

Course Programming in C^{++}

Exercise List 4

Deadline: 31.03.2017

Topic of this exercise are the use of defined operators, and the use of private fields.

1. Extend the **string** class, that can be downloaded with the exercise, with the following operators:

```
char operator [] ( size_t i ) const;
char& operator [] ( size_t i );
```

If you want these operators to check bounds, throw a `std::runtime_error`. Don't return a nonsense value. It is also possible to check nothing.

2. Add operators

```
void operator += ( char c ); // Append character at the end.
void operator += ( const string& s );
```

Be careful with self assignment for the second operator! Test both with **valgrind**. The first operator is inefficient if you repeatedly add one character, because the string gets reallocated all the time. In class **stack** of List 3, this problem is solved by always doubling the size of the stack. You don't need to worry about efficiency in this task.

3. Extend the **string** class with one of
`string operator + (const string& s1, const string& s2)`, or
`string operator + (string s1, const string& s2)`.
You can make use of `operator +=`.
4. Check for the absence of memory leaks, using code that contains the addition operator, the new assignment operators. Use **valgrind**. Make sure that self assignment `s+=s` is included in the tests.
5. Add the following to the string class:

```

using iterator = char* ;
using const_iterator = const char* ;
const_iterator begin( ) const { return p; }
const_iterator end( ) const { return p + len; }
iterator begin( ) { return p; }
iterator end( ) { return p + len; }

```

Write, in your main program:

```

s = "this is a string";
std::cout << s << "\n";
for( char& c : s )
    c = toupper(c);
std::cout << s << "\n";

```

It works because the range-for calls **begin()** and **end()**

Also rewrite `std::ostream& operator << (std::ostream& out, const string& s)` using **range-for**.

6. Add the following operators:

```

bool operator == ( const string& s1, const string& s2 );
bool operator != ( const string& s1, const string& s2 );
bool operator < ( const string& s1, const string& s2 );
bool operator > ( const string& s1, const string& s2 );
bool operator <= ( const string& s1, const string& s2 );
bool operator >= ( const string& s1, const string& s2 );

```

to **string**. Try to avoid writing too much repeated code.