Lab 3.4.5 A foretaste of system programming – obtaining the current date

Objectives

Familiarize the student with:

- making use of standard system services;
- understanding an exemplary system function's interface.

Scenario

We want to tell you a secret: we were going to ask you to write a program which would be able to count all the days you have lived from your birthday till tomorrow. It should be easy – in fact, you have all the necessary components already. It would be like building a house using factory-made blocks.

But there's one disadvantage: the program may want to ask the user about the current date. This would be a piece of nonsense – a computer asking a human for data it knows perfectly well itself. So we've changed our plans and now we want you to write a function returning the structure of previously defined type *Date*, but filled with the current date (at the moment of function invocation). Of course, the validity of that variable is limited. You should use it as quickly as possible and don't store it for a long time, as dates are likely to change. Don't freeze time in your code – ask your computer for a date every time you need it.

Here are our requirements for the function:

- its name is "today" (just like that);
- it uses no arguments at all;
- it returns a structure of type *Date*, filled with the current date;
- it should be **mute**.

You may be a bit puzzled now, but don't worry. We'll show you how to deal with the computer's calendar. It's not rocket science.

Take a look at the code below. It uses an old and traditional technique based on a set of classical functions derived from POSIX systems – a bit ugly, but efficient and portable.

```
#include <iostream>
#include <ctime>

using namespace std;

int main(void) {
 time_t t      = time(NULL);
 tm     tl     = *localtime(&t);

 cout << tl.tm_year+1900 << "-" << tl.tm_mon+1 << "-" << tl.tm_mday << endl;
 return 0;
}
```

There are several issues that require clarification:

- note the inclusion of the "ctime" header file: it contains prototypes for functions operating on dates and times; we're going to use two of them;
- the "time_t" type is used to represent time in a very weird way: it's the number of seconds elapsed since 0:00AM Jan 1, 1970. Strange, isn't it?
- the "time(NULL)" function invocation returns the current time known to your computer (if your computer is wrong on this matter, the result will be wrong too);
- the "tm" type is a structure intended to store time information in a more convenient way than the one used by "time_t"; it contains fields describing the year, month, day, etc.
- the "localtime()" function invocation converts "time_t" data into a "tm" structure and returns a **pointer** to an internal structure of a "tm" type; the structure is filled with the current time every time the function is invoked; as the function returns a pointer, we need to dereference it (*) and copy it into our private variable;
- three fields of the "tm" structure are interesting for us:
  - tm_year: stores the number of years since 1900, so you have to add 1900 to get the current year;

- tm_mon: stores the number of the current month, but we have to warn you: "0" means January (add 1 to get a usable value);
- tm_mday: stores the number of the current day in the month – fortunately, there are no further traps, so you can use it "as-is".

Now you can complete the snippet below – as it should show the current date, so we aren't able to provide any test output. Check it yourself.

```
#include <iostream>
#include <ctime>

struct Date {
 int year;
 int month;
 int day;
};

Date today(void) {

 // Insert your code here

}

int main(void) {
 Date t = today();
 cout << t.year << "-" << t.month << "-" << t.day << endl;
 return 0;
}
```