total field width for printing the number and reserves 6 spaced for printing it. The digit 2 means 2 digits are to be printed after the decimal point. If the floating point number to be printed requires less than 6 spaces then it will be right justified and extra spaces will appear at the left side of the number.

The execution of this program is shown in Fig, when user choice is 3.



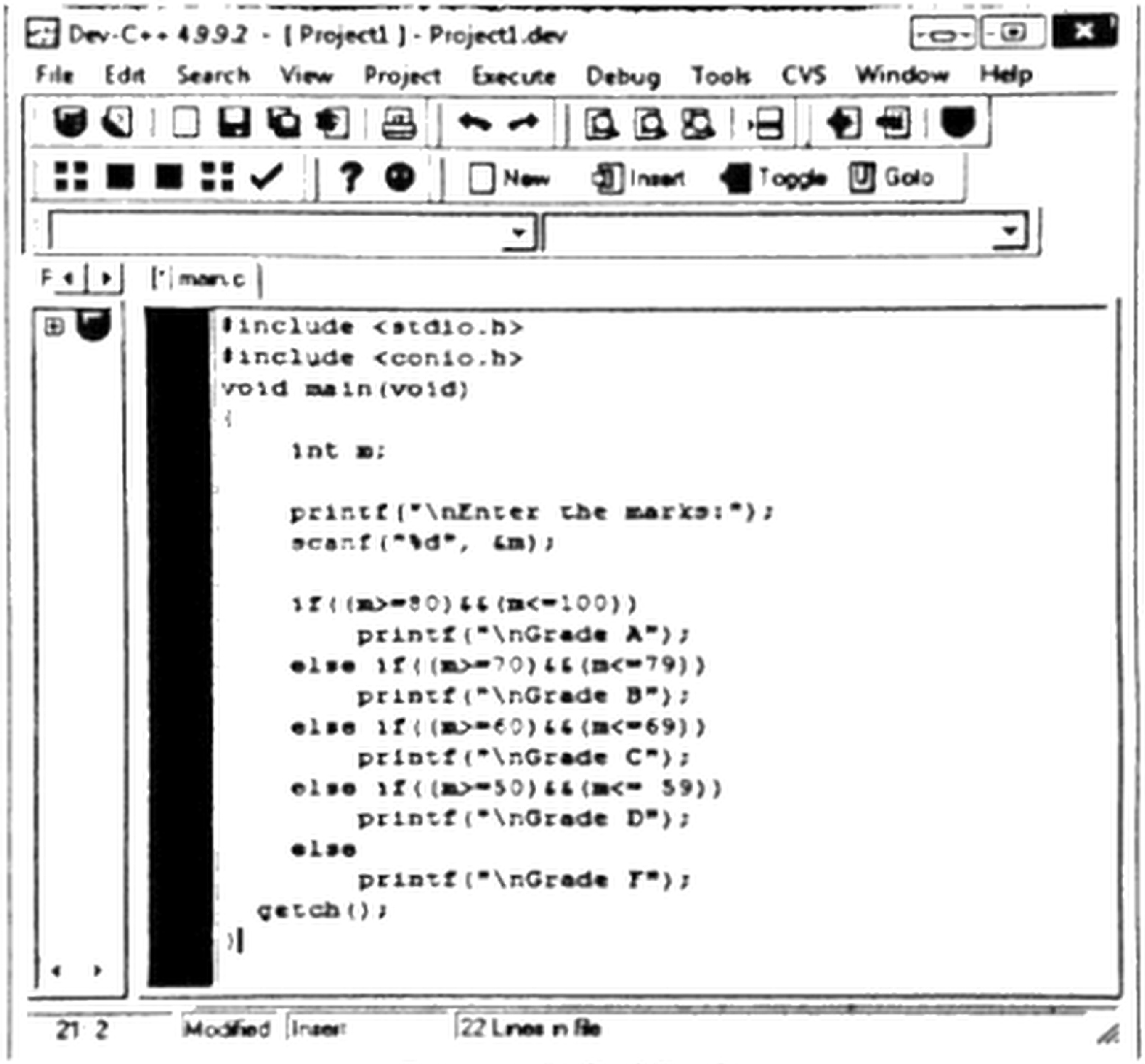
```
Program to perform arithmetic operations.

1. Addition
2. Subtraction
3. Multiplication
4. Division

Enter your choice:3

Enter 2 numbers:10.5,6.2

Product= 65.10
```

