Exercises for the Lecture Series

"Object-Oriented Programming for Scientific Computing"

Dr. Ole Klein

ole.klein@iwr.uni−heidelberg.de

To be handed in on 16. 05. 2017 before the lecture

EXERCISE 1   POINTERS

Let `i` have the type `int`, and `p` the type `int *`. Which of the following expressions are correct, which are incorrect? Also list the types of the correct ones. Answer without taking concrete values for `i` and `p` into account.

- `i + 1`
- `*p`
- `*p + 3`
- `&i == p`
- `i == *p`

- `&p`
- `p + 1`
- `&p == i`
- `**(&p)`
- `*p + i > i`

2 Points

EXERCISE 2   DESTRUCTOR

Which of the following statements are true? The destructor of a class `C` is accountable for. . .

- . . . cleaning up all objects of the class `C`.
- . . . cleaning up objects of the class `C` on the heap.
- . . . cleaning up all components of objects of the class `C`.
- . . . cleaning up components of objects of the class `C` that are on the heap.
- `delete` is just a special way of calling the destructor: let `x` be of type `C*`, then `delete x;` is the same as `(*x).~C()`

Explain your reasoning, since the correctness of the statements is at least partially subject to interpretation.   *2 Points*

EXERCISE 3   NEW & DELETE

1. Why is:

```
int* get_int1 ()
{
  int* p;
  p = new int;
  return p;
}
```

a reasonable method to create a reference to a new `int` variable, while in contrast

```
int* get_int2 ()
{
  int i;
  int* p = &i;
  return p;
}
```

is completely unsuitable?

2. Assume the following definitions and commands have been executed:

```
int* p;
p = new int;
*p = 17;
```

What happens when

```
p = 0;
delete p;
```

or

```
delete p;
p = 0;
```

is executed afterwards? Which of the snippets is sensible, which isn't, and why?

4 Points

EXERCISE 4   LINKED LIST

Using the simple example of a chained list we will practice the interaction of constructors, destructors and pointers.

We want to program a linked list, which can store an arbitrary number of values of type `int`. A list consists of an object of class `List`, which refers to a sequence of objects of class `Node`. The list elements are stored in a component `int value` within each node and a pointer `Node* next` points to the next node. The end of the list is designated by the pointer `next` having the value `0`.

1. What is special about a pointer having the value `0`?

2. Implement the class `Node`. Make sure that all member variables are always initialized.

3. Implement the class `List` with the following methods:

```cpp
class List
{
public:
    List ();                              // create an empty list
    ~List ();                             // clean up the list and all nodes
    Node* first() const;                  // return a pointer to the first entry
    Node* next(const Node* n) const;      // return a pointer to the node after n
    void append (int i);                  // append a value to the end of the list
    void insert (Node* n, int i);         // insert a value before n
    void erase  (Node* n);                // remove n from the list
};
```

   `List` must also store the beginning of the list, where would you place it in the class declaration? The `next` pointer should be `private` to ensure that the list structure isn't accidentally changed outside of class `List`. The member `value` is `public` to allow read and write access from outside the class. The line
   friend class `List;`
   has to be inserted into the declaration of the class `Node` to give the `List` class access to the `next` pointer. Additionally make sure that the destructor deletes all allocated `Node` objects.

4. Test your implementation with the following program:

```cpp
int main ()
{
    List list;
    list.append(2);
    list.append(3);
    list.insert(list.first(), 1);
    for (Node* n = list.first(); n != 0; n = list.next(n))
        std::cout << n->value << std::endl;
    return 0;
}
```

5. What happens if one copies the list? And what happens if both lists are deleted?

```cpp
int main ()
{
    List list;
    list.append(2);
...
    List list2 = list;
    return 0;
}
```

12 Points