# C Programming Workshop

## 

Alphabets--🡪 Words🡪 Sentences🡪Paragraph

Alphabets -🡪 Constants, variables-🡪 Instructions 🡪Program

Digits, Symbols Keywords

## C Keywords

Keywords are predefined, reserved words used in programming that have special meanings to the compiler.

These keywords cannot be used as identifiers (in other words) names of variables, functions, etc.

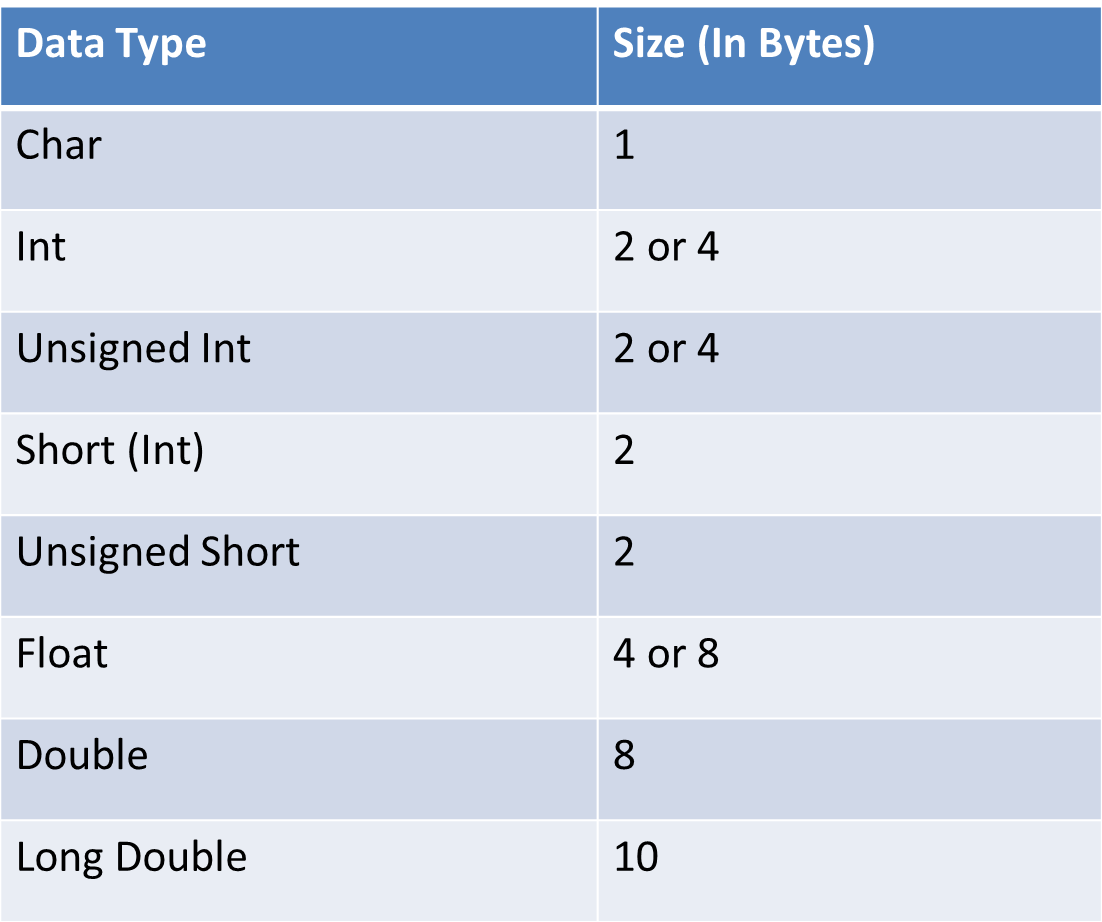
## Identifiers

### Rules for naming identifiers

1. A valid identifier can have letters (both uppercase and lowercase letters), digits and underscores.
2. The first letter of an identifier should be either a letter or an underscore.
3. You cannot use keywords like int, while etc. as identifiers.
4. There is no rule on how long an identifier can be. However, you may run into problems in some compilers if the identifier is longer than *31 characters.*

## Variables

In programming, a variable is a container to hold data. A variable can be of various data types.



long a; //stores large integers  
long long b; //stores long long int (very large integers)  
long double c; // stores a large number with great precision i.e more decimal points

short d;

### signed vs unsigned?

unsigned int x;  
int y; //signed int

an unsigned integer has a range of 0 to 4,294,967,295

signed int is simply int w range -2^31-1 to 2^31 or simply, to +2billion +**2billion**

In the case of chars, which are only 1 byte, the range of an unsigned char is 0 to 256, while the range of a signed char is -127 to 127.

