Н. А. МОИСЕЕВА

ЯЗЫКИ И ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

ОМСК 2019

Министерство транспорта Российской Федерации
Федеральное агентство железнодорожного транспорта
Омский государственный университет путей сообщения

Н. А. Моисеева

ЯЗЫКИ И ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Утверждено методическим советом университета
в качестве учебно-методического пособия
к выполнению самостоятельной работы по дисциплинам
«Информатика», «Информационные технологии»,
«Основы программирования», «Прикладное программирование»

Омск 2019

УДК 004.42:004.43(075.8)
ББК 32.973.2я73
М74

Языки и технологии программирования: Учебно-методическое пособие к выполнению самостоятельной работы / Н. А. Моисеева; Омский гос. ун-т путей сообщения. Омск, 2019. 30 с.

Пособие содержит краткие теоретические сведения о языках программирования и эволюции их развития, описание наиболее известных классификаций языков программирования. Представлены сведения о системах, технологиях программирования и этапах их развития. Описаны базовые технологии программирования, являющиеся основополагающими при разработке программного обеспечения.

Предназначено для студентов первого и второго курсов всех специальностей и направлений подготовки очной и заочной форм обучения, может быть использовано в качестве самоучителя для любых категорий пользователей.

Библиогр.: 3 назв. Табл. 3. Рис. 6.

Рецензенты: доктор пед. наук, профессор З. В. Семенова;
канд. техн. наук, доцент А. Г. Малютин.

© Омский гос. университет
путей сообщения, 2019

ОГЛАВЛЕНИЕ

Введение 5
1. Языки и системы программирования 6
1.1. Базовые понятия 6
1.2. Поколения языков программирования 6
1.3. Классификация языков программирования 8
1.4. Системы программирования 11
1.5. Контрольные вопросы 14
2. Технологии программирования 14
2.1. Понятие «технология программирования». Этапы развития технологии программирования 14
2.2. Структурное программирование 17
2.3. Модульное программирование 19
2.4. Объектно-ориентированное программирование 20
2.5. контрольные вопросы 23
3. Тестовые вопросы 23
Библиографический список 25
Приложение. Обзор и краткая характеристика языков программирования 26

ВВЕДЕНИЕ

На современном этапе научно-технического прогресса и интенсивной информатизации общества невозможно представить высококвалифицированного специалиста, не владеющего информационными технологиями. Современный специалист любого профиля должен уметь получать, обрабатывать и использовать информацию с помощью компьютера, а также средств телекоммуникаций, в том числе иметь представление о языках программирования и владеть основами базовых технологий программирования. Знание языков и технологий программирования необходимо для понимания создания программных средств и приобретения базовых навыков их разработки.

Опыт ведения реальных разработок и совершенствование программных средств постоянно переосмысливается программистами, в результате чего появляются новые технологии и методы их реализации, которые, в свою очередь, служат фундаментом более современных средств разработки программного обеспечения. В основе любой такой технологии лежат базовые технологии программирования: структурная, модульная и объектно-ориентированная.

В настоящем учебно-методическом пособии рассматриваются основные теоретические сведения о языках программирования и эволюции их развития, наиболее известные классификации языков программирования, а также назначение и состав системы программирования. В издании представлены сведения о технологиях программирования и этапах их развития, описаны базовые технологии программирования, которые являются основополагающими при разработке программного обеспечения, приведены обзор и краткая характеристика различных языков программирования.

Данное учебно-методическое пособие предназначено для студентов 1-го и 2-го курсов очной и заочной форм обучения всех специальностей и направлений подготовки при изучении раздела «Алгоритмизация и программирование» в рамках дисциплин «Информатика», «Информационные технологии», «Основы программирования» и «Прикладное программирование» в соответствии с федеральными государственными образовательными стандартами высшего образования. Может быть использовано при изучении основных сведений о языках и технологиях программирования в качестве самоучителя любыми категориями пользователей.

1. ЯЗЫКИ И СИСТЕМЫ ПРОГРАММИРОВАНИЯ

1.1. Базовые понятия

Программирование – раздел информатики, изучающий методы и приемы составления программ для компьютеров. Под программированием понимается также процесс создания компьютерных программ.

Компьютерная программа – набор команд, позволяющих компьютеру выполнить поставленную задачу. Команды, предназначенные для компьютера, необходимо записывать с помощью языка программирования (ЯП).

Язык программирования – формализованный искусственный язык для представления алгоритма решения задачи в форме, пригодной для выполнения компьютером [1].

ЯП является ядром системы программирования (подразд. 1.4), в среде которой осуществляется разработка программного обеспечения (ПО). Весь процесс разработки ПО рассматривается как программирование, а людей, занимающихся этим видом деятельности, называют **программистами**. Разработку средств системного ПО и систем программирования принято называть **системным программированием**; разработку прикладных программ – **прикладным программированием**.

1.2. Поколения языков программирования

Выделяют пять поколений ЯП [1, 2], которые отражают их эволюцию (табл. 1).

Таблица 1

Поколения языков программирования

Поколение (период)	Языки программирования	Краткая характеристика
1	2	3
Первое (начало 50-х гг.)	Машинные (машинно-зависимые)	Ориентированы на использование в конкретной ЭВМ. Сложны в освоении. Требуют хорошего знания организации и функционирования архитектуры конкретного типа ЭВМ
Второе (конец 50-х – начало 60-х гг.)	Ассемблеры, макроассемблеры	Более удобны в использовании, но по-прежнему машинно-зависимы

Окончание табл. 1

1	2	3
Третье (60-е гг. – по настоящее время)	Процедурные ЯП высокого уровня	Не зависят от архитектуры ЭВМ. Используются мощные синтаксические конструкции. Ориентированы на алгоритм. Создание и выполнение программы осуществляется в системе программирования. Используются для решения задач из любых предметных областей
Четвертое (начало 70-х гг. – по настоящее время)	Объектно-ориентированные, параллельные ЯП высокого уровня, скриптовые языки, языки запросов	Предназначены для реализации крупных проектов, повышают их надежность и скорость создания. Ориентированы на специализированные области применения
Пятое (середина 90-х гг. по настоящее время)	Декларативные, визуальные объектно-ориентированные ЯП высокого уровня	Ориентированы на создание графического интерфейса пользователя[1], интеллектуальных систем и технологий. Сюда относят языки искусственного интеллекта, а также многие объектно-ориентированные ЯП, реализуемые как системы автоматического создания программ с помощью визуальных средств разработки

Обзор и краткая характеристика наиболее значимых ЯП в истории программирования и его актуального состояния представлены в приложении.

Подавляющее большинство ЯП высокого уровня (ЯПВУ) успешно применяется и в настоящее время. Структуру любого ЯПВУ можно отобразить с помощью схемы, представленной на рис. 1.

