**Entendiendo el Patrón**  
Cada bit binario puede representar dos estados (0 o 1). A medida que aumentamos los bits, los estados posibles crecen exponencialmente:

* **1 bit** → 21=22^1 = 2 estados (0,1)
* **2 bits** → 22=42^2 = 4 estados (00, 01, 10, 11)
* **3 bits** → 23=82^3 = 8 estados (000, 001, ..., 111)

**Regla General**  
El número de combinaciones posibles con NN bits es 2N2^N.

**Conclusión**  
Con NN bits, se pueden representar **2N2^N** estados distintos.

1010101010 en base 2

1(512) + 0(256) + 1(128) + 0(64) + 1(32) + 0(16) + 1(8) + 0(4) + 1(2) + 0(1) = 682 en base 10

11111 en base 2

1(16) + 1(8) + 1(4) + 1(2) + 1(1) = 31 en base 10

10000000 en base 2

1(128) + 0(64) + 0(32) + 0(16) + 0(8) + 0(4) + 0(2) + 0(1) = 128 en base 10

100100100 en base 2

1(256) + 0(128) + 0(64) + 1(32) + 0(16) + 0(8) + 1(4) + 0(2) + 0(1) = 292 en base 10

127 en base 10

127 ÷ 2 = 63, residuo 1

63 ÷ 2 = 31, residuo 1

31 ÷ 2 = 15, residuo 1

15 ÷ 2 = 7, residuo 1

7 ÷ 2 = 3, residuo 1

3 ÷ 2 = 1, residuo 1

1 ÷ 2 = 0, residuo 1

Resultado: 1111111 en base 2

246 en base 10

246 ÷ 2 = 123, residuo 0

123 ÷ 2 = 61, residuo 1

61 ÷ 2 = 30, residuo 1

30 ÷ 2 = 15, residuo 0

15 ÷ 2 = 7, residuo 1

7 ÷ 2 = 3, residuo 1

3 ÷ 2 = 1, residuo 1

1 ÷ 2 = 0, residuo 1

Resultado: 11110110 en base 2

1025 en base 10

1025 ÷ 2 = 512, residuo 1

512 ÷ 2 = 256, residuo 0

256 ÷ 2 = 128, residuo 0

128 ÷ 2 = 64, residuo 0

64 ÷ 2 = 32, residuo 0

32 ÷ 2 = 16, residuo 0

16 ÷ 2 = 8, residuo 0

8 ÷ 2 = 4, residuo 0

4 ÷ 2 = 2, residuo 0

2 ÷ 2 = 1, residuo 0

1 ÷ 2 = 0, residuo 1

Resultado: 10000000001 en base 2

354 en base 10

354 ÷ 2 = 177, residuo 0

177 ÷ 2 = 88, residuo 1

88 ÷ 2 = 44, residuo 0

44 ÷ 2 = 22, residuo 0

22 ÷ 2 = 11, residuo 0

11 ÷ 2 = 5, residuo 1

5 ÷ 2 = 2, residuo 1

2 ÷ 2 = 1, residuo 0

1 ÷ 2 = 0, residuo 1

Resultado: 101100010 en base 2

Números enteros (Integer)

C: int (puede ser short int, long int o unsigned int para variar su tamaño)

Java: int, byte, short, long

Python: int (sin límite de tamaño)

Números decimales (Flotantes)

C: float, double, long double

Java: float, double

Python: float (similar a double en otros lenguajes)

Caracteres (Letras)

C: char (almacena un solo carácter)

Java: char (usa Unicode, por lo que ocupa 2 bytes)

Python: No tiene un tipo char, usa cadenas de un solo carácter con str

Cadenas de texto

C: char[] (arreglo de caracteres) o char \* (puntero a una cadena)

Java: String

Python: str

Booleanos (Verdadero o Falso)

C: No tiene un tipo booleano nativo, se usa int donde 0 es falso y cualquier otro valor es verdadero

Java: boolean con valores true o false

Python: bool con valores True o False

Otros tipos importantes

C: void para funciones sin retorno, enum para enumeraciones, struct y union para agrupación de datos

Java: enum para enumeraciones, ArrayList y HashMap para estructuras de datos avanzadas

Python: list, tuple, dict y set para colecciones de datos

Entero int ocupa 4 bytes

Flotante float ocupa 4 bytes

Booleano bool ocupa 1 byte

Cadena de texto de 10 caracteres char[10] ocupa 10 bytes

Paso 1 calcular el tamaño de un registro

Cada conjunto de datos almacenado ocupa la suma de 4 más 4 más 1 más 10 lo que da un total de 19 bytes

Paso 2 determinar la cantidad de registros

Se almacena la información cada 10 segundos durante 24 horas

El número de registros por hora se obtiene dividiendo 3600 segundos entre 10 lo que da 360 registros por hora

El número total de registros en 24 horas se obtiene multiplicando 360 por 24 lo que da 8640 registros

Paso 3 calcular el espacio total

Se multiplica 8640 registros por 19 bytes por registro lo que da un total de 164160 bytes o aproximadamente 160.31 KB

Con esta actividad entendí cómo se organiza la información en la memoria de un computador y la relevancia de conocer las dimensiones de los diversos tipos de datos. Aprendí que cada tipo de dato tiene un tamaño determinado en bytes, y que en el caso de los datos extensos, de hecho, hay que calcular el espacio que va a ocupar. Vi cómo puede afectar el uso de la memoria la frecuencia de almacenamiento, así como la cantidad de registros, lo cual es muy importante para la optimización de los programas y para cuidar los recursos.