CS1160
Lab 2: Variables and Operators

C Keywords and Identifiers

Keywords are predefined; reserved words used in programming that have special meanings to the compiler. For example: int, constant, case, else, for, do, return.

Identifier refers to name given to entities such as variables, functions, structures etc. Identifiers must be unique.

Note: C is a case sensitive language, all keywords must be written in lowercase, and identifiers must always be written the same way it was defined in.

Rules for naming identifiers:

- The identifier can have letters, digits, and underscore.
- The first letter of an identifier should be either a letter or an underscore.
- Keywords cannot be used as identifiers.
- No space is allowed within the identifier.
- There is no rule on how long an identifier can be. However, some compilers show errors if the identifier is longer than 31 characters.

Variables

A variable is a container (storage place) that has memory allocated to it and is used to store various types of data. Variables must be defined before they can be used within a program.

Variable declaration include the type of the variable and –optionally- its initial value.

- A typical variable declaration is of the form:

```
Data_type    variable_name;
```

- To define multiple variables at once:

```
Data_type    variable1_name, variable2_name, variable3_name;
```

- The most common variable types used in C:

Data type	Usage	Example
int	store an integer	int year = 2022;
float	store decimal numbers (its memory allocation is 4 bytes)	float pi = 3.14;
double	store decimal numbers, with bigger memory allocation that is equal to 8 bytes	double pi = 3.14;
char	store a single character	char grade = 'A';

Note: C is a strongly typed language. This means that the variable type cannot be changed once it is declared.

Const: is a keyword used to define a variable whose value cannot be changed. Example:

```
const float  pi = 3.14;
```

Input and Output Operation

- The function printf () is a library function used to send formatted output to the screen.
- Variable output

```
int year  =  2022;
printf ("Year =  %d", year);
```

- The function scanf () reads formatted input from the standard input such as keyboard

```
int year;
scanf("%d", &year);
printf ("Year =  %d",year);
```

- Taking multiple input in one line

```
int first;
float second;
scanf("%d %f", &first, &second);
```

Specifiers used in C for input and output purposes, so the compiler can understand what type of data is in a variable.

Data type	Format Specifier
int	%d or %i
float	%f
double	%lf
char	%c

Example

1. Write a C program to do the following:
    - Declare a variable called radius.
    - Read the value of the radius from the keyboard.
    - Compute the area (A)of the circle based on the following formula:

$$A = \pi r^2$$

Where π=3.14

```c
#include <stdio.h>

int main() {
    float radius, area;
    const float PI = 3.14;

    // Read radius from user
    printf("Enter the radius of the circle: ");
    scanf("%f", &radius);

    // Compute area
    area = PI * radius * radius;

    // Display result
    printf("The area of the circle is: %.2f\n", area);

    return 0;
}
```

Operators

An **operator** is a symbol that tells the compiler to perform specific mathematical or logical functions on a value or a variable.

1. **Arithmetic Operators:**

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

Operator	Meaning
+	addition
-	subtraction
*	multiplication
/	division
%	Reminder after division (modulo division)

2. **Relational Operators:**

   A relational operator checks the relationship between two operands. If the relation is true, it returns 1; if the relation is false it returns value 0.

   Relational operators are used in decision-making and loops.

Operator	Meaning	Example
==	Equal to	a == b
>	Greater than	a > b
<	Less than	a < b
!=	Not equal to	a != b
>=	Greater than or equal to	A > = b
<=	Less than or equal to	a <= b

3. Logical Operators:

An expression containing logical operator returns either 0 or 1 depending upon whether expression results true or false. Logical operators are commonly used in decision making.

Operator	Meaning	Example
&&	Logical and, true only if all operands are true.	(a=5) && (b > 3)
||	Logical or, true only if either one operand is true.	(a=5) || (b > 3)
!	Logical not, true only if the operand is 0	a !=b

4. Assignment Operators:

An assignment operator is used for assigning a value to a variable. The most common assignment operator is =.

