In [7]:

```python
import pandas as pd

# Crear lista con los valores de la variable independiente (Examen 1)
df = pd.read_csv('Valhalla23.csv')

# Crear lista con los valores de la variable independiente (Examen 1)
Celsius = df['Celsius'].tolist()
print("Celsius:", Celsius)

# Crear lista con los valores de la variable dependiente (Examen 2)
Valks = df['Valks'].tolist()
print("Valks:", Valks)

# Crear lista con los hiper-parámetros iniciales (thetas)
theta0 = 1
theta1 = 1
print("Theta0:", theta0)
print("Theta1:", theta1)

# Cargar el valor del learning rate (alpha)
alpha = 0.0005
print("Alpha:", alpha)
```

```
Celsius: [61.472, 70.579, -7.3013, 71.338, 43.236, -10.246, 7.8498, 34.688,
75.751, 76.489, -4.2387, 77.059, 75.717, 28.538, 60.028, -5.8114, 22.176,
71.574, 59.221, 75.949, 45.574, -16.429, 64.913, 73.399, 47.874, 55.774,
54.313, 19.223, 45.548, -2.8813, 50.605, -16.817, 7.6923, -15.383, -10.287,
62.346, 49.483, 11.71, 75.022, -16.555, 23.874, 18.156, 56.552, 59.52,
-1.3127, 28.976, 24.559, 44.631, 50.936, 55.469, 7.6025, 47.97, 45.51,
-3.7388, -8.1002, 29.836, 75.974, 14.039, 38.527, 2.3812, 55.127, 5.5095,
30.596, 49.908, 69.09, 75.929, 34.722, -6.1376, -5.0706, 5.7508, 64.072,
```

```
5.4282, 61.428, 4.3525, 72.926, 14.998, -0.34047, 5.1084, 41.604, 27.329,
15.166, 63.083, 38.526, 34.972, 71.719, 8.5839, 55.72, 55.373, 18.045, 36.782,
-12.415, -14.605, 33.08, 57.917, 73.401, -7.0094, 36.882, 26.939, -18.81,
13.712]
Valks: [-139.74, -156.6, 73.269, -165.42, -75.835, 83.437, 24.68, -55.108,
-182.82, -183.46, 61.973, -171.99, -175.83, -30.998, -142.49, 66.37, -12.882,
-150.58, -117.99, -174.92, -81.557, 103.46, -142.02, -156.32, -105.25,
-133.38, -110.95, -3.1829, -80.032, 58.486, -104.7, 98.051, 26.448, 98.098,
79.143, -122.73, -92.412, 16.143, -171.65, 99.744, -15.686, -1.8801, -129.87,
-111.75, 54.276, -40.934, -23.508, -85.997, -94.78, -115.04, 25.08, -95.258,
-87.107, 60.614, 74.247, -41.73, -186.09, 8.7644, -62.516, 42.159, -100.09,
32.198, -49.374, -108.59, -140.64, -166.94, -50.735, 69.075, 64.106, 31.984,
-127.1, 33.215, -134.07, 36.214, -178.19, 1.3718, 51.101, 35.183, -79.775,
-27.032, 8.774, -148.49, -65.58, -54.496, -182.57, 23.682, -121.09, -128.06,
-7.4412, -62.04, 84.882, 91.536, -56.911, -107.37, -169.76, 69.632, -71.24,
-34.255, 106.43, 9.1011]
Theta0: 1
Theta1: 1
Alpha: 0.0005
```

In [8]:

```python
# Crear función lambda para la función de hipótesis

h0 = lambda theta0, theta1, x : theta0 + theta1 * x

# Calcular el total de muestras a partir de los datos (n)
n = len(Celsius)
```

