C++ Programming Methods

Assignment 1, Date Problems

Bas Terwijn <b.terwijn@uva.nl>

In this assignment we ask the user for information about his/her day of birth and process this. Here we practice using basic data types, control flow statements and functions. Write your own algorithms to get familiar with C++, do **not** use existing code or algorithms.

Age

Ask the user for his/her birth year (range 1900-2100), birth month (range 1-12), and birth day (range 1-31, or depending on month and year) and check if it is in the valid range. Design and implement the algorithms for these functions:

```
  int yearsOld(int currentYear,int currentMonth,int currentDay,
               int birthYear,int birthMonth,int birthDay);

  int monthsOld(int currentYear,int currentMonth,int currentDay,
                int birthYear,int birthMonth,int birthDay);
```

to calculate how many years old and separately how many months old the user is and report the result to the user. Use the following example code to get the current date:

```
#include <time.h>
#include <iostream>
using namespace std;

int main()
{
  time_t currentTime;
  time(&currentTime);
  tm* timePtr = localtime(&currentTime);
  cout<<" year:" << timePtr->tm_year+1900
      <<" month:"<< timePtr->tm_mon+1
      <<" day:"  << timePtr->tm_mday <<endl;
}
```

A user is defined to be 1 month older in the next month only when the day number is larger than the birth day (for example a person born on January 5 is a month old on February 6 through March 5). Similarly a user is defined to be 1 year older in the next year when the month number is larger than the birth month, or when the month is equal and the day number is larger than the birth day (for example a person born on January 5 1990 is a year old on February 6 1991 through February 5 1992).

Check User Input

Make user that the user enters a valid date (1990-13-32 has an invalid month and also an invalid day) by asking again when the user hasn't. Take into account that different months have different number of days and a year is a Leap Year when it is divisible by 4, but not when it is divisible by 100, except when it is divisible by 400.

Day of the Week

Also ask the user on what day of the week he/she was born by having the user select a number from a list:

```
1: Monday
2: Tuesday
.: ......
7: Sunday
```

Design and implement your own algorithm for the function:

```
  int dayOfTheWeek(int birthYear,int birthMonth,int birthDay);
```

which checks if the day of the week the user selected is consistent with the birth date provided earlier and report this to the user. For your implementation select a single reference date of which you know the day of the week (for example 1800-01-01 was a Wednesday) and base the day of the week of any date on this reference date by computing by how many days they are separated.

Testing

Use the rand() function to test your software by generating random birth dates (year, month, days) and a random day-of-the-weeks and use your dayOfTheWeek() function to see if it is correct. Compute the ratio of correct random guesses over 10000 tries, this ratio should be close to 1/7 or something is wrong. Thereby use the srand() function to seed the random generator with the current time so that different executions of your program do not produce the same pseudo random results:

```
#include <stdlib.h>
#include <time.h>
#include <iostream>
using namespace std;

int main()
{
  srand(time(NULL));
  cout<< rand()%1000 <<endl;
  cout<< rand()%1000 <<endl;
}
```

In addition use the source code in “testCode.cpp” to test your implementation for correctness. Your implementation has to pass all tests. You will probably write some tests yourself when testing your algorithms. Make a habit of writing your own tests in such a way that they can be easily run again later.

Warnings

If you compile your source code using the “g++” compiler you can add flags to let the compiler warn you about things in your source code that could be the cause of incorrect results. This can save you a lot of time. For example use this in the shell to compile the “main.cpp” file and receive warnings if the compiler finds suspicious things in your code:

```
$ g++ -pedantic -Wall -Wextra main.cpp
```

See Warning-Options for more details and many more warnings. If instead your IDE compiles your source code for you it’s worth your time to google or read the manual in order to configure your IDE to give you warnings.

Algorithm Design

Try to keep your algorithms as simple and readable as possible which helps to get them correct. This isn’t simple at all, but you will get better at this with practice. What can help is to write a first draft (possibly first in some high-level informal language you make up yourself) and to incrementally improve it by fixing problems you identify. Decompose the problem in many small functions that each solve a particular part of the problem. Test often and test different parts separately before testing them together. Start from scratch when you get stuck and try to avoid adding unnecessary complexity or special cases.

Submission

Submit your solution before the deadline to blackboard. Add to each solution:

- your name, student number and the name of the assignment
- references to the source of any algorithm or code that you did not create yourself
- operating system and compiler that was used to test the code