UNIVERSITATEA TEHNICĂ A MOLDOVEI

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Методические указания к лабораторным работам

Chişinău
2024

UNIVERSITATEA TEHNICĂ A MOLDOVEI

FACULTATEA CALCULATOARE, INFORMATICĂ ȘI MICROELECTRONICĂ
DEPARTAMENTUL INFORMATICĂ ȘI INGINERIA SISTEMELOR

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Методические указания к лабораторным работам

Chişinău
Editura „Tehnica-UTM”
2024

CZU 004.4(076.5)
Б 896

Lucrarea a fost discutată și aprobată pentru editare la ședința Consiliului Facultății Calculatoare, Informatică şi Microelectronică din 13.12.2023, proces-verbal nr.3.

Данные методические указания предназначены 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 задания с примерами решения для иллюстрации принципов наследования, полиморфизма и абстракции в действии.

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

Текст делится на главы, которые содержат теоретическое обоснование практических заданий и технические рекомендации по их выполнению с помощью языка C++. Также студентам предлагается семь лабораторных работ для закрепления изученного материала и проверки полученных навыков на практике.

Автор: ассистент, Брынзан, Л. В.

Рецензент: доцент, доктор Фалько, Н. С.

DESCRIEREA CIP A CAMEREI NAȚIONALE A CĂRȚII DIN RM

Брынзан, Л. В.

Объектно-ориентированное программирование: Методические указания к лабораторным работам / Брынзан Л. В.; Universitatea Tehnică a Moldovei, Facultatea Calculatoare, Informatică și Microelectronică, Departamentul Informatică și Ingineria Sistemelor.

– Chișinău: Tehnica-UTM, 2024. – 155, [1] p.

Aut. indicat pe verso f. de tit. – Bibliogr.: p. 129-131 (28 tit.). – 100 ex.

ISBN 978-9975-64-378-8.

Содержание

Введение.....3
1. Абстрактные типы данных.....7
1.1. Теоретическая информация..... 7
1.2. Практические задачи для закрепления теории.....9
Лабораторная работа 1.....10
2. Сокрытие данных и инкапсуляция.....14
2.1. Теоретическая информация..... 14
2.2. Практические задачи для закрепления теории.....19
3. Конструкторы.....21
3.1. Теоретическая информация..... 21
3.2. Практические задачи для закрепления теории.....29
Лабораторная работа 2.....31
4. Перегрузка функций и операторов.....35
4.1. Теоретическая информация..... 35
4.2. Практические задачи для закрепления теории.....42
Лабораторная работа 3.....45
5. Итераторы..... 49
5.1. Теоретическая информация..... 49
Лабораторная работа 4.....53
6. Наследование и полиморфизм.....57
6.1. Теоретическая информация..... 57
6.2. Практические задачи для закрепления теории.....64
7. Полиморфизм подтипов.....66
7.1. Теоретическая информация..... 66
7.2. Практические задачи для закрепления теории.....71
Лабораторная работа 5.....74
8. Шаблоны типов.....77

8.1. Теоретическая информация..... 77
Лабораторная работа 6.....82
9. Множественное наследование..... 85
9.1. Теоретическая информация..... 85
9.2. Практические задачи для закрепления теории.....90
Лабораторная работа 7.....93
10. Примеры решений для практических задач..... 107
10.1. Задачи из пункта 1..... 107
10.2. Задачи из пункта 2.....111
10.3. Задачи из пункта 3.....113
10.4. Задачи из пункта 4.....118
10.5. Задачи из пункта 6.....119
10.6. Задачи из пункта 7..... 121
10.7. Задачи из пункта 9..... 125
Библиография.....129
Приложения..... 132

Bun de tipar 03.01.24 | Formatul hârtiei 60x84 1/16
Hârtie ofset. Tipar RISO | Tirajul 100 ex.
Coli de tipar 9,75 | Comanda nr. 04

MD-2004, Chișinău, bd. Ștefan cel Mare și Sfânt, 168, UTM
MD-2045, Chișinău, str. Studenților, 9/9, Editura „Tehnica-UTM”

Введение

Когда говорят об абстракции в обыденности, подразумевают что-то эфемерное, идеальное, не существующее в физическом мире. Между тем абстракция является очень важным инструментом в программировании. Вот знакомая всем запись, $y = f(x)$. Здесь есть несколько абстрактных элементов. Переменные x и y сами по себе являются абстракциями возможных конкретных значений. Функция f является абстрактным механизмом преобразования значения переменной x в значение переменной y. Знак равенства указывает на то, что в данном контексте применение функции f к аргументу равносильно значению переменной y. Это позволяет сокращать записи математических выражений или значительно их упрощать. Например, $4x^4 - x^2 - x + 2$, выражение выше можно упростить с помощью абстракции, $y = 2x^2 + x + 1$, тогда изначальное выражение можно переписать, $y^2 - y + 2$.

Скрыв подробности функции в переменной y можно сосредоточиться на работе с квадратичной функцией, при необходимости раскрывая определение переменной y. Использование абстракции в выражениях называется *комбинацией* [Башкин].