## Variable declaration refers to the part where a variable is first declared or introduced before its first us For ex: int x;

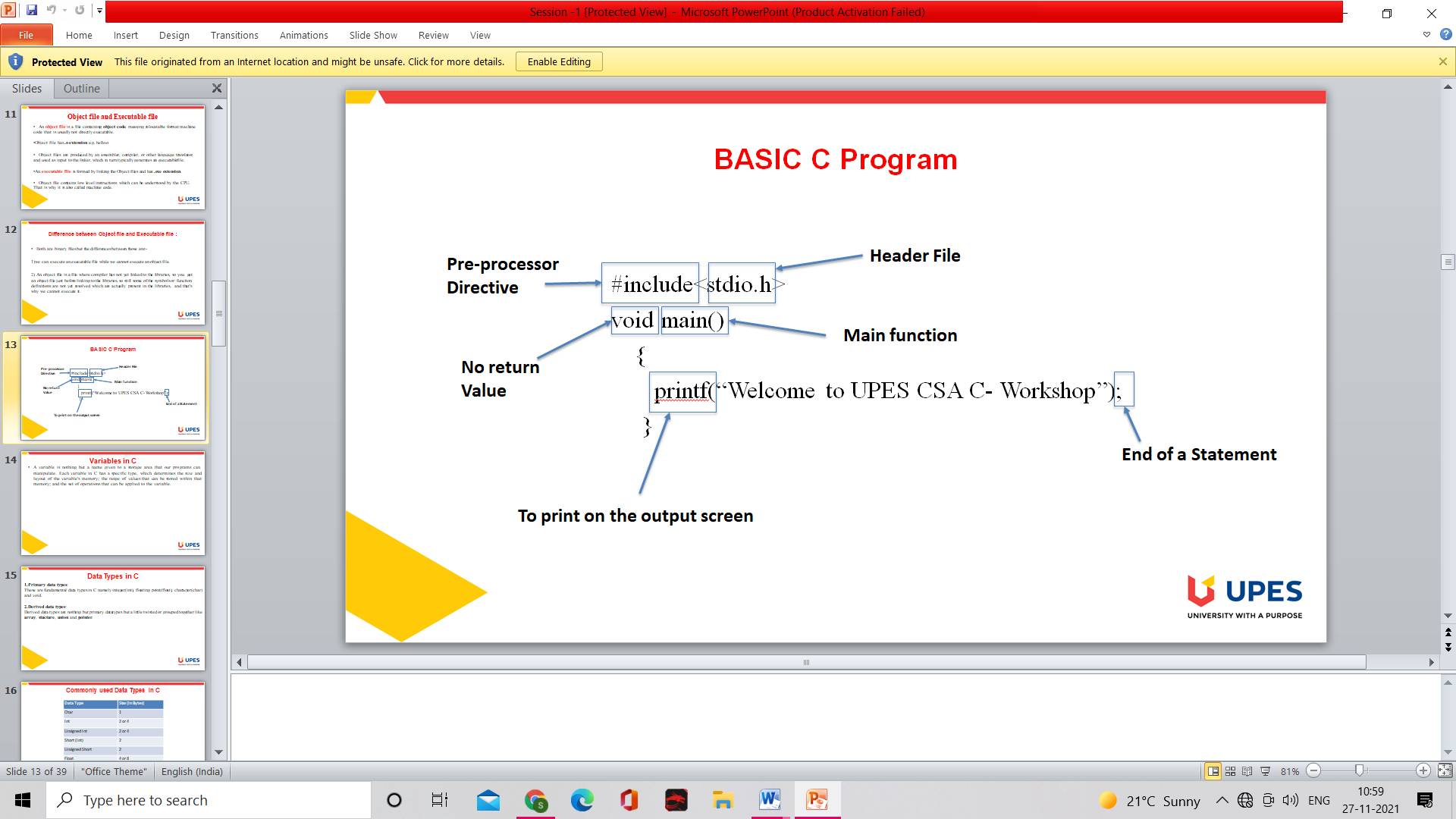
## Variable definition is the part where the variable is assigned a memory location and a value. For ex: x = 10;

