Hochschule Esslingen
University of Applied Sciences

Objektorientierte Programmierung mit C++:

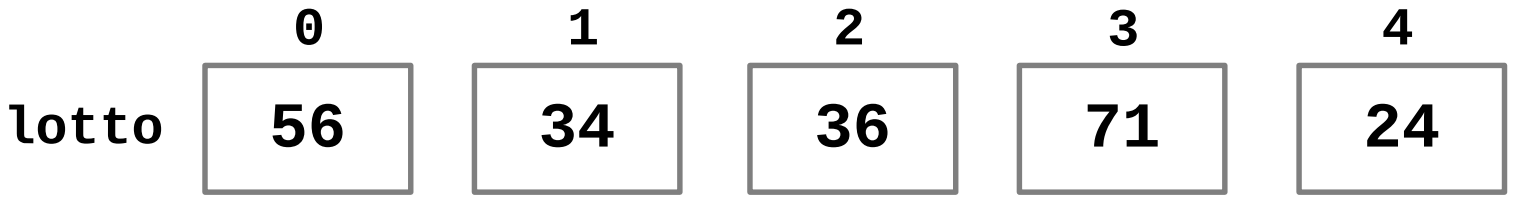
Build-in Arrays & dynamische Arrays

A. Freymann

Fakultät Informationstechnik, Hochschule Esslingen

Build-in Array

- Ein Array ist eine Kombination Variablen gleichen Typs.
- Die Elemente des Arrays werden über ihren Index angesprochen.
- Ein Array muss vor dem Benutzen deklariert werden
  - **`type name [elements];`**
  - ***`type: primitiver / komplexer Datentyp`***
  - ***`elements: Länge des Arrays`***
  - ***`int lotto [5] = {56, 34, 36, 71, 24}`***



Build-in Array – Deklaration & Definition

- Deklaration
  - `int foo [5];`
- Definition / Initialisierung ohne Werte
  - `int foo [5] = {};`
- Definition / Initialisierung mit Werte und Länge
  - `int foo [5] = { 16, 2, 77, 40, 12071 };`
- Definition / Initialisierung nur mit Werte ohne Länge
  - `int foo [] = { 16, 2, 77, 40, 12071 };`
- Auch möglich / *universal initialization*:
  - `int foo[] { 10, 20, 30 };`

