Lineare Suche

>> Die **Lineare Suche** ist ein einfacher Algorithmus. Die Idee des Algorithmus besteht darin, zur Suche nach einem Element in einem Array das Array von vorne nach hinten zu durchlaufen und jedes Element mit dem gesuchten Wert zu vergleichen.

(Ein Algorithmus ist eine schrittweise Anleitung zur Durchführung einer Aufgabe.)

Pseudocode

Wiederhole, beginnend beim ersten Element:

> Wenn das aktuelle Element das Gesuchte ist, stoppe.
>
> Andernfalls gehe zum nächsten Element.

Beispiel gesucht: 9

11	23	8	14	30	9	6	17	22	28	25	15	7	10	19

Wiederhole, beginnend beim ersten Element:

Wenn das aktuelle Element das Gesuchte ist, stoppe.

Andernfalls gehe zum nächsten Element.

Beispiel gesucht: 9

11	23	8	14	30	9	6	17	22	28	25	15	7	10	19

Wiederhole, beginnend beim ersten Element:

Wenn das aktuelle Element das Gesuchte ist, stoppe.

Andernfalls gehe zum nächsten Element.

Beispiel gesucht: 9

11	23	8	14	30	9	6	17	22	28	25	15	7	10	19

Wiederhole, beginnend beim ersten Element:

Wenn das aktuelle Element das Gesuchte ist, stoppe.

Andernfalls gehe zum nächsten Element.

Beispiel gesucht: 9

Wiederhole, beginnend beim ersten Element:

Wenn das aktuelle Element das Gesuchte ist, stoppe.

Andernfalls gehe zum nächsten Element.

Beispiel gesucht: 9

Wiederhole, beginnend beim ersten Element:

- Wenn das aktuelle Element das Gesuchte ist, stoppe.
- Andernfalls gehe zum nächsten Element.

Beispiel gesucht: 9

Wiederhole, beginnend beim ersten Element:

- Wenn das aktuelle Element das Gesuchte ist, stoppe.
- Andernfalls gehe zum nächsten Element.

Beispiel gesucht: 9

Wiederhole, beginnend beim ersten Element:

- Wenn das aktuelle Element das Gesuchte ist, stoppe.
- Andernfalls gehe zum nächsten Element.

Beispiel gesucht: 9

Wiederhole, beginnend beim ersten Element:

- Wenn das aktuelle Element das Gesuchte ist, stoppe.
- Andernfalls gehe zum nächsten Element.

Beispiel gesucht: 50

11	23	8	14	30	9	6	17	22	28	25	15	7	10	19

Wiederhole, beginnend beim ersten Element:

Wenn das aktuelle Element das Gesuchte ist, stoppe.

Andernfalls gehe zum nächsten Element.

Beispiel gesucht: 50

11	23	8	14	30	9	6	17	22	28	25	15	7	10	19

Wiederhole, beginnend beim ersten Element:

Wenn das aktuelle Element das Gesuchte ist, stoppe.

Andernfalls gehe zum nächsten Element.

Beispiel gesucht: 50

11	23	8	14	30	9	6	17	22	28	25	15	7	10	19

Wiederhole, beginnend beim ersten Element:

Wenn das aktuelle Element das Gesuchte ist, stoppe.

Andernfalls gehe zum nächsten Element.

Beispiel gesucht: 50

Wiederhole, beginnend beim ersten Element:

- Wenn das aktuelle Element das Gesuchte ist, stoppe.
- Andernfalls gehe zum nächsten Element.

Beispiel gesucht: 50

11	23	8	14	30	9	6	17	22	28	25	15	7	10	19

Wiederhole, beginnend beim ersten Element:

- Wenn das aktuelle Element das Gesuchte ist, stoppe.
- Andernfalls gehe zum nächsten Element.

Beispiel gesucht: 50

Wiederhole, beginnend beim ersten Element:

- Wenn das aktuelle Element das Gesuchte ist, stoppe.
- Andernfalls gehe zum nächsten Element.

Beispiel gesucht: 50

Wiederhole, beginnend beim ersten Element:

- Wenn das aktuelle Element das Gesuchte ist, stoppe.
- Andernfalls gehe zum nächsten Element.

Beispiel gesucht: 50

Wiederhole, beginnend beim ersten Element:

- Wenn das aktuelle Element das Gesuchte ist, stoppe.
- Andernfalls gehe zum nächsten Element.