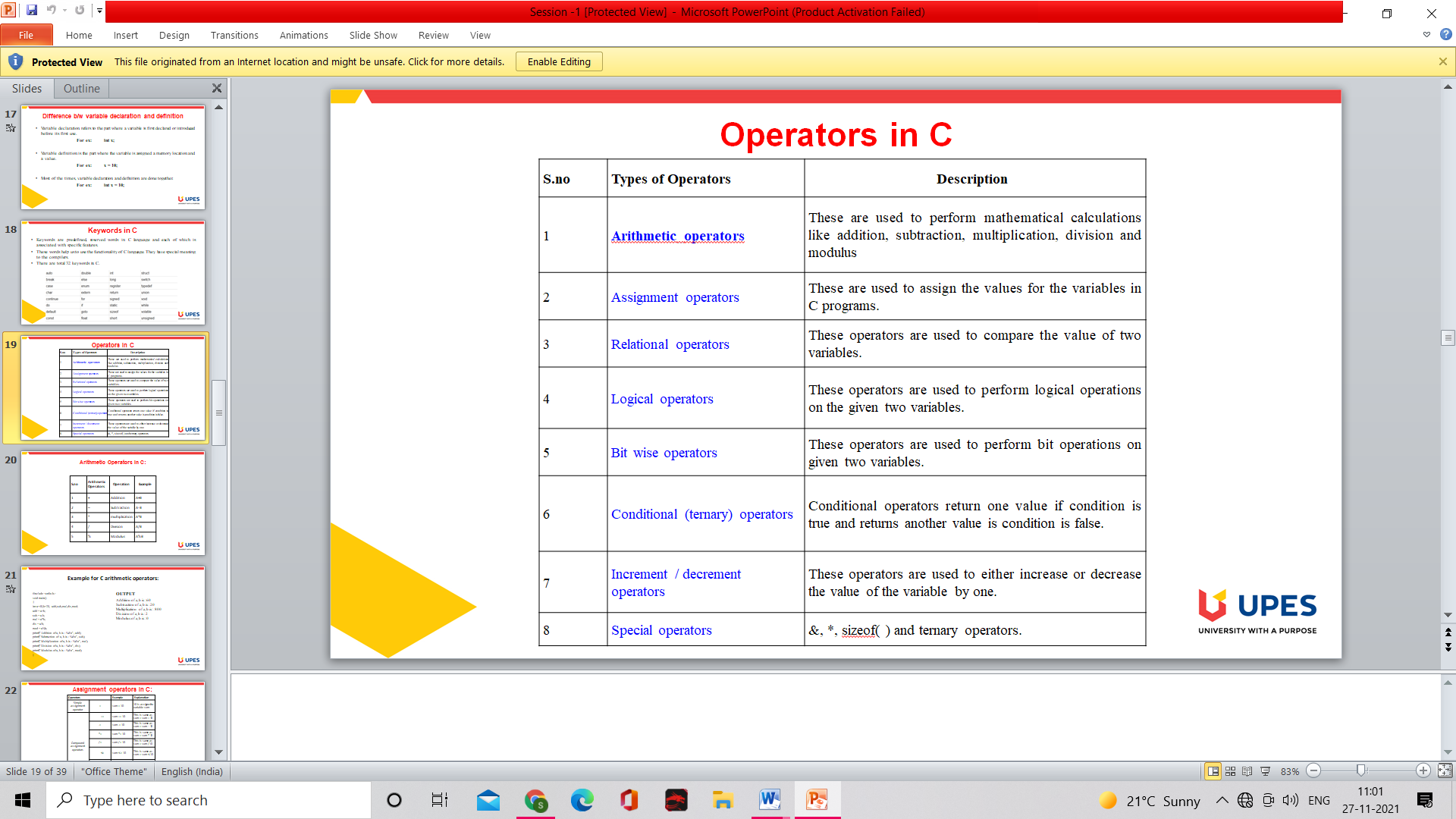
## Most of the times, variable declaration and definition are done together.

