on the economic systems of the countries of the world, depending on the starting conditions and the current level of development. It is obvious that the factors themselves change over time.

REFERENCES

1. The IMD World Competitiveness Yearbook // IMD. [Електронний ресурс]. – URL: https://www.imd.org/centers/world-competitiveness-center/rankings/world-competitiveness/
2. Wall, W.P. (2022) Global Competitiveness [Електронний ресурс]. – URL: https://doi.org/10.1007/978-981-16-7755-7
3. World Talent Ranking 2021 // IMD. 2021. [Електронний ресурс]. – URL: https://www.imd.org/centers/worldcompetitiveness-center/rankings/world-talent-competitiveness/

P. Forkert, M. Sydorova, Yu. Honcharova

MODERN ARCHITECTURE OF DYNAMICALLY TYPED PROGRAMMING LANGUAGEVMS

Dynamically typed programming languages such as Python, Ruby, and JavaScript have become increasingly popular in recent years. Their dynamic nature offers several advantages over their statically typed counterparts, such as rapid prototyping, easy debugging, and the ability to dynamically generate code at runtime. However, it also comes with a performance cost, as the interpreter has to perform additional operations during program executions, for example, type checking and memory management. This work aims to research the ways how modern dynamically typed programming languages are implemented to achieve their performance metrics.

A lot of research that attempted to mitigate the performance cost [1–3] was conducted during the last several decades. Nowadays language virtual machines (VMs) feature sophisticated garbage collectors, efficient bytecode interpreters, and just-in-time (JIT) compilers, that convert frequently executed code into machine

code during runtime. Such architectures allow VMs to achieve performance close to that of compiled statically typed languages such as C or C++. Furthermore, modern VMs use adaptive compilation techniques that dynamically optimize the code based on runtime profiling data, allowing for further performance improvements.

Javascript as the only programming language supported in browsers currently features the most performant and efficient implementations:

1. V8 [4], developed by Google for use in their Chrome browser. It is now also used in Microsoft Edge and Opera.
2. SpiderMonkey [5], developed by Mozilla for their Firefox browser.
3. JavaScriptCore [6], developed by Apple for the Safari browser.

The architecture of all those VMs, while different in the details and actual code, is surprisingly similar – they all feature an efficient interpreter and several (usually two) JIT compilers, which allow them to balance between their latency and throughput.

When the application starts execution happens in the interpreter mode, a VM does not spend time compiling and optimizing code, therefore the feedback happens faster, but the code is executed slowly. In addition to code execution interpreter also collects additional metrics about the running code: which types occur at which call site, how many times each method or function was called, etc.

When some particular function or section of code is executed enough times – the VM compiles it with the first JIT compiler, which is usually called "baseline compiler" – it does not do many optimizations, but the compilation itself is fast. At this point, the code continues collecting metrics about its execution.

If a section of code executes even more times – then it is compiled by the next JIT compiler, usually called "optimizing compiler" because it applies different optimizations so the code is more efficient to execute. At this stage, the generated code after optimizations can reach the performance of compiled statically typed languages or even overperform them.

REFERENCES

1. Clifford, Daniel & Payer, Hannes & Stanton, Michael &Titzer, Ben. Mementomori: dynamical location-site-based optimizations.2015.doi: 10.1145/2887746.2754181.

2. A. Parravicini, R. Mueller. The Cost of Speculation: Revisiting Overheads in the V8 JavaScript Engine.2021. doi: 10.1109/IISWC53511.2021.00013.
3. M. Chevalier-Boisvert, M. Feeley. Interprocedural Type Specialization of JavaScript Programs Without Type Analysis. 2016.
4. V8's TurboFan. https://v8.dev/docs/turbofan
5. Warp: Improved JS performance in Firefox 83. 2020. https://hacks.mozilla.org/2020/11/warp-improved-js-performance-in-firefox-83/
6. Speculation in JavaScriptCore. 2020. https://webkit.org/blog/10308/speculation-in-javascriptcore/

B. Garazha, O. Baylova, O. Aliseienko

THE ROLE OF PLANNING IN BUSINESS ACTIVITY

In modern conditions, to raise innovative business methods, before opening a new entrepreneurial business or improving an existing one, it is necessary to determine the needs of the ever-changing market by its nature. When developing new products, the dynamics of market trends should be taken into account. Therefore, before opening a new business or expanding an existing one it is necessary to solve the following important questions: if itis worth investing in the project, if it stands the competition and what its uniqueness is? Business planning answers all these questions and helps to avoid any economic risks.

Mostly, entrepreneurs planning a new business seek inspiration from one of these four sources:1) the previous work experience; 2) education or training; 3) nature hobbies, abilities or other personal interests; 4) understanding of the existence of unsatisfied needs or opportunities for market expansion. Sometimes the first reason is the work experience of a relative or a friend. As practice shows, today almost all beginner entrepreneurs analyze and plan the business before starting it.

A business plan is a carefully prepared document that discloses all aspects of any business venture that is started [2].

A business plan is a concise, available, accurate and easy-to-understand image of a planned business, the most important tool for studying various circumstances, which allows you to choose the most desired result and find out the ways to achieve it.