Execution of Program 1

Program 2: The following program reads marks of a student and prints his letter grade according to the following scheme.

Marks	**Grade**
80~100	A
70~79	B
60~69	C
50~59	D

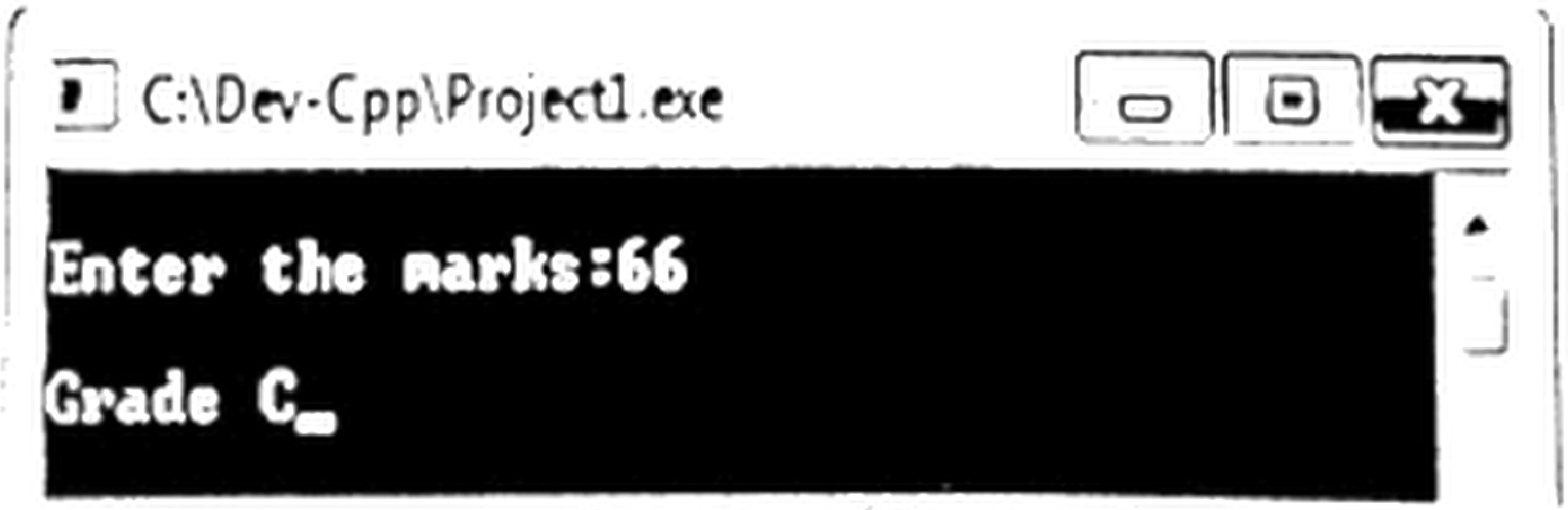
Below 50	F


Dev-C++ 4.9.9.2 - [Project1] - Project1.dev
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
[*] main.c
21: 2
Modified
Insert
22 Lines in file

```
#include <stdio.h>
#include <conio.h>
void main(void)
{
     int m;

     printf("\nEnter the marks:");
     scanf("%d", &m);

     if((m>=80)&&(m<=100))
          printf("\nGrade A");
     else if((m>=70)&&(m<=79))
          printf("\nGrade B");
     else if((m>=60)&&(m<=69))
          printf("\nGrade C");
     else if((m>=50)&&(m<= 59))
          printf("\nGrade D");
     else
          printf("\nGrade F");
   getch();
}
```