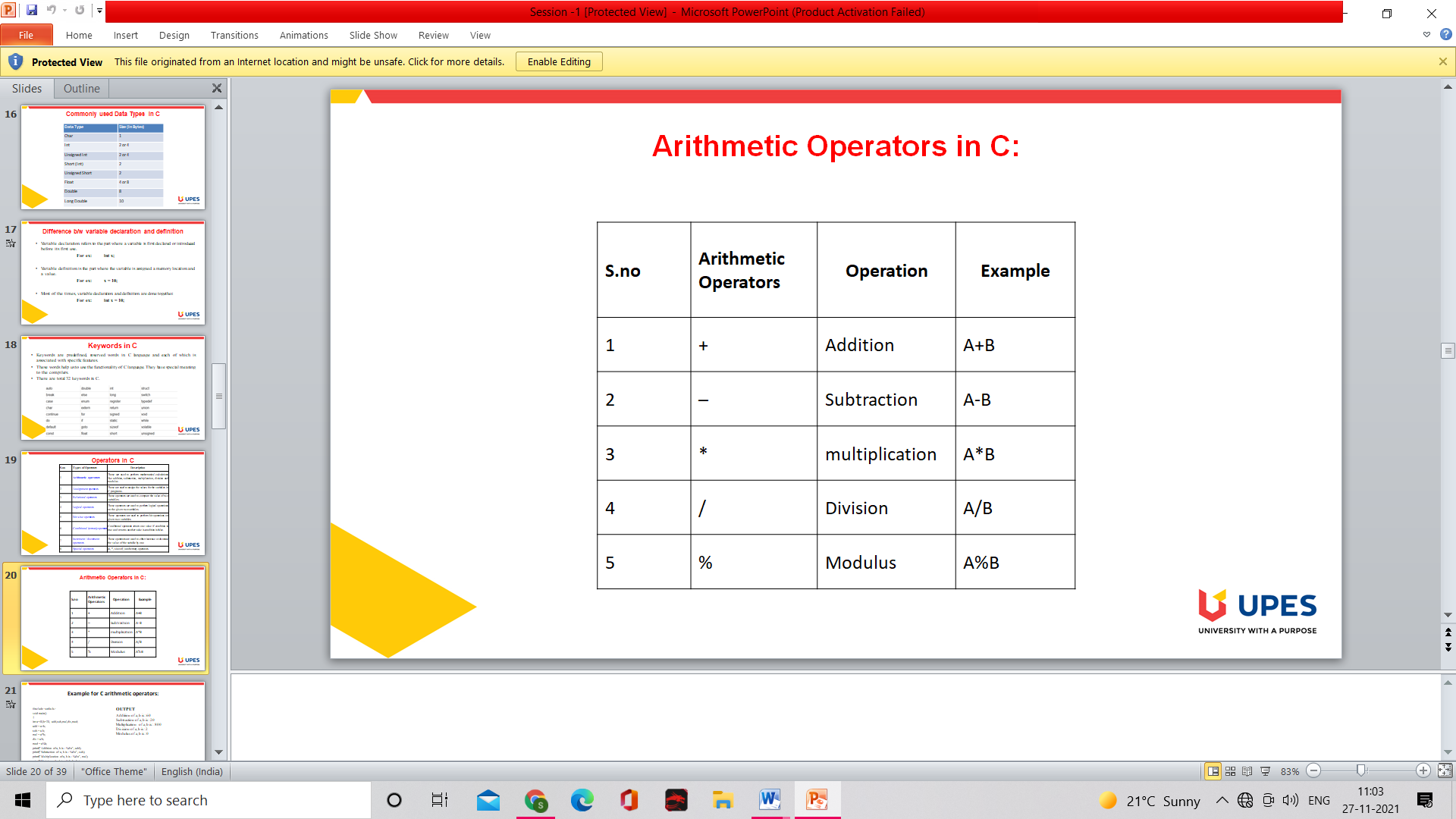
## For ex: int x = 10;