Operator	Example	Same as
=	a = b	a = b
+=	a += b	a = a + b
-=	a -= b	a = a - b
*=	a *= b	a = a * b
/=	a /= b	a = a / b
%=	a %= b	a = a % b

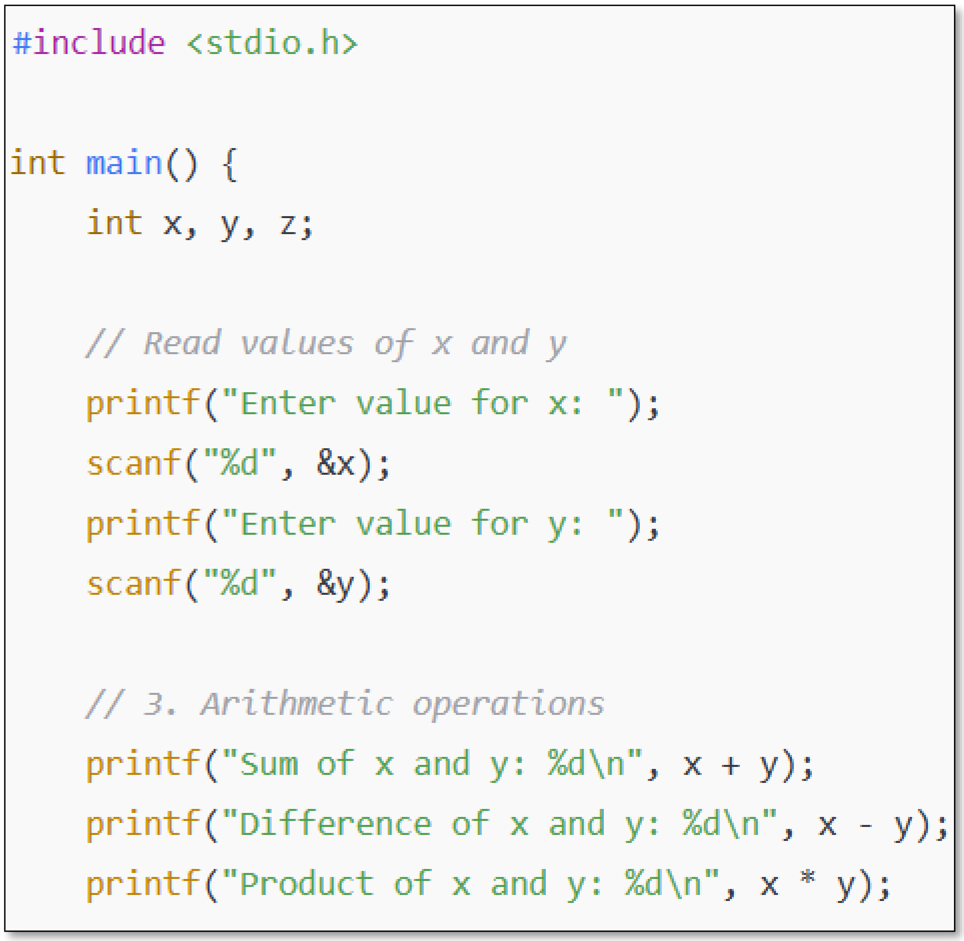
5. Increment and Decrement Operators:

Operators increment ++ and decrement - - are used to change the value of an operand (constant or variable) by 1. These two operators are unary operators, meaning they only operate on a single operand.