Program to find Grade

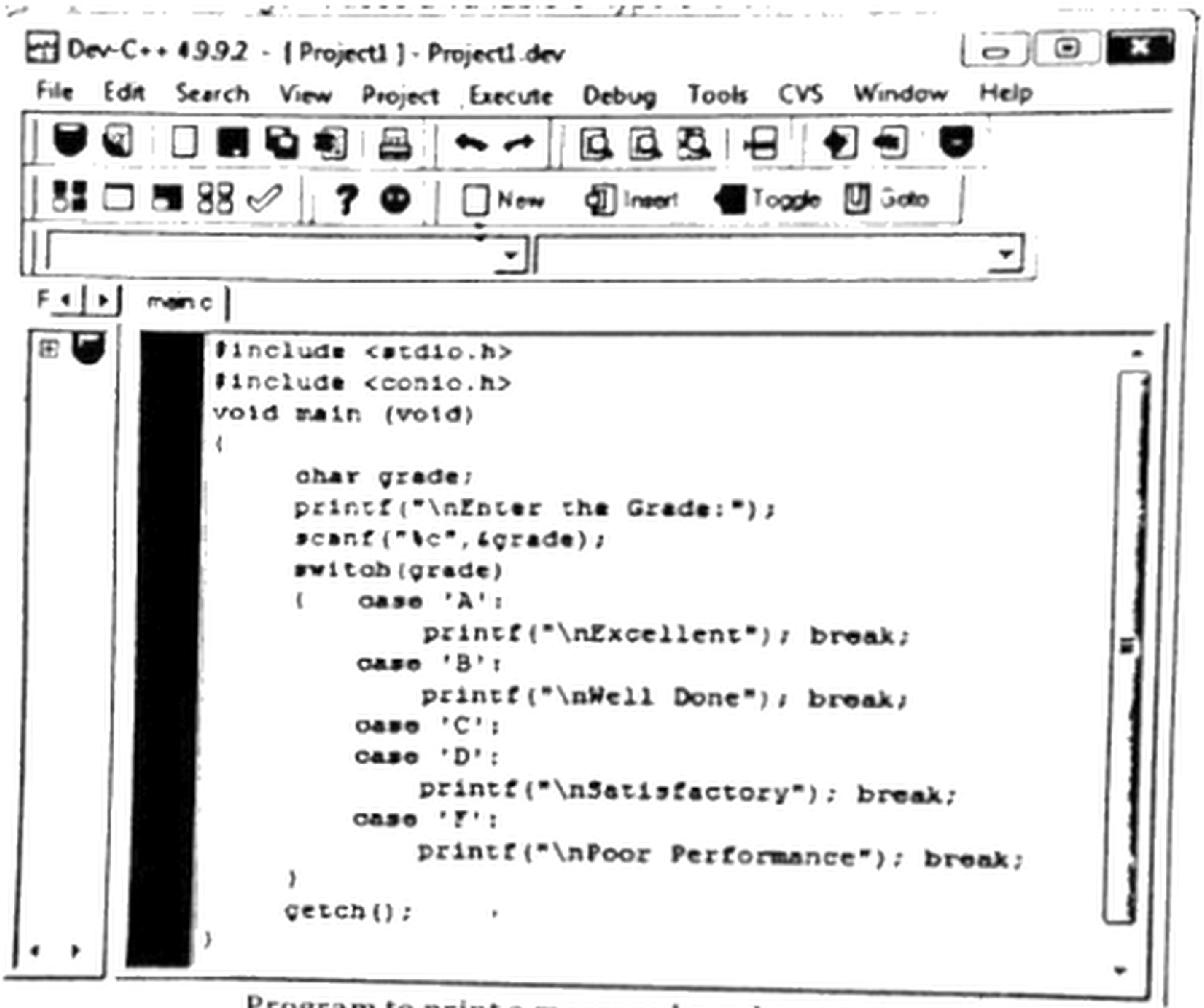
- When the above program is executed, it will promptly the user to enter marks . It will check in which range the marks fall and then accordingly print the grade.
- If none of the conditions is true then the statement following else will be executed and it will print the message "Grade F". The execution of the program is shown in Fig. if the marks entered by the user are 66. Since 66 falls in the range 60 to 69, so the grade displayed is C.



```
Enter the marks:66
Grade C_
```

