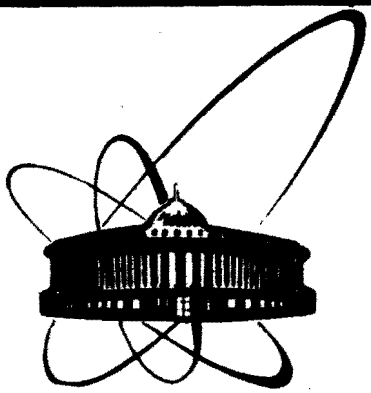


объединенный
институт
ядерных
исследований
дубна

Р11-88-859

В.Шуберт, Х.Юнгклауссен

ХАРАКТЕРИСТИКИ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ И АРХИТЕКТУР ЭВМ

ТЕНДЕНЦИИ РАЗВИТИЯ

Направлено в журнал
"Программирование"

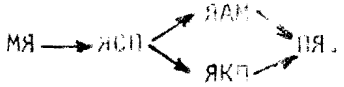
1988

1. ПОСТАНОВКА ВОПРОСА

Массовое производство ЭВМ привело к необходимости поиска новых методов общения конечных пользователей /КП/ с ЭВМ и к появлению целого ряда новых языков программирования /ЯП/. Обращение с ЭВМ должно быть доступным практически любому человеку, подобно тому, как это имеет место относительно вождения легковых автомобилей. Это означает, что язык общения с ЭВМ должен быть очень простым, точнее, похожим на язык, привычный человеку, т.е. КП. Это может быть язык математики в том объеме, в котором КП ее использует. Но если КП хочет использовать ЭВМ не как вычислитель, а, например, как средство управления, как справочник, как эксперт в своей области или советчик при принятии решений, то для формулирования поручений требуется язык, похожий на профессиональный язык КП, т.е. специальный язык, приспособленный к данной предметной области.

Возникают противоречивые требования. С одной стороны нужен простой, но высокоспециализированный язык конечного пользователя /ЯКП/; с другой стороны, требуется универсальность услуг ЭВМ, т.е. возможность обслуживания специалистов различных областей /в том числе и физиков-теоретиков, и экспериментаторов/. Удовлетворение этих двух требований при затрате как можно меньших средств - одна из центральных проблем информатики в настоящее время[1,11]. Путь к ее решению заключается в создании проблемно-ориентированного инструментария /ПОИ/, т.е. комплекса инструментов для решения типичных задач той или иной области. При таком подходе задача программиста состоит не в разработке отдельных прикладных программ, а в разработке ПОИ, а также технологии и рабочих станций для создания и использования ПОИ[2,3].

В настоящей статье не рассматриваются конкретные ПОИ или технология их создания, а поставлен вопрос об общих свойствах языков, участвующих при проектировании ПОИ. При этом будем различать языки пяти типов: машинные языки /МЯ/, языки системного программирования /ЯСП/, языки анализа и моделирования /ЯАМ/ предметной области, языки общения КП с ЭВМ /ЯКП/ и профессиональный язык КП /ПЯ/. В МЯ непосредственно отражается архитектура ЭВМ. Говорят, что уровень языка тем ниже, чем язык ближе к МЯ. Если по этому признаку упорядочить перечисленные языки, получится следующий частичный порядок по возрастающему расстоянию от МЯ:



/1/

Если ЯАМ в достаточной мере формализован, так что при переводе с ЯАМ на ЯСП недоразумения и ошибки маловероятны, то говорят о языке спецификации. ЯП высокого уровня, но легко усваиваемый, который соответствует требованиям, предъявляемым к ЯКП, часто называют языком 4-го поколения[12,13].

Труд программиста, аналитика и КП при создании и пользовании ПОИ облегчается по мере приближения указанных в /1/ языков друг к другу; при этом под расстоянием языков следует понимать их отличие в лексическом, синтактическом и семантическом отношении.

В центре внимания настоящей работы находятся расстояния между ЯСП и ЯАМ и, хотя и в меньшей мере, между МЯ и ЯСП. Будем искать ответ на следующий вопрос: в каком направлении должны развиваться языковые средства ЯСП, чтобы уменьшить расстояние между ЯСП и ЯАМ? Кроме того, рассматривается вопрос о приближении МЯ к ЯСП путем приспосабливания архитектуры ЭВМ к ЯСП.

