Exercise Week 08

GianAndrea Müller
mailto:muellegi@student.ethz

April 25, 2018

Time Schedule

- 20' Repetition Pointer und Übung
- 10' Iterators
- 5' Using
- 20' Self-assessment test
- 20' Korrektur

Learning Objectives

- Verständnis der Funktionalität von Pointern
- Kenntnis von Iteratoren

Repetition Pointers

```
1  int a = 5;
2  int * aptr = &a;
3  *aptr = *aptr + 1; // a = 6;
4
5  int liste[10];
6  int * index = &liste[0]; // = liste;
7
8  int i = 0;
9  while ( index < liste + 10 ){
10   cout<<*index<<*(liste+i)<<liste[i];
11   index++; i++;
12 }
```

Zusammenfassung Pointer

Operatoren

- Neuer Pointer: `<type> * name;`
- Dereferenzierung: `*(some_pointer)`
- Referenzierung: `&some_variable`
- Arithmetik: `++some_pointer`

Nutzen

- Dynamische Speicherverwaltung (später)
- Zeiger statt Index (effizienter)
- Call/Return by Reference

Unterschied: Pointer und Referenzen

- Ein Pointer kann neu zugeordnet werden.
- Ein Pointer kann auf `NULL` zeigen.
- Die Adresse eines Pointers kann verwendet werden.
- Es gibt keine Referenzarithmetik.

Merksatz

Eine Referenz ist ein konstanter Pointer, der nicht dereferenziert werden muss.

Konstante Pointer

```
1 int i = 0;
2 int * iptr = &i;
3 const int * icptr = &i;
4 int const * ic2ptr = &i;
5 int * const iptrc = &i;
6 const int * const iptrc2 = &i;
```

Konstante Pointer

Lies von rechts nach links! Lies $*$ als 'Pointer auf'.

Lesen von Pointern

- `int * iptr = &i;`
  iptr: Pointer auf int.
- `const int* icptr = &i;`
  icptr: Pointer auf const int.
- `int const * ic2ptr = &i;`
  icptr2: Pointer auf const int.
- `int * const iptrc = &i;`
  iptrc: const Pointer auf int.
- `const int * const icptrc = &i;`
  icptrc: const Pointer auf const int.

Exercise 8_1 5'

Write a function that outputs part of an int array. You are not allowed to pass the number of elements to be printed.

```
1 void print_part_of_array(arg1, arg2, arg3)
    {
2   //print elements in defined range
3 }
```

Solution 8_1

```
1 //PRE: beginning and end have to enclose a
      valid part of an array
2 //POST: The part of the array is printed
3 void print_part_of_array(int * beginning,
     int * end){
4   for(int * current = beginning; current
       <= end; current++){
5     cout<<*current<<" ";
6   }
7   cout<<endl;
8 }
```

Iterators

```
1  #include <vector>
2
3  vector<int> vec = {8,3,1,4,6,9};
4  //C++ 11 syntax! Choose C++14 on codeboard
       .io!
5
6  vector<int>::iterator itb = vec.begin();
7  vector<int>::iterator ite = vec.end();
8
9  cout<<*itb<<" "<<*ite<<endl;
```

Iterators

Iterators

```
1  #include <vector>
2
3  // int a[] = {8,3,1,4,6,9};
4  // for (int *p = a; p!= a + 6; ++p)
5  //    cout<<*p;
6
7  vector<int> vec = {8,3,1,4,6,9};
8  //C++ 11 syntax! Choose C++14 on codeboard
       .io!
9
10 for(vector<int>::iterator it=vec.begin();
       it<vec.end(); ++it)
11    cout<<*it;
```

Using

```
1 using intvec = std::vector<int>;
2 using intvecit = std::vector<int>::
    iterator;
3
4 intvec vec = {1,2,3,4,5,6,7};
5
6 for (intvecit it = vec.begin(); it < vec.
    end(); ++it)
7   std::cout<< *it;
```