Execution of Program 2

Program 3: The program in Fig, prompts the user to enter a grade and prints appropriate message. It uses a variable of type character as switch variable.

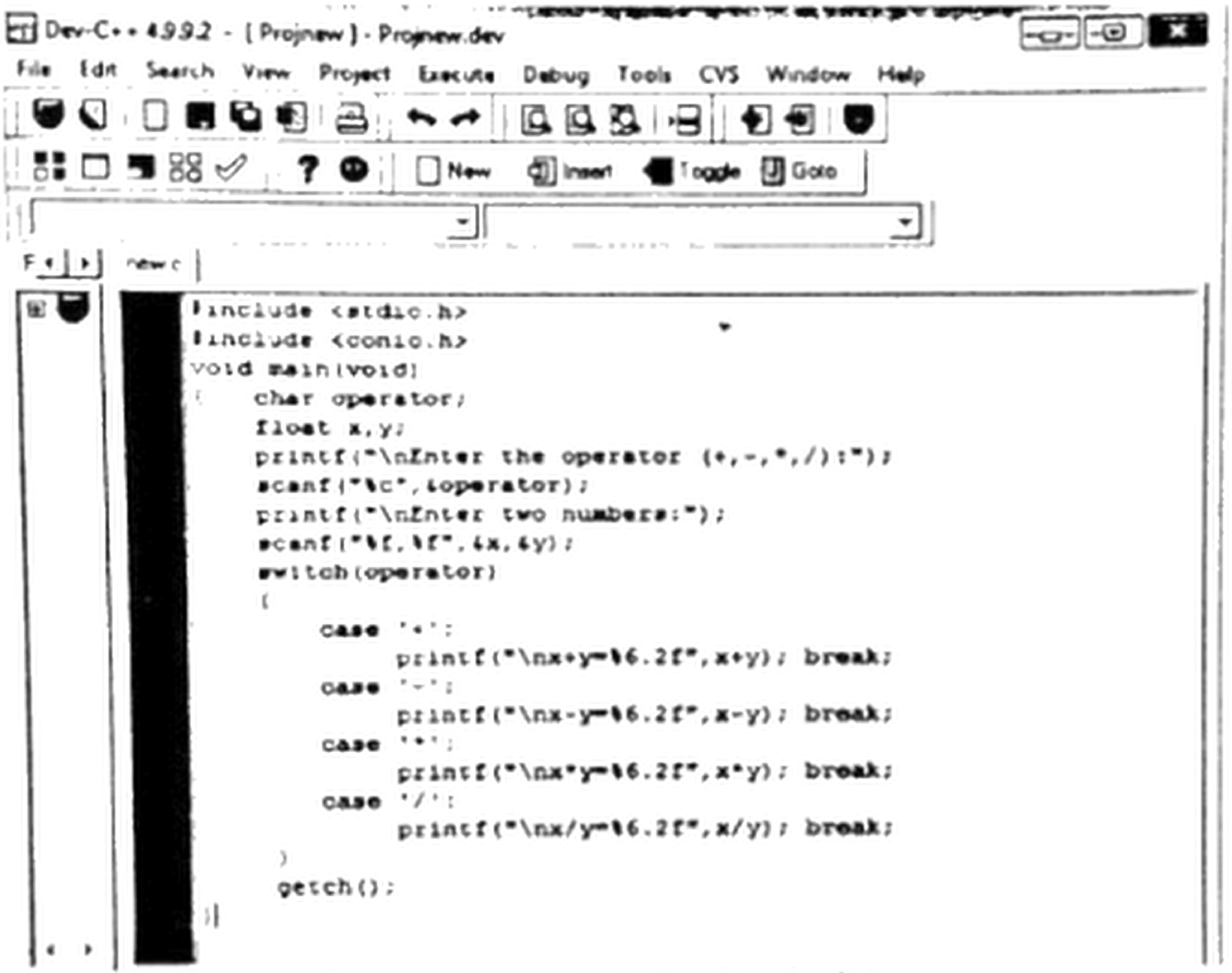


```
#include <stdio.h>
#include <conio.h>
void main (void)
{
      char grade;
      printf("\nEnter the Grade:");
      scanf("%c",&grade);
      switch(grade)
      {    case 'A':
                printf("\nExcellent"); break;
           case 'B':
                printf("\nWell Done"); break;
           case 'C':
           case 'D':
                printf("\nSatisfactory"); break;
           case 'F':
                printf("\nPoor Performance"); break;
      }
      getch();
}
```

