0.1. Busqueda Lineal

proc contiene (in t1 :$seq\langle\mathbb{Z}\rangle$, in x:$\mathbb{Z}$, out result:Bool) {
Pre {true}
Post {$result = \text{true} \leftrightarrow (\exists i : \mathbb{Z})(0 \leq i < |s| \wedge_L s[i] = x)$}
}

El invatiante del ciclo y la funcion variante:

$$I \equiv 0 \leq i \leq |s| \wedge_L (\forall j : \mathbb{Z})(0 \leq j < i \longrightarrow_L s[j] \neq x) \qquad fv = |s| - i$$

En c++ la implementacion es

```
bool contiene(vector<int> &s, int x){
    int i = 0;
    while( i < s.size() && s[i] != x ){
        i = i+1;
    }
    return i < s.size();
}
```

El tamaño de cada secuencia delimitada la cantidad de iteraciones y, tambien, si el elemento esta contenido en la misma Si buscamos que el tiempo de ejecucion sea el maximo, el elemento no debe estar contenido, eso representa el **peor caso** en tiempo de ejecucion El tiempo de ejecucion de la busqueda lineal es de $O(n)$

0.2. Busqueda Binaria

Suponiento que la secuencia esta **ordenada**

proc contieneOrdenada (in t1 :$seq\langle\mathbb{Z}\rangle$, in x:$\mathbb{Z}$, out result:Bool) {
Pre {$ordenado(s)$}
Post {$result = \text{true} \leftrightarrow (\exists i : \mathbb{Z})(0 \leq i < |s| \wedge_L s[i] = x)$}
}

El invatiante del ciclo y la funcion variante:

$$I \equiv 0 \leq i < j < |s| \wedge_L s[i] \leq x < s[j] \qquad fv = j - i - 1$$

Y la funcion variante
Le implementacion en C++ es

```
bool contieneOrdenada(vector<int> &s, int x){
    if(s.size() == 0){
        return false;
    } else if(s.size() == 1){
        return s[0] = x;
    } else if(x < s[0]){
        return false;
    } else if(x ≥ s[s.size() - 1]){
        return s[s.size()-1] == x;
    } else {
        int low = 0;
        int hight = s.size() -1;
        while(low + 1 < high){
            int mid = (low+high) / 2;
            if(s[mid] ≤ x){
                low = mid;
            }else{
                high = mid;
            }
        }
        return s[low] = x;
    }
}
```

La complejidad de la busqueda binaria es $O(log_2|s|)$ o $O(logn)$