JMEA Journal of Modern Educational Achievements
2023, Volume 5 https://scopusacademia.org/

ECONOMICAL SYSTEM AND DIFFERENT METHODS

Erdonov Mukhammadamin Erdon o'g'li

Assistant, Samarkand Institute of Economics and service

Khakimov Damir Ulugbekovich

Samarkand economy and service

Student of the Institute

ANNOTATION

Are you aware of the different types of economic systems? I bet you only know of one or two categories. This article explores the various forms of economic systems their application, and their merits and demerits. Follow through this guide so you can gauge your knowledge of economic systems and their differences.

A program is an expression of an idea. A programmer starts with a general idea of a task for the computer to perform. Presumably, the programmer has some idea of how to perform the task by hand, at least in general outline. The problem is to flesh out that outline into a complete, unambiguous, step-by-step procedure for carrying out the task. Such a procedure is called an "algorithm." (Technically, an algorithm is an unambiguous, step-by-step procedure that always terminates after a finite number of steps. We don't want to count procedures that might go on forever.) An algorithm is not the same as a program. A program is written in some particular programming language. An algorithm is more like the idea behind the program, but it's the idea of the steps the program will take to perform its task, not just the idea of what the task needs to accomplish in the end. When describing an algorithm, the steps don't necessarily have to be specified in complete detail, as long as the steps are unambiguous and it's clear that carrying out the steps will accomplish the assigned task. An algorithm can be expressed in any language, including English. Of course, an algorithm can only be expressed as an actual program if all the details have been filled in. So, where do algorithms come from? Usually, they have to be developed, often with a lot of thought and hard work. Skill at algorithm development is something that comes with practice, but there are techniques and guidelines that can help. I'll talk here about some techniques and guidelines that are relevant to "programming in the small," and I will return to the subject several times in later chapters.

An economic system is a methodology of producing, allocating resources as well as distributing trade goods and services in a society. These systems are then utilized in the control of the main aspects of production like labour, information resources and finally capital. The process by which rare resources are produced and spread in an economy has an effect on the form of economic system that is utilized. There are four main types of economic systems frequently experienced, they consist of the traditional system, mixed economic structure, command, and mixed economic system. All this four have different advantages and disadvantages. What is the purpose of an economic system? Read on to know the economic systems examples in each classification.

Suppose you have a task in mind that you want the computer to perform. One way to proceed is to write a description of the task, and take that description as an outline of the algorithm you want to develop. Then you can refine and elaborate that description, gradually adding steps and detail, until you have a complete algorithm that can be translated directly into programming language. This method is called stepwise refinement, and it is a type of top-down design. As you proceed through the stages of stepwise refinement, you can write out descriptions of your algorithm in code—informal instructions that imitate the structure of programming languages without the complete detail and perfect syntax of actual program code. As an example, let's see how one might develop the program from the previous section, which computes the value of an investment over five years. The task that you want the program to perform is: "Compute and display the value of an investment for each of the next five years, where the initial investment and interest rate are to be specified by the user." You might then write—or more likely just think—that this can be expanded as:

Get the user's input Compute the value of the investment after 1 year Display the value Compute the value after 2 years Display the value Compute the value after 3 years Display the value Compute the value after 4 years.

Display the value Compute the value after 5 years Display the value

Two final notes on this program: First, you might have noticed that the first term of the sequence—the value of N input by the user—is not printed or counted by this program. Is this an error? It's hard to say. Was the specification of the program careful enough to decide? This is the type of thing that might send you back to the boss/professor for clarification. The problem (if it is one!) can be fixed easily enough. Just replace the line "counter = 0" before the while loop with the two lines:

It would be nice if, having developed an algorithm for your program; you could relax, press a button, and get a perfectly working program. Unfortunately, the process of turning an algorithm into Java source code doesn't always go smoothly. And when you do get to the stage of a working program, it's often only working in the sense that it does something. Unfortunately not what you want it to do.

After program design comes coding: translating the design into a program written in Java or some other language. Usually, no matter how careful you are, a few syntax errors will creep in from somewhere, and the Java compiler will reject your program with some kind of error message. Unfortunately, while a compiler will always detect syntax errors, it's not very good about telling you exactly what's wrong. Sometimes, it's not even good about telling you where the real error is. A missing or extra brace can be one of the hardest errors to find in a large program. Always, always indent your program nicely. If you change the program, change the indentation to match. It's worth the trouble. Use a consistent naming scheme, so you don't have to struggle to remember whether you called that variable interestrate or interestRate. In general, when the compiler gives multiple error messages, don't try to fix the second error message from the compiler until you've fixed the first one. Once the compiler hits an error in your program, it can get confused, and the rest of the error messages might just be guesses. Maybe the best advice is: Take the time to understand the error before you try to fix it. Programming is not an experimental science.

When your program compiles without error, you are still not done. You have to test the program to make sure it works correctly. Remember that the goal is not to get the right output for the two sample inputs that the professor gave in class. The goal is a program that will work correctly for all reasonable inputs. Ideally, when faced with an unreasonable input, it should respond by gently chiding the user rather than by crashing. Test your program on a wide variety of inputs. Try to find a set of inputs that will test the full range of functionality that you've coded into your program. As you begin writing larger programs, write them in stages and test each stage along the way. You might even have to write some extra code to do the testing—for example to call a subroutine that you've just written. You don't want to be faced, if you can avoid it, with 500 newly written lines of code that have an error in there somewhere.

REFERENCES

1.Karshieva U. IMPROVEMENT OF THE SYSTEM OF BREEDING AND SEED PRODUCTION OF SOFT WHEAT FOR IRRIGATED LANDS OF UZBEKISTAN //International Journal of Early Childhood Special Education. – 2022. – T. 14. – №. 7.

2.Karshieva U. Improving the System of Selection and Seed Production of Soft Wheat for Irrigated Lands of Uzbekistan //International Journal on Integrated Education. – T. 2. – №. 6. – C. 240-242.