Program to print a message based on grade

- This program prompts the user to enter a grade, that is 'A', 'B', 'C', 'D', OR 'F' and stores it in the variable grade. It will print the message 'Excellent' for grade 'A' and "Well Done" for grade 'B.
- If the user enters grade 'C' or 'D', the same message "Satisfactory" will be printed. In the absence of statements after a case, the control falls right through one case to the case below and this makes it easy for several values of switch variable to execute the same code.
- Finally, if the user enter grade 'F', the program will print the message "Poor Performance".
- In this program there is no default keyword. Therefore, the whole switch statement simply terminates when there is no match.

Program 4: The program in Fig. creates a simple calculator to perform addition, subtraction, multiplication or division.


Dev-C++ 4.9.9.2 - [Projnew] - Projnew.dev
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
new c

```
#include <stdio.h>
#include <conio.h>
void main(void)
{    char operator;
     float x,y;
     printf("\nEnter the operator (+,-,*,/):");
     scanf("%c",&operator);
     printf("\nEnter two numbers:");
     scanf("%f,%f",&x,&y);
     switch(operator)
     {
         case '+':
              printf("\nx+y=%6.2f",x+y); break;
         case '-':
              printf("\nx-y=%6.2f",x-y); break;
         case '*':
              printf("\nx*y=%6.2f",x*y); break;
         case '/':
              printf("\nx/y=%6.2f",x/y); break;
      }
      getch();
}
```

Program to create a simple calculator

- When the above program is executed, it will ask the user to enter the type of operation to be performed on two numbers. Then it will ask for two numbers to be entered.
- Based on the operator, it will perform addition, subtraction, multiplication or division and print the result.

Q6. What is the advantage and limitation of switch statement?

Ans: Advantage and Limitation of Switch Statement:

