**УДК 378.14:004.43**

# ДВА ТИПА ЯЗЫКОВ ПРОГРАММИРОВАНИЯ

**Кобелев И.А.[1], Иванова Л.В.[1], Чекушина В.Е.[1]**

*[1]Елабужский Институт Казанского Федерального Университета, 423604, Татарстан, г. Елабуга, ул. Казанская, 89, e-mail: ivanova.lw@yandex.ru*

**В статье рассматривается проблема эффективного обучения программированию в вузе, так как фундаментальность подготовки в области программирования во многом определяется классификацией языков программирования. На данный момент существует большое разнообразие, как языков программирования, так и их классификаций. В работе предложена следующая классификация языков программирования: императивные и функциональные. Императивные языки программирования включают четыре группы: первая - ассемблер, интерпретаторы Бейсик, Фортран; вторая – Паскаль; третья – Си; четвертая – специализированные языки программирования. Возможным критерием классификации функциональных языков может являться – использование функций: узкого класса (Пролог) и разных классов (Хаскел). Построенная классификация используется авторами в рамках преподавания таких дисциплин, как: «Программирование», «Языки и методы программирования», «Искусственный интеллект», «Теория алгоритмов»**

Ключевые слова: программирование, язык программирования, классификация языков программирования, императивные, функциональные, аппликативные, объектно-ориентированные языки программирования.

# TWO TYPES OF PROGRAMMING LANGUAGES

**Kobelev I.A.[1], Ivanova L.V.[1], Chekushina V.E.[1]**

*[1]Elabuga Institute of the Kazan Federal University, 423604, Tatarstan, s. Elabuga, str. Kazan 89, e-mail: ivanova.lw@yandex.ru*

**The problem of effective teaching programming in high school, as a fundamental training in programming is largely determined by the classification of languages programming. At the moment, there are a variety of languages as a programming, and their classifications. The paper proposed the following classification of programming languages: imperative and functional.. Imperative programming languages include four groups: first - assembler, Basic interpreters, Fortran; second - Pascal; third - C; fourth - specialized programming languages. Possible criterion for classification of functional languages can be - use functions: small class (Prolog) and different classes (Haskell). Built classification used by the authors through the teaching of subjects such as: "Programming", "Languages and Programming Techniques", "Artificial Intelligence", "Theory of Algorithms"**

Keywords: programming, programming language, classification of languages Pro-program, imperative, functional, applicative, object-oriented programming languages.

Программирование – раздел информатики, занимающийся разработкой средств решения задач на компьютере и созданием программного обеспечения, с помощью которого реализуется вычислительный процесс и обмен информацией с внешним миром. Из всего многообразия языков программирования активно применяется только часть их, другие же стали историей языков программирования. В конструкциях различных языков программирования много общего, они классифицируются по своим возможностям, конструктивным особенностям и сферам применения. Программирование из искусства постепенно превращается в промышленное изготовление программ, для чего создаются разнообразные технологии программирования.

Наш многолетний опыт преподавания программирования в школе и в вузе показал, что начинать изучение языков программирования целесообразнее с их классификации, так как

это во многом определяет содержание обучения [4]. В результате анализа существующих классификаций мы четко выделяем две группы языков программирования.

Программа - это алгоритм, исполнителем которого может быть компьютер. Во многих учебниках по программированию или алгоритмизации не обращается достаточного внимания на следующую двойственность алгоритма:

1. Алгоритм может быть исполнен.
2. Алгоритм может быть записан.

В языке Бейсик четко выделены команды list – просмотреть запись текста программы и run – исполнить эту программу Язык программирования - возможность реализации записи алгоритма для компьютера.

В настоящее время количество языков программирования огромно. Имеется потребность их классификации. Попыток классифицировать их принималось много, например, в книге Т. Пратта и М.Зелковича [8] языки программирования разделены на четыре группы:

- императивные;
- аппликативные;
- основанные на системах правил;
- объектно-ориентированные.

Приведены описания этих групп языков, но четких признаков, по которым тот или иной язык программирования относится к определенной группе не выделено.

Приведенный в качестве примера языка, основанного на системах правил, Пролог вполне может рассматриваться как функциональный язык, в котором используются функции, возвращающие логическое значение. С другой стороны, совершенно не определено словосочетание «система правил». В любом языке программирования есть некоторая система правил, например, правил оформления записей.

Считаем также несостоятельным выделение в отдельную группу объектно-ориентированные языки. Мы также выделяем эти языки, но как подгруппу алгоритмических языков, поскольку имеется достаточно много общего с другими языками основной группы. Представляется, что все объектно-ориентированные языки программирования, есть набор библиотек самостоятельных программ, неважно, на каком языке написанных изначально, но они исполняются операционной системой. Фактически, объектно-ориентированный язык предоставляет возможность программисту использовать в своих проектах уже готовые объекты операционной системы. Просто правила конструирования новых проектов имеют в основе правила записи языков прародителей.