In [9]:

```python
# Calcular delta para theta0 y para cada muestra
```

```python
delta = [h0(theta0, theta1, Celsius[i]) - Valks[i] for i in
range(len(Celsius))]
print(delta)

# Calcular delta para theta1 y para cada muestra
deltax = [delta[i] * Celsius[i] for i in range(len(Celsius)) ]
print(deltax)

# Calcular sumatorias y promedio

sumdelta = sum(delta)
sumdeltax = sum(deltax)
print(sumdelta)
print(sumdeltax)
```

```
[202.21200000000002, 228.17899999999997, -79.5703, 237.75799999999998,
120.071, -92.68299999999999, -15.8302, 90.79599999999999, 259.571, 260.949,
-65.2117, 250.049, 252.54700000000003, 60.536, 203.518, -71.18140000000001,
36.058, 223.154, 178.21099999999998, 251.86899999999997, 128.131, -118.889,
207.933, 230.719, 154.124, 190.154, 166.263, 23.4059, 126.58, -60.3673,
156.305, -113.868, -17.7557, -112.481, -88.43, 186.076, 142.895, -3.433,
247.67200000000003, -115.299, 40.56, 21.036099999999998, 187.422, 172.27,
-54.5887, 70.91, 49.067, 131.628, 146.716, 171.50900000000001, -16.4775,
144.228, 133.617, -63.352799999999995, -81.3472, 72.566, 263.064,
6.2745999999999995, 102.043, -38.7778, 156.217, -25.6885, 80.97, 159.498,
210.73, 243.869, 86.457, -74.21260000000001, -68.1766, -25.233200000000004,
192.172, -26.786800000000003, 196.498, -30.8615, 252.11599999999999,
14.626199999999999, -50.44147, -29.0746, 122.379, 55.361000000000004,
7.392000000000001, 212.573, 105.106, 90.468, 255.289, -14.098099999999999,
177.81, 184.433, 26.486200000000004, 99.822, -96.297, -105.141, 90.991,
166.287, 244.161, -75.6414, 109.12199999999999, 62.194, -124.24000000000001,
5.6108999999999999]
[12430.376064000002, 16104.645640999997, 580.9666313900001,
16961.180203999997, 5191.389756, 949.630018, -124.26390396, 3149.531648,
19662.762821000004, 19959.728061, 276.4128327899999, 19268.525891,
19122.101199, 1727.576368, 12216.778504, 413.66358796000003,
799.6222079999999, 15972.024395999999, 10553.833630999998, 19129.198680999998,
5839.442194, 1953.2273809999997, 13497.554828999999, 16934.543881,
```

```
7378.532376, 10605.649196, 9030.242319, 449.93161569999995, 5765.46584,
173.93630149, 7909.814525, 1914.918156, -136.58217111000002,
1730.2952229999999, 909.6794100000002, 11601.094296, 7070.873285,
-40.200430000000004, 18580.848784, 1908.7749450000001, 968.32944,
381.93143159999994, 10599.088944, 10253.510400000001, 71.65858649,
2054.6881599999997, 1205.0364530000002, 5874.689267999999, 7473.126176000001,
9513.432721000001, -125.27019374999999, 6918.617160000001, 6080.909669999999,
236.86344863999997, 658.92858944, 2165.079176, 19986.024336000002, 88.0891094,
3931.4106610000003, -92.33769736, 8611.774559000001, -141.53079075,
2477.35812, 7960.226184, 14559.3357, 18516.729301, 3001.959954,
455.48725376000004, 345.69626795999994, -145.11108656000002, 12312.844384,
-145.40410776000002, 12070.479143999999, -134.32467875, 18385.811416,
219.36374759999998, 17.1738072909, -148.52468664, 5091.455916,
1512.9607690000003, 112.10707200000002, 13409.742559, 4049.313756,
3163.846896, 18309.071791, -121.01668058999999, 9907.5732, 10212.608509,
477.94347900000014, 3671.652804, 1195.527255, 1535.584305, 3009.9822799999997,
9630.844179, 17921.661561, 530.20082916, 4024.637603999999, 1675.444166,
2336.9544, 76.93666079999998]
8181.8485299999975
615621.9098332411
```

