МОУ СШ № 2

РАССМОТРЕНО	СОГЛАСОВАНО	УТВЕРЖДЕНО
на заседании ШМО учителей математики и информатики	заместитель директора по УВР	директором МОУ СШ № 2
Фефелова Н. Н.	Вахрина Е. Е.	Долгушина Т. В.
Протокол № 1 от «29» августа 2023 г.	Протокол № 1 от «31» августа 2023 г.	Приказ № 197 от «31» августа 2023 г.

МУНИЦИПАЛЬНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ " СРЕДНЯЯ ШКОЛА № 2"

Подписан: МУНИЦИПАЛЬНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ " СРЕДНЯЯ ШКОЛА № 2"
DN: C=RU, S=Ярославская область, STREET=мкр. Чкаловский д. 53, L=Переславль-Залесский, T=директор, O="МУНИЦИПАЛЬНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ""СРЕДНЯЯ ШКОЛА № 2""", ОГРН=1027601050751, СНИЛС=04570801648, OID.1.2.643.100.4=7608009105, ИНН=760801102410, E=glavbyx_ssh2@mail.ru, G=Татьяна Владимировна, SN=Долгушина, CN="МУНИЦИПАЛЬНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ""СРЕДНЯЯ ШКОЛА № 2"""
Основание: я подтверждаю этот документ своей удостоверяющей подписью
Местоположение: место подписания
Дата: 2023.09.07 14:01:21+03'00'
Foxit Reader Версия: 10.1.1

РАБОЧАЯ ПРОГРАММА

(ID 1475313)

Курса внеурочной деятельности «Введение в функциональное программирование»

для обучающихся 10 классов

Переславль-Залесский, 2023 год

Пояснительная записка

Курс «Программирование» направлен на обучение школьников программированию в функциональном стиле. Такой подход обеспечивает формирование иного стиля мышления и написания алгоритмов, более кратких и выразительных по сравнению с традиционно принятым императивным стилем. Это связано, прежде всего, с использованием понятия «функции» в математическом смысле, что позволяет более эффективно создавать программы расчётного характера, а также с возможностью проведения вычислений на уровне символов, а не чисел. В работе описаны преимущества, которые даёт предложенный подход, а также его основные особенности.

Способы рассуждения, которые применяются при решении тех или иных задач, в частности, в программировании, неформально принято делить на императивный и функциональный. Это связано с особенностями человеческого разума: кому-то удобно планировать свою деятельность последовательно по шагам, а кому-то удобно описать в деталях желаемый результат деятельности, не уделяя внимания пути достижения этого результата.

Одним из первых этапов развития программирования было создание ассемблерных языков, инструкции которых были напрямую связаны с архитектурой компьютера, на котором они исполнялись. В дальнейшем были созданы императивные языки высокого уровня (BASIC, Pascal, C, включая и объектно-ориентированные). Главным свойством таких языков являлось то, что программа представляла собой последовательное исполнение инструкций, оперирующих с памятью, и итеративные циклы. Вызовы функций и процедур, даже рекурсивные, не избавляли такие языки от явной императивности [9].

Программирование в императивном стиле подразумевает описание процесса вычисления в виде последовательности инструкций, возможно, с хранением промежуточных результатов. Этот подход в программировании влияет и на создание языков программирования, где написание программ представляет собой высокоуровневое кодирование инструкций, ориентированных на то, как решить задачу на компьютере. Каждая инструкция в такой программе является указанием на то, что нужно сделать на очередном шаге вычислений [1].

Программирование в функциональном стиле позволяет описывать более абстрактные и понятные решения задач (при этом приходится жертвовать скоростью вычислений). Создание функциональных программ заключается в написании утверждения, которое и является результатом программы, а весь процесс вычисления промежуточных значений игнорируется. Идея функционального подхода заключается в

том, что описывается сама задача, тот результат, который необходимо получить, а не процесс получения этого результата [1].

Основные методы, применяемые при написании функциональных программ — декомпозиция задачи на подзадачи, а также описание нерешенных проблем через уже решенные. Эти методы используются и в процедурном подходе, но не являются обязательными. Таким образом, элементы функционального стиля программирования присутствуют и в процедурном стиле.

Преимущества функциональных программ в том, что:

1) они обычно оказываются короче своих императивных аналогов;
2) функциональные программы легче поддаются формальному анализу;
3) параллельная реализация таких программ выполняется проще [1].

Владение как процедурным, так и функциональным стилем описания вычислительных процессов позволяет более широко рассматривать проектирование и разработку программных систем. К сожалению, при обучении программированию на уроках информатики рассматривается только процедурный подход, и это приводит к известному перекосу в сторону процедурного стиля, то есть к рассмотрению проектирования и реализации программных средств только с одной стороны.

Распространение и популяризация парадигмы функционального программирования позволит осуществлять более серьёзную и разностороннюю подготовку специалистов в области информационных технологий, а овладение основными приемами функционального подхода (декомпозиция и выражение нерешенных проблем через уже решенные) способствует развитию умения решать некоторые задачи в области управления.