С точки зрения проектирования сверху вниз создание ПОИ заключается в определении ЯКП и его пошаговом сведе́нии /переводе/ к нижележащим языкам вплоть до МЯ. Язык или языковое средство будем называть реализованными, если их сведе́ние /перевод/ к МЯ осуществлено; в другом случае будем их называть нереализованными или гипотетическими. Если не оговорено противное, будем считать, что ЯСП - реализован, а ЯАП - нереализован. Теперь поставленный выше вопрос можно сформулировать так: какие средства ЯСП следует реализовать для того, чтобы к ним легко могли быть сведены гипотетические средства ЯАМ? С этой точки зрения будем рассматривать развитие языков программирования.

2. ХАРАКТЕРИСТИКИ ЕСТЕСТВЕННЫХ ЯЗЫКОВ

Самым прямым решением проблемы реализации всех возможных ЯКП, казалось бы, является реализация естественного языка /ЕЯ/. Однако этого недостаточно для того, чтобы ЭВМ стала партнером и помощником КП. Ведь язык - не генератор мыслей и идей, а лишь средство их материализации и передачи. Более важны, чем язык - знание и умение партнера, т.е. интеллектуальные возможности ПОИ. Тем не менее, краткое рассмотрение основных свойств ЕЯ будет полезно, поскольку наблюдается определенное сходство между ЕЯ и ЯП относительно их главной задачи, а также относительно закономерностей их развития.

Язык - это средство коммуникации между партнерами в целях их кооперации. Для кооперации необходимы общая модель окружающего мира и согласование действий партнеров в этом мире. Следовательно, язык должен служить для описания мира и для описания или предписания действий партнеров, включая их цели. Основная /но не наименьшая/ синтаксическая и семантическая единица языка - это предложение, точнее, повествовательное, вопросительное и повелительное предложения. Первые два служат для передачи опыта, т.е. личной модели мира; последнее служит для передачи предписания действия. Предписания бывают процедурного типа, если они детально описывают последовательность /процесс/ действия,и непроцедурного типа,если они описывают результат действия. С формальной точки зрения, все предложения, кроме повелительных процедурного типа, являются предикатами в смысле исчисления предикатов. Поэтому ЕЯ можно называть предикативным языком с императивными /процедурно-повелительными/ примесями.

Семантика ЕЯ привязана к внешнему миру, точнее, семантика ЕЯ есть модель внешнего мира. В этом смысле будем говорить о внешней семантике. Отвлекаясь от ЕЯ, отметим, что, когда говорят о семантике ЯП, то обычно имеют ввиду процессы или состояния внутри ЭВМ, вызванные теми или иными оборотами ЯП. В этом смысле будем говорить о машинной семантике. Иначе обстоит дело в случае чистой математики. Однозначность и достоверность математических выводов и высказываний достигается благодаря абстракции от любой семантики, кроме формальной семантики самого математического языка, т.е. той, которая включена в определение формального языка как алгебраическая структура. Формальному и машинному языкам присваивается внешняя семантика путем интерпретации ее человеком. Формальному языку присваивается машинная семантика путем его реализации аппаратными средствами.

Языковая модель мира /предметной области/ - это множество высказываний о свойствах объектов /вещей, понятий/ и отношениях между ними. Посредством отношений образуются сети и иерархии объектов /понятий/. Важное значение для повышения выразительных способностей ЕЯ имеют отношения: часть - целое /напр., пиджак - костюм/; частный случай - общий случай, или подкласс - суперкласс /напр., пиджак - одежда/. В дальнейшем эти два вида отношений будем называть отношением компоновки и отношением обобщения. Они образуют иерархии компоновки и обобщения объектов. Опускаясь по иерархии компоновки, объекты постепенно раскладываются на все более простые /элементарные/ составные части.

