ALGORITHM FOR SOFTWARE DEVELOPMENT

Isoqova Adiba,
Termez State University,
Student of the Faculty of Information Technology.
E-mail: isabida010120@mail.ru

Today I would like to tell you about some code development algorithm, be it software or a game. I do not claim to be innovative, revolutionary, or any other privilege of this development method. I believe that any approach has a right to life.

Recently, I began to notice a certain algorithm that I use when writing code. I confess - I have not read about this anywhere, although, even if it did slip, I do not remember it. Therefore, if I violated someone's rights, I apologize.

On reflection, I came to the conclusion that this is a great method to minimize bugs even before writing the entire program.

The algorithm itself

1. Initially, the program is designed in the head or on paper; we develop the architecture of the application.
2. Next, from all the future code, we select the core of the program, from which we will build on in the future, break it down into small subtasks and get down to work. After completing the subtask, you can rest and / or optimize your code.
3. After writing the kernel, go for a walk or do something unrelated to what you were just doing.
4. After rest, we look through the architecture for optimization. If you don't like something, change it. If nothing new comes to your mind, it does not matter, perhaps you initially thought of a good application architecture.
5. Start developing new functionality by breaking it down into smaller pieces. After completing the small part, you can rest and / or optimize the code.
6. After writing new functionality, go, take a walk or do something unrelated to what you were just doing.
7. After resting, take another look at the architecture of the application, and change it if necessary. After - optimize the code.
8. Repeat from 5 to 7 points when adding each new functionality before writing the program.

Let's go through the points

You can read about the development of application architecture on the Internet, but the main concept, in my opinion, is the development of basic classes, groups of classes and their interaction with each other.

By the core of the program, I mean the main functionality of the application, for example: in the game - the engine for positioning objects in the game world, in the drawing program - the simple functionality of the canvas and brush, in the program for creating and editing text - the component that formats and displays text, etc.

Breaks between small and large tasks are essential. First, so that the code you write can be digested in your head. Secondly, in order for you to rest, because, as you know, the best rest is to change one activity to another. Thirdly, to warm up the body.

Better to optimize with less code and more often than everything and very rarely. This approach is good for its simplicity and ease of optimization. Optimization means everything that is included in code optimization, for example, refactoring, performance optimization.

Optimizing code after a rest, rather than before it, has the advantage of being fresh.

Conclusion

The uniqueness of this method, I believe, is in breaking down the programming into small tasks, looking at the architecture of the program from the outside, optimizing the code and saving, every time, after completing one assigned task. Plus, with this approach, the application is operational at almost any point in time. And if you did not survive, and your lights were turned off, the loss will not be big, it will even benefit you, because you can rethink the previously written code and optimize it again.

Perhaps the only drawback of this approach is the coding time. But it also pays off due to fewer bugs during further development.

For beginners, I would not recommend optimizing the code so often, so as not to lose interest in basic development. But people with healthy perfectionism will especially love this method! The main thing is not to overdo it and not make optimization a goal.