Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Lecturer: Prof. B. Gärtner

Informatik für Mathematiker und Physiker HS16

Exercise Sheet 9

Submission deadline: 15:15 - Tuesday 22th November, 2016
Course URL: http://lec.inf.ethz.ch/ifmp/2016/

Short Summary

	Array	Vector
Array/Vector Initialization	`int my_arr[] = {1,2,3};`	`// 1 2 3` `std::vector<int> my_vec (3, 1);` `// 1 1 1`
Iterator to Begin	`int* ptr = my_arr;`	`std::vector<int>::iterator itr = my_vec.begin();`
const	`const int* cptr = my_arr;`	`std::vector<int>::const_iterator citr = my_vec.begin();`
Iterator to Past-the-End	`int* ptr = my_arr + 3;`	`std::vector<int>::iterator itr = my_vec.end();`
const	`const int* cptr = my_arr + 3;`	`std::vector<int>::const_iterator citr = my_vec.end();`
to Next Elt	`++ptr`	`++itr`
to Previous Elt	`--ptr`	`--itr`
Distance	`ptr1 - ptr2`	`itr1 - itr2`
Comparisons	`ptr1 < ptr2`	`itr1 < itr2`
	`ptr1 != ptr2`	`itr1 != itr2`
	…	…

Assignment 1 – Understanding Pointers (4 points)

a) What does the following code output?

```
int a[] = {9, 8, -5};
int* p = &a[2];
std::cout << (p - a) << "\n";
```

b) What does the following code output?

```
unsigned int a[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
for (unsigned int* p = a; p < a + 10; ++p)
  std::cout << *p << " ";
```

c) What does the following code output? [Skript-Aufgabe 115a]

```
int a[] = {5, 6, 2, 3, 1, 4, 0};
int* p = a;
do {
  std::cout << *p << " ";
  p = a + *p;
} while (p != a);
```

d) What does the following code output? [based on Exam Summer 2016, ex.4]

```
char text[] = {'I', 'f', 'm', 'm', 'p'};
for (char* it = text; it < text + 5; ++it)
  *it = *it-1;
for (char* it = text; it < text + 5; ++it)
  std::cout << *it;
```

This exercise can be handed in via Codeboard! However, if you prefer, you can also hand in your solutions on paper as before.

Submission: https://codeboard.ethz.ch/ifmp16E9T1

Assignment 2 - Using Pointers and Iterators (4 points)

For technical reasons, the output shall include a star before the first and after the last `int` (separated from the actual output by at least one space) where indicated in the example boxes.

a) Write a program `every_other.cpp` which inputs 10 `int`s from the user and then outputs every second element (starting from index `0`) using pointers or iterators. **You are not allowed to use `[]` in this exercise to access elements.**

I/O-Examples (Explanation: http://lec.inf.ethz.ch/ifmp/2016/codeboard.html)

```
0 1 2 3 4 5 6 7 8 9
* 0 2 4 6 8 *
```

Submission: https://codeboard.ethz.ch/ifmp16E9T2a

b) Write a program `reverse.cpp` which inputs 10 `int`s from the user and then outputs these in reverse order using pointers or iterators. **You are not allowed to use `[]` in this exercise to access elements.**

I/O-Examples (Explanation: http://lec.inf.ethz.ch/ifmp/2016/codeboard.html)

```
0 1 2 3 4 5 6 7 8 9
* 9 8 7 6 5 4 3 2 1 0 *
```

Submission: https://codeboard.ethz.ch/ifmp16E9T2b

c) Implement a function

```
// PRE: [begin, end) is a valid non-empty range.
// POST: returns the average of the elements in [begin, end)
double average (const double* begin, const double* end);
```

to compute the average of the elements in an array with elements of type `double`. Use this function in a program `average.cpp` to compute and output the average of 20 numbers that are input by the user. **You are not allowed to use `[]` to access elements of the array in this exercise.**

I/O-Examples (Explanation: http://lec.inf.ethz.ch/ifmp/2016/codeboard.html)

```
1 2 3 4 5 6 7 8 9 10 1 2 3 4 5 6 7 8 9 10
5.5
```

Submission: https://codeboard.ethz.ch/ifmp16E9T2c

d) Implement a function

```
// PRE: [begin, end) is a valid range
// POST: fills the range with the following pattern
//         1 2 2 3 3 3 4 4 4 4 ...
//       The pattern is cut off as soon as the range is full.
void n_times_n_pattern (Vit begin, Vit end);
```

to fill the range with the specified pattern.[1] Use this function in a program `pattern.cpp` which inputs a number `n` from the user and then outputs the pattern for a range of length `n`. **You are not allowed to use `[]` in this exercise to access elements.** Furthermore, `Vit` is given according to the following `typedef`:

```
typedef std::vector<int>::iterator Vit;
```

I/O-Examples (Explanation: http://lec.inf.ethz.ch/ifmp/2016/codeboard.html)

```
16
* 1 2 2 3 3 3 4 4 4 4 5 5 5 5 5 6 *
```

Submission: https://codeboard.ethz.ch/ifmp16E9T2d

Assignment 3 – Sorting Ranges (4 points)

[Skript-Aufgabe 112]

a) Implement the following function[1]

```
// PRE: [begin, end) is a valid range
// POST: the elements in [begin, end) are in ascending order
void sort (Vit begin, Vit end);
```

For this implementation you may not use `std::sort`. And `Vit` is defined according to the following:

```
typedef std::vector<int>::iterator Vit;
```

b) Write a program `sort_range.cpp` in which you use your function from above to sort input `int`-values into ascending order and then output the sorted numbers. The user first inputs the number of values and afterwards the actual values. For technical reasons, the output shall include a star before the first and after the last `int` (separated from the actual output by at least one space).

[1] In programming contexts we usually include the empty range when talking about valid ranges unless mentioned otherwise.

I/O-Examples (Explanation: http://lec.inf.ethz.ch/ifmp/2016/codeboard.html)

```
5
3 2 5 3 4
* 2 3 3 4 5 *
```

Submission: https://codeboard.ethz.ch/ifmp16E9T3

Assignment 4 – Interpreting Recursive Calls (4 points)

[parts from: Skript-Aufgabe 122]

Find PRE- and POST-conditions for the following recursive functions!

a) Find PRE- and POST-conditions for

```
bool f (const int n) {
  if (n == 0) return false;
  return !f(n-1);
}
```

b) Find PRE- and POST-conditions for

```
void g (const int n) {
  if (n == 0) {
    std::cout << "*";
    return;
  }
  g(n-1);
  g(n-1);
}
```

c) Find PRE- and POST-conditions for

```
void h (const int* begin, const int* end, int t) {
  if (begin == end)
    std::cout << t << "\n";
  else {
    h (begin+1, end, t);
    h (begin+1, end, t + *begin);
  }
}
```

This Codeboard-program might give you some hints.

This exercise can be handed in via Codeboard! However, if you prefer, you can also hand in your solutions on paper as before.

Submission: https://codeboard.ethz.ch/ifmp16E9T4

Challenge – Cellular Automata (8 points)

The shorter programs above may be just the right thing to learn how to program, but you are probably in for something larger... Interested? Then take this route.