КЛАССИФИКАЦИЯ И ХАРАКТЕРИСТИКА СОВРЕМЕННЫХ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ

Гедранович Валентина Васильевна

28 июня 2012 г.

Аннотация

Глава 15 из УМК: **Гедранович, В.В.** Основы компьютерных информационных технологий: учеб.-метод. комплекс / В.В. Гедранович, Б.А. Гедранович, И.Н. Тонкович. – 2-е изд., стереотип. – Минск: Изд-во МИУ, 2011. – 344 с.

1.1 Языки программирования. Классификация языков программирования

Функционирование ЭВМ осуществляется на основе принципа программного управления. Программа, представляющая собой последовательность команд, реализующих алгоритм решения задачи, вводится в память ЭВМ, после чего начинается ее автоматическое выполнение с первой команды. После каждой выполненной команды машина автоматически переходит к выполнению следующей команды, и так до тех пор, пока не встретится команда, предписывающая закончить вычисления.

Структура команды ЭВМ в простейшем случае включает в себя две части: операционную и адресную. Операционная часть содержит код операции (сложить, вычесть, ...). Адресная часть содержит адреса ячеек памяти; в них хранятся значения операндов, с которыми надо выполнить заданную операцию. В зависимости от числа адресов, указанных в команде, различают одно-, двух-, трехадресные команды.

Физические принципы работы электронных устройств ЭВМ таковы, что компьютер может воспринимать команды, состоящие только из единиц и нулей, т.е. машинный код. На начальной стадии развития ЭВМ человеку было необходимо составлять программы на языке, понятном компьютеру, в машинных кодах. Каждая команда состояла из кода операций и адресов операндов, выраженных в виде различных сочетаний единиц и нулей.

Как показала в дальнейшем практика общения с компьютером, такой язык громоздок и неудобен. При пользовании им легко допустить ошибку, записав не в той последовательности 1 или 0. Программу очень трудно контролировать. Кроме того, при программировании в машинных кодах надо хорошо знать внутреннюю структуру ЭВМ, принцип работы каждого блока. И самое плохое в таком языке, что программирование в машинных кодах требует от программиста много времени, труда, повышенного внимания.

Это привело к необходимости найти такое средство, которое позволит более просто наладить общение человека и компьютера. И такое средство было найдено: различные символические языки и соответствующие им трансляторы (***системы программирования***).

Язык программирования – формализованный язык для описания алгоритма решения задачи на компьютере.

Для автоматизации программирования разрабатывался для каждой ЭВМ свой автокод (или Ассемблер). Этот язык в полной мере повторяет набор команд машинного языка и появился лишь для упрощения программирования на машинном коде (рисунок 15.1).

```
TITLE Hello world in win32. Tasm

VERSION T310
Model use32 Flat,StdCall

start_code segment byte public 'code' use32
begin:
 Call MessageBox, 0, offset sHallo, offset caption, 0
 Call ExitProcess, 0
start_code Ends

start_data segment byte public 'data' use32

sHallo  db 'Hello world',0
caption db "Hi",0

start_data Ends
End begin
```

Рисунок 15.1 – Пример программы на Ассемблере

Дальнейшее развитие языковых средств шло по пути создания машинно-независимых языков, позволяющих писать программы на любой доступной ЭВМ с предусмотренной возможностью переноса на более совершенную архитектуру.

В мире насчитывается несколько сотен символических языков программирования различных структур и возможностей, которые могут быть классифицированы по различным признакам.

