1. **Primary (Basic) Data Types**

   int – stores integers
   Syntax: int x;
   Example: int age = 20;

   float – stores decimal numbers (single precision)
   Syntax: float y;
   Example: float price = 10.5;

   double – stores decimal numbers (double precision)

   Syntax: double z;
   Example: double pi = 3.14159;

   char – stores single characters
   Syntax: char c;
   Example: char grade = 'A';

2. **Derived Data Types**

   Array – collection of similar data types
   Syntax: int a[5];
   Example: int marks[3] = {10,20,30};

   Pointer – stores memory address
   Syntax: int *p;
   Example: int *ptr = &age;

   Structure – group of different data types

   Syntax:

```
struct student {
   int id;
   char name[20];
};
```

   Union – stores different data types in the same memory location
   Syntax:union item {

```
   int x;
   float y;};
```

3. **User-Defined Data Types**

   typedef – defines a new name for an existing type
   Example:

   typedef int number;
   number a = 10;

   enum – stores integer constants
   Example:

   enum week {Mon, Tue, Wed};

Operators
Operators are symbols used to perform operations on variables and values.

Types of Operators in C

1. **Arithmetic Operators**
   + - * / %
   Example: c = a + b;
2. **Relational Operators**
   == != > < >= <=
   Example: a > b
3. **Logical Operators**
   && || !
   Example: a > b && b > c
4. **Assignment Operators**
   = += -= *= /= %=
5. **Increment/Decrement Operators**
   ++ --
   Example: a++;
6. **Bitwise Operators**
   & | ^ << >> ~
7. **Conditional (Ternary) Operator**
   ?:
   Example: result = (a > b) ? a : b;

8. **Special Operators**

    sizeof

    & (address-of)

    * (pointer dereference)

    . and -> (structure/ pointer member access)