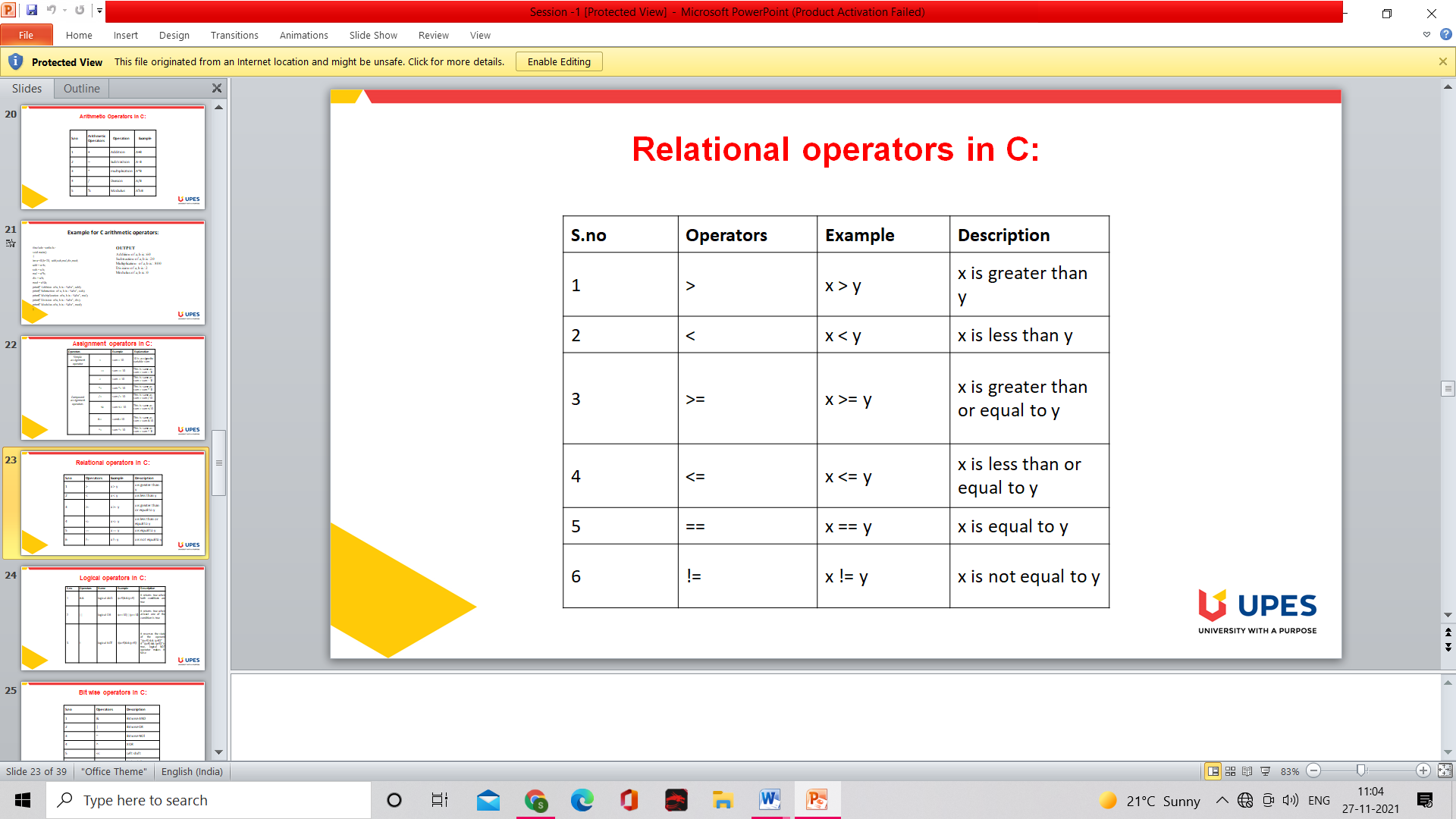
## scanf() vs printf()

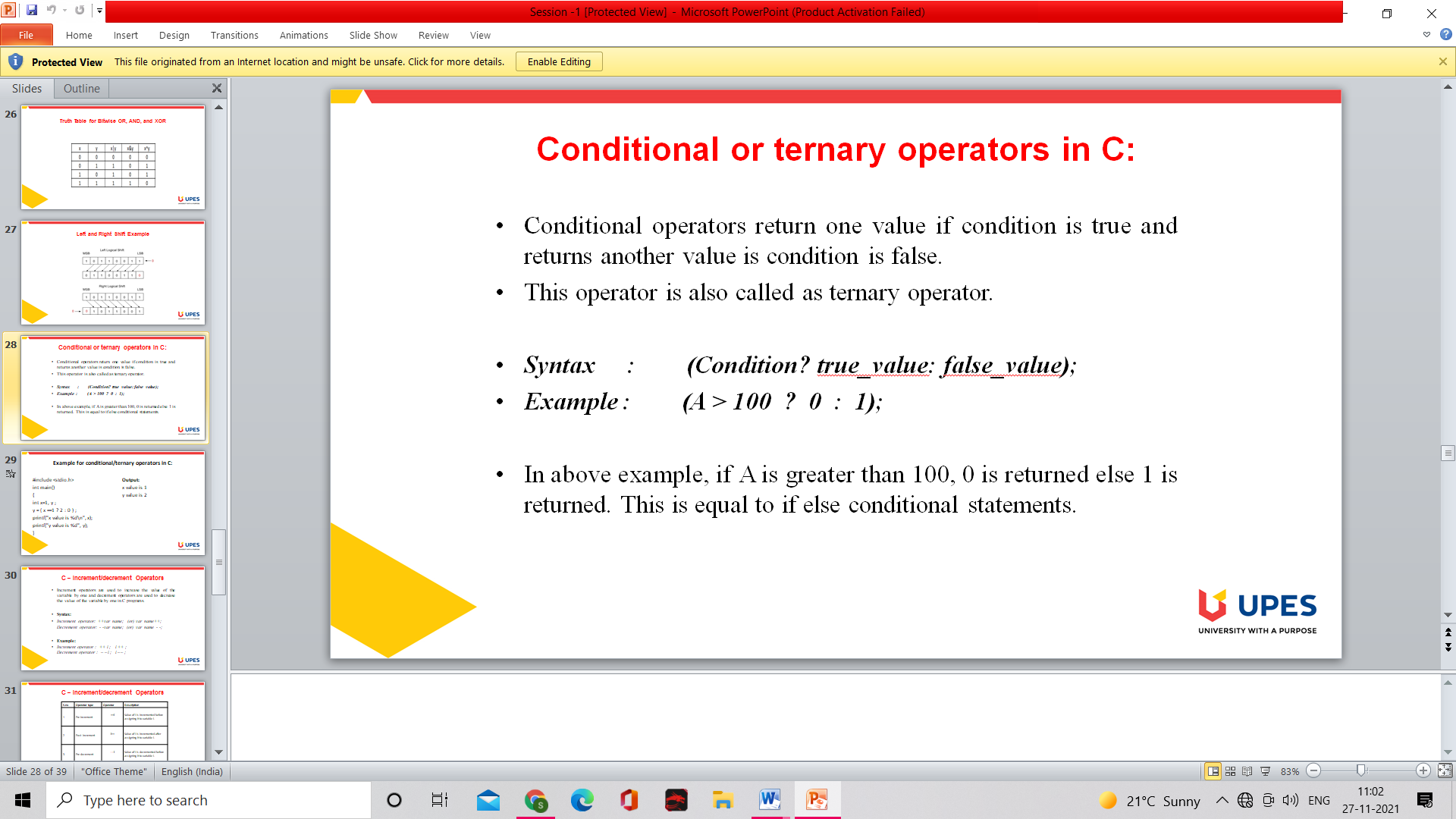
* To take input from the user, we use scanf() function
* To display any output to the user, we use printf() function.
* these functions are defined in stdio.h header file.