Каждый ЯПВУ, равно как и «естественный» язык (русский, английский и т. д.), имеет алфавит, лексику (словарный запас), грамматику, синтаксис и семантику.

Алфавит – фиксированный набор символов, допускаемых для составления текста программы на ЯП.

[1] Графический интерфейс пользователя (англ. *graphical user interface, GUI*) – разновидность пользовательского интерфейса, в котором элементы интерфейса (меню, кнопки, списки и т. п.), представленные пользователю на дисплее, исполнены в виде графических изображений.

Лексика – совокупность правил образования цепочек символов (лексем), образующих идентификаторы (переменные и метки), ключевые слова, операции и другие лексические компоненты языка.

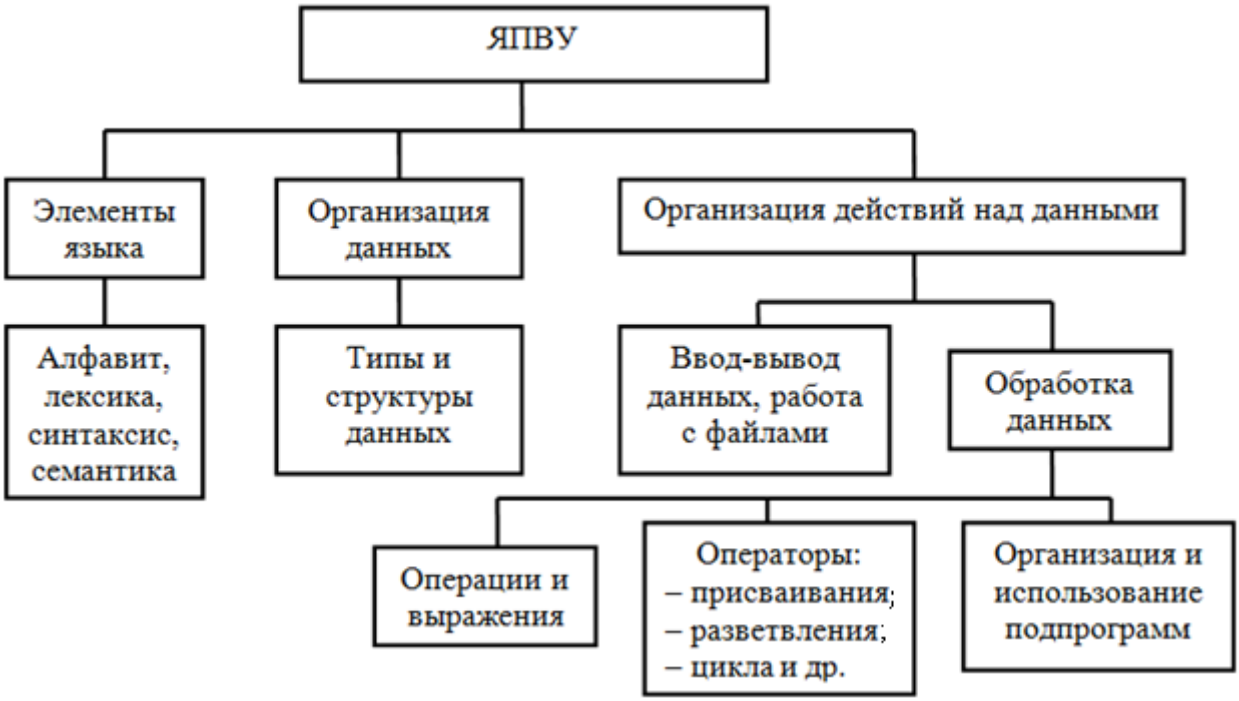

Рис. 1. Структура ЯПВУ

Синтаксис – система правил записи допустимых конструкций ЯП из букв алфавита.

Семантика – система правил однозначного толкования отдельных языковых конструкций, позволяющих воспроизвести процесс обработки данных.

1.3. Классификация языков программирования

Строгой классификации ЯП не существует, поэтому в зависимости от классификационного признака можно провести их условную классификацию (табл. 2).

Рассмотрим более подробно классификацию ЯП по *парадигме программирования*, или по *функциональному признаку*.

Парадигма программирования – совокупность идей и понятий, определяющих стиль или подход к программированию. Иными словами, парадигма программирования определяет способ организации программы, т. е. принцип ее построения. Основные парадигмы программирования представлены в табл. 2.

В настоящее время существует четкое разделение ЯП на процедурные (императивные) и непроцедурные (декларативные). Отдельным классом выделяют объектно-ориентированные ЯП, хотя они содержат элементы процедурного программирования.

Таблица 2

Классификация языков программирования

Признак классификации	Группы		Примеры
Близость к аппаратному обеспечению ЭВМ	Языки низкого уровня[2] (машинно-ориентированные)		Машинные коды, Автокод, Ассемблер
	Языки высокого уровня (машинно-независимые)		Basic, C, Pascal
	Языки сверхвысокого уровня		Алгол-68, APL
Парадигма программиро-вания	Процедурные (императивные)	Операционные	Ассемблер, Fortran, Basic, C
		Структурные	Pascal, Modula, ADA
	Непроцедурные (декларативные)	Функциональные	Lisp, Haskell, APL
		Логические	Prolog, Planner
	Объектно-ориентированные		SmallTalk, Simula, Java, C++

Процедурная (императивная) парадигма программирования описывает процесс вычисления в виде инструкций, которые необходимо выполнить, а результат задается только способом его получения при помощи некоторой процедуры, представляющей собой определенную последовательность действий.

Процедурные ЯП требуют от программиста детального описания того, как решать задачу, т. е. формулировки алгоритма и его записи. При этом ожидаемый результат не указывается. Основные понятия языков этой группы – оператор (команда) и данные. В процедурном подходе операторы объединяются в процедуры.

[2] «Низкий» уровень языка означает, что операторы этого ЯП близки к машинному коду и ориентированы на команды процессора конкретного типа.

Считается, что основой всех ЯП являются процедурные языки, поскольку в основе работы ЭВМ на самом низком уровне лежит возможность исполнять только примитивные команды, явно указывающие, что делать процессору. Языки декларативного типа можно рассматривать как надстройки над процедурными ЯП. В этих ЯП заложены и реализованы определенные математические модели, позволяющие более эффективно программировать некоторые специфические типы задач.

Непроцедурная (декларативная) парадигма программирования описывает результат, а не то, как его получить. При использовании декларативного языка программист указывает исходные информационные структуры, взаимосвязи между ними и то, какими свойствами должен обладать результат. Функциональные и логические ЯП получили широкое применение в системах автоматизированного проектирования, в моделировании, системах искусственного интеллекта.