In [10]:

```python
# Actualizar theta0
theta0 = theta0 - alpha * sumdelta/n
print(theta0)

# Actualizar theta1
theta1 = theta1 - alpha * sumdeltax/n
print(theta1)

estimaciones = [h0(theta0, theta1, x) for x in Celsius]

# Crear un DataFrame de pandas para mostrar los resultados en una tabla
tabla = pd.DataFrame({'Celsius': Celsius, 'Valks Estimado': estimaciones})
print(tabla)
```

```
0.95909075735
-2.078109549166206
    Celsius  Valks Estimado
0   61.4720     -126.786459
1   70.5790     -145.711803
2   -7.3013       16.131992
3   71.3380     -147.289088
4   43.2360      -88.890054
..      ...             ...
95  -7.0094       15.525392
96  36.8820      -75.685746
97  26.9390      -55.023102
98 -18.8100       40.048331
99  13.7120      -27.535947

[100 rows x 2 columns]
```

In [11]:

```python
import matplotlib.pyplot as plt
plt.scatter(Celsius, Valks)
plt.scatter(Celsius, [h0(theta0, theta1, x) for x in Celsius], color='red')
plt.xlabel('Celsius')
plt.ylabel('Valks')
plt.title('Regresión Lineal')
plt.show()
```

100 ITERACIONES

In [12]:

```
import pandas as pd
import matplotlib.pyplot as plt

# Crear lista con los valores de la variable independiente (Examen 1)
df = pd.read_csv('Valhalla23.csv')

# Crear lista con los valores de la variable independiente (Examen 1)
Celsius = df['Celsius'].tolist()
print("Celsius:", Celsius)

# Crear lista con los valores de la variable dependiente (Examen 2)
Valks = df['Valks'].tolist()
print("Valks:", Valks)

# Crear lista con los hiper-parámetros iniciales (thetas)
theta0 = 1
theta1 = 1
print("Theta0:", theta0)
print("Theta1:", theta1)

# Cargar el valor del learning rate (alpha)
alpha = 0.005
print("Alpha:", alpha)

# Crear función lambda para la función de hipótesis

h0 = lambda theta0, theta1, x : theta0 + theta1 * x

# Calcular el total de muestras a partir de los datos (n)
n = len(Celsius)

# 100 ITERACIONES
for i in range(100):
  # Calcular delta para theta0 y para cada muestra
  delta = [h0(theta0, theta1, Celsius[i]) - Valks[i] for i in
```

```
range(len(Celsius))]
  # Calcular delta para theta1 y para cada muestra
  deltax = [delta[i] * Celsius[i] for i in range(len(Celsius)) ]
  # Calcular sumatorias
  sumdelta = sum(delta)
  sumdeltax = sum(deltax)
  # Actualizar theta0
  theta0 = theta0 - alpha * sumdelta/n
  # Actualizar theta1
  theta1 = theta1 - alpha * sumdeltax/n

print("Theta0 final:", theta0)
print("Theta1 final:", theta1)
print(h0(theta0, theta1, 61.47200))
estimaciones = [h0(theta0, theta1, x) for x in Celsius]

