Проверена 9.2.84
Отпр. в редакцию записку Когана
В ред. "Наука и жизнь" 54 1684
Получили продиктованное по телефону 28.2.84

209-74

ЯЗЫК ИЛИ ЛЕКСИКОН?

Одна из главных задач, стоящих перед нашим обществом сегодня, - увеличение производительности труда. Нужно создавать условия, при которых она поднимается. Одно из таких условий - творческая атмосфера, привычка к творческому труду. Широкое внедрение ЭВМ способствует этому. ЭВМ в руках инженера резко расширяет сферу посильных для него творческих задач.

До последнего времени бытовало мнение, что работа с ЭВМ - это удел избранных, посвященных. Освоить машину нелегко. При этом, образно говоря, человек должен идти навстречу машине. Теперь направление движения меняется. Машина идет навстречу человеку, современные ЭВМ становятся все более удобными в работе.

В числе таких машин - программируемые калькуляторы. Но и они не сегодня-завтра уступят место на столе инженера персональным ЭВМ, которые возьмут на себя огромную часть рутинной работы - расчеты, оформление документов, упростят поиск информации и т.д.

Готовиться к этому процессу нужно заранее. Считаю, что большую роль в этом может сыграть постоянная рубрика "Человек с микрокалькулятором", цель которой - привить читателям журнала вкус к работе с ЭВМ, собирать и распространять опыт использования вычислительной техники. В этой связи, пожалуй, первоочередным является вопрос о языке, на котором было бы удобно записывать программы для программируемых микрокалькуляторов.

Язык программирования - это язык для описания данных (информации) и алгоритмов (программ) их обработки на ЭВМ. Это замкнутая совокупность определенных символов и правил, диктующих, как

209-75

с помощью этих символов записывать алгоритмы для ЭВМ.

Число существующих языков программирования перевалило за несколько тысяч и, наверное, скоро будет сравнимо с количеством человеческих языков. Но программистов не покидает мечта об общем языке программирования, на котором можно было бы не только учить программированию, но и описывать программы, годные для использования на любых типах ЭВМ.

Главные вехи развития мечты - языки "Алгол-60", "Кобол", "ПЛ/I". Сейчас появился еще один - "Ада". Каждый из этих языков имеет много достоинств, но, увы, и много недостатков.

"Алгол-60", например, может претендовать на роль латыни в алгоритмическом мире. Строгий и одновременно удобный, наглядный, он наложил отпечаток практически на все появившиеся после него языки. Основной его недостаток - оторванность от машины. Трудности учета требований конкретных ЭВМ ограничивают его употребление.

Напор вавилонского столпотворения в алгоритмическом мире сегодня сдерживают "Фортран" и "Бейсик". Живучесть их объясняется хорошей реализацией на наиболее распространенных типах машин и простотой. Но простота эта сродни простоте языка человека каменного века. Что касается языка "Ада" (в США он широко рекламируется как последний алгоритмический язык XX века), то, по мнению многих специалистов, "все, что связано с "Адой", прекрасно, кроме самого языка".

Создатели каждого из языков программирования сталкиваются с диалектическим противоречием. Обусловлено оно тем, что всякий язык программирования имеет две стороны. С одной стороны, он должен быть максимально понятен человеку и удобен для записи алгоритмов. Очень хорошо, если этот язык включает обширную ма-

209-76

тематическую символику, широкий набор алгоритмических конструкций. Короче говоря, идеальным представляется язык, используемый математиками и дополненный еще целым рядом понятий из других наук, задачи которых решаются с его помощью: физики, химии, экономики и т.д.

С другой стороны (это вторая сторона языка), он должен быть максимально "понятен" машине. Ведь программы-то мы пишем для нее! С этой точки зрения, чем он экономичней, чем меньше в нем символов и понятий, тем лучше. Идеальным здесь предоставляется язык команд - единственный, который машина понимает сама, без перевода.

Вот мы и подошли к камню преткновения. Для того чтобы машина поняла какой-нибудь другой язык, кроме своего, нужен переводчик. Переводчик, или, как его называют, транслятор, - программа, написанная на машинном языке и переводящая тексты другого языка в машинные команды.

Представьте себе для сравнения англо-русский словарь, в который включены еще все допустимые сочетания слов. С помощью такого словаря человек, владеющий только русским языком, смог бы прочесть (и понять!) любую английскую книгу. Роль такого словаря и выполняет транслятор.

Понятно, что составить такой словарь - дело безнадежное. Вот если бы язык, для перевода с которого он нужен, был не английским, а какой-либо другой, попроще, насчитывающий мало слов и немного словосочетаний - тогда создание описанного словаря было бы делом реальным и книги, написанные на нем, мы читали бы без труда. Но... на таком языке вряд ли мог быть написан "Гамлет".

Постоянное стремление разрешить эти противоречия порождает все новые языки программирования. Одни из них обладают широким набором средств для записи алгоритмов, но трансляторы с них очень

209-77

сложны, транслированные программы громоздки и мало эффективны. Другие языки, наоборот, позволяют создавать простые трансляторы и эффективные транслированные программы, но достигается это за счет резкого обеднения языка.

Кстати, языки программирования для микрокалькуляторов, состоящие просто-напросто из команд, - наглядное тому подтверждение. Удовлетворительные по "второму качеству", они не выдерживают никакой критики с точки зрения первого.

Противоречивость требований, которым должен удовлетворять универсальный язык, приводит к важному выводу: создание такого языка невозможно. Мы и дальше вряд ли сможем повлиять на языковое разнообразие. И отсюда второй вывод: универсальная методология программирования не может ориентироваться на конкретный язык.

Каков же выход из этого положения? В качестве альтернативы единому языку мы выдвигаем понятие о языковой среде, которую называем лексиконом программирования.

В отличие от языка программирования лексикон - открытая система, которая может развиваться и дополняться, подобно языку, на котором мы общаемся. Лексикон должен содержать стандартную математическую символику алгебры, теории множеств, математической логики. Он должен позволять не только записывать программы, но и давать характеристики их функциональных свойств. С этой точки зрения он будет максимально приближен к человеческому языку. С другой стороны, он должен иметь четкую структуру и систему обозначений, допускающую точное толкование его конструкций. И с этой точки зрения он будет приближаться к языку программирования. Лексикон должен допускать формальное преобразование программы в конструкции любого языка программирования.

209-78

Публикуемая статья В.Милехина может рассматриваться как попытка создать систему обозначений для сочинения и записи программ для микрокалькуляторов. С лексиконом ее роднит то, что она предназначена для употребления человеком, который тут же играет роль транслятора. Подобно лексикону, эта нотация открыта для расширений и новых обозначений.

"Промик" далек от идеала - неудачна нотация, употребляются символы, отсутствующие как в программировании, так и в математике, смешиваются названия клавиш и операторов, нечетка терминология. Тем не менее он представляет безусловный интерес в качестве первого опыта. Кроме того, автор уже пользуется им на практике, обосновывая его право на существование.

Мы в Вычислительном центре СО АН СССР собираемся приступить к созданию "Микролексикона", ориентируя его в первую очередь на владельцев программируемых микрокалькуляторов. Надеемся, что он окажется полезным всем, кто по роду своей работы связан с программированием. Рассчитываем, что в процессе работы мы будем постоянно получать помощь от читателей журнала, расширяя таким образом авторский коллектив. Как только работа закончится, она будет опубликована в журнале.

А.ЕРШОВ
член-корреспондент АН СССР
(г.Новосибирск)