Build-in Array - Auslesen

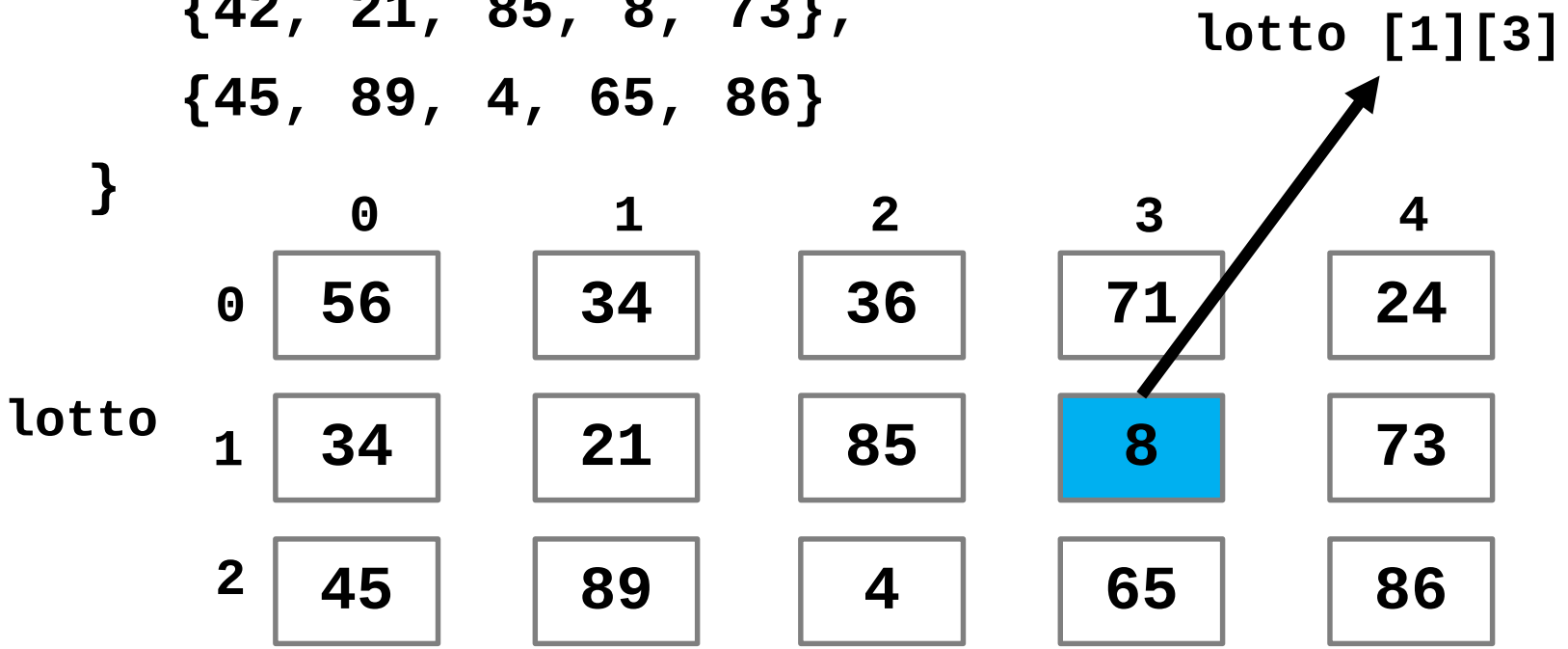
- Möglichkeiten des Auslesens des Arrays
- Über den Index mittels einer Schleife

```
#include <iostream>
using namespace std;


int foo[] = { 16, 2, 77, 40, 12071 };
int n, result = 0;

int main(){
        for (n = 0; n<5; ++n){
                result += foo[n];
        }
        cout << result;
        return 0;
}
```

Build-in Array - Multidimensional

- Ein multidimensionales Array mehrere Dimensionen

```
int lotto [3][5];          //Ohne Intitialisierung
int lotto [3][5] = {       //Mit Intitialisierung
   {56, 35, 36, 71, 24},
   {42, 21, 85, 8, 73},
   {45, 89, 4, 65, 86}
}
```



Build-in Array – Multidimensional - Auslesen

- Möglichkeiten des Iterierens eines Multidimensionales Arrays mittels Doppelschleife

```
const int WIDTH = 5;
const int HEIGHT = 3;

int jimmy[HEIGHT][WIDTH] = {};
int n, m;
int main() {
    for (n = 0; n<HEIGHT; n++){
        for (m = 0; m<WIDTH; m++) {
                jimmy[n][m] = (n + 1)*(m + 1);
        }
    }
}
```

Build-in Array – Dynamisch

- Was tun, wenn die Grösse des Arrays erst während dem Programmablauf bekannt wird?
- Lösung: Einsatz eines Pointers mit **new**
  - 1. Pointer, der auf ein Datentyp zeigt, der im Array gespeichert wird
    - `int * lotto;`
  - 2. Mit **new** Speicher direkt mit Angabe der Grösse des Arrays allokieren
    - `lotto = new int[10];`
  - 3. Mit dem zurückgelieferten Pointer kann das Array normal benutzt werden.
    - `lotto[0] = 2778; // cout << lotto[0] << endl;`

Container Array – Deklaration & Definition

- Sie bieten viele Operationen für die Verwaltung von Elementen (z.B. Hinzufügen oder Auslesen)
- Container bedeutet in diesem Zusammenhang, dass dieser die Daten innerhalb des Containers (arrays) im Speicher verwaltet.