Опускаясь по иерархии обобщения, объекты /понятия/ постепенно уточняются добавлением к ним все новых свойств. При этом нижележащий объект наследует все свойства вышележащего объекта. Для дальнейших рассуждений важно отметить, что образование иерархии понятий на самом деле не является свойством языка, а есть свойство мышления. Тот факт, что уровни компоновки и обобщения тем не менее являются важными характеристиками языковых средств, причем не только ЕЯ, но и ЯП /см. рис. 1 и 2/, свидетельствует о том, что в характеристиках языка отражается образ человеческого мышления.

В литературе, /а также в предыдущей главе нашей работы/ понятие уровня языка и языковых средств используется в более общем смысле, куда входит и уровень компоновки, и уровень обобщения.

3. СТИЛИ ПРОГРАММИРОВАНИЯ

Характеристики ЯП связаны с архитектурой ЭВМ. При анализе этой связи исходим из следующего обстоятельства. Любая, сколь угодно сложная операция, которую может выполнить ЭВМ, компонуется из элементарных операций по определенным правилам управления. Для этого необходимы соответствующие аппаратные средства, а именно, операторы /материальные носители или исполнители операций/, средства управления /носители или исполнители правил управления/ и места для хранения операндов /носители данных/. Эти средства являются компонентами композит-операторов, т.е. носителей композит-операций /сложных операций/. Компоновка операторов графически изображена на рис. 1 под нижней штрих-пунктирной линией. Петли указывают на рекурсивный характер процесса компоновки, приводящий к образованию иерархии операторов.

Компоновка операторов может осуществляться чисто аппаратными средствами, но может быть продолжена языковыми /программными/ средствами. При переходе от аппаратной к программной компоновке /при переходе в среднюю часть рис. 1 между штрих-пунктирными линиями/ вводятся названия для композит-операторов и для их компонентов; при этом можно абстрагироваться от материальных носителей. Для того, чтобы ЭВМ могла компоновать требуемую композит-операцию, необходимо точно описать причинную или временную последовательность операций-компонентов. Этим двум возможностям соответствуют описания композит-операции в форме плана передачи данных /ППД/ или в /более приемлемой/ форме плана передачи управления /ППУ//4/:

$$ППД = (Op;\ R_{пд})$$

$$ППУ = (K;\ R_{пу}),$$

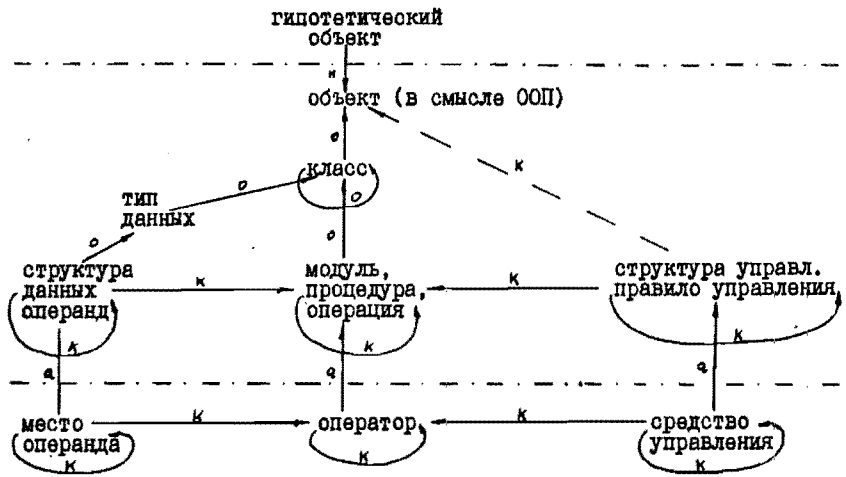


Рис.1.Компоновка и обобщение аппаратных и языковых средств: $a \xrightarrow{k} b$ - a часть от b; $a \xrightarrow{u} b$ - a - интерпретация от b; $a \xrightarrow{o} b$ - a частный случай или подкласс от b; $a \xrightarrow{a} b$ - b - результат абстракции от носителя a.