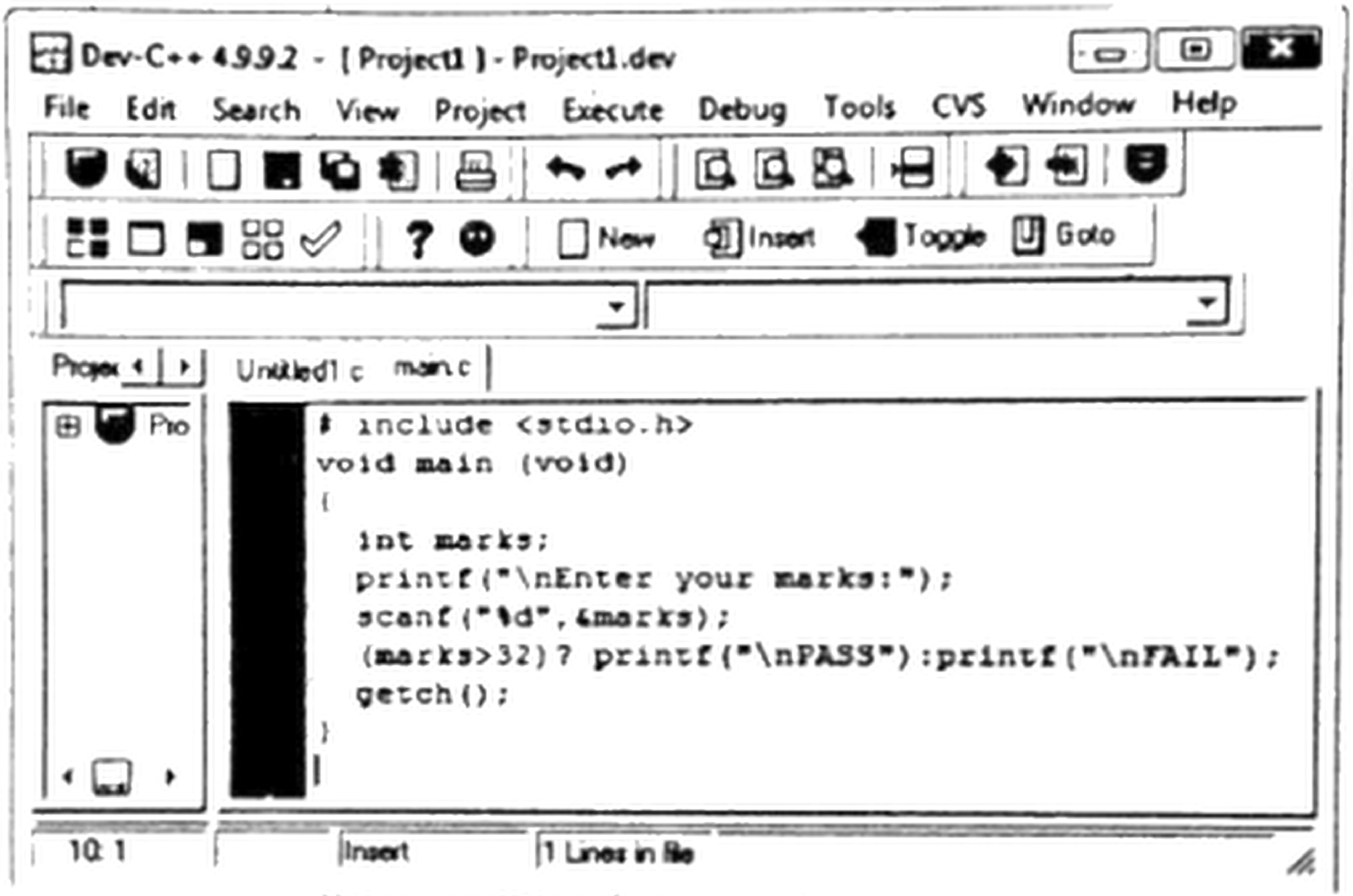
- The switch statement allows a variable to be compared against a list of constant values. When there is a match to a case, the statement following that case will execute until a break statement is reached. This makes the logic of program and easy to understand.
- The switch statement has a limitation. It is not allowed to use relational operators in the expression of switch statement.

For example, to check for passing marks, the condition (marks > 32) cannot be used in witch statement. It is also not possible to check for a range such as ((marks>=70)&&((marks<=80)) in a switch statement, for this, the programmer has to use if, if-else or else-if statement.

Q7. Explain the use of conditional operator in program.

Ans: Using Conditional Operator in Program:

Program: The program in Fig, reads marks and prints the message "PASS" or "FAIL" using conditional operator instead of **if-else** statement. Conditional operator was explained in the previous unit.

