UNIVERSITATEA TEHNICĂ A MOLDOVEI

FACULTATEA CALCULATOARE, INFORMATICĂ ȘI MICROELECTRONICĂ
DEPARTAMENTUL INFORMATICĂ ȘI INGINERIA SISTEMELOR

# ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Курс-конспект

EDITURA TEHNICA-UTM

**2026**

**CZU 004.4(075.8)**
**Б 896**

Lucrarea a fost discutată și aprobată pentru editare la ședința Consiliului Facultății Calculatoare, Informatică și Microelectronică, proces-verbal nr. 8 din 25.06.2026

Данный конспект предназначен для того, чтобы помочь студентам раскрыть суть основных принципов, входящих в понятие “объектно-ориентированное программирование”. В тексте содержится как теоретический материал, необходимый для работы с этими принципами, так и практические примеры на нескольких языках программирования, включая **Smalltalk**, **Simula** и **C++**–для иллюстрации принципов наследования, полиморфизма и инкапсуляции в действии.

Работа в первую очередь предназначена для студентов второго курса, изучавших основы программирования на языке **C**, а также владеющих базовыми знаниями о фундаментальных структурах данных и алгоритмах.

Текст освещает пятнадцать тем, соответствующих лекциям, которые читаются в ходе курса “Объектно-ориентированное программирование”, и делится на двенадцать глав. Каждая глава содержит теоретическое обоснование практических и индивидуальных заданий курса. Конспект также включает в себя технические рекомендации по применению языка **C++** для создания объектно-ориентированных систем.

Автор: ассистент, Брынзан, Л. В.

Рецензент: доцент, доктор Фалько, Н. С.

**DESCRIEREA CIP A CAMEREI NAȚIONALE A CĂRȚII DIN RM**

**Брынза Л. В.**

Объектно-ориентированное программирование: Курс-конспект / Брынзан Л. В.; Universitatea Tehnică a Moldovei, Facultatea Calculatoare, Informatică şi Microelectronică, Departamentul Informatică şi Ingineria Sistemelor. – Chișinău: Tehnica-UTM, 2026. – 199 p.: fig., Bibliogr.: p. 194-196 (24 tit.). – 50 ex. Text: nemediat.

**ISBN 978-9975-64-670-3** 

## Введение

Сложно сказать когда именно фраза “объектно-ориентированный” была впервые применена в контексте компьютерной программы, но доподлинно известно, кем и почему это было сказано. Математик Алан Кёртис Кей, который работал в исследовательской лаборатории PARC[1] корпорации Xerox, занимался проектированием и реализацией экспериментальных компьютерных систем, в связи с чем ему приходилось работать с новыми компьютерными разработками и новыми языками программирования. В 1967 году–под впечатлением от компьютерной системы **Sketchpad**, языка программирования **Simula** и нескольких маленьких экспериментальных проектов–А. Кей начал работу над собственной системой **FLEX**[2], которая включала в себя высокоуровневый язык программирования на базе компилятора, реализованного на микросхеме. Основной целью проекта было создание такой системы, которая была бы легче в освоении, чем существовавшие на тот момент компьютеры и операционные системы, поддерживавшие языки программирования **Fortran** и **Algol**. Ядро системы составлял “процессо-ориентированный” подход к написанию программ.

По задумке А. Кея на любом компьютере процессы должны выполняться параллельно, так как разные задачи напрямую друг от друга не зависят, а следовательно и не должны друг друга блокировать. Например, ввод или вывод данных никак не связан с обработкой не зависимой от этих данных информации. Отсюда возникла необходимость дать программисту самому возможность создавать параллельные процессы в своих программах. Т.е. создание параллельных процессов должно быть составной–даже основной–частью самого языка программирования. Это позволяет писать

[1]Palo Alto Research Center (сегодня–Future Concepts).

[2]FLEX основывался на языке **Euler** Никлауса Вирта, а тот, в свою очередь, был интерпретацией языка **Lisp** на базе языка **Algol**.

программы разной степени сложности, используя рекурсию, “события” и процессы для описания таких программ.