Вывод о том, что объектно-ориентированный язык программирования просто представляет собой возможность подключения к своим проектам ресурсов операционной систе-

мы, подтверждается тем, что во многих учебниках по таким языкам раздел о создании собственных объектов представлен очень слабо, а в некоторых учебниках вообще отсутствует. С другой стороны, уделяется большое внимание подключению и вопросам совместимости библиотек программ.

Предлагается разделить все языки программирования четко на две группы: императивные и функциональные.

Таблица 1

Классификация языков программирования

| | **Императивные** | **Функциональные** |
|---|---|---|
| **Синонимы** | процедурные, алгоритмические | аппликативные |
| **Память** | Система "склада". Изначально "склад" заполняется исходными данными, получаемыми непосредственно (с помощью специальной операции присваивания) или извне (с устройств ввода). Далее, данные берутся из одного "помещения" склада, обрабатываются (изменяются) и размещаются в другом. Результат - появление требуемых данных в указанном "помещении" склада (заполнение видеопамяти обеспечивает вывод ответа на мониторе ПК). | Отсутствие "склада". Для получения данных один "специалист" требует их непосредственно от другого "специалиста", тот от следующего. |
| **Переменная** | Фиксированное помещение выше упомянутого "склада". Всегда ***глобальны***, но придуманы способы "скрытия" отдельных "помещений", например системы "видимости" имен переменных, позволяющие иметь локальные переменные. | Информация, передаваемая от одного процесса обработки другому. Существует лишь для момента процесса передачи. Всегда ***локальны***. |
| **Теоретическая база** | Принципы фон-Неймана, машина Тьюринга. | Исчисление Черча, вычислимые функции. |
| **Режим работы** | Компиляционный. Высокое быстродействие. | Интерпретационный. Режимы реального времени. |
| | Описание действий (процедурность). | Описание возможностей (декларативность). |

Передаваемые (через "склад" или нет) данные обладают определенными свойствами (типы данных). По этому параметру императивные и функциональные не различаются.

Дальнейшую классификацию проводим для каждой группы отдельно.

Императивные языки программирования классифицируем по работе с переменными и подпрограммами, тенденция – приближение по возможностям к функциональным языкам.

Таблица 2

Классификация императивных языков программирования

| Переменные | Подпрограммы | Проблемы |
|---|---|---|
| Явно глобальные. Только базовые типы данных. | Слабо представлены | Необходимость следить за тем, чтобы в одну переменную не попадали разные данные, переприсваивание перед вызовом подпрограмм и после их исполнения.<br>Фиксирование некоторых имен переменных, работающих одинаково, например, до сих пор счетчики в циклах обозначаются i, j. |
| Появление областей видимости | Разделяются на процедуры и функции | Возможность применения рекурсии.<br>Появление сложных (структурированных) данных, которые пока невозможно возвращать как значения функций. |
| Хранение подпрограмм | Унификация подпрограмм – только функции. | Активное применение указателей.<br>Возвращение структурированных объектов подпрограммами, как адресов этих объектов.<br>Появление проблемы «очистки мусора». |
| Объединение в единое целое данных и средств их обработки (объекты). | | |
| Объектно-ориентированные языки программирования | | Использование программными системами уже имеющихся средств – объектов операционной системы. |

В языки программирования первой подгруппы получение информации обеспечивается, в основном, присваиванием. Появляется режим пакетной обработки информации. По сути, получение пакета – обобщение операции присваивания.

Для работы этих языков программирования операционная система не нужна. Средства языка достаточны для замены операционной системы. Много персональных компьютеров («Ямаха», «БК») выпускались с встроенным языком программирования, таким, как Бейсик или Фокал.

К языкам этой группы можно отнести ассемблер, интерпретаторы Basic, Фортран. Язык ассемблера можно и отделить, как имеющий слабую типизацию данных, «машинно-зависимый» язык. Фортран тоже занимает особую позицию, но с другой стороны: имеет достаточно серьезную типизацию данных.