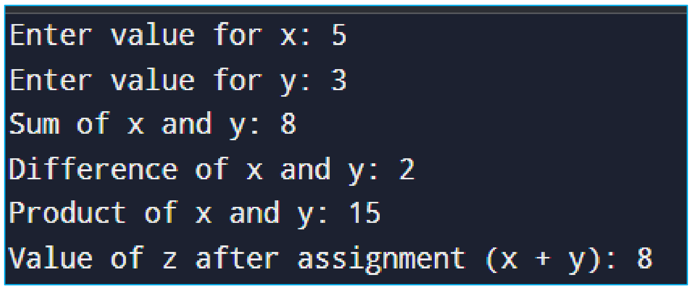
Example

Write a C program to demonstrate the use of arithmetic, relational, logical, increment, and decrement operators. The program should:

1. Declare three integer variables: x, y, and z.s
2. Read the values of x and y from the user.
3. **Arithmetic operations:**
    - Compute and print the **sum**, **difference** and **product** of x and y.

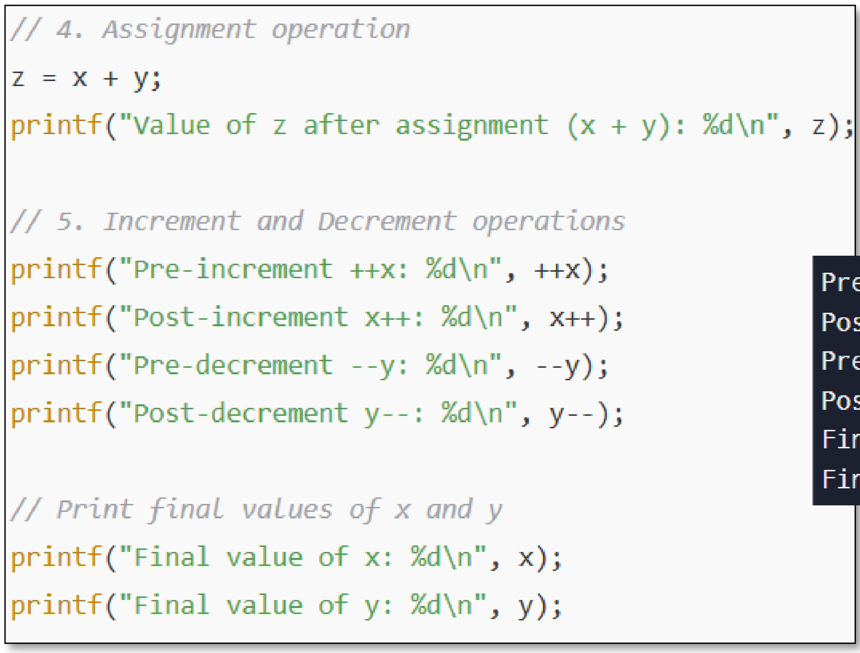


```c
#include <stdio.h>

int main() {
    int x, y, z;

    // Read values of x and y
    printf("Enter value for x: ");
    scanf("%d", &x);
    printf("Enter value for y: ");
    scanf("%d", &y);

    // 3. Arithmetic operations
    printf("Sum of x and y: %d\n", x + y);
    printf("Difference of x and y: %d\n", x - y);
    printf("Product of x and y: %d\n", x * y);
```

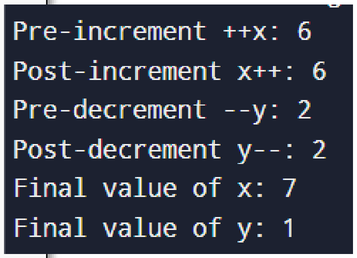


```
Enter value for x: 5
Enter value for y: 3
Sum of x and y: 8
Difference of x and y: 2
Product of x and y: 15
Value of z after assignment (x + y): 8
```

4. **Assignment operation:**
   - Assign the result of (x + y) to z and display it.
5. **Increment and Decrement operations:**
   - print the result of **pre-increment** ++x and **post-increment** x++.
   - print the result of **pre-decrement** --y and **post-decrement** y--.
   - After all operations, print the **final values** of x and y.
6. **Relational operation:**
   - Display the result of the expression x <= y.