Языковые конструкции *функциональных* ЯП состоят из элементарных функций, на основе которых программист должен создавать более сложные функции, необходимые для решения поставленной задачи. Языки *логического* программирования основаны на разделе дискретной математики, изучающей принципы логического вывода информации на основе заданных фактов и правил вывода.

Объектно-ориентированное программирование – парадигма программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования.

На сегодняшний день для создания программ используются все рассмотренные парадигмы программирования, но наибольшее распространение получило объектно-ориентированное программирование. При этом наблюдается интеграция подходов, поскольку они существуют и развиваются в рамках ЯП, чтобы обеспечивать удобные механизмы решения различных прикладных задач.

Большинство современных ЯП аккумулируют в себе элементы и приемы нескольких подходов, и тем не менее их можно классифицировать по основному их ядру, реализующему приемы определенной парадигмы программирования. В настоящее время существует ряд ЯП, в которых объединены несколько парадигм:

– процедурная и объектно-ориентированная: C++, C#, Delphi, Java и др.;

– процедурная, функциональная и объектно-ориентированная: JavaScript, Perl и др.;

– логическая и объектно-ориентированная: Object Prolog.

1.4. Системы программирования

Система программирования – комплекс языковых и программных средств, предназначенных для автоматизации процесса составления, отладки программы и подготовки ее к выполнению на компьютере.

В системе программирования (СП) с помощью ЯП создается текст программы, описывающий разработанный алгоритм. Процесс создания программы в СП включает в себя следующее:

1) ввод и редактирование текста программы на соответствующем ЯП;

2) трансляция исходного кода программы;

3) отладка программы.

Для выполнения этих действий применяются специальные средства, которые входят в состав стандартной СП:

1) текстовый редактор;

2) библиотеки стандартных подпрограмм и функций;

3) транслятор с ЯПВУ;

4) редактор связей (компоновщик);

5) отладчик.

Текстовый редактор необходим для создания и редактирования исходного кода программы на ЯП.

Библиотеки стандартных подпрограмм – совокупность часто используемых подпрограмм в виде готовых объектных модулей.

Транслятор предназначен для преобразования программы, написанной на ЯП, в программу на машинном языке. Такой перевод программы с ЯП на язык машинных кодов называется **трансляцией,** а программа называется **исходным модулем** (рис. 2). Существует два вида трансляторов:

интерпретатор производит пооператорную обработку и выполнение исходного модуля;

компилятор преобразует всю программу в модуль на машинном языке, после этого программа записывается в память компьютера и лишь потом исполняется. Сам процесс трансляции в этом случае называется **компиляцией**.

Редактор связей (компоновщик) связывает полученные после трансляции объектные модули и подключенные библиотечные подпрограммы и функции, затем создает готовый к исполнению загрузочный модуль (EXE-файл) (см. рис. 2).

В качестве входной информации для редактора связей трансляторы применяют исходные модули и формируют в результате своей работы **объектный модуль.**

Этот модуль содержит текст программы на машинном языке и дополнительную информацию, обеспечивающую настройку модуля по месту его загрузки и объединение этого модуля с другими независимо оттранслированными модулями в единую программу. С помощью программы, называемой **загрузчиком**, для выполнения программы загрузочный модуль помещается в оперативную память.

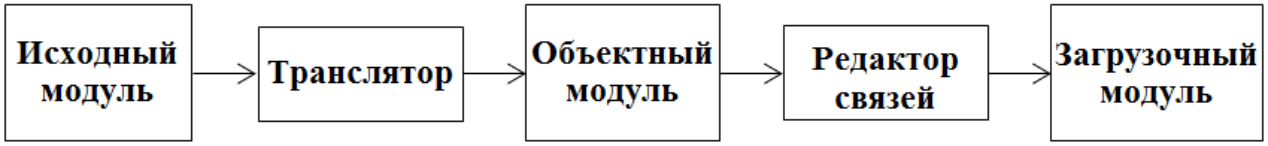

Рис. 2. Процесс создания программы, готовой к исполнению

Отладчик – инструмент для поиска и исправления ошибок в программе.

СП различаются по тому, какой ЯП они реализуют. Ведущими разработчиками СП являются компании Embarcadero, Microsoft.

Если компоненты СП имеют общий интерфейс пользователя и представляют собой части одного приложения, то такая СП называется **интегрированной.** Часто ее называют **IDE (Integrated Developed Environment – интегрированная среда разработки)**. В стандартную поставку IDE, как правило, входят текстовый редактор, библиотеки стандартных функций, компилятор, редактор связей, а также специализированный текстовый редактор, в котором выделяются ключевые слова различными цветами и шрифтами. Все этапы создания программы в IDE автоматизированы: после того, как исходный текст программы введен, его компиляция и сборка осуществляются одним нажатием клавиши [2]. Примеры IDE: JetBrains, NetBeans, PascalABC.NET.

Дальнейшее развитие вычислительной техники и появление графических интерфейсов приложений привели к созданию **среды RAD (Rapid Application Development – среды быстрого проектирования)**, в которой программирование по сути заменяется проектированием. В RAD-среде программист проектирует некоторую часть с применением визуальных средств (элементов управления), например, экранную форму как основу интерфейса будущего приложения (рис. 3). Исходный текст программы, ответственный за работу этих элементов, генерируется автоматически с помощью RAD-среды.

Результатом визуального проектирования является заготовка будущей программы, в которую уже внесены соответствующие коды. Подобный подход называется *визуальным программированием*.

