Programowanie Objektowe (angielsku)

Due: 24 March 2009

The purpose of this exercise is that you understand pointers, allocation and deallocation even better. The tasks are not so easy, so probably you will have two weeks to make them. Use recursion whenever possible, because it is nearly always easier.

Consider the following declaration:

```
struct digit
{
   char n;
   digit* next;
};
```

It can be used for building big integers. It will be not efficient, but we don't care about that. I advice that you put the least significant digit in front. This means that the number 102 will be represented by the list $[\prime 2\prime, \prime 0\prime, \prime 1\prime]$.

1. Write a function `digit* buildbigint( unsigned int x )`, that constructs a bigint from an unsigned integer.
2. Write a function `addto( const digit* , digit * )` that adds the first bigint to the second bigint.
3. Write a function `printbigint( const digit* )` . (The easiest way to do this, is by a recursive procedure)
4. Write a function `subtractfrom( const digit* , digit * )` that subtracts the first number from the second number. When the first bigint is bigger than the second bigint, the function may simply construct 0. You will need an additional function that removes leading zeroes.
5. Write a multiplication function `multiply( const digit* , const digit* , digit* );`
6. Write a function `deletebigint( digit* )` that deallocates a bigint.
7. Use the previous functions to compute 69!
8. Test the previous program (that computes 69!) for memory leaks (using **top** and a **for** loop), and ensure that there is none.