# Crear un DataFrame de pandas para mostrar los resultados en una tabla
tabla = pd.DataFrame({'Celsius': Celsius, 'Valks Estimado': estimaciones})
print(tabla)

plt.scatter(Celsius, Valks)
plt.scatter(Celsius, [h0(theta0, theta1, x) for x in Celsius], color='red')
plt.xlabel('Celsius')
plt.ylabel('Valks')
plt.title('Regresión Lineal')
plt.show()
```

```
Celsius: [61.472, 70.579, -7.3013, 71.338, 43.236, -10.246, 7.8498, 34.688,
75.751, 76.489, -4.2387, 77.059, 75.717, 28.538, 60.028, -5.8114, 22.176,
71.574, 59.221, 75.949, 45.574, -16.429, 64.913, 73.399, 47.874, 55.774,
54.313, 19.223, 45.548, -2.8813, 50.605, -16.817, 7.6923, -15.383, -10.287,
62.346, 49.483, 11.71, 75.022, -16.555, 23.874, 18.156, 56.552, 59.52,
-1.3127, 28.976, 24.559, 44.631, 50.936, 55.469, 7.6025, 47.97, 45.51,
-3.7388, -8.1002, 29.836, 75.974, 14.039, 38.527, 2.3812, 55.127, 5.5095,
30.596, 49.908, 69.09, 75.929, 34.722, -6.1376, -5.0706, 5.7508, 64.072,
5.4282, 61.428, 4.3525, 72.926, 14.998, -0.34047, 5.1084, 41.604, 27.329,
15.166, 63.083, 38.526, 34.972, 71.719, 8.5839, 55.72, 55.373, 18.045, 36.782,
-12.415, -14.605, 33.08, 57.917, 73.401, -7.0094, 36.882, 26.939, -18.81,
13.712]
```

```
Valks: [-139.74, -156.6, 73.269, -165.42, -75.835, 83.437, 24.68, -55.108,
-182.82, -183.46, 61.973, -171.99, -175.83, -30.998, -142.49, 66.37, -12.882,
-150.58, -117.99, -174.92, -81.557, 103.46, -142.02, -156.32, -105.25,
-133.38, -110.95, -3.1829, -80.032, 58.486, -104.7, 98.051, 26.448, 98.098,
79.143, -122.73, -92.412, 16.143, -171.65, 99.744, -15.686, -1.8801, -129.87,
-111.75, 54.276, -40.934, -23.508, -85.997, -94.78, -115.04, 25.08, -95.258,
-87.107, 60.614, 74.247, -41.73, -186.09, 8.7644, -62.516, 42.159, -100.09,
32.198, -49.374, -108.59, -140.64, -166.94, -50.735, 69.075, 64.106, 31.984,
-127.1, 33.215, -134.07, 36.214, -178.19, 1.3718, 51.101, 35.183, -79.775,
-27.032, 8.774, -148.49, -65.58, -54.496, -182.57, 23.682, -121.09, -128.06,
-7.4412, -62.04, 84.882, 91.536, -56.911, -107.37, -169.76, 69.632, -71.24,
-34.255, 106.43, 9.1011]
Theta0: 1
Theta1: 1
Alpha: 0.005
Theta0 final: 8.305557436551998e+92
Theta1 final: 4.934607093274349e+94
3.0342322281212633e+96
    Celsius  Valks Estimado
0   61.4720    3.034232e+96
1   70.5790    3.483627e+96
2   -7.3013   -3.594599e+95
3   71.3380    3.521081e+96
4   43.2360    2.134357e+96
..      ...             ...
95  -7.0094   -3.450558e+95
96  36.8820    1.820812e+96
97  26.9390    1.330164e+96
98 -18.8100   -9.273690e+95
99  13.7120    6.774639e+95

[100 rows x 2 columns]
```

