OPTIMIZED SOLUTION TO THE PROBLEM SIEVE OF ERATOSTHENES

Mihaela Ana Runceanu, Ecaterina Teodoroiu High College, Targu Jiu, ROMANIA
Adrian Runceanu, Constantin Brancusi University, Targu Jiu, ROMANIA

ABSTRACT: In this article we present a solution of Sieve of Eratosthenes implemented in C++ language. Our solution is integrated in application tool that we use for our students. Our application name is OPT (Online Programming Tool). This tool could be a very useful tool for teaching Computer Programming course.

KEY WORDS: Programming tool, E-learning, Learning Management System, Moodle, Massive Open Online Course

1. OPT (Online Programming Tool)

We present the user interface of the proposed learning/evaluation environment. We know that when designing a web component the usability is important. To highlight this aspect, we tried to follow some principles of design (design) of web home interfaces. The main feature of the interface that needs to be designed is to present a single visual model (layout) for all the users of that application. To design the user interface we started with the profiling activities (tasks) and use activities objects and activities actions (tasks events) [1]. Then we selected a suitable style of interaction and ultimately we created a screen aspect.

1.1 The Profiles and Activities of users of the application categories

At the beginning of the design process, has identified the following profiles actions:
- Administrator - a person responsible for the management of application configuration, such as user accounts and storage settings;
- Professor - a person responsible for a set of activities related to resource management, such as the creation of two types of resources: expository (e.g. video, PDF or HTML files) and assessment resources (programming exercises) and presentation archive resource (deposit) exercise. This class of users will also receive exercises solved by students and the automatically feedback generated by the evaluation system;
- Student - a person who browses resources and solve exercises.

We assume that users will have different profiles. On the one hand, many will be novices or for first time users, especially among students. After identifying users and usage profiles we proceeded to identify tasks that must be met on this interface. We have clearly identified the expository type resources and of evaluation so our activities objects, each with a number of actions related activities, according to user profiles. Actions on components resources include: viewing,

downloading, commenting solving and results (submission of suggestions and further explanation).

A typical learning pedagogical point of view is to assign an activity (homework) in a computer course. For example, when a student starts solving an exercise the component OPT (Online Programming Tool) automatically creates a project. A project contains the source code and the necessary files to build a program in a specified programming language. Thus it must be generated for the project to create a set of pre-set files. These files are related to the selected programming language. After the automatic creation of the project, the student reads the description of the year (the theme suggested by the teacher) and solves it in a specialized web editor (e.g. SciTE) [6]. The student must test locale the code by running tests offered by the teacher and he is encouraged to create other new tests. If the new tests are created, is performed a validation phase to verify that they meet the specifications defined by the teacher in the process of creation. The assessment report is returned by the CAE student. The student may submit repeatedly integrating feedback from CAE. At the end of this cycle, the OPT (Online Programming Tool) component reports the data solving exercise back to archive (store) the exercise is quite and transmit the results back to the LMS. [4]

1.2 Application interface

To define the application interface (screen layout) we looked for a balanced interaction style based on intuition and expressiveness. First we considered it important to use direct object type actions. Although it provides a convenient way to select objects it is not possible to associate all actions identified in the interaction basic tasks of the mouse (click, point and shoot). We used an appropriate form filling data entry for complex actions, such as searching, commenting and resolution. We defined application interface, establishing specific areas for action selection objects and task.

Figure 2 shows the structure of the user interface environment for learning computer programming with two main areas: the selection on the left and action in the middle. In the left section the user navigates through the filing structure to select tasks objects. In the action section the user performs task actions (e.g. viewing videos, solving exercises) on tasks objects selected. Secondary to this is the header, used for authentication and registration. As a general rule, all available task actions are activated, helping novice users to recognize which actions are available. However, some of these actions tasks can be executed directly over tasks objects selected without requiring additional data.