Если в качестве признака классификации взять синтаксис образования конструкций языков программирования, то их можно условно разделить на следующие классы:

- ***машинные языки*** (computer language) – языки программирования, воспринимаемые аппаратной частью компьютера (машинные коды);
- ***машинно-ориентированные языки*** (computer-oriented language) – языки программирования, которые отражают структуру конкретного типа компьютера (Ассемблеры);
- ***алгоритмические языки*** (algorithmic language) – не зависящие от архитектуры компьютера языки программирования для отражения структуры алгоритма (Паскаль, Фортран, Бейсик и др.);
- ***процедурно-ориентированные языки*** (procedure-oriented language) – языки программирования, где имеется возможность описания программы как совокупности процедур (подпрограмм);
- ***проблемно-ориентированные языки*** (universal programming language) – языки программирования, предназначенные для решения задач определенного класса (Лисп, РПГ, Симула и др.);
- ***интегрированные системы программирования***.

Если в качестве признака классификации взять принадлежность к одному из оформившихся к настоящему времени *стилей* программирования, каждому из которых соответствует своя собственная *модель вычислений*, то языки программирования можно условно разделить на следующие классы:

- **процедурные;**
- **функциональные;**
- **логические;**
- **объектно-ориентированные.**

Программа на ***процедурном языке программирования*** состоит из последовательности операторов (инструкций), задающих те или иные действия. Одним из важнейших квалификационных признаков процедурных языков является их ***уровень***, характеризующий степень близости языка программирования и машинного языка. За начало отсчета уровней принимается машинный язык, уровень которого равен нулю. Язык человека рассматривается как язык наивысшего уровня.

Некоторые языки программирования в порядке увеличения их уровня

Двоичный язык – в настоящее время программистами не применяется.

Шестнадцатеричный язык – упрощение за счет представления четырех двоичных цифр одной шестнадцатеричной. Используется в качестве дополнения к языкам высокого уровня для программирования критичных к времени выполнения фрагментов алгоритмов.

Язык Ассемблера – предназначен для представления в удобочитаемой символической форме программ, написанных на машинном языке.

Язык программирования C – разработан в начале 70-х годов. Сочетает достоинства современных высокоуровневых языков (в части структур данных и управляющих структур) и возможность доступа к аппаратным средствам машины на уровне языка Ассемблера.

Fortran (Formula Translator) разработан в 1956 г. Считается «рабочей лошадью» научных работников за счет своей «приспособленности» к ведению сложных вычислений и широко используется до настоящего времени, несмотря на свою ограниченность и «корявость».

Pascal – разработан в 1968 г. профессором Никлаусом Виртом. Язык назван в честь французского учёного Блеза Паскаля, внесшего вклад в развитие средств вычислительной техники.

Modula-2 – создан в 1978 г. Никлаусом Виртом для создания системного программного обеспечения. По существу – развитие Паскаля. Его особенности состоят в высокой *модульности программ* и наличии *средств описания параллельных процессов*.

Ada – разработан в 1979 г. по заказу Министерства обороны США для использования во встроенных системах с управляющими ЭВМ, что требует поддержки *режима реального времени*.

Logo – разработан с целью обучения детей и используется в настоящее время. Отличается простотой, но весьма богат возможностями, среди которых процедуры, графические средства и т.д.

Функциональные языки программирования. Программа на таком языке представляет собой совокупность описаний функций и выражения, которые необходимо вычислить. Функциональное программирование не использует концепцию памяти как хранилища значений переменных. Операторы присваивания отсутствуют, вследствие чего переменные обозначают не области памяти, а объект программы, что полностью соответствует понятию переменной в математике. Наличие стройной математической основы обеспечивает возможность использования алгебраических методов создания структуры, преобразования и исследования программ. Это в какой-то мере приближает их к описанию структуры мышления человека. Примером функционального языка является язык ***LISP*** (List Processing – обработка списков). Разработан и реализован в Массачусетском технологическом институте в 1959 г. Рассматривается специалистами как основной язык программирования систем искусственного интеллекта.

