# ОТ ПРОСТОГО К СЛОЖНОМУ: СИСТЕМА ПОСТЕПЕННО УСЛОЖНЯЮЩИХСЯ ЗАДАЧ ДЛЯ ОСВОЕНИЯ РАЗВЕТВЛЯЮЩИХСЯ АЛГОРИТМОВ В C#

**Тазетдинова Ю.А.**, к.ф.-м.н., доцент

**Тазетдинов Б.И.**, к.ф.-м.н., доцент

**Гималтдинов А.Ф.**, студент 1 курса ПИ

г. Бирск, Бирский филиал Уфимского университета науки и технологий

**Аннотация**. В статье рассматривается методика обучения начинающих программистов работе с разветвляющимися алгоритмами в C# через систему постепенно усложняющихся практических задач. Предлагаемый подход помогает преодолеть типичные трудности, связанные с переходом от знания синтаксиса к умению решать реальные задачи. Описана поэтапная система заданий – от базовых условий до комплексных логических структур – которая формирует устойчивые навыки алгоритмического мышления и готовит студентов к решению практических проблем программирования.

**Ключевые слова**: разветвляющиеся алгоритмы, C#, оператор if, логические операторы, switch-case, вложенные условия, алгоритмическое мышление.

**Введение**. Многие начинающие программисты успешно заучивают синтаксис операторов if, else if и switch, но сталкиваются с значительными трудностями, когда дело доходит до решения даже несложных задач. Они не могут декомпозировать условие задачи на отдельные логические шаги, выявить взаимосвязи между данными и корректно структурировать код. Традиционный подход, при котором студентам сразу предлагаются сложные многоуровневые условия, часто приводит к фрустрации и формированию устойчивых ошибок.

Цель работы – разработать и предложить систему постепенно усложняющихся задач для освоения разветвляющихся алгоритмов в C#, которая позволяет планомерно формировать навыки алгоритмического мышления и преодолевать разрыв между знанием синтаксиса и умением решать практические задачи.

Решением обозначенных проблем является принцип «от простого к сложному», реализованный через продуманную последовательность учебных заданий, которая обеспечивает последовательное освоение базовых конструкций ветвления и их постепенное усложнение до уровня решения комплексных практических задач.

**1. Базовое ветвление. Формирование интуиции.** Цель этого этапа – закрепить

понимание того, что программа может выполнять различные действия в зависимости от истинности или ложности простого условия.

**Задача 1.1**. **Определение знака числа**. Пользователь вводит число. Выведите сообщение «Число положительное», «Число отрицательное» или «Число равно нулю».

Это первое знакомство с конструкцией if-else. Задача учит проверять одно значение по нескольким взаимоисключающим условиям.

**Задача 1.2. Проверка на четность**. Запросите у пользователя целое число и выведите на экран, является ли оно четным.

Вводится понятие операции остатка от деления % и ее использование в логическом выражении. Закрепляется работа с единственным условием if-else.

**2. Логические операторы. Комбинирование условий**. На этом этапе студенты учатся описывать более сложные правила, комбинируя простые условия с помощью операторов && (И), || (ИЛИ), ! (НЕ).

**Задача 2.1. Проверка попадания в диапазон**. Напишите программу, которая проверяет, принадлежит ли введенное число промежутку от 10 до 100 включительно.

Идеальная задача для демонстрации логического оператора &&. Показывает, как проверить выполнение двух условий одновременно.

**Задача 2.2. Простейший калькулятор**. Пользователь вводит два числа и символ операции (+, -, *, /). Программа должна выполнить соответствующую операцию и вывести результат. Реализуйте проверку деления на ноль.

Задача требует использования как if для выбора операции, так и вложенного if для проверки корректности операции деления. Это первый шаг к вложенным структурам.

**3. Вложенные условия и оператор выбора switch**. Здесь формируется умение работать с иерархией условий и выбирать оптимальную конструкцию для читаемости кода.

**Задача 3.1. Определение времени года**. Пользователь вводит номер месяца (1-12). Программа определяет и выводит время года (зима, весна, лето, осень).

Задачу можно решить длинной цепочкой else if, но она идеально подходит для демонстрацию эффективности оператора switch-case. Позволяет обсудить критерии выбора между if и switch.

**Задача 3.2. Проверка авторизации с дополнительными условиями**. Программа запрашивает логин и пароль. Если логин и пароль верны, выводится «Доступ разрешен». Если логин верный, а пароль нет, выводится «Неверный пароль». Если логина нет в системе, выводится «Пользователь не найден».

Классический пример вложенного ветвления. Студенты учатся выстраивать логическую цепочку: сначала проверяется одно глобальное условие (существование логина), а внутри него – другое (корректность пароля).

**4. Комплексные задачи. Интеграция ветвлений в полноценные алгоритмы**. Финальный этап, на котором отрабатывается навык интеграции разветвляющихся алгоритмов в более сложные программы, включая циклы и обработку ввода.

**Задача 4.1. Поиск максимального из трех чисел**. Запросите у пользователя три целых числа и определите, какое из них является наибольшим.

Эта задача заставляет думать стратегически. Студенты должны решить, сравнивать ли все числа одновременно или использовать последовательные сравнения, что развивает алгоритмическое мышление.

**Задача 4.2. Простая валидация ввода в цикле**. Напишите программу, которая в цикле запрашивает у пользователя число, пока не будет введено число в диапазоне от 1 до 5. При вводе некорректного числа выводите соответствующее сообщение об ошибке.

Ключевая задача, показывающая практическое применение ветвлений. Условие if внутри цикла while используется для проверки корректности данных и управления потоком выполнения программы. Это мостик к теме циклов и обработки исключительных ситуаций.

**Заключение**. Предложенная система задач представляет собой не просто набор упражнений, а целостную методическую цепочку. Каждое новое задание вытекает из предыдущего, незначительно увеличивая сложность и вводя новый концептуальный элемент. Такой подход позволяет студенту постоянно находиться в «зоне ближайшего

развития», где задача является вызовом, но выполнимой. В результате у обучающегося формируется не фрагментарное знание синтаксиса, а целостное и глубокое понимание логики разветвляющихся алгоритмов, что служит прочным фундаментом для дальнейшего изучения программирования на C#.

**Литература**

1. Павловская Т.А. C#. Программирование на языке высокого уровня. – СПб.: Питер, 2009. – 432 с.

2. Информационно-справочная система ресурса Microsoft. URL: https://msdn.microsoft.com/uk-ua/library/ms173121(v=vs.90).aspx (дата обращения: 08.10.2025).

3. Ресурс METANIT.COM. URL: https://metanit.com/sharp/tutorial/ (дата обращения: 15.10.2025).

4. Ресурс internetурок. URL: https://interneturok.ru/lesson/algebra/podgotovka-k-ege/tema-

2-
uravneniya
-
i
-
neravenstva
/
kvadratnye
-
uravneniya
-
lineynye
-
uravneniya
-
praktika
(дата обращения:  16.10.2025).