Образовательная цель обучения по данному курсу – сформировать у обучающихся представление о функциональном подходе в программировании на основе решения задач.

Развивающая цель обучения по данному курсу – расширить кругозор обучающихся о подходах и методах, использующихся в программировании при решении задач различных типов.

Воспитательная цель обучения по данному курсу – способствовать формированию информационной культуры обучающихся при работе с текстами научного характера.

Курс рассчитан на обучающихся 10 классов, длительность курса 34 академических часа (1 час в неделю).

Тематическое планирование элективного предмета «Введение в функциональное программирование»

Тема	Общее количество часов	Количество часов на теорию	Количество часов на практику
Величины и выражения. Знакомство с интерпретатором Hugs. Вычисление выражений.	2	1	1
Функции. Композиция функций.	2	1	1
Типы данных в языке Haskell: логический, целочисленный, вещественный и символьный	6	2	4
Конструкции языка Haskell: охранные выражения, локальные определения, выбор варианта.	2	1	1
Рекурсивные функции. Локальные определения.	2	1	1
Списки, сопоставление с образцом, выражение списка, упорядоченные множества.	11	3	8
Полиморфные типы, классы типов, классы Eq, Ord.	3	2	1
Функции высшего порядка, бесконечные списки.	3	2	1
Интерактивный ввод-вывод информации. Работа с файлами.	3	1,5	1,5
Всего часов:	**34**	**14**	**20**

Поурочное планирование

1. Величины и выражения.
2. Закрепление материала по работе с интерпретатором Hugs
3. Функции в Haskell
4. Композиция функций
5. Числовые типы данных в языке Haskell
6. Решение задач с использованием целочисленного типа данных
7. Написание функций, использующих данные целого и логического типов
8. Практикум по созданию и тестированию функций
9. Решение задач с использованием условного оператора и охранных выражений
10. Создание функций с использованием оператора выбора вариантов (case)
11. Проверка знаний по теме «Написание функций, работающих с целым, вещественным и логическим типами данных»
12. Символьный тип данных
13. Рекурсивные функции
14. Решение задач с использованием рекурсии
15. Списки
16. Использование стандартных функций, определённых над списками
17. Написание функций, обрабатывающих списки
18. Диапазон
19. Сопоставление с образцом
20. Упорядоченные множества (кортежи) и списки, образованные из них
21. Выражение списка
22. Решение задач с использованием выражения списка
23. Практикум по созданию функций, обрабатывающих кортежи
24. Написание функций, обрабатывающих списки кортежей
25. Реализация функций над полиномами (контрольная работа)
26. Полиморфные типы данных
27. Классы типов
28. Класс Ord в Haskell
29. Функции высшего порядка
30. Некоторые функции высшего порядка из Prelude.hs
31. Проверочная работа по темам «Функции высшего порядка» и «Полиморфные функции»
32. Использование функций высшего порядка при работе со списками
33. Интерактивный ввод-вывод информации
34. Хранение информации в файлах

Результаты обучения по курсу

По окончании данного курса обучающиеся **должны знать и понимать** определения следующих понятий:

- тип данных,
- функция,
- выражение,
- вычисление выражения,
- рекурсивная функция,
- полиморфный тип данных,
- функция высшего порядка.

По окончании данного курса обучающиеся должны **уметь**:

- проектировать типы данных, используемых в функциях;
- описывать функции на языке Haskell;
- разбивать поставленную задачу на подзадачи, реализовывать функции, соответствующие подзадачам;
- повторно использовать код написанных функций.

По окончании данного курса обучающиеся должны **применять на практике**

- навыки вычисления рекурсивных функций;
- навыки вычисления алгебраических выражений с помощью функциональных программ.

Литература для учителя

1. Филд А., Харрисон П. Функциональное программирование: пер. с англ. М.: Мир, 1993, 637 с., ил.
2. Практика функционального программирования. Выпуск 2. Сентябрь 2009. Алгебраические типы данных и их использование в программировании. Душкин Р. с. 96 – 121.
3. Практика функционального программирования. Выпуск 3. Сентябрь 2009. Элементы функциональных языков. Кирпичёв Е. с. 97 – 233.
4. Роганова Н. А. Функциональное программирование: Учебное пособие для студентов высших учебных заведений – М.: ГИНФО, 2002. – 260 с.
5. Paul Hudak: The Haskell School of Expression. Cambridge University Press, New York, 2000, 416 pp.
6. Paul Hudak, John Peterson, Joseph H. Fasel: A Gentle Introduction to Haskell. www.haskell.org/tutorial
7. Ричард Берд. Введение в функциональное программирование на примере языка Haskell.
8. Практика функционального программирования. Выпуск 1.Июль 2009. Функции и функциональный подход. Душкин Р. с. 17 – 31.
9. Душкин Р.В. Лабораторный практикум по функциональному программированию «Инструментальное средство Hugs 98 для программирования на языке Haskell» Учебное пособие. – М.: Типография МИФИ, 61 с.