Логическое программирование. Логика и программирование долгое время были непересекающимися областями исследований. Только в 1973 впервые было опубликовано описание *языка* ***PROLOG*** (PROgramming in LOGic – программирование в терминах логики). Центральным понятием в логическом программировании является *отношение.* Программа представляет собой совокупность определений отношений между объектами и цели. В логическом программировании нужно только специфицировать факты, на которых алгоритм основывается, а не определять последовательность шагов, которые требуется выполнить. Логические программы отличаются принципиально низким быстродействием, так как вычисления осуществляются методом проб и ошибок (посредством поиска с возвратами). В настоящее время для ПК существует около двух десятков реализации PROLOG'а, некоторые из них оформлены в виде интегрированных сред.

Объектно-ориентированное программирование. Корни объектно-ориентированного программирования уходят в одну из ветвей логики, в которой первичной является не отношение, а объект. Прототипом объектно-ориентированного программирования явился язык SIMULA-67. Этот стиль программирования характеризуется богатыми графическими возможностями и средой программирования, развитой модульной структурой программ. Именно модульность упрощает разработку сложных программных продуктов. Как пример объектно-ориентированного языка можно назвать **Visual Basic** и **Delphi**.

1.2 Характеристика некоторых языков программирования. Понятие системы программирования

Язык С – это самый распространённый язык программирования. На нём написано больше программ, чем на любом другом. Подавляющее большинство профессиональных программистов владеют им. Исторически этот язык неотделим от операционной системы UNIX, которая в наши дни переживает своё второе рождение. 60-е годы были эпохой становления операционных систем и языков программирования высокого уровня. Язык с самого начала создавался так, чтобы на нём можно было писать системные задачи. Разработчики языка – Кеннет Томсон и Деннис Ричи. Но поскольку в языке не хватало высокоуровневых средств (абстрактных типов данных и объектов, обработки исключений) в начале 80-х годов Бьерн Страуструп стал разрабатывать расширение языка **C** под условным названием «**C** с классами». Первый коммерческий транслятор «**C++**» появился в 1983 году. Одна из главных целей создания C++ – увеличить процент повторного использования уже написанного кода. Когда появился язык Java, на него обратили очень пристальное внимание, так как он был близок по синтаксису C++ и показался знакомым многим программистам. Однако он не стал, как опасались некоторые, «убийцей C++».

```
#include <iostream.h>

main()
{
    cout << "Hello World!" << endl;
    return 0;
}
```

Рисунок 15.2 – Текст программы на C++

Pascal и Delphi

Pascal. Язык Паскаль, названный в честь французского математика и философа Блеза Паскаля (1623–1662), был создан как учебный язык программирования в 1968–1971 годах Никлаусом Виртом в Высшей технической школе (ETH) в Цюрихе. Этот язык быстро превратился из средства для обучения студентов программированию в инструмент для создания новых программных проектов. Одно из достоинств языка – лаконичность. Язык был создан в то время, когда языков высокого уровня было не много, к тому же все они,

в отличие от языка Pascal, были созданы для решения конкретных задач. Целью работы Н. Вирта было создание языка, который:

- строился бы на небольшом количестве базовых понятий,
- имел бы простой синтаксис,
- допускал бы перевод программ в машинный код простым компилятором.

```
Program HelloWorld(output);
begin
  writeLn('Hello, World!');  { оператор вывода строки }
end.
```

Рисунок 15.3 – Фрагмент программы на Pascal

Delphi – это не что иное, как **Visual Pascal**. Благодаря созданию этой среды программирования простые программы (Windows-приложения) может писать практически любой пользователь.

Basic. Это одни из самых старых языков программирования. Его создатели – Джон Кемени и Том Куртц, работавшие в Дортмундском колледже в 1964 году. Свой язык они назвали по первым буквам слов «Beginner's All Purpose Symbolic Instructions Code». Интерпретатор Basic был первым программным продуктом фирмы Microsoft, основанной Полом Аленом и Уильямом Гейтсом в 1975 году. В дальнейшем он не только поставлялся как программа, но и зашивался в ПЗУ компьютеров.

```
10 INPUT I
20 IF I >= 0 THEN PRINT "Positive number or null" : GOTO 40
30 PRINT "Negative number"
40 END
```

Рисунок 15.4 – Фрагмент программы на Basic'е

В середине 80-х годов фирма Microsoft разработала QuickBASIC. Это был уже компилятор, а не интерпретатор. Вообще языков Basic несколько сотен. После появления Windows и визуальных средств разработки программ был создан **Visual Basic**.