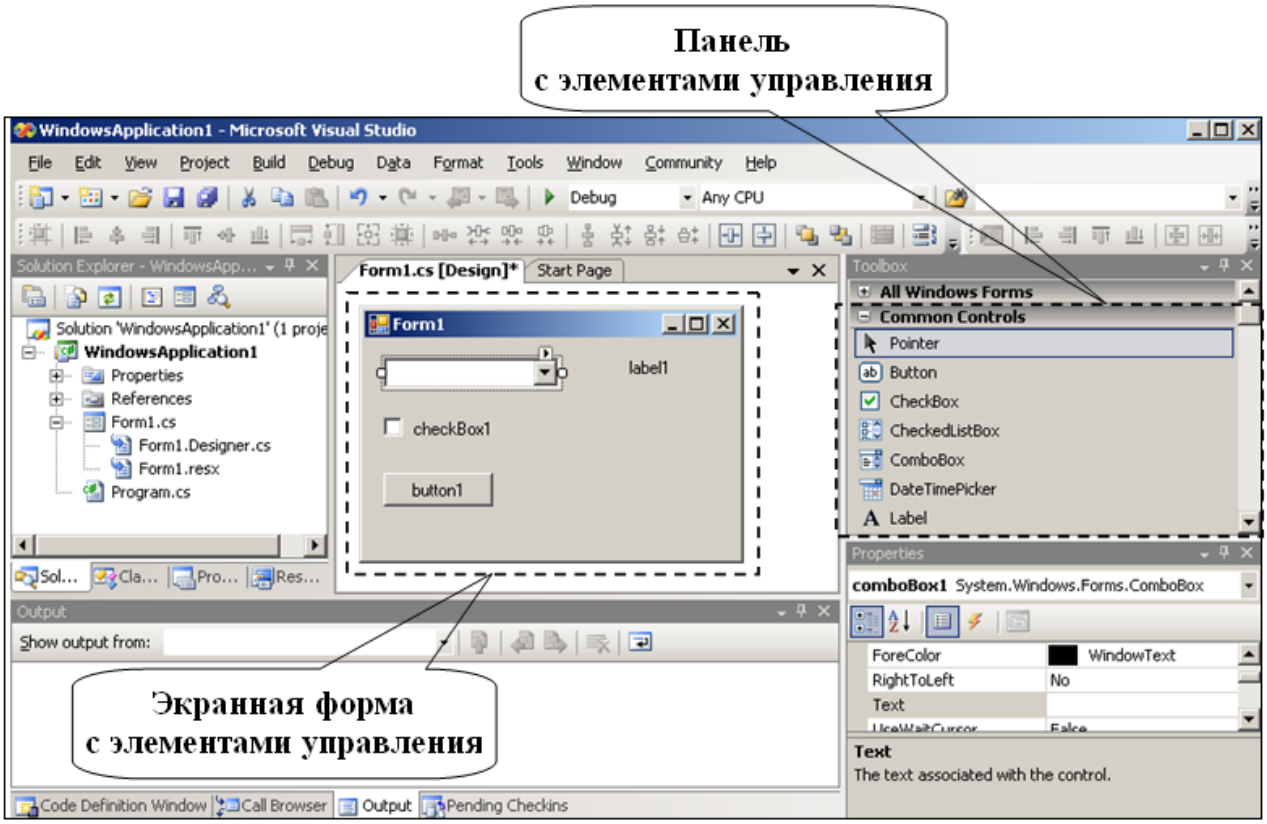

Рис. 3. Пример реализации концепции визуальной среды RAD

RAD-системы используют ЯП третьего и четвертого поколений. В табл. 3 сопоставлены наиболее популярные ЯП и соответствующие им визуальные среды быстрого проектирования программ для операционной системы Windows.

Таблица 3

Примеры наиболее популярных ЯП

Наименование ЯП	Визуальная среда быстрого проектирования
Basic	Microsoft Visual Basic
Pascal	Embarcadero Delphi, Lazarus
C++	Embarcadero C++ Builder, Microsoft Visual C++
Java	Embarcadero JBuilder, Microsoft Visual J++
Prolog	Visual Prolog

Существуют также среды визуального программирования, предназначенные для нескольких ЯП, например, Microsoft Visual Studio включает в себя такие ЯП, как Visual Basic, Visual C++, Visual C#, Visual Fox Pro.

1.5. Контрольные вопросы

1) Дайте определение языка программирования.

2) Сколько поколений языков программирования выделяют? Дайте им краткую характеристику.

3) Опишите классификации языков программирования.

4) Что такое парадигма программирования? Дайте краткую характеристику каждой парадигме программирования.

5) Что такое система программирования?

6) Что такое транслятор? Какой процесс называется трансляцией?

7) Что такое компилятор и чем он отличается от интерпретатора?

8) В чем различие между языками высокого и низкого уровня?

9) Какие компоненты входят в состав системы программирования?

10) Что такое интегрированная среда разработки?

2. ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

2.1. Понятие «технология программирования». Этапы развития технологии программирования

Технология программирования – совокупность методов и средств, используемых в процессе разработки ПО. Иначе говоря, технология программирования – способ создания программы. В процессе разработки программных средств используются различные технологии программирования (ТП).

Исторически в развитии ТП выделяют несколько этапов.

I этап. **«Стихийное» программирование (с момента появления первых ЭВМ до середины 60-х гг. XX в.)**. Развитие программирования шло по пути замены машинных языков ассемблерами, а затем процедурными ЯП и использования подпрограмм, что повысило производительность труда программиста.

Слабое место такой архитектуры программы – при увеличении количества подпрограмм возрастала вероятность искажения части данных, которые не делились на глобальные[3] и локальные[4] данные подпрограмм. Это предопределило возникновение «кризиса программирования» в 60-е гг. XX в. Выход из этого кризиса – переход к структурному программированию.

II этап. **Структурное программирование (60-е – 70-е гг. XX в.).** Структурный подход к программированию представлял собой совокупность технологических приемов, охватывающих выполнение всех этапов разработки ПО. В основе структурного подхода лежит *декомпозиция* (разбиение на части) сложных систем с целью последующей их реализации в виде отдельных небольших подпрограмм. Структурный подход к программированию требовал представления задачи в виде иерархии подзадач простейшей структуры. Рекомендовался специальный метод проектирования алгоритмов – **метод пошаговой детализации,** когда проектирование алгоритма разбивается на ряд последовательных шагов.

Дальнейший рост сложности и размеров разрабатываемого ПО потребовал развития структурирования данных. Как следствие этого в ЯП появляется возможность определения пользовательских типов данных. Одновременно усилилось стремление разграничить доступ к глобальным данным программы, чтобы уменьшить количество ошибок, возникающих при работе с глобальными данными.

В результате появилась и стала развиваться **технология модульного программирования,** которая предполагает выделение групп подпрограмм, использующих одни и те же глобальные данные, в отдельно компилируемые модули. Структурный подход в сочетании с модульным программированием позволяет получить надежные программы, размер которых не превышает 100000 операторов.
Однако при увеличении размера программы возросла сложность межмодульных интерфейсов, ошибки в которых трудно обнаружить по причине раздельной компиляции модулей (ошибки выявляются только при выполнении программы). Для разработки ПО большого объема был предложен *объектный подход* к программированию, в результате чего появилась технология объектно-ориентированного программирования (ООП).

---

[3] Глобальные данные – это данные, которые описываются в основной программе перед описанием подпрограмм. Областью действия глобальных данных является любая подпрограмма. Это означает, что глобальные данные могут участвовать в каких-либо операциях любой подпрограммы.

[4] Локальные данные – это данные, которые описываются и используются в пределах одной подпрограммы. Использование этих данных за пределами подпрограммы невозможно.

III этап. **Объектно-ориентированное программирование (с середины 80-х до конца 90-х гг. XX в.)**. ООП определяется как технология создания сложного ПО, основанная на представлении программной архитектуры в виде совокупности объектов, каждый из которых является экземпляром определенного типа (класса), а классы образуют иерархию объектов с наследованием свойств.