где Op - множество операций-компонентов, $R_{пд}$ - отношение передачи данных /множество передач операндов между операциями/, K - множество команд k, причем k = (op, od_1, od_2, ...), где op Op; od_i - операнд операции op, а $R_{пу}$ - отношение передачи управления /множество передач управления /активации/ между командами/. Программы, написанные в виде ППД или ППУ, называются программами потока данных или потока управления соответственно.

ППУ как последовательность команд представляет собой так называемую императивную форму записи программ, которая является естественной формой /естественным стилем/ программирования фон-неймановских ЭВМ /ФН-ЭВМ/. Действительно, поскольку в ФН-ЭВМ все данные хранятся в центральной памяти, то для каждой операции надо указать местонахождение операндов, т.е. программу следует записать в виде ППУ. Отсюда возникают известные трудности, связанные с привязкой данных к памяти, характерные для императивных ЯП. В случае человеческого мышления этих трудностей не существует /по крайней мере, пока память не подведет/. Необходимое содержание памяти вызывается подсознательно путем ассоциации. При выполнении вычислений в голове не надо специально адресовать операнды. Это отражается в форме записи на бумаге. Ведь математики формулируют свои вычисления, как правило, в виде ППД, используя функциональную форму записи. В математических выражениях, так же как и в программах, написанных на так называемых функциональных ЯП /напр., LISP или APL/, операнды /кроме исходных/ не указываются явно, они отсутствуют. Следовательно, отсутствует и понятие структуры типа данных в этих языках. Однако все эти проблемы возникают тогда, когда функциональный язык реализуется на ФН-ЭВМ, но решать их должен не прикладной, а системный программист.

Таблица. Стили программирования

Обозначение	Название стиля	Запись предложения	Языки /см. рис. 2/
И	императивный	b = op a	FORTRAN, Pascal
Ф	функциональный /аппликативный/	f(a)	LISP, APL
Л	логический /реляциональный/	P(a,b)	PROLOG
О	объектно-ориентированный /директивный/16//]	b ← m(a)	Smalltalk

Семантика предложения

И - присвоить величине b значение, получаемое при выполнении операции op над a или над значением, присвоенным a.

Л - значение функции f(a) при x = a.

Ф - значение предиката P, если a и b постоянные, или значение a и/или b, при которых P - истинный, если a и/или b - переменные.

О - директива /сообщение/ объекту b, применить метод m к объекту a.

В таблице перечислены еще два стиля программирования, кроме императивного и функционального, называемые логическим и объектно-ориентированным стилями. Логическое программирование - это частный случай предикативного, непроцедурного программирования, отличающийся формой записи, заимствованной у исчисления предикатов. Основными элементами логического программирования являются не операнды и операции, а объекты и отношения, что приведет к определенному сходству с ЕЯ. Можно говорить о том, что логическое программирование - это формализация неформального моделирования мира, о котором шла речь выше, т.е. модели-

рование на ЕЯ в виде совокупности объектов и отношений. Объектно-ориентированное программирование /ООП/ будет рассмотрено ниже. В таблице приведены синтаксис предложения, записанного в том или другом стиле, и соответствующая семантика. Более подробное обсуждение основных концепций и стилей программирования можно найти в работах/5,14-17/. Принципы ООП рассматриваются в/5,15,16,18,19/.

4. ПРИБЛИЖЕНИЕ ЯЗЫКА ПРОГРАММИРОВАНИЯ К ЯЗЫКУ МОДЕЛИРОВАНИЯ

Обратимся теперь к вопросу о развитии средств ЯП в целях приближения ЯСП к ЯАМ. Ответ на этот вопрос можно найти в предыдущих рассуждениях и выводах. Естественно предположить, что изобретатель новых языковых средств пользуется методами, подсказываемыми собственным мышлением, а именно - методами компоновки и обобщения. Провильность этого предположения демонстрируется средней частью рис. 1, между штрих-пунктирными линиями, где показаны средства ЯП и отношения компоновки и обобщения между ними. Комментируем эти отношения /стрелки/ на примере модуля, класса и объекта.