А. Кей часто приводил в качестве примера такой системы сеть "Интернет", в которой отдельные компьютеры могут взаимодействовать друг с другом только посредством отправки сообщений другим компьютерам, оставаясь независимыми вычислительными объектами, данные которых скрыты от остальных компьютеров в одной с ними сети. В этом заключалась сущность “объектов” в контексте системы, проектируемой им.

**Объект** изначально–это описание вычислительного процесса или семейства процессов, которое скрывает свои данные от внешней среды и может работать независимо от других объектов, взаимодействуя с ними с помощью сообщений, отправленных согласно какому-то протоколу. Концепцию “объектно-ориентированного подхода” в программировании примерно тогда же первым начал использовать в своей работе именно А. Кей. Как он говорил намного позже [1], “одним из идеалов в 1960-х для вычислительных систем было не просто использовать их в сфере образования, а создать новое мышление, которое стало необходимым после индустриальной революции с появлением проблемы масштабирования в контексте человеческого бытия. Эйнштейн заметил, что мышление, которое создает для нас проблемы, не поможет нам решить эти проблемы. Нужно другое мышление. Это глубокая мысль!”

Данная цитата может служить лейтмотивом этого курса. Похожую мысль высказал другой известный специалист, лауреат премии имени Алана Тьюринга Джон Бакус [2].

“Широко распространенные сегодня языки программирования становятся все больше, но при этом не становятся мощнее. Врожденные дефекты в самом основании делают их опухшими и слабыми: примитивный стиль программирования, унаследованный от своего общего

предка–компьютера Фон Неймана, сильная привязка семантики к изменениям состояний в памяти, разделение всего программирования на мир выражений и мир утверждений, невозможность эффективно использовать мощные инструменты для комбинации существующих программ в новые, отсутствие в них полезных математических свойств–для того, чтобы делать правильные умозаключения о компьютерных программах.”

Обе цитаты особо подчеркивают необходимость нового способа мышления для написания хороших компьютерных программ. Объектно-ориентированное программирование было попыткой сделать такое мышление более естественным и оттого–более распространенным. Одной из основных целей данного курса как раз и является обоснование необходимости и важности такого подхода к программированию программного обеспечения.

1. За последние 50 лет в рамках объектно-ориентированной парадигмы было написано огромное количество кода.
2. Многие современные языки программирования были спроектированы вокруг идеи “объектно-ориентированности”.
3. Многие популярные идиомы (приемы), используемые в дизайне языков программирования или в конечных программах, пришли из объектно-ориентированного дизайна.
4. Многие компании, производящие программное обеспечение, зациклены на объектно-ориентированном дизайне и требуют его знания и применения от своих разработчиков.
5. Объектно-ориентированный дизайн требует особого подхода к архитектуре приложений.

## Содержание

## Библиография


1. KAY, A.C. Why the Real Computer Revolution Never Happened, 2025. https://www.youtube.com/watch?v=MbEZ-DC0L-g (accessed 2026-04-20).
2. BACKUS, J. Can Programming Be Liberated from the Von Neumann Style? A Functional Style and Its Algebra of Programs. In Programming Languages; Horowitz, E., Ed.; Springer Berlin Heidelberg: Berlin, Heidelberg, 1983; pp 146–174. https://doi.org/10.1007/978-3-662-09507-2_10.
3. БРЫНЗАН, Л.В. Объектно-ориентированное программирование: Методические указания к лабораторным работам; Ghiduri metodologice; Universitatea Tehnică a Moldovei: Chișinau, 2024. ISBN: 978-9975-64-378-8.
4. СТЕПАНОВ, А.А. Наибольшая общая мера последние 2500 лет, 2010. https://www.youtube.com/watch?v=zwucsB2EfXc (accessed 2026-04-20).
5. SUSSMAN, G.J. Flexible Systems: The Power of Generic Operations, 2016. https://www.youtube.com/watch?v=JbyAcZf6tds (accessed 2026-04-20).
6. KAY, A.C. The Early History of Smalltalk. SIGPLAN Not. 1993, 28 (3), pp. 69–95. https://doi.org/10.1145/155360.155364.
7. NYGAARD, K.; DAHL, O.-J. The Development of the SIMULA Languages. In History of programming languages; Wexelblat, R. L., Ed.; ACM: New York, NY, USA, 1978; pp 439–480. https://doi.org/10.1145/800025.1198392.
8. KAY, A.C.; GUZDIAL, M. Alan Kay on Moti's "Objects Ever?" CACM Article. Computing Ed Research–Guzdial's Take. https://computinged.wordpress.com/2010/09/15/alan-kay-on-motis-objects-ever-cacm-article/ (accessed 2026-04-20).