Основное преимущество ООП по сравнению с модульным программированием – более естественная декомпозиция ПО, которая значительно облегчает его разработку. Кроме того, объектный подход предлагает новые способы организации программ, основанные на механизмах наследования, полиморфизма, композиции. Развитие объектного подхода в технологии программирования привело к созданию сред визуального программирования.

IV этап. **Компонентный подход и CASE-технологии (с середины 90-х гг. XX в. по настоящее время**). Компонентный подход предполагает построение ПО из отдельных компонентов, т. е. физически отдельно существующих частей ПО.

Основы компонентного подхода были разработаны компанией Microsoft, начиная с технологии OLE (Object Linking and Embedding – связывание и внедрение объектов), которая применялась в ранних версиях операционной системы Windows для создания составных документов. Ее развитием стало появление COM-технологии (Component Object Model – компонентная модель объектов), а затем ее распределенной версии DCOM, на основе которых были разработаны компонентные технологии, решаются различные задачи разработки ПО. Среди них выделяют OLE-automation – технологию создания программируемых приложений, обеспечивающую доступ к внутренним службам этих приложений. На основе OLE-automation создана технология ActiveX, предназначенная для создания ПО как сосредоточенного на одном компьютере, так и распределенного.

Компонентный подход лежит также в основе технологии CORBA (Common Object Request Bracer Architecture – общая архитектура с посредником обработки запросов объектов). Программное ядро CORBA реализовано для всех основных аппаратных и программных платформ и обеспечивает создание ПО в гетерогенной вычислительной сети[5].

[5] Гетерогенная компьютерная сеть – вычислительная сеть, соединяющая ЭВМ и другие устройства с различными операционными системами или протоколами передачи данных. Например, гетерогенной является локальная вычислительная сеть, соединяющая ЭВМ под управлением операционных систем Windows, Linux и MacOS.

Важнейшая особенность современного этапа ТП – широкое использование компьютерных технологий создания и сопровождения программных систем на всех этапах их жизненного цикла. Эти технологии получили название **CASE-технологий (ComputerAided Software/System Engineering – разработка программного обеспечения/программных систем с использованием компьютерной поддержки)**. Сегодня существуют CASE-технологии, поддерживающие как структурный, так и объектный, в том числе компонентный, подходы к программированию. Практически все промышленно производимое сложное ПО разрабатывается с использованием CASE-средств.

В настоящее время компонентный подход в совокупности с технологией ООП и CASE-технологией применяется в разработке программных средств.

Рассмотрим базовые ТП более подробно.

2.2. Структурное программирование

Структурное программирование – методология разработки ПО, в основе которой лежит представление программы в виде иерархической структуры блоков. Идеи структурного программирования появились в начале 70-х гг. XX в. в компании IBM, в их разработке участвовали известные ученые Э. Дейкстра, Х. Милс, Э. Кнут, С. Хоор.

Рассмотрим принципы технологии структурного программирования.

1. Любая программа представляет собой структуру, построенную из трех *базовых управляющих конструкций*: следование (т. е. линейная последовательность действий), ветвление и цикл.

2. Отказ от средств управления последовательностью выполнения управляющих конструкций, например, *оператора безусловного перехода GoTo*;

3. Исходный код программы имеет *модульную структуру*. Это значит, что программа разбита на более мелкие единицы, т. е. подпрограммы (функции и процедуры). Комбинируя эти подпрограммы, удается формировать итоговый алгоритм уже не из простых операторов, а из законченных блоков кода, имеющих определенную смысловую нагрузку, причем обращаться к таким блокам можно по названиям.

4. Разработка программ с подпрограммами ведется по двум методикам.

Нисходящее проектирование (программирование «сверху вниз») – методика разработки программ, при которой сначала определяются цели решения проблемы, после чего идет последовательная детализация, завершающаяся детальной программой (метод пошаговой детализации).

Сначала выделяются несколько подпрограмм, решающих самые глобальные задачи (например, инициализация данных, главная часть и завершение), потом каждый из этих модулей детализируется на более низком уровне, разбиваясь на небольшое число других подпрограмм, и так происходит до тех пор, пока вся задача не окажется реализованной. В данном случае программа конструируется иерархически (сверху вниз): от главной программы к подпрограммам самого нижнего уровня, причем на каждом уровне используются только простые последовательности инструкций, циклы и условные разветвления.

Восходящее проектирование (программирование «снизу вверх») – методика разработки программ, начинающаяся с разработки подпрограмм анализа того, как решения этих задач могут обеспечить решение более сложной задачи. Такой подход стал возможным, поскольку в настоящее время имеются готовые программные модули, предназначенные для решения большого числа частных задач. Такая методика является менее предпочтительной по сравнению с нисходящим программированием, так как часто приводит к нежелательным результатам, переделкам и увеличению времени разработки.

К положительным свойствам структурного программирования относят

повышение надежности программ (благодаря хорошему структурированию при проектировании программа легко поддается тестированию и не создает проблем при отладке);

повышение эффективности программ (структурирование программы позволяет легко находить и корректировать ошибки, а отдельные подпрограммы можно переделывать (модифицировать) независимо от других);

уменьшение времени и стоимости программной разработки;

улучшение читабельности программ.

Структурное программирование используется при создании средних по размеру приложений (несколько тысяч строк исходного кода).

Структурное программирование реализовано в таких ЯП, как Ada, C, Basic, Fortran, Pascal и др.

Дальнейшим развитием структурного программирования является модульное программирование.

2.3. Модульное программирование

Модульное программирование предполагает выделение группы подпрограмм, использующих одни и те же глобальные данные, в отдельно компилируемые программные модули (библиотеки подпрограмм). В модульном программирования под **модулем** подразумевают комплекс подпрограмм, который находится в отдельном файле и может быть использован другими программами. Одни модули вызывают другие, начиная с самого верхнего первого модуля, называемого корневым модулем, или головной программой.

Модуль характеризуют:

1) один вход и один выход (на входе модуль получает определенный набор исходных данных, выполняет содержательную обработку и возвращает один набор результатных данных);

2) функциональная завершенность (модуль выполняет перечень операций для реализации каждой отдельной подпрограммы);

3) логическая независимость (результат работы модуля зависит только от исходных данных, но не зависит от работы других модулей);

4) слабые информационные связи с другими программными модулями (обмен информацией между модулями должен быть минимизирован);

5) относительно небольшой по размеру и сложности.

Рассмотрим принципы модульного программирования.

1. Используется *принцип модульности*, согласно которому большие и сложные программы разрабатываются и отлаживаются по модулям, которые затем объединяются в единый комплекс.