Составными частями модуля, так же как и процедуры /в плане рис. 1 не делается различия между понятиями операции, процедуры и модуля/, являются операции-компоненты, структуры данных и структуры управления. Понятия модуля и типа данных объединяются в обобщенном понятии класса. Это обобщение означает переход к объектно-ориентированному стилю программирования/19/. Класс - это, по-существу, алгебраическая структура, точнее, это статический программный объект в виде алгебраического определения абстрактного типа данных. При обработке программы инстанциируются экземпляры /инстанции/ класса. Классы образуют иерархию обобщения, по которой передаются свойства классов по наследству сверху вниз,как это имеет место в ЕЯ.Понятия класса и инстанции объединяются в самом обобщенном понятии - объект.

Введением классов и объектов как стандартных средств программирования обеспечивается приближение ЯСП к ЯАМ. Язык программиста и язык моделирующего аналитика становятся еще более близкими друг к другу путем введения отношения соседства или знакомства между объектами. Таким образом, объект становится активным элементом /"актором"/20/, который может послать сообщения /директивы, см. табл/ соседним объектам; иными словами, объект может обратиться к своим знакомым с просьбой о помощи при выполнении полученной им директивы. Это означает,что в понятие объекта /актора/ входят элементы управления /см. пунктирную стрелку на рис. 1/. Это означает некоторую децентрализацию управления. Одновременно вносится элемент программирования потока данных, а также предикативного программирования, поскольку программа состоит из взаимодействующих объектов, связанных между собой отношением соседства, компоновки и обобщения. Это именно те гипотетические понятия, т.е. нереализованные языковые средства, которые аналитик или КП использует при моделировании предметной области.

Для реализации модели, созданной аналитиком, в виде машинной модели предметной области, надо свести гипотетические объекты к программным объектам, т.е. надо интерпретировать абстрактные объекты ЯСП конкретными объектами ЯАМ, и, тем самым, свести внешнюю семантику ЯАМ к машинной семантике ЯСП. Именно в этом состоит идея ООП, и это имеют ввиду, когда характеризуют ООП лозунгом: каждому объекту действительности - свой программный объект! Отсюда понятно, что ООП приобретает растущую популярность, а также растущее значение для развития технологии программирования. В настоящее время ведутся работы по обеспечению разрабатываемой в ОИЯИ Ф-технологии объектно-ориентированными средствами/6/.

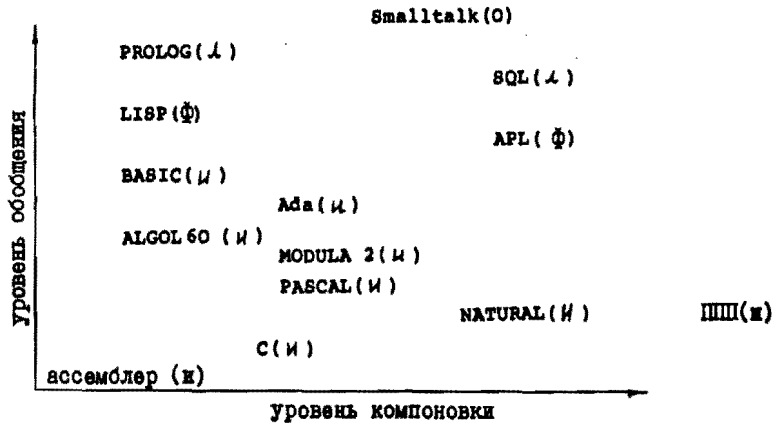


Рис. 2. Упорядочение языков по уровням компоновки и обобщения; в скобках указан стиль программирования /см. табл. /.

На рис. 2 показана попытка двухмерного упорядочения ряда ЯП по максимальным уровням компоновки и обобщения их стандартных языковых средств. Утверждения рисунка не претендуют на общепризнанную истину. Рисунок предлагается, скорее, в качестве предмета обсуждения. Мы ограничимся некоторыми комментариями.

Первые ЯКП, которые появились /хотя и не под этим названием/, - были языки для работы с системами и пакетами прикладных программ /ППП/, разработанными для различных областей применения. С самого начала важной областью применения ППП явилась физика высоких энергий[7,8,21]. С точки зрения нашей классификации, ППП является комплексом операторов очень высокого уровня компоновки, но низкого уровня обобщения. Структуры данных ППП, например, списки результатов измерений, также могут быть очень высокого уровня компоновки. Структуры управления, наоборот, обычно очень простые. Часто они вообще отсутствуют, т.е. ЯКП является командным языком. Такой подход соответствует императивному стилю программирования.