9. KAY, A.C. What aspects of Lisp influenced Smalltalk?. Quora. https://www.quora.com/What-aspects-of-Lisp-influenced-Smalltalk (accessed 2026-04-20).
10. KAY, A.C. What did Alan Kay mean by “assignment” in The Early History of Smalltalk? Software Engineering, Stack Exchange. https://softwareengineering.stackexchange.com/questions/81197/what-did-alan-kay-mean-by-assignment-in-the-early-history-of-smalltalk (accessed 2026-04-20).
11. KAY, A.C. Joe Armstrong Interviews Alan Kay, 2016. https://www.youtube.com/watch?v=fhOHn9TClXY (accessed 2026-04-20).
12. STROUSTRUP, B. The Essence of C++, 2024. https://www.youtube.com/watch?v=ZXc_z1sNbfA (accessed 2026-04-20).
13. STROUSTRUP, B. A History of C++: 1979–1991. In The second ACM SIGPLAN conference on History of programming languages; ACM: Cambridge Massachusetts USA, 1993; pp 271–297. https://doi.org/10.1145/154766.155375.
14. KAY, A.C. Computer Software. Sci Am 1984, 251 (3), pp. 3–9. https://doi.org/10.1038/scientificamerican0984-188.
15. STROUSTRUP, B. An Overview of C++. SIGPLAN Not. 1986, 21 (10), pp. 7–18. https://doi.org/10.1145/323648.323736.
16. MARTIN, R.C. The Last Programming Language, 2011. https://www.youtube.com/watch?v=P2yr-3F6PQo (accessed 2026-04-20).
17. KAY, A.C. Dr. Alan Kay on the Meaning of “Object-Oriented Programming.” Persistent uniform resource locator. https://userpage.fu-berlin.de/~ram/pub/pub_jf47ht81Ht/doc_kay_oop_en (accessed 2026-04-20).
18. CARDELLI, L.; WEGNER, P. On Understanding Types, Data Abstraction, and Polymorphism. ACM Comput. Surv. 1985, 17 (4), pp. 471–523. https://doi.org/10.1145/6041.6042.

19. STEPANOV, A.A.; MCJONES, P. Elements of Programming, 3. printing.; Addison-Wesley: Upper Saddle River, NJ, 2010. https://doi.org/10.5555/1614221.
20. ARMSTRONG, J. The Mess We're In, 2014. https://www.youtube.com/watch?v=lKXe3HUG2l4 (accessed 2026-04-20).
21. MUSSER, D.R.; STEPANOV, A.A. Generic Programming. In Symbolic and Algebraic Computation; Gianni, P., Ed.; Springer Berlin Heidelberg: Berlin, Heidelberg, 1989; Vol. 358. https://doi.org/10.1007/3-540-51084-2_2.
22. KALB, J. Object-Oriented Program: Best Practices, 2020. https://www.youtube.com/watch?v=c0lutJECNUA (accessed 2026-04-20).
23. LOVEJOY, A. Smalltalk: Getting The Message. Smalltalk.org. 2007. https://grumpyrabbit.substack.com/p/smalltalk-getting-the-message (accessed 2026-04-20).
24. MURATORI, C. The Big OOPs: Anatomy of a Thirty-Five-Year Mistake, 2025. https://www.youtube.com/watch?v=wo84LFzx5nI (accessed 2026-04-21).