2. Соблюдается *принцип нисходящего проектирования* задачи «сверху вниз» – сначала определяются состав и подчиненность функций, а затем набор программных модулей, реализующих эти функции. При программировании «сверху вниз» алгоритмы и данные делятся на модули. Некоторые из модулей являются стандартными и поставляются в составе ЯП, к примеру, вычисление элементарных математических функций квадратный корень, логарифм и т. д. Программист может не только использовать стандартные библиотеки процедур, но и создавать свои собственные модули.

3. *Стандартизация интерфейса* (взаимодействия) между отдельными программными модулями является основной чертой модульного программирования. Например, при использовании библиотечных подпрограмм всегда возникает проблема их состыковки с головной программой. Для облегчения этой работы были разработаны стандарты, которые позволяют записывать библиотечные подпрограммы в форме, максимально облегчающей такую состыковку.

Связи между модулями осуществляются через специально разрабатываемый интерфейс, в то время как доступ к реализации модуля (телу подпрограмм) запрещен.

К преимуществам модульного программирования относят следующее:

– большую программу могут разрабатывать одновременно несколько программистов, что позволяет раньше закончить задачу;

– возможность отдельной отладки каждого модуля;

– использование одного и того же модуля для разных наборов входных данных;

– облегчается процесс написания и тестирования программ, уменьшается стоимость их сопровождения.

Узким местом модульного программирования является то, что при увеличении размера программы возрастает сложность межмодульных интерфейсов, и с некоторого момента предусмотреть взаимовлияние отдельных частей программы становится практически невозможным.

Модульного программирование реализовано в таких ЯП, как Pascal, C, Python, Perl и др.

2.4. Объектно-ориентированное программирование

ООП – методология программирования, основанная на представлении программы в виде совокупности объектов. В рамках объектно-ориентированного подхода программа представляет собой совокупность связанных объектов, каждый объект – это набор некоторых данных и набор действий, которые он умеет делать.

Взаимосвязь основных понятий ООП представлена на рис. 4.

Объект – совокупность свойств, методов их обработки и событий, на которые этот объект может реагировать и которые, как правило, приводят к изменению свойств объекта[6].

Для описания объекта используется класс, который определяет свойства и методы объекта, принадлежащие данному классу.

Класс – шаблон, на основе которого создается конкретный программный объект, описываются свойства и методы, определяющие поведение объектов этого класса. Каждый конкретный объект, имеющий структуру этого класса, называется *экземпляром класса*. Например, от класса «кассир железнодорожной кассы» могут быть получены экземпляры класса: «кассир кассы № 1», «кассир кассы № 2» и т. д.

[6] Объект «Забор» имеет свойства: «Длина», «Высота», «Цвет». Метод «Окраска» меняет цвет забора. Событие «Дождь» может изменить цвет забора.

Свойства – перечень характеристик (атрибутов) объекта, присущих объектам (например, размер шрифта, цвет фона и др.). Через свойства реализуется возможность получения доступа к информации, которая хранится в объекте.

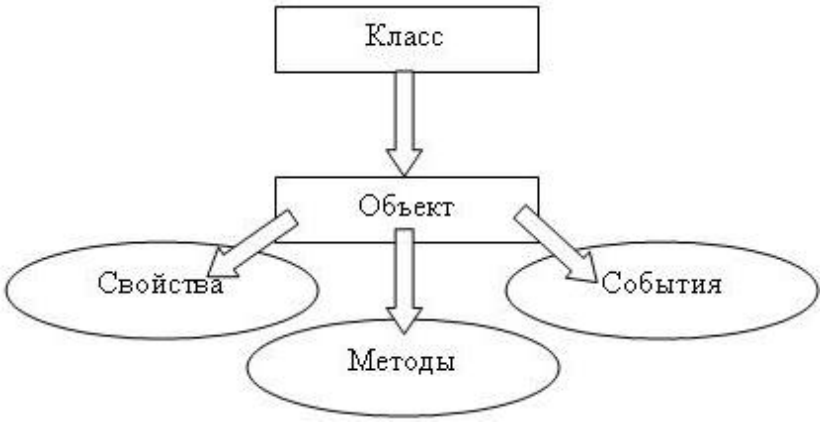

Рис. 4. Взаимосвязь основных понятий ООП

Метод – действия, которые выполняет объект. Методы объекта выполняются при наступлении заранее определенных событий, например, однократное нажатие левой кнопки мыши, вход в поле ввода, выход из поля ввода, нажатие определенной клавиши и т. п.

Событие – характеристика класса объекта, описывающая внешнее воздействие, на которое реагирует объект этого класса во время работы приложения. В ООП различают внешние и внутренние события. *Внешние события* генерируются пользователем (например, клавиатурный ввод или нажатие кнопки мыши, выбор пункта меню, запуск макроса); *внутренние события* – операционной системой или работающей программой.

Рассмотрим основные принципы ООП.

1. **Инкапсуляция** – принцип, согласно которому любой класс должен рассматриваться как «черный ящик», т. е. пользователь класса может видеть и использовать только свойства и методы класса, а его внутренняя реализация скрыта от пользователя. В ООП доступ к объекту возможен только через его методы и свойства.

Рассмотрим в качестве иллюстрации принципа инкапсуляции кнопку – типичный объект, присутствующий в интерфейсе большинства программных средств. Кнопка имеет некоторые свойства (например, ширина, высота и надпись) и характеризуется определенным поведением, т. е. она может быть нажата пользователем, после чего будут происходить определенные действия. Соединение таких свойств и поведения в одном объекте называется инкапсуляцией.

2. **Наследование** – организация классов, которая предусматривает создание новых классов на базе существующих и позволяет классу-потомку наследовать все свойства класса-родителя. Наследование поддерживает концепцию иерархии классов.

Например, класс-родитель Транспорт можно описать такими базовыми свойствами, как *цвет* и *модель*. При разделении данного класса на наземные, морские и воздушные транспортные средства мы видим, что класс-потомок наследует свойства класса Транспорт, но имеет определенный набор свойств, характеризующих специфику этого транспортного средства (рис. 5).