FORTRAN. Это старейший язык программирования. В начале 50-х годов он был разработан исследовательской группой под руководством Джона Бэкуса. Его название происходит от 2 слов: **FOR**MULA **TRAN**SLATION. Первая версия системы **FORTRAN** для компьютера IBM была выпущена в начале 1957 г. Характерной чертой языка была специфическая форма записи программ. Текст программы записывался строками фиксированной длины по 80 знаков, что соответствует размеру перфокарты. Очень важную роль играют в языке метки. Язык постепенно избавлялся от недостатков. Так появился **FORTRAN IV**, затем в 1977 г. – **FORTRAN 77**, в 1991 г. – очередной стандарт **FORTRAN 91**.

```
C     Hello World in Fortran

      PROGRAM HELLO
      WRITE (*,100)
      STOP
  100 FORMAT (' Hello World! ' /)
      END
```

Рисунок 15.5 – Текст программы на Fortran'е

ADA. По сложности этот язык сравнивают с C++. Назван в честь леди Ады Августы Лавлейс (дочери Байрона), работавшей вместе с Чарльзом Бэббиджем и разрабатывавшей программы для его «аналитической машины». Она по праву считается первым в мире программистом. Разработан язык в 1979 г. группой под руководством Жана Ишбиа в рамках конкурса, объявленного Министерством Обороны США, поскольку разработки в этом ведомстве велись до этого на многих языках и ни один из них не удовлетворял всем задачам. ADA – универсальный высокоуровневый язык программирования. Он – модульный и даже объектный, но не объектно-ориентированный. Как и все языки, он развивался. Мода на объектно-ориентированное программирование привела к созданию принципиально новой его версии ADA95.

```
-- Hello World in Ada

with Text_IO;
procedure Hello_World is

begin
  Text_IO.Put_Line("Hello World!");
end Hello_World;
```

Рисунок 15.6 – Текст программы на Ada

Понятие системы программирования

Чтобы вычислительная машина могла выполнить программу, написанную на каком-либо языке программирования, в её программном обеспечении должна быть ***программа-транслятор*** для этого языка.

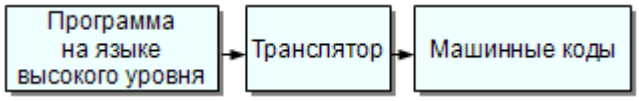

Рисунок 15.7 – Схема работы транслятора

Транслятор представляет собой программу, на основе которой машина преобразует вводимые в нее программы на машинный язык, поскольку вычислительная машина может выполнять программы, записанные только на языке машины, и алгоритмы, заданные на другом языке, должны быть перед их выполнением переведены на машинный язык.

Трансляторы бывают двух типов:

- ***интерпретаторы;***
- ***компиляторы.***

Интерпретатор переводит каждую команду программы с одновременным её выполнением и, если обнаруживает ошибку, сообщает о ней и прекращает выполнение программы.

Компилятор переводит всю программу целиком и в конце работы выдаёт список ошибок, если они обнаружены.

Также система программирования может включать в себя:

- интегрированную среду разработчика программ;
- отладчик;
- средства оптимизации кода программ;
- набор библиотек (возможно с исходными текстами программ);
- редактор связей;
- сервисные средства (утилиты) для работы с библиотеками, текстовыми и двоичными файлами;
- справочные системы;
- документатор исходного кода программы;
- систему поддержки и управления проектом программного комплекса.