Может показаться, что рис. 2 не отражает в достаточной мере обобщающих возможностей таких языков, как, например, Modula-2 или Ada. Надо, однако, иметь ввиду следующее обстоятельство. Любой универсальный ЯП дает, в принципе, возможность программисту реализовать средства, уровень компоновки или обобщения которых выше уровня стандартных средств данного языка[17]. Но не созданными программистом средствами, а стандартными средствами определяется уровень языка. Например, используя средства языка Ada, можно без особого труда реализовать классы в смысле Smalltalk и создать средства для ООП. Однако создатели языка Ada не сделали главного шага в направлении ООП, т.е. отождествление модулей и типов данных на более высоком уровне обобщения не было осуществлено[19].

5. ПРИБЛИЖЕНИЕ МАШИННОГО ЯЗЫКА К ЯЗЫКУ ПРОГРАММИРОВАНИЯ

В заключение рассмотрим вкратце возможности приближения МЯ и ЯСП путем создания новых, не фон-неймановских ЭВМ /НФН-ЭВМ/. Если пренебречь фактором времени, то нет необходимости в таких машинах. Любой язык, любой стиль программирования может быть реализован на однопроцессорной ФН-ЭВМ. Но при разработке и использовании больших систем необходимый объем человеческого и машинного времени становится проблематичным. Для решения этой проблемы можно идти по пути создания операторов высокого уровня компоновки аппаратными средствами, например, в виде неоднородных или однородных многопроцессорных систем и систолических полей, а также по пути аппаратной реализации языков высокого уровня и различных стилей программирования. Были созданы, например, так называемые LISP-машины и PROLOG-машины и разрабатывается Smalltalk-машина[5].

Несмотря на определенные успехи в указанных направлениях, пока трудно предсказать, в какой мере ЭВМ может стать настоящим партнером человека, его интеллигентным помощником и умным советчиком. Сомнения могут возникнуть, если всмотреться в глубокое отличие между мозгом и ЭВМ относительно их структуры и работы, а также относительно их "машинных" языков. Если считать, что ЕЯ является "машинным" языком мозга, то по аналогии с машинным языком ЭВМ надо предположить, что в характеристиках ЕЯ, о которых речь шла во 2-й главе, отражаются характеристики структуры и работы мозга. Иными словами, образование сетей и иерархий объектов мышления /понятий/ должно иметь свое "аппаратное" /т.е. нейронное/ соответствие, например, в виде статических структур или структур возбуждения нейронной сети. Правильность этого заключения пока не доказана экспериментально. Но предположение о внутреннем представлении /кодировке/ объектов мышления в виде возбуждений нейронной сети лежит в основе многочисленных работ в области нейрофизиологии и психологии[9,22-24], и в определенной мере подтверждается полученными результатами. Немаловажную роль играют при этом также работы по моделированию работы мозга на ЭВМ или аппаратными средствами. Прежде чем несколько подробнее об этом говорить, остановимся еще раз на том фундаментальном несоответствии, которое все ярче вырисовывалось в течение всей предыдущей дискуссии. Речь идет о несоответствии между иерархией понятий, характерной для человеческого мышления, и иерархией операторов, характерной для архитектуры ЭВМ.

Основным принципом проектирования архитектуры ЭВМ, не только ФН-ЭВМ, но и НФН-ЭВМ /имеются ввиду ЭВМ процессорного, а не сетевого типа, см. ниже/, является принцип иерархии операторов, т.е. из элементарных булевых операторов компонуются комбинационные схемы, из них - операторы для выполнения арифметических операций, дальше - операторы для вычисления различных стандартных функций. Как это ни странно, несмотря на то, что человек хочет использовать ЭВМ как "мыслящую" машину, он проектирует ее как "вычислительную" машину в точном смысле этого слова, по старым традициям тех времен, когда ЭВМ служила исключительно для проведения трудоемких расчетов.