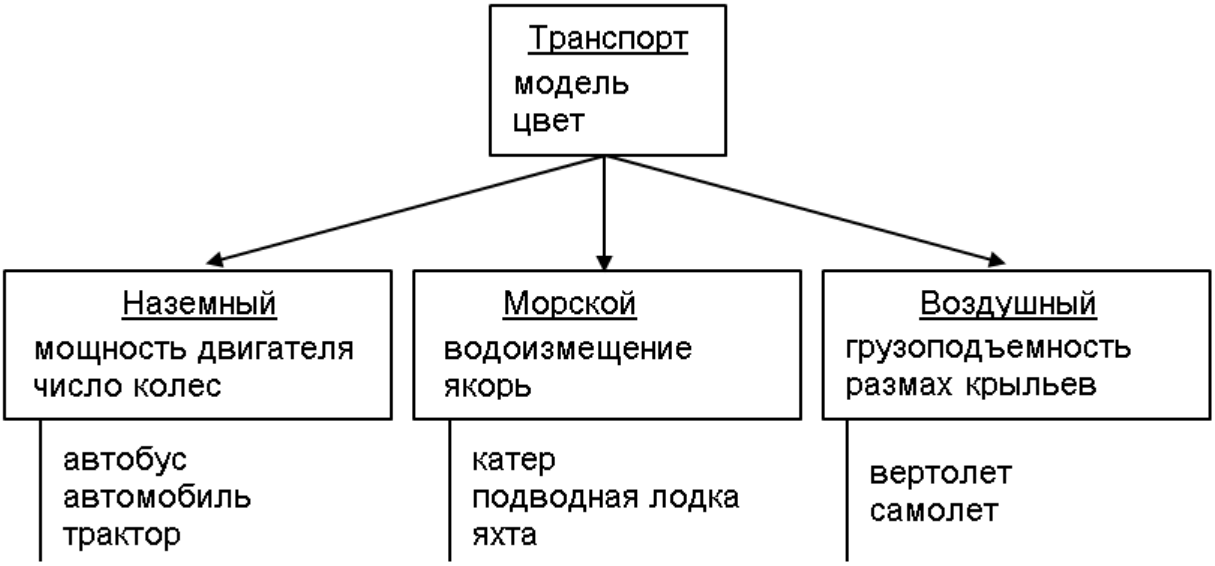

Рис. 5. Пример реализации принципа наследования

3. **Полиморфизм** – способность объектов с одинаковой спецификацией иметь различную реализацию. Например, метод «нарисовать» должен по-разному реализовываться для родственных классов «точка», «прямая», «круг», «прямоугольник».

Например, в программе рисования графических примитивов, написанной на языке ООП, при вызове одного и того же метода Draw класс Rect рисует прямоугольник, а класс Round – круг (рис. 6).

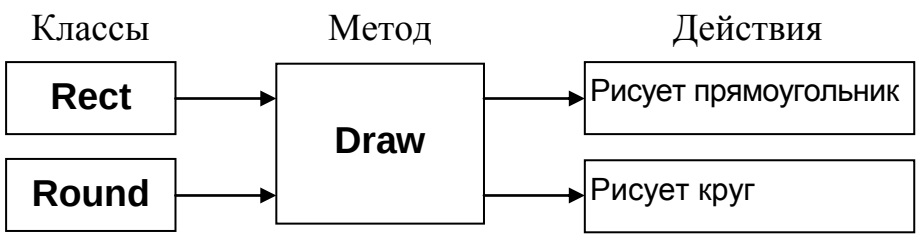

Рис. 6. Пример реализации принципа полиморфизма

Программное средство, созданное с помощью ООП, содержит объекты (элементы управления) с их характерными свойствами. Как правило, работа с программой осуществляется с помощью экранной формы и элементов управления, которые содержат методы обработки, вызываемые при наступлении определенных событий. Экранные формы используются для выполнения заданий и перехода от одного компонента программного средства к другому.

Современными языками ООП являются C++, Java и др. С середины 90-х гг. многие языки ООП реализуются как системы визуального проектирования, в которых интерфейсная часть программного продукта создается в диалоговом режиме, практически без написания программных операторов. К объектно-ориентированным системам визуального проектирования относятся Visual Basic, Delphi, C++ Builder, Visual C++.

2.5. Контрольные вопросы

1) Что понимается под технологией программирования?

2) Опишите этапы развития технологии программирования.

3) Что такое структурное программирование? Приведите примеры ЯП, поддерживающих структурное программирование.

4) Назовите базовые управляющие конструкции, применяемые в структурном программировании.

5) Что такое модульное программирование?

6) Чем характеризуется модуль?

7) Что такое ООП? Приведите примеры языков ООП.

8) Дайте определение объекта, свойства, класса объектов.

9) Что такое событие?

10) Перечислите и опишите принципы ООП.

3. ТЕСТОВЫЕ ВОПРОСЫ

Вопрос № 1 (один верный)

Пошаговая детализация постановки задачи, начиная с наиболее общей проблемы, характеризует…

Варианты ответов:

1) программирование «сверху вниз»;

2) проектирование «от частного к общему»;

3) метод объектной декомпозиции;

4) поиск логической взаимосвязи.

Вопрос № 2 (один верный)

Редактор связей необходим в системе программирования для …

Варианты ответов:

1) формирования исполнимого кода из объектных кодов программы и подключенных библиотечных функций;

2) последовательного выполнения операторов исходного текста программы;

3) перевода исходного текста программы в машинный код;

4) получения файла с исходным текстом программы, который содержит набор стандартных символов для записи алгоритма.

Вопрос № 3 (один верный)

Для технологии ООП верно утверждение, что …

Варианты ответов:

1) в качестве основных элементов программы используются классы и объекты;

2) внутреннее описание класса отражает абстракцию поведения объектов класса;

3) в качестве основных элементов программы используются процедуры;

4) внешнее описание класса (интерфейс) отражает структуру объекта.

Вопрос № 4 (один верный)

Основным принципом структурного программирования является…

Варианты ответов:

1) разбиение задачи на шаги и решение шаг за шагом;

2) использование оператора GOTO для определения структуры программы;

3) использование композиции трех базовых управляющих конструкций;

4) использование большого количества подпрограмм.

Вопрос № 5 (несколько верных)

Структурное программирование основывается на принципах …

Варианты ответов:

1) модульного программирования;

2) программирования «справа налево»;

3) программирования «сверху вниз»;

4) проектирования «слева направо».

Библиографический список

1. Зыков С. В. Программирование / С. В. Зыков. М.: Юрайт, 2018. 320 с.

2. Красновидов А. В. Теория языков программирования и методы трансляции / А. В. Красновидов / УМЦ ЖДТ. М., 2016. 176 с.

3. Трофимов В. В. Информатика: В 2 т. / В. В. Трофимов. М.: Юрайт, 2018. Т. 2. 406 с.

ОБЗОР И КРАТКАЯ ХАРАКТЕРИСТИКА ЯЗЫКОВ ПРОГРАММИРОВАНИЯ

Ada[7] – ЯП, ориентированный на применение в системах реального времени и предназначенный для автоматизации задач управления процессами и (или) устройствами, например, в бортовых (корабельных, авиационных и др.) ЭВМ. Разработан по инициативе министерства обороны США в 1980-х гг.

