Überblick

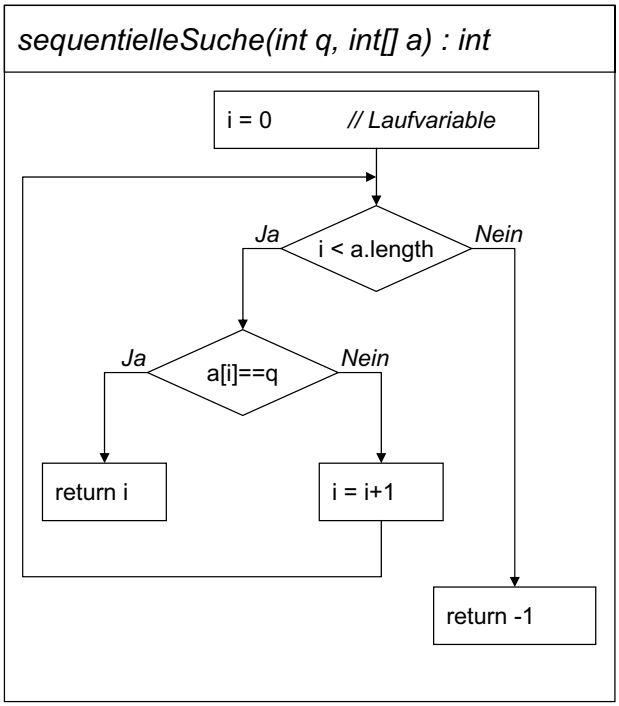
4. Imperative Programmierung
4.1 Grunddatentypen und Ausdrücke
4.2 Imperative Variablenbehandlung
4.3 Anweisungen, Blöcke und Gültigkeitsbereiche
4.4 Klassenvariablen
4.5 Reihungen
4.6 (Statische) Methoden
4.7 Kontrollstrukturen
4.8 ... putting the pieces together ...

Beispiel: Suche in einem Array

- Aufgabe:
  - Sei A ein Array der Länge n mit Werten aus $\mathbb{N}$ und $q \in \mathbb{N}$.
  - Keiner der Werte kommt mehrfach vor.
  - Suche q in A und gib, falls die Suche erfolgreich ist, die Position i mit $A[i] = q$ aus. Falls die Suche erfolglos ist, gib den Wert -1 aus.
- Einfachste Lösung: *Sequentielle Suche*
  - Durchlaufe A von Position $i = 0, \ldots, n-1$.
  - Falls $A[i] = q$, gib i aus und beende den Durchlauf.
  - Falls q nicht gefunden wird, gib -1 aus.

Beispiel: Suche in einem Array

- Algorithmus:



Beispiel: Suche in einem Array

- Java-Methode:

```
public static int sequentielleSuche(int q, int[] a)
{
  for(int i = 0; i < a.length; i++)
  {
    if(a[i]==q)
    {
      return i;
    }
  }
  return -1;
}
```

Beispiel: Suche in einem Array

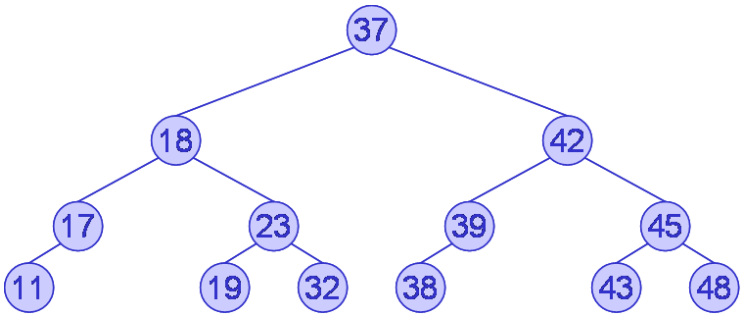
- Anzahl der Vergleiche (= Analyse der Laufzeit):
  - Erfolgreiche Suche: n Vergleiche maximal, $n/2$ im Durchschnitt.
  - Erfolglose Suche: n Vergleiche.
- Falls das Array sortiert ist, funktioniert auch folgende Lösung: *Binäre Suche*
  - Betrachte den Eintrag $A[i]$ mit $i = n/2$ in der Mitte des Arrays
    - Falls $A[i] = q$, gib i aus und beende die Suche.
    - Falls $A[i] > q$, suche in der linken Hälfte von A weiter.
    - Falls $A[i] < q$, suche in der rechten Hälfte von A weiter.
  - In der jeweiligen Hälfte wird ebenfalls mit binärer Suche gesucht.
  - Falls die neue Hälfte des Arrays leer ist, gib -1 aus.