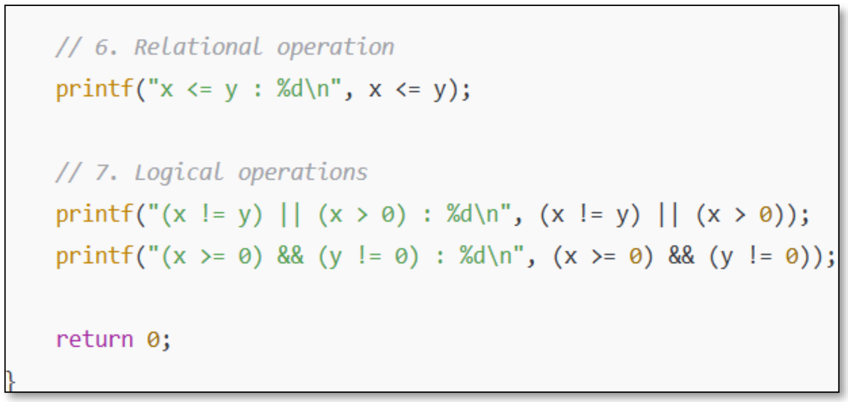


```c
// 4. Assignment operation
z = x + y;
printf("Value of z after assignment (x + y): %d\n", z);

// 5. Increment and Decrement operations
printf("Pre-increment ++x: %d\n", ++x);
printf("Post-increment x++: %d\n", x++);
printf("Pre-decrement --y: %d\n", --y);
printf("Post-decrement y--: %d\n", y--);

// Print final values of x and y
printf("Final value of x: %d\n", x);
printf("Final value of y: %d\n", y);
```

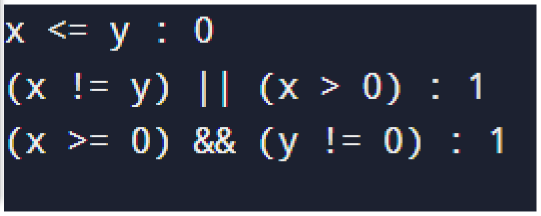


```
Pre-increment ++x: 6
Post-increment x++: 6
Pre-decrement --y: 2
Post-decrement y--: 2
Final value of x: 7
Final value of y: 1
```

7. **Logical operation:**
   - Display the result of the expression (x != y) || (x > 0).
   - Display the result of the expression (x >= 0) && (y != 0)



```c
    // 6. Relational operation
    printf("x <= y : %d\n", x <= y);

    // 7. Logical operations
    printf("(x != y) || (x > 0) : %d\n", (x != y) || (x > 0));
    printf("(x >= 0) && (y != 0) : %d\n", (x >= 0) && (y != 0));

    return 0;
}
```



```
x <= y : 0
(x != y) || (x > 0) : 1
(x >= 0) && (y != 0) : 1
```

Tasks

1. Write a C program to do the following:

- Declare a variable called `side`.
- Read the value of the side of a square from the keyboard.
- Compute the area (A) of the square based on the following formula:

$$A = \text{side}^2$$

3. Write a C program to demonstrate arithmetic, relational, logical, and increment/decrement operators.

The program should:

- Declare three integer variables: p, q, and r.
- Read the values of p and q from the user.
- **Arithmetic operations:**
  - Compute and Print the **sum**, **difference**, **product**, and **quotient** of p and q.
- **Assignment operation:**
  - Assign the result of (p * q) to r and display it.
- **Increment and Decrement operations:**
  - Display the result of **pre-increment** ++p and **post-increment** p++.
  - Print the result of **pre-decrement** --q and **post-decrement** q--.
  - After all operations, display the **final values** of p and q.
- **Relational operation:**
  - Display the result of the expression p!= q.
- **Logical operation:**
  - Display the result of (p > 0) && (q < 5)
  - Print the result of (p == q) || (q > 0)

4. Write a program that defines an **integer X** and **float Y** then compute the following equation:

$$\boldsymbol{y = x^3 + 5\,/\,3}$$