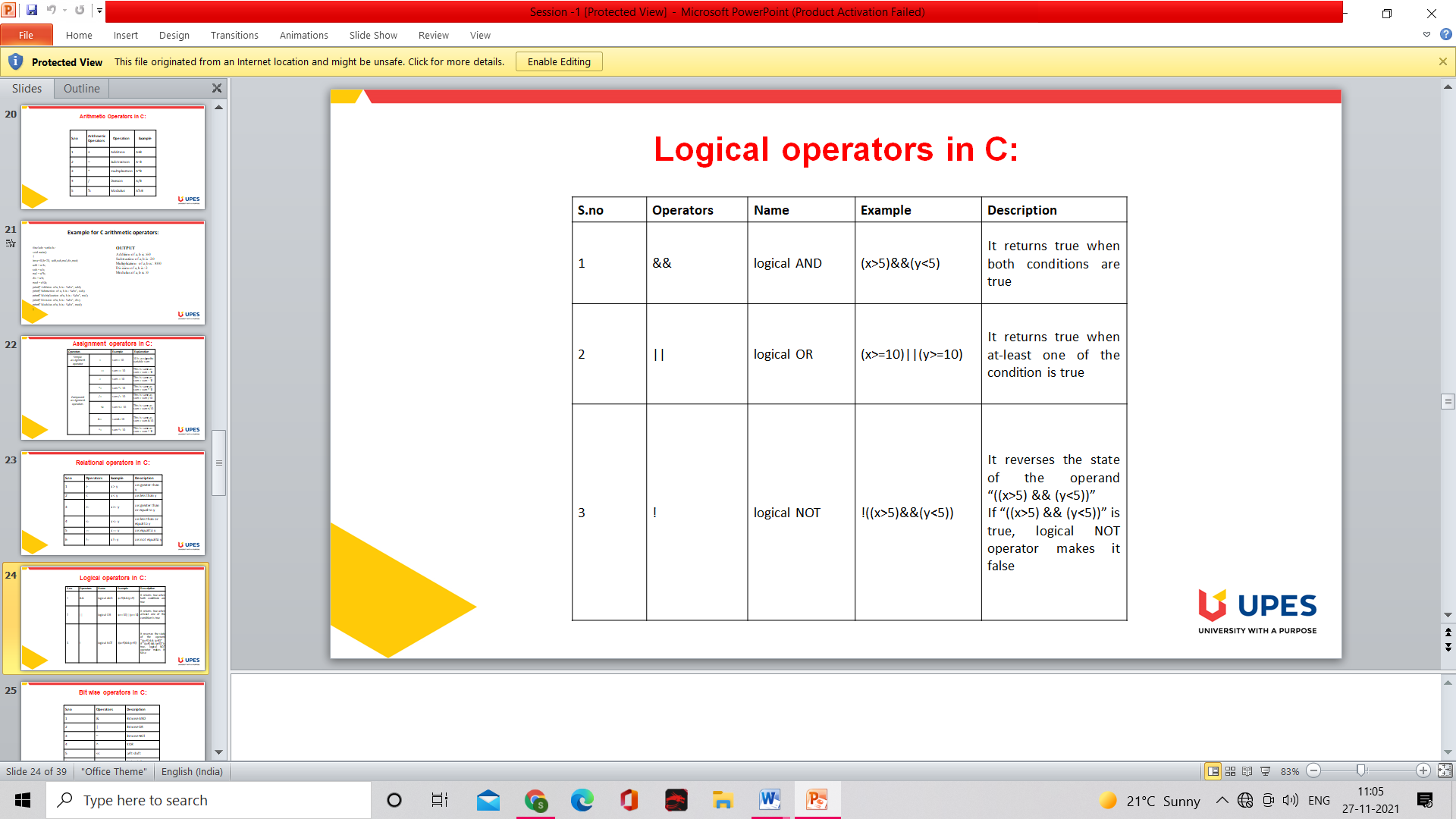












## Bitwise Operator

* In ALU of the CPU, mathematical operations like: addition, subtraction, multiplication and division are done in bit-level. To perform bit-level operations in C programming, bitwise operators are used.

|  |  |
| --- | --- |
|  |  |

## Bitwise AND operator &

I assume you guys are familiar with logic gates? Easy stuff right?

The output of bitwise AND is 1 if the corresponding bits of two operands is 1. If either bit of an operand is 0, the result of corresponding bit is evaluated to 0.

Let us C the bitwise AND operation of two integers 12 and 25.

### Example #1: Bitwise AND

#include <stdio.h>  
int main()  
{  
 int a = 12, b = 25;  
 printf("Output = %d", a&b);  
 return 0;  
}

**Output**

Output = 8

But, how?

I assume you all are familiar with binary expansion of a number?

12 = 00001100 (In Binary)  
25 = 00011001 (In Binary)  
  
Bit Operation of 12 and 25  
 00001100  
& 00011001  
 \_\_\_\_\_\_\_\_  
 00001000 = 8 (In decimal)

## Bitwise OR operator |

The output of bitwise OR is 1 if at least one corresponding bit of two operands is 1. In C Programming, bitwise OR operator is denoted by |.

12 = 00001100 (In Binary)  
25 = 00011001 (In Binary)  
  
Bitwise OR Operation of 12 and 25  
 00001100  
| 00011001  
 \_\_\_\_\_\_\_\_  
 00011101 = 29 (In decimal)

### Example #2: Bitwise OR

#include <stdio.h>  
int main()  
{  
 int a = 12, b = 25;  
 printf("Output = %d", a|b);  
 return 0;  
}

**Output**

Output = 29

## Bitwise XOR (exclusive OR) operator ^

The result of bitwise XOR operator is 1 if the corresponding bits of two operands are opposite. It is denoted by ^.

12 = 00001100 (In Binary)  
25 = 00011001 (In Binary)  
  
Bitwise XOR Operation of 12 and 25  
 00001100  
^ 00011001  
 \_\_\_\_\_\_\_\_  
 00010101 = 21 (In decimal)

### Example #3: Bitwise XOR

#include <stdio.h>  
int main()  
{  
 int a = 12, b = 25;  
 printf("Output = %d", a^b);  
 return 0;  
}

**Output**

Output = 21

## 

## Shift Operators in C programming

There are two shift operators in C programming:

* Right shift operator //divide by 2^x, where x is number of shifts
* Left shift operator. // multiply by 2^x , where x is number of shifts

### Right Shift Operator

Right shift operator shifts all bits towards right by certain number of specified bits. It is denoted by >>.

212 = 11010100 (In binary) // think about dragging this chain of bits to the right  
212>>2 = 00110101 (In binary) [Right shift by two bits] //to maintain the 8 bit structure, we add 0s to the front  
212>>7 = 00000001 (In binary)  
212>>8 = 00000000   
212>>0 = 11010100 (No Shift)