Beispiel: Suche in einem Array

Beispiel: A:

11	17	18	19	23	32	37	38	39	42	43	45	48

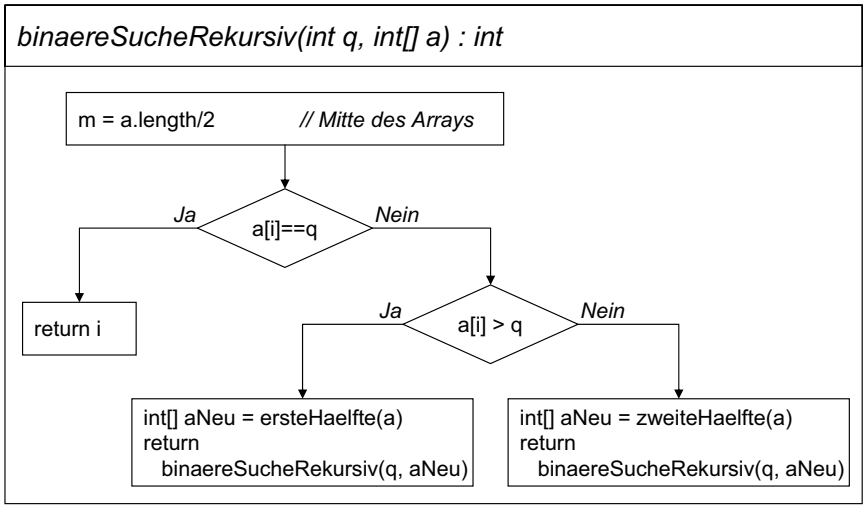
- Entscheidungsbaum:



- Analyse der Laufzeit: Maximale Anzahl von Vergleichen entspricht Höhe h des Entscheidungsbaums: $h = \lceil \log_2(n+1) \rceil$.
- Vergleich der Verfahren:
  - $n = 1.000$:
    Sequentielle Suche: 1.000 Vergleiche – Binäre Suche: 10 Vergleiche
  - $n = 1.000.000$:
    Sequentielle Suche: 1.000.000 Vergleiche – Binäre Suche: 20 Vergleiche

Beispiel: Suche in einem Array

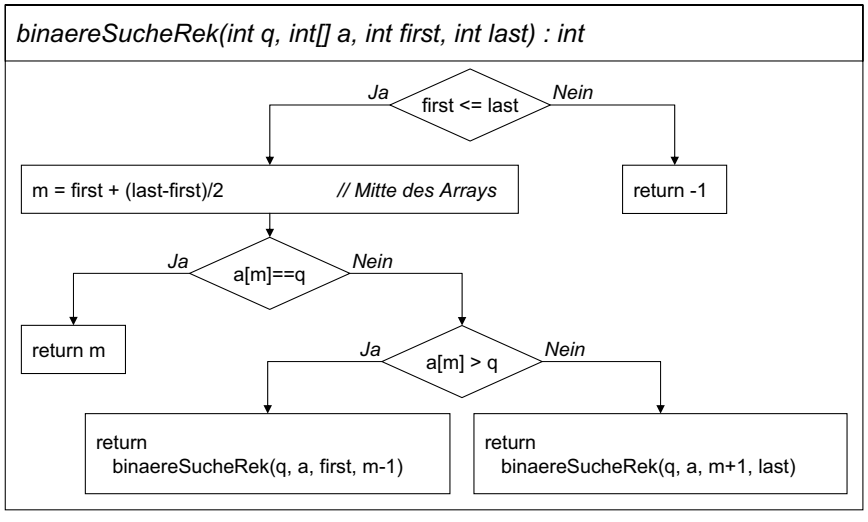
Rekursiver Algorithmus:

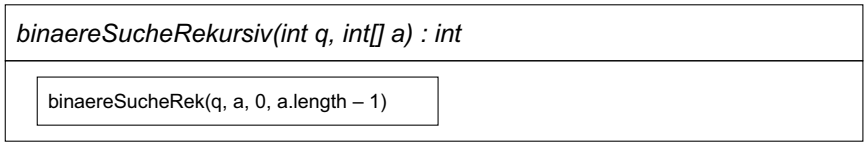


Probleme?