Currently, the developed system runs under Linux. For this we chose Apache Tomcat, a common platform for hosting web services. We used Java to develop this application. For this reason subsequent modification of the application is easy and adapts to any platform execution. We respected XHTML 1.0 standard for developing the main web page application. So that the proposed system is platform independent and is executable to any user in a browser.

System interface was designed to be very simple in use. When you enter the site you will see the main page with a menu on the header. There you can choose between three options: Home, Resources and Problems. Home briefly explains the proposed application and resources. In Resources you can find general information on programming systems used in competitions or training and resources on learning general programming. The proposed exercises (option Problems) are actually a list of issues with difficulty levels from easy to difficult.

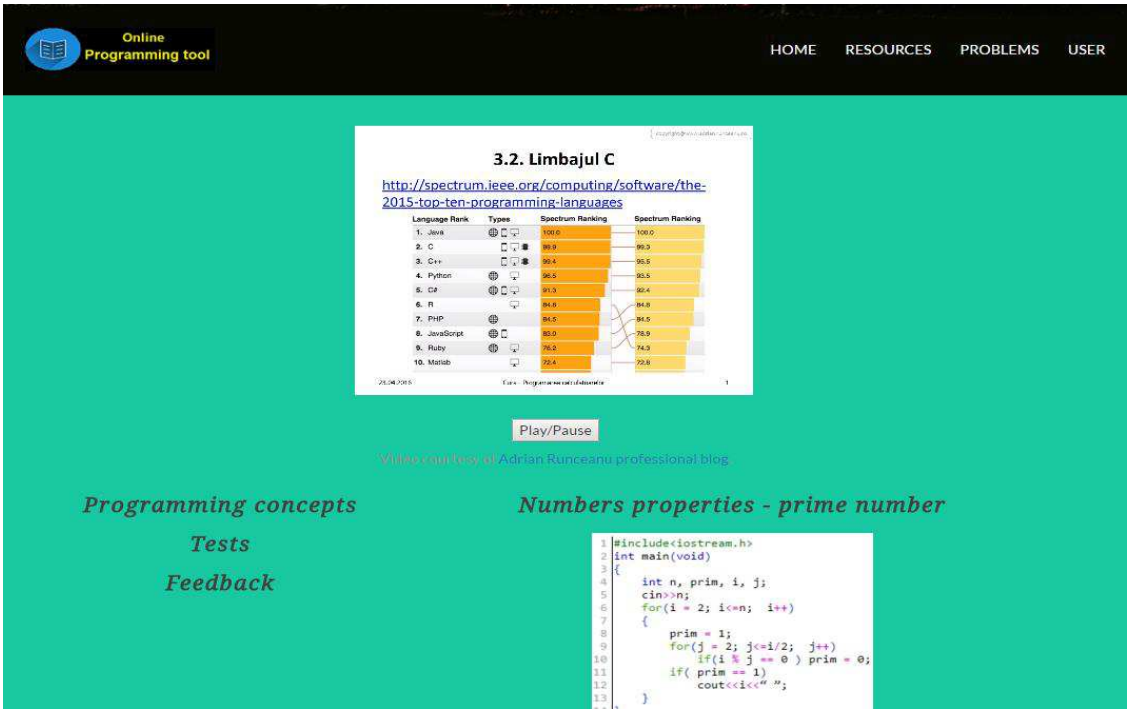


Figure 1. Application layout

Each student who was logged into the application can choose a problem, he can write code in the main area proposed as a solution and then sends the answer. After sending the solution, the application returns one of the responses: "Accepted", "Wrong Answer", "Runtime Error" or "Time Limit Exceeded".

The Programming Concepts option covers four stages in the learning process of programming: simple data and basic operations; control structures (selection and loops); subprograms; and data types (arrays and structures). For each of these stages we developed a set of 45 problems. Problems have been carefully chosen and developed so gradually grow the difficulty in five levels of difficulty, each of them having five problems. The difficulty levels range from very easy (we expect at least 95% of our students will solve the five problems) to very heavy (no more than 5% of our students could solve problems in this category). This five levels of difficulty are present in each set of problems; right in step with the core issues (phrases), there are four or five difficult problems, and also the datasets on structures or algorithmic techniques there are problems easy to solve.