C-programming

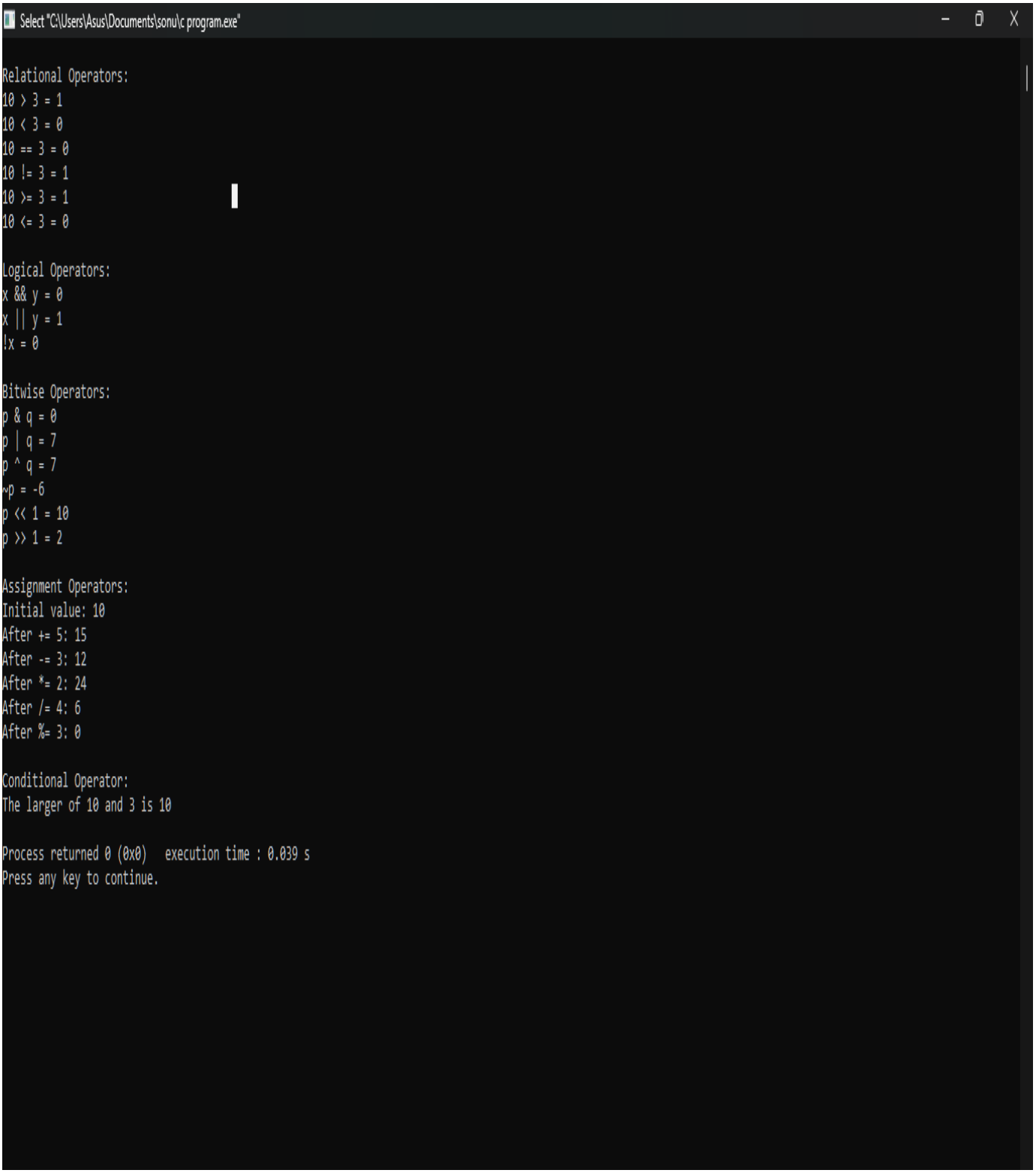
```
#include <stdio.h>
int main() {
/* Demonstrating Data Types */
int integer = 25; /* Integer data type */
float decimal = 12.34; /* Float data type */
double largeDecimal = 1234.5678; /* Double data type */
char character = 'A'; /* Character data type */
int boolean = 1; /* Boolean (using int as C has no explicit boolean in std C89) */
/* Printing data types */
printf("Data Types Demonstration:\n");
printf("Integer: %d\n", integer);
printf("Float: %.2f\n", decimal);
printf("Double: %.4lf\n", largeDecimal);
printf("Character: %c\n", character);
printf("Boolean (1 for true, 0 for false): %d\n\n",
boolean);
/* Demonstrating Arithmetic Operators */
int a = 10, b = 3;
printf("Arithmetic Operators:\n");
printf("Addition: %d + %d = %d\n", a, b, a + b);
```

```
printf("Subtraction: %d - %d = %d\n", a, b, a - b);
printf("Multiplication: %d * %d = %d\n", a, b, a * b);
printf("Division: %d / %d = %d\n", a, b, a / b);
printf("Modulus: %d %% %d = %d\n\n", a, b, a % b);
/* Demonstrating Relational Operators */
printf("Relational Operators:\n");
printf("%d > %d = %d\n", a, b, a > b);
printf("%d < %d = %d\n", a, b, a < b);
printf("%d == %d = %d\n", a, b, a == b);
printf("%d != %d = %d\n", a, b, a != b);
printf("%d >= %d = %d\n", a, b, a >= b);
printf("%d <= %d = %d\n\n", a, b, a <= b);
/* Demonstrating Logical Operators */
int x = 1, y = 0;
printf("Logical Operators:\n");
printf("x && y = %d\n", x && y);
printf("x || y = %d\n", x || y);
printf("!x = %d\n\n", !x);
/* Demonstrating Bitwise Operators */
int p = 5, q = 2; /* 5 = 0101, 2 = 0010 in binary */
printf("Bitwise Operators:\n");
printf("p & q = %d\n", p & q); /* AND */
printf("p | q = %d\n", p | q); /* OR */
printf("p ^ q = %d\n", p ^ q); /* XOR */
printf("~p = %d\n", ~p); /* Complement */
printf("p << 1 = %d\n", p << 1); /* Left Shift */
printf("p >> 1 = %d\n\n", p >> 1); /* Right Shift */
```

```
/* Demonstrating Assignment Operators */
int assign = 10;w
printf("Assignment Operators:\n");
printf("Initial value: %d\n", assign);
assign += 5;
printf("After += 5: %d\n", assign);
assign -= 3;
printf("After -= 3: %d\n", assign);
assign *= 2;
printf("After *= 2: %d\n", assign);
assign /= 4;
printf("After /= 4: %d\n", assign);
assign %= 3;
printf("After %%= 3: %d\n\n", assign);
/* Demonstrating Conditional Operator */
int max = (a > b) ? a : b;
printf("Conditional Operator:\n");
printf("The larger of %d and %d is %d\n", a, b, max);
return 0;

}
```

Output


Select "C:\Users\Asus\Documents\sonu\c program.exe"
Relational Operators:
10 > 3 = 1
10 < 3 = 0
10 == 3 = 0
10 != 3 = 1
10 >= 3 = 1
10 <= 3 = 0
Logical Operators:
x && y = 0
x || y = 1
!x = 0
Bitwise Operators:
p & q = 0
p | q = 7
p ^ q = 7
~p = -6
p << 1 = 10
p >> 1 = 2
Assignment Operators:
Initial value: 10
After += 5: 15
After -= 3: 12
After *= 2: 24
After /= 4: 6
After %= 3: 0
Conditional Operator:
The larger of 10 and 3 is 10
Process returned 0 (0x0) execution time : 0.039 s
Press any key to continue.