Beispiel gesucht: 50

11	23	8	14	30	9	6	17	22	28	25	15	7	10	19

Wiederhole, beginnend beim ersten Element:

- Wenn das aktuelle Element das Gesuchte ist, stoppe.
- Andernfalls gehe zum nächsten Element.

Beispiel gesucht: 50

11	23	8	14	30	9	6	17	22	28	25	15	7	10	19

Wiederhole, beginnend beim ersten Element:

Wenn das aktuelle Element das Gesuchte ist, stoppe.

Andernfalls gehe zum nächsten Element.

Beispiel gesucht: 50

11	23	8	14	30	9	6	17	22	28	25	15	7	10	19

Wiederhole, beginnend beim ersten Element:

- Wenn das aktuelle Element das Gesuchte ist, stoppe.
- Andernfalls gehe zum nächsten Element.

Beispiel gesucht: 50

11	23	8	14	30	9	6	17	22	28	25	15	7	10	19

Wiederhole, beginnend beim ersten Element:

- Wenn das aktuelle Element das Gesuchte ist, stoppe.
- Andernfalls gehe zum nächsten Element.

Beispiel gesucht: 50

11	23	8	14	30	9	6	17	22	28	25	15	7	10	19

Wiederhole, beginnend beim ersten Element:

Wenn das aktuelle Element das Gesuchte ist, stoppe.

Andernfalls gehe zum nächsten Element.

Beispiel gesucht: 50

Wiederhole, beginnend beim ersten Element:

- Wenn das aktuelle Element das Gesuchte ist, stoppe.
- Andernfalls gehe zum nächsten Element.

Beispiel gesucht: 50

11	23	8	14	30	9	6	17	22	28	25	15	7	10	19

Wiederhole, beginnend beim ersten Element:

Wenn das aktuelle Element das Gesuchte ist, stoppe.

Andernfalls gehe zum nächsten Element.

Beispiel gesucht: 50

11	23	8	14	30	9	6	17	22	28	25	15	7	10	19

Wiederhole, beginnend beim ersten Element:

Wenn das aktuelle Element das Gesuchte ist, stoppe.

Andernfalls gehe zum nächsten Element.

Fälle

Worst Case:
Gesuchtes Element kommt gar nicht vor.
Dann müssen wir alle Elemente prüfen.

Best Case:
Gesuchtes Element ist das allererste Element. Dann benötigen wir nur einen Vergleich.

Laufzeit-komplexität

$O(n)$

$\Omega(1)$

Vor- und Nachteile

Vorteile:

Einfach zu verstehen und zu implementieren

Funktioniert mit unsortierten Arrays

Nachteile:

Ineffizient für große Datenmengen

Durchschnittliche Laufzeit wächst linear mit der Arraygröße

EXTRAS IN 3 MINUTEN

FRAGEN – ANTWORTEN – RÄTSEL
UND KURZE ZUSAMMENFASSUNG

```c
#include <cs50.h>
#include <stdio.h>
#define ARRAY_SIZE 10000

bool linear_search(int arr[], int x)
{
    for (int i = 0; i < ARRAY_SIZE; i++)
    {
        if (arr[i] == x)
            return true;
    }
    return false;
}
```

```c
int main()
{
    int numbers[ARRAY_SIZE];
    for (int i = 0; i < 63; i++)
    {
        numbers[i] = i + 1;
    }
    if (linear_search(numbers, 64))
    {
        printf("gefunden!\n");
    }
    return 0;
}
```

```c
bool linear_search(int arr[], int size, int x)
{
    for (int i = 0; i < size; i++)
    {
        if (arr[i] == x)
            return true;
    }
    return false;
}


// Im Hauptprogramm:
if (linear_search(numbers, 63, 64))
{
    printf("gefunden!\n");
}
```

Lineare Suche: Element für Element von links nach rechts durchgehen

Array-Größe als Parameter statt global definieren

Komplexität:
O(n) im Worst Case,
Ω(1) im Best Case

Unterschied zwischen deklarierter und genutzter Array-Größe beachten

Einfach zu implementieren, aber ineffizient bei großen Datenmengen

Vorsicht bei nicht-initialisierten Arrays in C:Werte sind undefiniert