The first set of problems is about expressions. They must be solved without using repetitive instructions or conditional instructions (alternative) and cannot be used structured data types as arrays. The difficulty level is set in that case, meaning an issue considered difficult in this work stage certainly it would be considered easy if you could solve using iteration structures or arrays. To present this idea of classifying problems we present some of utterances used in each of the five (5) stages [3]:

- Very easy problems: To add two integers; To read and to display a string; To convert a word in uppercase and lowercase and vice versa; To turn a sum of money in a certain currency to another; A temperature to convert from Celsius to Fahrenheit and vice versa; To transform into various units of measure some value entered as the date of entry.

- Easy Problems: To be a term in the Fibonacci sequence; Compute after a square matrix; Compute the lowest and highest value of the two numbers; to check

certain properties of numbers (primes, perfect numbers, palindrome numbers).
- Medium problems: To sort the elements of a vector; Compute the smallest and largest value in a vector of numbers of a particular type; To eliminate certain values in a vector; Calculate the sum and product values of the 4 areas of a square matrix.
- Difficult problems: problems that require the use of programming techniques such as backtracking, divide and conquer, recursion, or dynamic memory allocation.
- Very difficult problems: Crossing a certain tree; Finding Partial Least Cost tree; the maze problem (requires the method backtracking in plan).

2. SOLUTION TO SIEVE OF ERATOSTHENES

In mathematics, the sieve of Eratosthenes (Ancient Greek: κόσκινον Ἐρατοσθένους, kóskinon Eratosthénous), one of a number of prime number sieves, is a simple, ancient algorithm for finding all prime numbers up to any given limit. It does so by iteratively marking as composite (i.e., not prime) the multiples of each prime, starting with the multiples of 2.[1]
The multiples of a given prime are generated as a sequence of numbers starting from that prime, with constant difference between them that is equal to that prime.[1] This is the sieve's key distinction from using trial division to sequentially test each candidate number for divisibility by each prime.[2]
The sieve of Eratosthenes is one of the most efficient ways to find all of the smaller primes. It is named after Eratosthenes of Cyrene, a Greek mathematician; although none of his works have survived, the sieve was described and attributed to Eratosthenes in the Introduction to Arithmetic by Nicomachus.[3]

REQUIREMENT
Given n natural numbers less than 1,000,000, determine how many of them are primes.

INPUT
The input file eratostene.in contains on its first line the number n, then the n numbers arranged on several lines and separated by spaces.

OUTPUT
The output file eratostene.out contains the number C, representing the number of values that were read primes.