Преодоление несоответствия между понятийной и операционной иерархиями полностью лежит на плечах программиста. Для облегчения этой задачи создаются все новые языковые средства возрастающего уровня обобщения. Однако имеющееся несоответствие таким путем не устраняется по существу и продолжает действовать как главное препятствие на пути к искусственному интеллекту. Первые попытки порвать с принципом иерархии операторов делаются при разработке архитектуры ЭВМ 5-го поколения. Но при этом сохраняется традиционный для ЭВМ принцип разделения функций обработки и хранения информации, т.е. сохраняется главное структурное различие между ЭВМ и мозгом.

Для более радикального преодоления фундаментального несоответствия между характером мышления человека и архитектурой ЭВМ придется, по-видимому, отказаться от имитации работы мозга на языковом уровне и перейти к имитации на аппаратном уровне, т.е. построить искусственные нейронные сети. Первые успешные теоретические и экспериментальные работы в этом направлении были проведены Розенблаттом[25]. После некоторого затишья его идеи заново оживились за последние годы[26,10]. В настоящее время существуют работающие системы под названием нейронных или сетевых ЭВМ /в отличие от обычных, процессорных ЭВМ/. Подробный обзор по этим и смежным вопросам можно найти в[27].

Возможности и области применения сетевых ЭВМ /например, распознавание образов, медицинская и техническая диагностика/ пока довольно ограничены. Не исключено, что это ограничение имеет принципиальный характер и является следствием так называемой статической кодировки. Дело в том, что в сетевых ЭВМ применяется, как правило, статический метод кодировки, используемый в обычных ЭВМ, где объекты /операнды, данные/, с которыми работает ЭВМ, кодируются в виде статических состояний носителей /например, регистров/. В отличие от этого, результаты нейрофизиологических экспериментов свидетельствует о том, что мозг применяет динамическую кодировку, т.е. объекты мышления /содержание сознания/ кодируются не статическими,а динамическими, точнее, стационарными состояниями возбуждения нейронной сети, проявляющимися при исследовании посредством электродов как последовательности импульсов[9]. От структуры нейронной сети, которая медленно меняется под влиянием работы мозга, зависит то, какие состояния возбуждения возможны. С этой точки зрения мышление - никогда не прекращающаяся динамика возбуждения живой нейронной сети. В отличие от этого, в случае статической кодировки, работа сетевой ЭВМ состоит из отдельных процессов, каждый из которых возбуждается извне и кончается в некотором конечном статическом состоянии. Это поведение соответствует работе обычного оператора, например, комбинационной схемы или карманного калькулятора, если отвлечься от подробностей внутренних процессов.

Исходя из настоящего состояния вычислительной техники, можно ожидать, что в недалеком будущем будет реализована комбинация ЭВМ процессорного и сетевого типов со статической кодировкой, в то время как ЭВМ с динамической /импульсной/ кодировкой пока относится скорее к области фантазии.

Авторы выражают благодарность Н.Н.Говоруну и Е.П.Жидкову за поддержку этой работы, Ю.Л.Гоголицыну, В.А.Загребнову, В.Г.Иванову, И.М.Иванченко, Ю.Л.Каминскому, А.А.Корнейчуку, Г.А.Ососкову, В.А.Пономареву, Ю.В.Столярскому и П.П.Сычеву - за полезные обсуждения и ценные замечания.