Глобальность переменных приводит к необходимости большого количества переприсваиваний. Перед каждым входом в подпрограмму (GWbasic, Qbasic) выполняется заполнение переменных подпрограммы значениями из переменных основной программы и обратная передача информации после завершения подпрограмм. Недопустимость одноименности переменных в программах и подпрограммах является причиной ошибок в больших программах и затрудняет использование чужих подпрограмм. Во избежание этих трудностей, в языки программирования второй группы, каждой подпрограмме предоставляется своя область

определения, в том числе и имен. Появляется иерархия. При необходимости значения переменных берутся из области имен самой подпрограммы, если переменная в подпрограмме не объявлена, ищем в области определения подпрограммы, вызвавшей данную подпрограмму. Оказывается возможным применение рекурсии. В задачнике Пильщикова В.Н.[7] знаком повешенной трудности отмечено задание на вычисление значения Sin(Sin(…Sin(x))). Для языка Паскаль, поддерживающего рекурсию, данное задание трудностей не представляет. По-видимому, автор перерабатывал задания для языка Фортран, где, из-за недопустимости рекурсивных вызовов, при выполнении задания возникают проблемы.

Компиляционный режим работы требует наличия в памяти кода всех подпрограмм программы, но оперативная память компьютеров не безгранична. Появляется возможность создания оверлейных программ, когда часть оперативной памяти предоставляется для хранения отдельных модулей кода программы, заменяемых по необходимости.

Примерами языков программирования второй подгруппы являются: QBasic, Паскаль.

Во всех рассмотренных языках сохраняется четкое разделение элементов программы на команды и выражения. И при выполнении программ, и при вычислении выражений часто приходится выполнять одинаковые действия, что организуется в виде подпрограмм. Подпрограммы – элементы программ получили название «процедуры», фактически, после объявления – отдельные команды программы. Подпрограммы – элементы выражений, называют функциями. С развитием теории алгоритмов, появляется отождествление понятий «алгоритм» и «функция». Как только мы представим исполнение программы вычислением некоторого результата (результатом, кроме данных, может быть и действие), а исходные данные сопоставим с аргументами, разница между процедурами и функциями стирается.

Представителем языка программирования третьей подгруппы можем считать язык Си.

Представителем языков программирования четвертой подгруппы можем считать языки dBASE, FoxPro.

Работа с объектами появляется в очередных версиях языка Паскаль, Си и других.

Языки группы объектно-ориентированных, как уже отмечалось выше, по сути одно и то же. Традиционными остались формы записи основных алгоритмических структур и объявление переменных, как у предшественников, но других различий почти нет. Basic → VisualBasic, Pascal → Delphi, C → C++→C#. Существенную часть программ на этих языках составляют объекты системных библиотек, одних и тех же.

Попытаемся классифицировать функциональные языки. Возможен критерий - по используемым функциям.

Таблица 3

Классификация функциональных языков программирования

| | **Функции** | **Примеры** |
|---|---|---|
| 1 | Узкого класса | В языке программирования «Пролог» имеем функции, возвращающие результат только логического типа. |
| 2 | Разных классов | Язык программирования «Хаскел» работает с различными функциями. |

В математике известно заявление Пратта Т., что любое высказывание математики может быть записано предложениями «Если …, то …». Существует так же раздел «математическая логика», включающий «исчисление предикатов». Двоичная система, выдающая результат «да» или «нет», лежит в основе практически всех наук, от арифметики, до юриспруденции. Как видим, существуют широко используемые, замкнутые системы логических функций. Компьютерной реализацией оболочки подобной системы можем считать язык Пролог.

Если в алгоритмических языках мы описываем необходимые для получения результата действия, с указанием порядка их выполнения (детерминированность), то в пролог-системе мы просто перечисляем действия, которые можно выполнить (декларативность). Появляется новая категория – цель. Цель представляет собой так же некоторое действие. Основным вопросом программирования на языке Пролог является: возможно ли достичь поставленной цели, не выходя за рамки имеющихся правил? Сам путь достижения цели, важный пользователю, для пролог-системы вторичен.

Реализация поиска результата сопоставима с поиском пути в графе. В алгоритмических языках, для работы с иерархическими графами известен алгоритм перебора с возвратом. Используя этот алгоритм, можно составить список действий, выполнив который получаем результат. Базой функциональных языков является конструкция «список», основанная на понятии «монада». Список – есть набор из двух элементов: первый элемент - «голова», и «хвост» - снова такой же список остальных элементов. Для завершенности конструкции добавляем возможность «пустого» списка.

Первым из языков, предназначенным для работы со списками, считается Лисп. Лисп занимает двоякую позицию: с одной стороны, он может быть отнесен к алгоритмическим языкам, с другой, наличие аппарата работы со списками, позволяет решать задачи функциональных языков.

Языки Лисп и Пролог используются для создания систем искусственного интеллекта, представляющих собой наборы правил «Если …, то …», задачей которых является диагностика имеющейся ситуации и выполнение соответствующих действий для достижения цели.

Как в математике «математическая логика» не охватывает всех имеющихся возможностей, так и языков логического программирования недостаточно для решения различных задач. Появляются реализованные на функциональной основе возможности обработки не только логических данных.