Prueba tu implementación. Usando Traininig.cvs (60 datos) para predecir los resultados de test.cvs (40 datos)

Yo seleccioné los valores de alpha y datos de manera arbitraria, pero decidi aumentar el número de itteraciiones a 10,000 para que pueda arreglar las fallas del mismo.

In [26]:

```python
import pandas as pd
import matplotlib.pyplot as plt

# Cargar los datos de entrenamiento
df_train = pd.read_csv('Training.csv')

# Datos de entrenamiientto
Celsius_train = df_train['Celsius'].tolist()
Valks_train = df_train['Valks'].tolist()

# Cargar los datos de prueba
df_test = pd.read_csv('test.csv')

# Datos de prueba
Celsius_test = df_test['Celsius'].tolist()
Valks_test = df_test['Valks'].tolist()

# theta inicial
theta0 = 1
theta1 = 1

# Alpha
alpha = 0.0005

# Definir la función de hipótesis
h0 = lambda theta0, theta1, x : theta0 + theta1 * x

# Número n
n_train = len(Celsius_train)

# Gradiente descendente
for i in range(10000):
    delta = [h0(theta0, theta1, Celsius_train[i]) - Valks_train[i] for i in
range(n_train)]
    deltax = [delta[i] * Celsius_train[i] for i in range(n_train)]
    sumdelta = sum(delta)
    sumdeltax = sum(deltax)
    theta0 = theta0 - alpha * sumdelta/n_train
    theta1 = theta1 - alpha * sumdeltax/n_train

# Mostrar los parámetros finales
```

```
print("Theta0:", theta0)
print("Theta1:", theta1)

# Resultados de prueba
estimaciones_test = [h0(theta0, theta1, x) for x in Celsius_test]

# Todos los resultados
tabla_test = pd.DataFrame({'Celsius': Celsius_test, 'Valks Real':
Valks_test, 'Valks Estimado': estimaciones_test})
print(tabla_test)

# Graficar los datos de prueba y la línea de regresión
plt.scatter(Celsius_test, Valks_test, label='Datos Reales')
plt.scatter(Celsius_test, estimaciones_test, color='red',
label='Estimaciones')
plt.xlabel('Celsius')
plt.ylabel('Valks')
plt.title('Regresión Lineal - Conjunto de Prueba')
plt.legend()
plt.show()

# Costo para el conjunto de entrenamiento
costo_train = sum([(h0(theta0, theta1, Celsius_train[i]) -
Valks_train[i])**2 for i in range(n_train)]) / (2 * n_train)
print("Costo en el conjunto de entrenamiento:", costo_train)

# Costo para el conjunto de prueba
n_test = len(Celsius_test)
costo_test = sum([(h0(theta0, theta1, Celsius_test[i]) - Valks_test[i])**2
for i in range(n_test)]) / (2 * n_test)
print("Costo en el conjunto de prueba:", costo_test)
```

```
Theta0: 44.519622211687384
Theta1: -2.884562177380718
     Celsius  Valks Real  Valks Estimado
0    2.38120     42.1590       37.650903
1   55.12700   -100.0900     -114.497637
2    5.50950     32.1980       28.627127
3   30.59600    -49.3740      -43.736442
```

4	49.90800	-108.5900	-99.443107
5	69.09000	-140.6400	-154.774779
6	75.92900	-166.9400	-174.502299
7	34.72200	-50.7350	-55.638146
8	-6.13760	69.0750	62.223911
9	-5.07060	64.1060	59.146083
10	5.75080	31.9840	27.931082
11	64.07200	-127.1000	-140.300046
12	5.42820	33.2150	28.861642
13	61.42800	-134.0700	-132.673263
14	4.35250	36.2140	31.964565
15	72.92600	-178.1900	-165.839959
16	14.99800	1.3718	1.256959
17	-0.34047	51.1010	45.501729
18	5.10840	35.1830	29.784125
19	41.60400	-79.7750	-75.489703
20	27.32900	-27.0320	-34.312578
21	15.16600	8.7740	0.772352
22	63.08300	-148.4900	-137.447214
23	38.52600	-65.5800	-66.611020
24	34.97200	-54.4960	-56.359286
25	71.71900	-182.5700	-162.358293
26	8.58390	23.6820	19.758829
27	55.72000	-121.0900	-116.208182
28	55.37300	-128.0600	-115.207239
29	18.04500	-7.4412	-7.532302
30	36.78200	-62.0400	-61.580344
31	-12.41500	84.8820	80.331462
32	-14.60500	91.5360	86.648653
33	33.08000	-56.9110	-50.901695
34	57.91700	-107.3700	-122.545565
35	73.40100	-169.7600	-167.210126
36	-7.00940	69.6320	64.738672
37	36.88200	-71.2400	-61.868800
38	26.93900	-34.2550	-33.187598
39	-18.81000	106.4300	98.778237
40	13.71200	9.1011	4.966506

```
Costo en el conjunto de entrenamiento: 25.976834257305747
Costo en el conjunto de prueba: 30.88295587243797
```

descarga del archivo a pdf

In [14]:

```
import os

# Montar el drive
from google.colab import drive
drive.mount('/content/drive')
# Listar archivos en el directorio MyDrive/7_Semestre
os.listdir('/content/drive/MyDrive/7_Semestre')
```

In [ ]:

```
!jupyter nbconvert --to html
"/content/drive/MyDrive/7_Semestre/Módulo2.ipynb"
```

```
[NbConvertApp] Converting notebook
/content/drive/MyDrive/7_Semestre/Módulo2.ipynb to html
[NbConvertApp] Writing 740573 bytes to
/content/drive/MyDrive/7_Semestre/Módulo2.html
```