ЛИТЕРАТУРА

1. Дракин В.И., Попов Э.В., Преображенский А.Б. - Общение конечных пользователей с системами обработки данных. М.: Радио и связь, 1988.
2. Железнова К.М., Корнейчук А.А., Шарапова Э.В. - ОИЯИ, Р11-85-905, Дубна, 1985.
3. Иванченко И.М., Седых Ю.В. - ОИЯИ, Р10-87-898, Дубна, 1987.
4. Юнгклауссен Х. - ОИЯИ, Р11-86-542, Дубна, 1986.
5. Фути К., Судзуки Н. - Языки программирования и схемотехника СБИС. М.: Мир, 1988.
6. Шуберт В. - ОИЯИ, Р11-88-721, Дубна, 1988.
7. Говорун Н.Н., Иванов В.Г., Стриж Т.А. - ОИЯИ, Р10-11612, Дубна, 1978.
8. Ежела В.В., Куликов В.А., Никитин С.Г. - ИФВЭ, 83-65, Серпухов, 1983.
9. Бехтерева Н.П. и др. - Нейрофизиологические механизмы мышления. Л.: Наука, 1985.
10. Загребнов В.А., Чвыров А.С. - ОИЯИ, Р17-88-407, Дубна,1988.
11. Martin J. - Application Development without Programmers. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1982.
12. Martin J. - Fourth Generation Languages. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1985.
13. Chorofas D.N. - Fourth and Fifth Feneration Programming Languages. McGraw-Hill, New York, 1986.
14. Ghezzi C., Jazayeri M. - Programming Language Concepts. John Wiley, 1982.
15. Horowitz E. - Fundamentals of Programming Languages. Berlin, Heidelberg, New York: Springer, 1984.
16. Schefe P. - Informatik - Eine Konstruktive Einführung. Mannheim, Wien, Zürich. Bibliogr. Inst., 1985.
17. Kurbel K. - Programmierstil. Berlin, Heidelberg, New York: Springer, 1985.
18. Ingalls D.H.H. - Byte, 1981, 6, 8, p.286.
19. Meyer B. - Object-Oriented Design. First European Software Engineering Conference. Strasbourg, 8-11 September, 1987.
20. Hewitt C. - Artificial Intelligence, 1977, 8, p.323.
21. Brun R. et al. - HBOOK users guide. CERN, DD/EE/81-1, Geneva, 1984.
22. Edelman G.M., Mountcastle V.B. - The Mindful Brain. NIT Press, Cambridge, Massachusetts, London, 1978. Русский пе-

ревод: Эделмен Дж., Маунткасл В. - Разумный мозг. М.: Мир, 1981.
23. Palm, Aertsen (Ed.) - Brain Theory. Berlin, Heidelberg, New York: Springer, 1986.
24. Klix F., Hagendorf H. (Ed.) - Human Memory and Cognitive Capabilities. Amsterdam: Nort-Holland, 1986.
25. Rosenblatt F. - Principles of Neurodynamics. Washington: Spartan Books, 1962. Русский перевод: Розенблатт Ф.,- Нейродинамика. М., 1965.
26. Hopfield J.J. - Proc. Nat. Acad. Sci. USA, 1982,79,p.2554; 1984, 81, p.3088.
27. McClelland J.L., Rumelhart D.E. - Ravallel Distributed Processing. Exploration in the Microstructure of Cognition. MIT - Press, Cambridge, Massachusetts, London, 1986.

Рукопись поступила в издательский отдел
14 декабря 1988 года.

Шуберт В., Юнгклауссен Х. Р11-88-859

Характеристики языков программирования и архитектур ЭВМ. Тенденции развития

Рассматриваются направления развития языков программирования с точки зрения их приближения к естественным языкам и к образу человеческого мышления. Языки классифицируются по уровням компоновки и обобщения языковых средств, по стилю программирования. Особое внимание уделяется объектно-ориентированному программированию. Коротко обсуждается вопрос о перспективах сетевых /нейронных/ ЭВМ со статической или динамической кодировкой.

Работа выполнена в Лаборатории вычислительной техники и автоматизации ОИЯИ.

Препринт Объединенного института ядерных исследований. Дубна 1988

Перевод автора

Schubert V., Jungklaussen H. P11-88-859

Programming Languages and Computer Architecture. Tendencies of Characteristics Development

The trends of programming languages with the point of view of their approach to natural languages and the human thought are considered. Languages are classified in the levels of composition and generalization of the language means and in the programming style. Attention is especially paid to the object-oriented programming. The question of the perspectives of the network /neuron/ computers with statical and dynamical coding is briefly discussed.

The investigation has been performed at the Laboratory of Computing Techniques and Automation, JINR.

Preprint of the Joint Institute for Nuclear Research. Dubna 1988