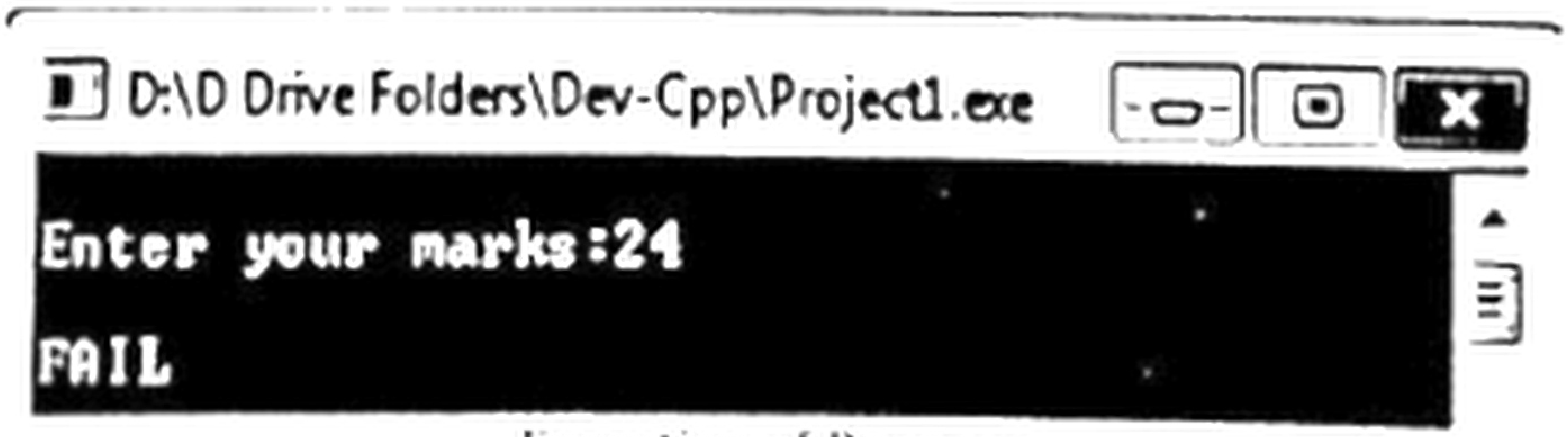

Dev-C++ 4.9.9.2 - [Project1] - Project1.dev
File Edit Search View Project Execute Debug Tools CVS Window Help
New Insert Toggle Goto
Project
Untitled1 c
main.c
Pro
10: 1
Insert
1 Lines in file

```
# include <stdio.h>
void main (void)
{
  int marks;
  printf("\nEnter your marks:");
  scanf("%d",&marks);
  (marks>32)? printf("\nPASS"):printf("\nFAIL");
  getch();
}
```

Using conditional operator in a program

- When the program is executed, it will ask the user to enter the marks and store it in the variable marks
- The condition (marks>32) will be evaluated.
- If the condition is true, that is, marks are greater than 32 then the first print statement will be executed and the message "PASS" will be printed.
- If the condition is false then the second print statement will be executed and the message "FAIL" will be printed.

Execution of the program is shown in Fig.



```
Enter your marks:24
FAIL
```

Execution of Program

KEY POINTS

- In a programming language, a control statement is an instruction which determines the sequence of execution of other statements in a program.
- A conditional statement is an instruction in a programming language that contains a condition based on which flow of execution of program statements is controlled
- The if statement is a control statement. When it is executed, the condition is evaluated. If it is true then the block of statement within the braces will be executed otherwise control will be transferred to the next statement.
- The **if-else** statement is a control statement. When it is executed, the condition is evaluated. If it is true then the block of statement following if will be executed otherwise the block of statements following else will be executed.
- The **if-else-if** statement is a control statement. When it is executed, the first condition is evaluated, if it is true then the block of statements following. If will be executed. If the condition is false then second condition after else if will be evaluated if any condition is true the block of statements following that else if will be executed. If none of the conditions is true then the block of statement following else will be executed automatically if it exists otherwise the control will be transferred to the next statement.
- The **switch** statement is used when multiple choices are given and one choice is to be selected. It is similar to else-if statement.
- A selections structure within another selection structure is known as nested selection structure