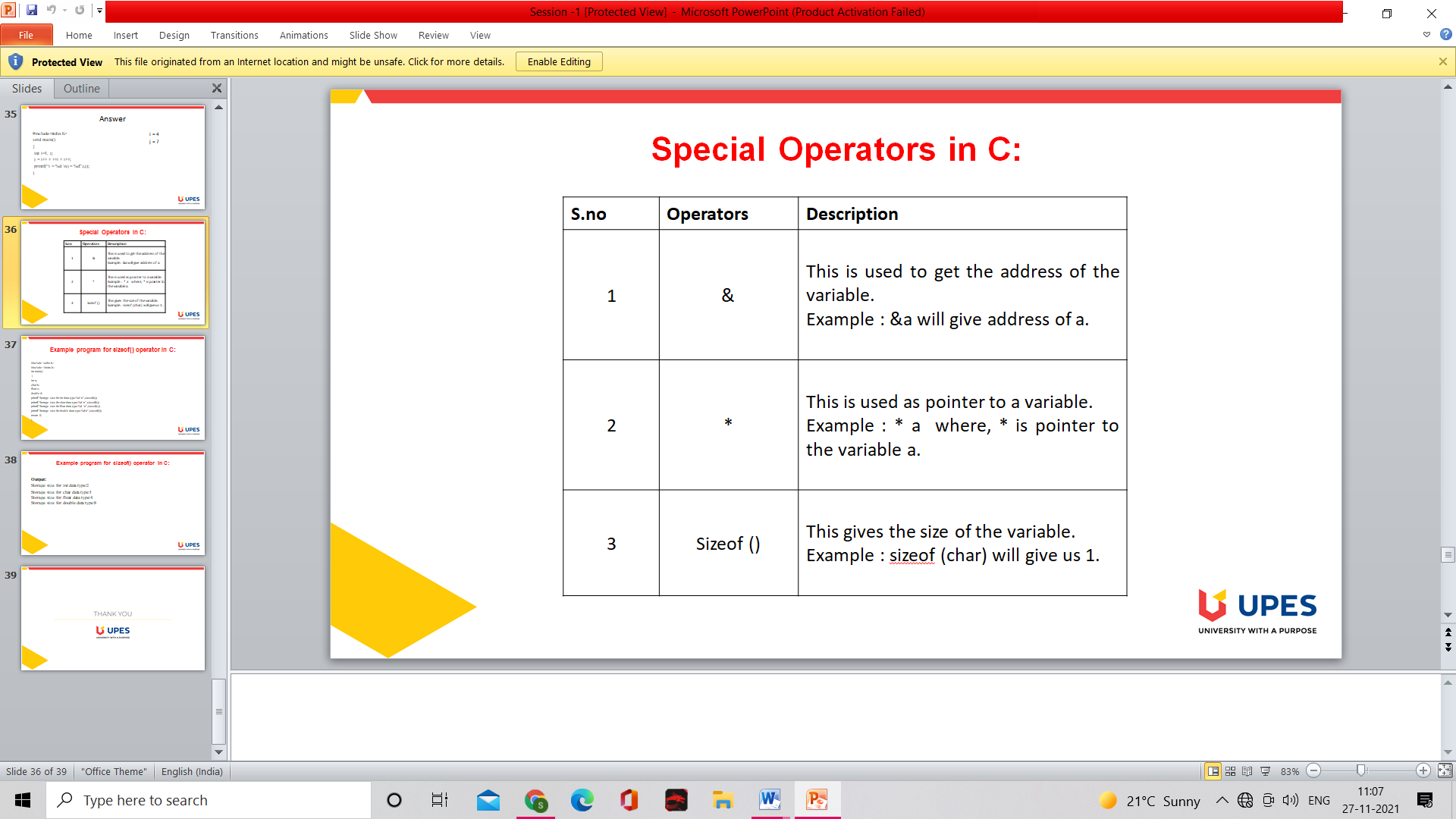
## Left Shift Operator

Left shift operator shifts all bits towards left by a certain number of specified bits. The bit positions that have been vacated by the left shift operator are filled with 0. The symbol of the left shift operator is <<.

212 = 11010100 (In binary)  
212<<1 = 110101000 (In binary) [Left shift by one bit]  
212<<0 = 11010100 (Shift by 0)  
212<<4 = 110101000000 (In binary) =3392(In decimal)

### Example

#include <stdio.h>  
int main()  
{  
 int num=212, i;  
 for (i=0; i<=2; ++i)  
 printf("Right shift by %d: %d\n", i, num>>i);  
  
 printf("\n");  
  
 for (i=0; i<=2; ++i)   
 printf("Left shift by %d: %d\n", i, num<<i);   
   
 return 0;  
}  
  
Right Shift by 0: 212  
Right Shift by 1: 106  
Right Shift by 2: 53  
  
Left Shift by 0: 212  
Left Shift by 1: 424  
Left Shift by 2: 848



e.g. :-

#include <stdio.h>

#include <limits.h>

int main()

{

int a;

char b;

float c;

double d;

printf("Storage size for int data type:%d \n",sizeof(a));

printf("Storage size for char data type:%d \n",sizeof(b));

printf("Storage size for float data type:%d \n",sizeof(c));

printf("Storage size for double data type:%d\n",sizeof(d));

return 0;

}