В настоящее время существует несколько функциональных языков программирования, например, Душкин Р. В. [1] выделяет:

- **Lisp (List processor).** Считается первым функциональным языком программирования. Нетипизирован. Содержит массу императивных свойств, однако, в общем поощряет именно функциональный стиль программирования. Существует объектно-ориентированный диалект языка — CLOS.
- **Scheme**. Диалект Lisp'а, предназначенный для научных исследований в области computer science. При разработке Scheme был сделан упор на элегантность и простоту языка.
- **ML (Meta Language).** Семейство строгих языков с развитой полиморфной системой типов и параметризуемыми модулями.
- **Miranda**. Разработан в качестве стандартного функционального языка, использовавшего отложенные вычисления. Имеет строгую полиморфную систему типов. Оказал большое влияние на разработчиков языка Haskell.
- **Haskell**. Один из самых распространённых нестрогих языков. Имеет очень развитую систему типизации. Несколько хуже разработана система модулей. Последний стандарт языка — Haskell-98.
- **Gofer** (GOod For Equational Reasoning). Упрощённый диалект Haskell'а. Предназначен для обучения функциональному программированию.
- **Clean**. Специально предназначен для параллельного и распределённого программирования. По синтаксису напоминает Haskell. Clean использует отложенные вычисления.

С некоторой натяжкой, к функциональным языкам программирования, можно отнести системы редактирования (работа в интерпретационном режиме со специализированными данными – тексты, изображения). Как примеры, назовем HTML, Flash.

Представление о классификации языков программирования необходимы, например, при переводе алгоритмов с одного языка на другой.

**Заключение**

Фундаментальная подготовка будущих учителей информатики в нашем вузе на физико-математическом факультете осуществляется с 1987 года [5], с 2011 года – бакалавров по направлениям подготовки: 230700.62 Прикладная информатика и профилю подготовки «Прикладная информатика в экономике», 010200.62 Математика и компьютерные науки (профиль - «Математическое и компьютерное моделирование»).

Предложенный в данной работе подход к классификации языков программирования на две группы: императивные и функциональные апробирован в рамках таких дисциплин, как: «Программирование», «Языки и методы программирования», «Практикум решения задач на ЭВМ», «Информатика и программирование», «Теория и методика обучения информатике»,

«Искусственный интеллект», «Теория алгоритмов» [6]. Важность выбора языка программирования в фундаментальной подготовке специалистов в области информатики и программирования, методические аспекты эффективного обучения программированию рассматривались авторами в таких статьях, как [2-4].

## Список литературы


1. Душкин Р. В. Функциональное программирование на языке Haskell. Текст лекций по курсу «функциональное программирование» М. 2001
2. Иванова Л.В., Чекушина В.Е. Методы и формы обучения программированию в вузе. //Сборник научных трудов SWorld. -Выпуск 3. Том 17. -Одесса: КУПРИЕНКО СВ., 2013. -ЦИТ:313-0324. -С. 18-22.
3. Иванова Л.В. Методические аспекты обучения программированию будущих учителей информатики. Материалы XII Международной научно-методической конференции «Информатика: проблемы, методология, технологии», Воронеж, 9-12 февраля 2012, том 2, 92-94
4. Иванова Л.В. Методические аспекты преподавания дисциплины «Информатика и программирование». //Сборник научных трудов SWorld. -Выпуск 1. Том 13. -Одесса: КУПРИЕНКО СВ, 2014 - ЦИТ: 114 - 509. С. 41-43
5. Лаптев В.В., Швецкий М.В. Методическая система фундаментальной подготовки в области информатики: теория и практика многоуровневого педагогического университетского образования. - Спб.: Издательство Санкт-Петербургского университета, 2000. – 508 с.
6. Лапчик М.П. и др. Методика преподавания информатики: Учеб. пособие для студ. пед. вузов/ М.П.Лапчик, И.Г.Семакин, Е.К. Хеннер; Под общей ред. М. П. Лапчика. — М.: Издательский центр «Академия», 2001. — 624 с.
7. Пильщиков В.Н. Сборник упражнений по языку Паскаль: Учеб. пособие для вузов. –М.: Наука. Гл. ред. физ.-мат. лит., 1989.-160 с.
8. Пратт Т., Зелкович М. Языки программирования: разработка и реализация / Под общей ред. А. Матросова. – СПб.: Питер, 2002. – 688 с.: ил.



**Рецензенты:**

Ахметов Л.Г., д.п.н., профессор кафедры теории и методики профессионального образования, ЕИ К(П)ФУ, г. Елабуга.

Мухаметшин А.Г., д.п.н., профессор кафедры педагогики и психологии, декан факультета педагогики и психологии, НИСПТР, г. Набережные Челны.