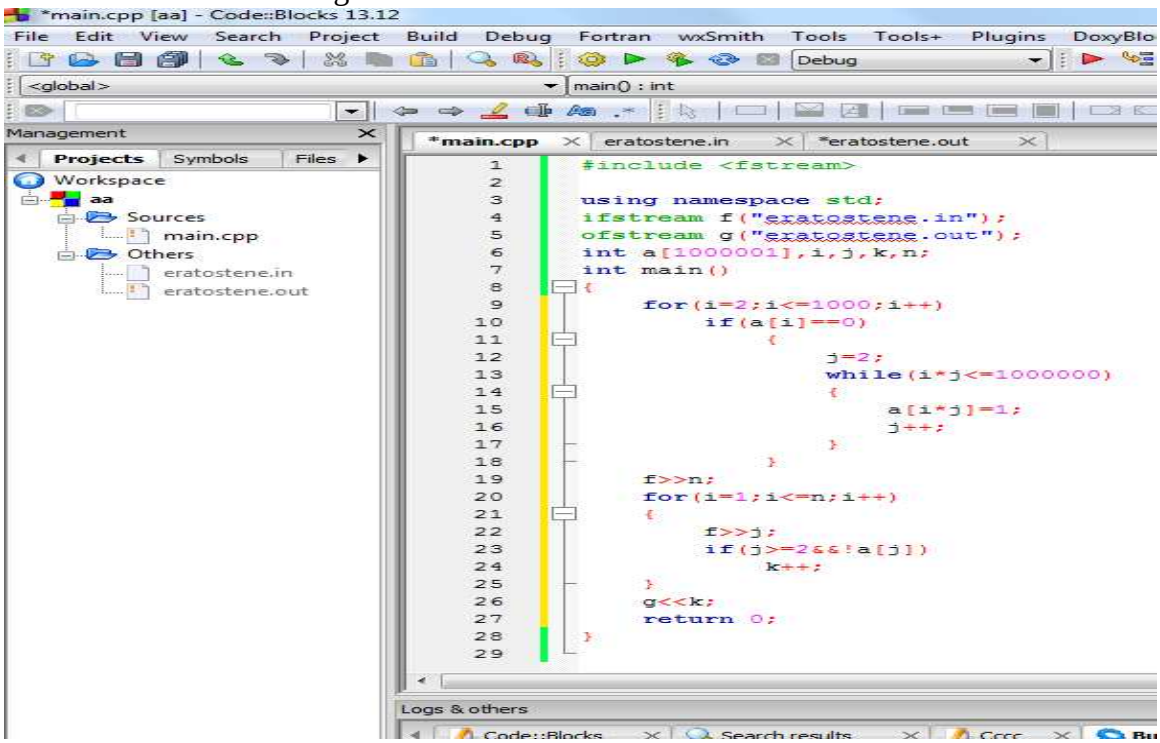
RESTRICTIONS AND SPECIFICATIONS
$1 \leq n \leq 1.000.000$

Example:
eratostene.in
6
12 18 19 25 29 7
eratostene.out
3

```
#include <fstream>

using namespace std;

ifstream f("eratostene.in");
ofstream g("eratostene.out");

int a[1000001],i,j,k,n;
int main()
{
        for(i=2;i<=1000;i++)
           if(a[i]==0)
              {
                 j=2;
                 while(i*j<=1000000)
                 {
                    a[i*j]=1;
                    j++;
                 }
              }
        f>>n;
        for(i=1;i<=n;i++)
        {
           f>>j;
           if(j>=2&&!a[j])
```

```
        k++;                                    }
}
g<<k;
return 0;
```

Figure 2 – Source code in Code Blocks



4. CONCLUSION

In this article we designed a MOOC (Massive Open Online Course) type application that can be integrated into LMS (Learning Management System) Moodle. In many universities are using MOOC platforms, approaching increasingly more courses for students. In this article we present a model of application adapted for an introductory teaching course called Computer Programming, for students of the technical faculties. In our futures work we implement another teaching course that could be use in Engineering Faculty - Electronics.

5. REFERENCES

[1] Ricardo Queirós, Design a Computer Programming Learning Environment for Massive Open Online Courses, Hershey, PA: IGI Global, doi: 10.4018/978-1-4666-7304-57-68, ISBN: 9781466673045

[2] R. Crandall, C. Pomerance. Prime Numbers: A Computational Perspective. Springer, 2nd ed., 2005

[3] Luis Llana, Enrique Martin-Martin, and Cristóbal Pareja-Flores. 2012. FLOP, a free laboratory of programming. In Proceedings of the 12th Koli Calling International Conference on Computing Education Research (Koli Calling '12). ACM, New York, NY, USA, 93-99. DOI=http://dx.doi.org/10.1145/2401796.2401807

[4] T. Cormen, C. Leiserson, R. Rivest, C. Stein. Introduction to Algorithms. The MIT Press, 3rd ed., 2009.

[5] F.J. Olver et al. (eds.). NIST Handbook of Mathematical Functions. National Institute of Standards and Technology and Cambridge University Press, 2010

[6] R. Sedgewick, K. Wayne. Algorithms. Addison-Wesley, 4th ed., 2011

[7] http://www.scintilla.org/SciTE.html

[8] http://www.runceanu.ro/adrian