В программировании абстракцию приходится применять похожим образом. Код ниже это иллюстрирует:

```
push    rbp
mov     rbp, rsp
mov     edi, OFFSET FLAT:.string "Hello, World!"
mov     eax, 0
call    printf
mov     eax, 0
pop     rbp
```

Неопытным взглядом сложно понять что именно происходит, особенно если не знать инструкции, которые используются в этом фрагменте. При этом, существует

механизм, создающий абстракцию этих инструкций, который приводит весь фрагмент к другому виду:

```
printf(“Hello, World!”);
```

Новая запись предельно краткая, но полностью скрывает происходящее в момент применения функции **printf** к аргументу. Тем не менее, такая форма является более предпочтительной, потому что в подавляющем большинстве случаев программиста не столько интересует конкретный набор инструкций, сколько окончательный результат. Под таким углом полезность абстракций очевидна.

Эту концепцию поначалу трудно представить, когда речь идет о *типах данных*. И тем не менее, программирование в значительной степени состоит из взаимодействий с различными уровнями абстракций типов. Для того, чтобы в этом убедиться, следует вернуться к понятию “переменная”, но теперь уже – в контексте программирования. Кто-то может сказать, что переменная – это имя в программе. Что будет не совсем точно, потому что “имя”, обладает единственным свойством – уникальностью. Очевидно, что переменные обладают и другими свойствами. Кто-то скажет, что это “хранилище”, которое принимает различные значения. Это опять не совсем точно в отношении переменных в программах, так как часто переменная не хранит значение, которое ей присваивается, при этом значение не может существовать в программе само по себе, ведь оно тоже абстрактно. Оно будет иметь некоторое конкретное представление. Например, число “42” является значением, оно будет представлено в памяти компьютера в виде двоичного ($2^5 + 2^3 + 2^1$), а в программе – еще и в виде десятичного ($4 * 10^1 + 2 * 10^0$) или шестнадцатеричного ($2 * 16^1 + 10 * 16^0$, $2A_{16}$) набора данных.

На основании этих уточнений можно выделить три основных момента, присущих любой переменной. Переменные обладают способом представления в конкретном языке (целое число, дробное число, текст и т.д.). Переменные обладают порядком хранения в памяти компьютера (занимаемые байты, а

также назначение каждого бита в соответствующих байтах). Наконец, переменные обладают непосредственно значением, которое определяет их полезность в программе. Таким образом мы столкнулись с тремя уровнями абстракции, совершая переход от математических концепций к их воплощению в памяти компьютера посредством какого-то языка программирования[1].

Так как математические абстракции принято классифицировать по каким-либо признакам для удобства (например, переменные разделяются на такие, которые представляют целые или дробные числа), такой подход применяется и в языках программирования. Классификация производится по свойствам, перечисленным выше, а именно: какое множество значений принимают данные, какую форму принимают значения из этого множества, какие операции возможны для значений из этого множества [Wirth]. То есть, *тип данных* – это совокупность представления какого-то значения в программе и в памяти компьютера, которая позволяет работать с конкретным значением в программе с помощью соответствующих операций. Внутри самой программы также есть “конкретные” сущности и абстрактные сущности. Первые ассоциируются с *объектами*, вторые – со значениями [McJones]. В общем смысле, объект – это некоторое отношение между типом данных и самими данными в памяти компьютера. Свойства объекта определяются его типом.

Другой смысл слова “абстрактный” – “удаленный”, то есть, рассматриваемый в отрыве от своего источника. Так разделение подходов в программировании на парадигмы (программирование может быть структурное, объектно-ориентированное, функциональное и т.д.) является абстракцией, потому что с точки зрения целевого устройства все эти парадигмы производят один и тот же машинный код, который исполняется независимо от того, какую идею

[1] Программирование можно считать одной из областей математики, языки программирования можно называть абстрактными компьютерами [Strachey].

вкладывал в него программист. Предназначение абстракций в программировании заключается в том, чтобы *проектировать программы, понимать их и проверять их логику на правильность на основании законов, которые управляют абстракциями* (например, на основании законов математики), не отвлекаясь на подробности реализации этих абстракций в конкретном компьютере.