Beispiel: Suche in einem Array

Alternativer rekursiver Algorithmus mit nur einem Array:





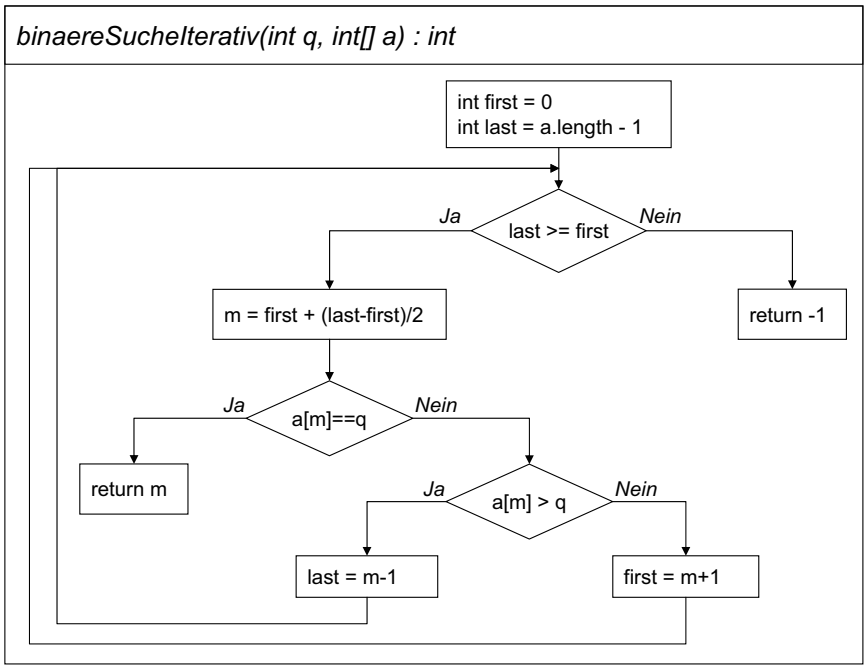
Beispiel: Suche in einem Array

Alternativer rekursiver Algorithmus in Java:

```
public static int binaereSucheRek (int q, int[] a, int first, int last)
{
  if (first > last)
  {
    return -1;  // leeres Array
  }
  int m = first + (last-first) / 2;
  if (q == a[m])
  {
    return m;   // q gefunden
  }
  else if (a[m] > q)
  {
    return binaereSucheRek(q, a, first, m-1);
  }
  else /* a[m] < q */
  {
    return binaereSucheRek(q, a, m+1, last);
  }
}
public static int binaereSucheRekursiv (int q, int[] a)
{
  return binaereSucheRek (q, a, 0, a.length - 1);
}
```


Beispiel: Suche in einem Array

Alternativer *iterativer* Algorithmus mit nur einem Array:



Beispiel: Suche in einem Array

Iterativer Algorithmus in Java:

```
public static int binaereSucheIterativ (int q, int[] a)
{
  int first = 0;
  int last = a.length - 1;
  while (last >= first)
  {
    int m = (last + first) / 2;
    if (q == a[m])
    {
      return m;
    }
    else if (q < a[m])
    {
      last = m - 1;
    }
    else /*q > a[m]*/
    {
      first = m + 1;
    }
  }
  return -1;  // not found
}
```


Zusammenfassung

Sie kennen jetzt die Grundkonzepte imperativer Programmierung:

- Grunddatentypen und ihre typischen Operationen als elementare Bestandteile der meisten Programmiersprachen,
- das Konzept von (lokalen und globalen) Variablen und Konstanten (und deren Unterschied) zur sequentiellen Verarbeitung von Information in imperativen Programmen,
- das Konzept von (sequentiellen) Anweisungen,
- Blöcke als Zusammenfassung mehrerer Anweisungen,

...

Zusammenfassung

...

- die Gültigkeitsbereiche verschiedener Arten von Variablen,
- Arrays (Reihungen) als grundlegende Datenstruktur zur Verwaltung einer Menge gleichartiger Werte im imperativen Programmierparadigma,
- Methoden (Prozeduren) als Mittel zur Abstrahierung von Algorithmen und einzelnen Berechnungsschritten sowie (durch die Verwendung formaler Parameter) von den konkreten Daten,
- typische imperative Kontrollstrukturen zur Implementierung von bedingten, kontrollierten, iterierten Anweisungen

und können somit im Prinzip einen Algorithmus imperativ in Java implementieren.