Container Array – Deklaration & Definition

- **`#include array`**
  - **`std::array<datatype,array_size> array_name;`**
- Deklaration
  - **`std::array<int, 5> n;`**
- Definition / Initialisierung mit Werte und Länge
  - **`std::array<int, 5> n = {1, 2, 3, 4, 5};`**
- Auch möglich / *universal initialization*:
  - ***`std::array<int, 5> n { {1, 2, 3, 4, 5} };`***

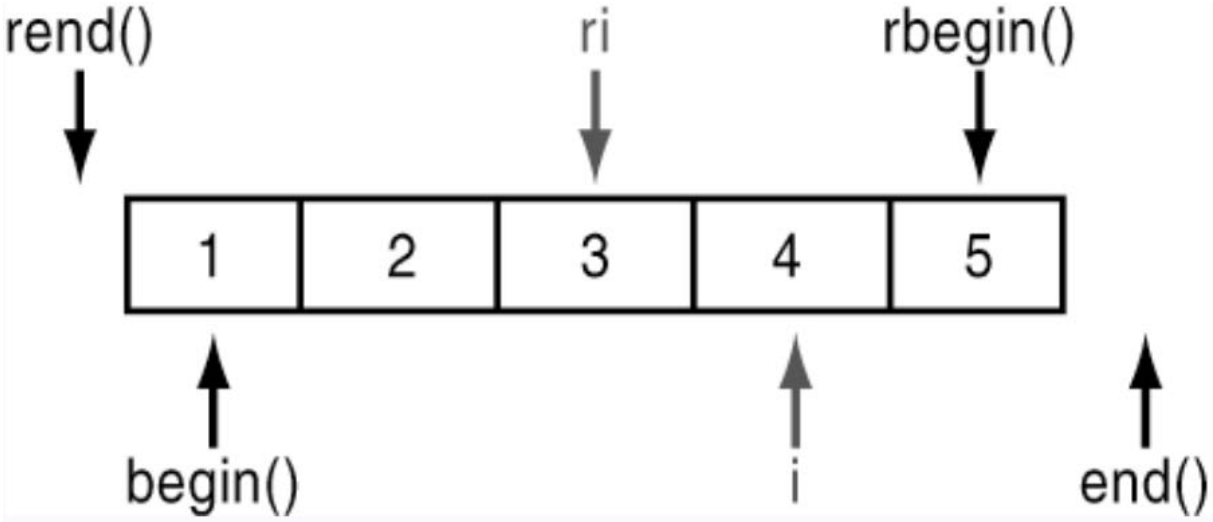
Container Array

- Array ist ein Objekt und hat viele Methoden
  - **size()**
  - **at()**
  - **back()**
  - **empty()**
  - **max_size()**
  - **fill()**
  - **iterators**

```
#include <iostream>
#include <array>
using namespace std;

int main() {
        array<int, 3> myarray{ 10,20,30 };
        for (int i = 0; i<myarray.size(); ++i{
                {++myarray.at(i);}
        for (int elem : myarray)
                {cout << elem << '\n';}
        myarray.empty();
}
```

Container Array - Iteratoren

- Ein Iterator zeigt auf ein Element innerhalb eines Arrays
  Es gibt verschiedene Methoden um einen Iterator zu bekommen
- Wird ein Iterator benutzt zeigt er nachfolgend auf das nächste Element (Iterator = er itertiert)



http://www.drdobbs.com/cpp/three-guidelines-for-effective-iterator/184401406

Container Array - Iteratoren

- **`array::begin`**
  - Iterator zeigt auf das erste Element
- **`array::end`**
  - Iterator zeigt auf das letzte Element
- **`array::rbegin`**
  - Iterator zeigt reverse auf das erste Element
- **`array::rend`**
  - Iterator zeigt reverse auf das letzte Element

Container Array - Iteratoren

```
#include <iostream>
#include <array>

int main(){
        std::array<int, 5> myarray = { 2, 16, 77, 34, 50};

        std::cout << "myarray contains:";
        for (auto it = myarray.begin(); it != myarray.end(); ++it){
                std::cout << ' ' << *it;
                std::cout << '\n';
        }
        return 0;
}
```