Библиография

1. Башкин, В. А., “Лямбда-выражения”, Лямбда-исчисление, ЯрГУ, Ярославль, 2018, с. 8.
2. Овчинников, А. В., “Множество, замкнутое относительно операции”, Теория групп. Лекционный курс, Москва, 2016, с. 10.
3. Степанов, А. А., “Наибольшая общая мера: последние 2500 лет” [сетевой ресурс], 2010, [посещен 24.08.2023]. Ссылка: https://www.youtube.com/watch?v=NfGeVRebiio.
4. ANSI, “ISO-IR-6: ASCII Graphic Character Set”, ANSI, December 1, 1975.
5. Brinkerhoff, D. A., “Introduction to files and I/O streams”, Streams, Object-Oriented Programming Using C++ [сетевой ресурс], Weber State University, 2015-2022, [посещен 24.08.2023]. Ссылка: https://icarus.cs.weber.edu/~dab/cs1410/textbook/14.Streams/introduction.html.
6. Cardelli, L., Wegner, P., “On Understanding Types, Data Abstraction, and Polymorphism”, Computing Surveys, Vol 17 n. 4, 1985, pp. 471-522.
7. Chu, I., “Dynamic Dispatch in Object Oriented Languages”, Concepts of Programming Languages [сетевой ресурс], Course notes, [посещен 24.08.2023]. Ссылка: https://condor.depaul.edu/ichu/csc447/notes/wk10/Dynamic2.htm.
8. Clarkson, M. R. et al., “Type Inference”, OCaml Programming: Correct + Efficient + Beautiful [сетевой ресурс], [посещен 24.08.2023]. Ссылка: https://cs3110.github.io/textbook/cover.html.
9. Cormen, T. H., et al., “Common divisors and greatest common divisors”, Number-Theoretic Algorithms, Introduction to Algorithms, 4th ed., 2022, p. 906.
10. Gill, P. E., et al., “Reference Counting”, Iotr documentation [сетевой ресурс], August 27, 2003, [посещен 24.08.2023].

Ссылка:
http://www.ccom.ucsd.edu/~peg/papers/iotrdoc/reference_counting.html.
11. Goldberg, A., Kay, A., “The Smalltalk World and Its Primitives”, Smalltalk-72 instruction manual, Palo Alto Research Center, Xerox, March, 1976, p.48.
12. Hinnant, H. E., et al., “A Brief Introduction to Rvalue References” [сетевой ресурс], June 12, 2006, [посещен 24.08.2023]. Ссылка:
https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2006/n2027.html.
13. IBM, “Restrictions on default arguments (C++ only)” [сетевой ресурс], March 23, 2021, [посещен 24.08.2023]. Ссылка:
https://www.ibm.com/docs/en/zos/2.1.0?topic=only-restrictions-default-arguments-c.
14. Knuth, D. E., “The Greatest Common Divisor”, Rational Arithmetic, Arithmetic, The Art of Computer Programming, vol. 2, 3rd ed., 1998, p. 338.
15. McJones, P., Stepanov, A. A., “Foundations. Values”, Elements of Programming, Semigroup Press, 2019, p. 2.
16. Moore, P., “Rational.hpp”, Boost Libraries Documentation [сетевой ресурс], [посещен 24.08.2023]. Ссылка:
https://www.boost.org/doc/libs/1_55_0/boost/rational.hpp.
17. Musser, D. R., Stepanov, A. A., “Generic Programming”, First International Joint Conference of ISAAC 88 and AAECC 6, 1988.
18. Polacek, M., “Understanding when not to std::move in C++” [сетевой ресурс], [посещен 24.08.2023]. Ссылка:
https://developers.redhat.com/blog/2019/04/12/understanding-when-not-to-stdmove-in-c.
19. Pomeranz, A., “Inheritance and access specifiers”, Inheritance [сетевой ресурс], Learn C++, March 24, 2022, [посещен 24.08.2023]. Ссылка:
https://www.learncpp.com/cpp-tutorial/inheritance-and-access-specifiers/.

20. Radford, M., ”C++ Interface Classes - An Introduction”, Overload 12(62) [сетевой ресурс], ACCU, August, 2004, [посещен 24.08.2023]. Ссылка: https://accu.org/journals/overload/12/62/radford_233/.
21. Ritchie, D.M., “The Development of the C Language”, History of Programming Languages, 2nd ed., ACM Press, 1996.
22. Standard C++ Foundation, “Inheritance – Multiple and Virtual Inheritance” [сетевой ресурс], Standard C++ Foundation Wiki, [посещен 24.08.2023]. Ссылка: https://isocpp.org/wiki/faq/multiple-inheritance.
23. Strachey, C., “Fundamental Concepts in Programming Languages”, Higher-Order and Symbolic Computation, 13, 2000, p. 13.
24. Stroustrup, B., Sutter, H., “C++ Core Guidelines” [сетевой ресурс], International Standardization Organization C++, 2022, [посещен 24.08.2023]. Ссылка: http://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines.
25. WG14 / N1256, “Types”, Concepts, Language, ISO/IEC 9899:TC3, Committee Draft, 2007, p. 33.
26. Whitney(a), T., et al., “delete operator”, Built-in operators, precedence and associativity [сетевой ресурс], C++ Language Reference, Microsoft Docs, August 3, 2021, [посещен 24.08.2023]. Ссылка: https://docs.microsoft.com/en-us/cpp/cpp/delete-operator-cpp?view=msvc-170.
27. Whitney(b), T., et al., “__interface”, Classes and Inheritance, C++ Language Reference [сетевой ресурс], Microsoft Docs, August 3, 2021, [посещен 24.08.2023]. Ссылка: https://docs.microsoft.com/en-us/cpp/cpp/interface?view=msvc-170.
28. Wirth, N., “Representation of Arrays, Records, and Sets”, Fundamental Data Structures, Algorithms and Data Structures, 1985, p. 21.