ALGOL (ALGOrithmic **L**anguage **–** алгоритмический язык**) –** ЯП, созданный в 1960 г., который был призван заменить FORTRAN, но из-за более сложной структуры не получил широкого распространения. В 1968 г. была создана версия ALGOL 68, по своим возможностям и сегодня опережающая многие ЯП, однако из-за отсутствия достаточно эффективных компьютеров для нее не удалось своевременно создать хорошие компиляторы.

Assembler – машинно-ориентированный ЯП, представляющий собой формат записи машинных команд, удобный для восприятия человеком. Этот ЯП используется для программирования в кодах процессора. Каждая модель процессора имеет свой набор команд и соответствующий ему язык ассемблера.

BASIC (Beginner's **A**ll-purpose **S**ymbolic **I**nstruction **C**ode – универсальный код символических инструкций для начинающих**)** – ЯП, разработанный в 1963 – 1964 гг., первоначально предназначался для обучения программированию. Отличается простотой, легко осваивается начинающими программистами благодаря наличию упрощенных конструкций языка FORTRAN и встроенных математических функций, алгоритмов и операторов.

C – многоцелевой ЯПВУ, разработанный Д. Ритчи для реализации операционной системы UNIX в начале 1970-х гг. В последующем приобрел высокую популярность среди системных и прикладных программистов. В С сочетаются преимущества современных ЯПВУ в части управляющих конструкций и структур данных с возможностями доступа к аппаратным средствам ЭВМ на уровне, который обычно ассоциируется с ЯП низкого уровня типа языка Assembler.

---

[7] Назван по имени английского математика Ады Августы Байрон (Лавлейс), жившей в XIX в., дочери поэта Байрона и первой женщины-программиста.

C++ – объектно-ориентированное расширение языка Си, созданное Б. Страуструпом в 1980 г. на базе языка С. Оказал огромное влияние на другие ЯП (Java, C# и др.). C++ активно используется для разработки ПО. Область его применения включает в себя создание операционных систем, разнообразного прикладного ПО, драйверов устройств, высокопроизводительных серверов, развлекательных приложений (например, игр).

C# (C Sharp) – язык ООП, который был разработан в конце 90-х гг. фирмой Microsoft. В нем воплотились лучшие идеи С и C++, а также положительные свойства Java. Предназначен для эффективного создания web-приложений.

Java – язык ООП, разработанный компанией Sun Microsystems в 1995 г. на основе C++. Первая серьезная программа, написанная на этом языке, – браузер Всемирной паутины. Особое внимание в развитии Java уделяется двум направлениям: поддержка всевозможных мобильных устройств и микрокомпьютеров, встраиваемых в бытовую технику, и создание платформно-независимых программных модулей, способных работать на серверах в глобальных и локальных сетях с различными операционными системами.

JavaScript – это язык сценариев, разработанный в 1995 г. Предназначен для создания приложений в Internet и придания интерактивности web-страницам. Поддерживает различные стили программирования (процедурный, объектно-ориентированный и функциональный). JavaScript используется как встраиваемый ЯП для программного доступа к объектам web-приложений.

HTML (**H**yper**T**ext **M**arkup **L**anguage – язык разметки гипертекста) – разработан в 1992 г. Используется для подготовки web-страниц, распределенных в глобальной сети Internet.

FORTRAN (**FOR**mula **TRAN**slation – переводчик формул) – первый ЯПВУ, имеющий транслятор. Разработан фирмой IBM в 1956 г. для описания алгоритмов решения вычислительных задач (научно-технических и математических расчетов). Являясь стандартизированным ЯП, легко переносится на различные платформы. FORTRAN продолжает активно использоваться во многих организациях.

LISP (**LIS**t **P**rocessing language – обработка списков) – алгоритмический язык, предназначенный для манипулирования линейными списками данных. LISP является вторым в истории (после FORTRAN) высокоуровневым ЯП, который используется и в настоящее время. Этот язык применяется преимущественно для решения задач, связанных с искусственным интеллектом.

LOGO – ЯП, разработанный в 1970-х гг. для целей обучения математическим понятиям. Используется в школах и пользователями ПК при написании программ для создания чертежей на экране монитора и управления перьевым графопостроителем.

PASCAL – процедурно-ориентированный ЯП, разработанный в конце 1960-х гг. Н. Виртом для обучения программированию в университетах. Последующие доработки (Turbo Pascal, Object Pascal и др.) позволили сделать PASCAL хорошим универсальным ЯП для написания больших и сложных программ. С 2007 г. язык Object Pascal фирмы Borland официально называется Delphi.

PHP (**P**ersonal **H**ome **P**age Tools – инструменты для создания персональных web-страниц) – скриптовый ЯП общего назначения, разработанный в 1995 г., для генерации HTML-страниц на web-сервере и работы с базами данных. В настоящее время применяется для разработки web-приложений и поддерживается подавляющим большинством хостинг-провайдеров; является одним из лидеров среди скриптовых ЯП для создания динамических web-сайтов.

Perl (**P**ractical **E**xtraction and **R**eport **L**anguage – практический язык извлечений и отчетов) – скриптовый ЯП общего назначения, используется для выполнения широкого спектра задач, включая системное администрирование, web-разработку, сетевое программирование, игры, биоинформатику, разработку графических пользовательских интерфейсов.

PL/1 (Programming **L**anguage One – язык программирования номер один**)** – первый многоцелевой универсальный язык, разработанный в США фирмой IBM в 1963 – 1966 гг. Этот ЯП предназначен для решения задач в области вычислительной техники (исследования и планирования вычислительных процессов, моделирования, решения логических задач и исследования логических схем, разработки систем математического обеспечения. При разработке PL/1 были широко использованы основные понятия и средства языков FORTRAN, ALGOL.

PROLOG (**PRO**gramming in **LOG**ic – программирование на логике) – ЯП декларативного типа. Был разработан в 1971 г. в университете г. Марсель (Франция). В основе языка лежит аппарат математической логики. Предназначен для разработки систем и программ искусственного интеллекта.

SQL (Structured **Q**uery **L**anguage – структурированный язык запросов**)** – ЯП специального назначения, разработанный в 1970-х гг. для управления данными в реляционных системах управления данными (СУБД). Практически в каждой СУБД помимо поддержки языка SQL имеется также свой уникальный язык, ориентированный на особенности этой СУБД.

Учебное издание

МОИСЕЕВА Наталья Александровна

ЯЗЫКИ И ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Учебно-методическое пособие

Редактор Н. А. Майорова

***

Подписано в печать 22.01.2019. Формат 60 × 84 $^{1}/_{16}$.
Офсетная печать. Бумага офсетная. Усл. печ. л. 1,9. Уч.-изд. л. 2,1.
Тираж 250 экз. Заказ .

**

Редакционно-издательский отдел ОмГУПСа
Типография ОмГУПСа

*

644046, г. Омск, пр. Маркса, 35