1.3 Тенденции развития технологий и языков программирования

Появление новых поколений ЭВМ обусловлено расширением сферы их применения, требующей более производительной и надежной вычислительной техники. В настоящее время стремление к реализации новых потребительских свойств ЭВМ стимулирует работы по созданию новых и усовершенствованию имеющихся языков программирования, которые будут удовлетворять качественно новым *функциональным требованиям*:

- работать с базами знаний в различных предметных областях и организовывать на их основе системы искусственного интеллекта;
- обеспечивать простоту применения ЭВМ путем реализации эффективных систем ввода-вывода информации голосом, диалоговой обработки информации с использованием естественных языков, устройств распознавания речи и изображения;
- упрощать процесс создания программных средств путем автоматизации синтеза программ.

В настоящее время ведутся интенсивные работы как по созданию ЭВМ пятого поколения традиционной (неймановской) архитектуры, так и по созданию и апробации перспективных архитектур и схемотехнических решений в области программирования. Развитие технологий и языков программирования с высоким параллелизмом во многом определяется элементарной базой, степенью развития параллельного программного обеспечения и методологией распараллеливания алгоритмов решаемых задач.

Проблема создания эффективных систем *параллельного программирования*, ориентированных на высокоуровневое распараллеливание алгоритмов вычислений и обработки данных, представляется достаточно сложной и предполагает дифференцированный подход с учетом сложности распараллеливания и необходимости синхронизации процессов во времени.

Наряду с развитием архитектурных и схемотехнических решений ведутся работы по совершенствованию *технологий производства интегральных схем* и по созданию принципиально новых *элементных баз*, основанных на оптоэлектронных и оптических принципах.

Важным направлением развития вычислительных и программных средств является ***интеллектуализация ЭВМ***, связанная с наделением ее элементами интеллекта, интеллектуализацией интерфейса с пользователем и т.д.

Работа в данном направлении, затрагивая, в первую очередь, программное обеспечение, потребует и создания ЭВМ определенной архитектуры, используемой в системах управления базами знаний, – *компьютеров баз знаний*, а также других подклассов ЭВМ. При этом ЭВМ должна обладать способностью к обучению, производить ассоциативную обработку информации и вести интеллектуальный диалог при решении конкретных задач.

Современные языки программирования похожи друг на друга: каждый из них содержит конструкции (операторы, типы данных и другие), имеющие аналоги в других языках программирования. В то же время идентичность языков далеко не полная. Каждый из них содержит конструкции, присущие только ему (даже похожих конструкций в других языках не наблюдается).

Конструкции современных языков имеют общее содержание (семантику), но различный порядок следования компонент (синтаксис) и разные ключевые слова (лексику). Таким образом, различные языки предоставляют программисту одинаковые возможности (при различном внешнем виде программ).

Сравнивая между собой конструкции современных языков программирования и выделив их общую составляющую, можно описать (не создать, а именно описать уже существующий de facto!) «универсальный» язык программирования (правда, только на семантическом уровне).

Существующая ныне система стандартизации языков программирования не способствует выполнению этой задачи. Главная проблема состоит в том, что при описании стандарта семантическая составляющая не отделена от синтаксиса и лексики. Кроме того, при модернизации стандартов комитеты ISO/ANSI предпочитают скорее добавлять в язык новые возможности, чем исключать редко используемые, что приводит к неоправданному синтаксическому расширению языков.

Если попытаться начертить схему пересечения семантики языков программирования, то можно получить изображение, приведенное на рисунке 15.8.

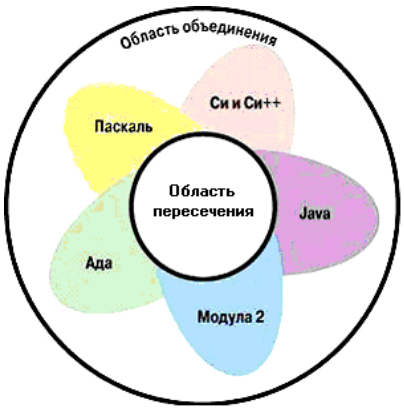

Рисунок 15.8 – Области пересечения и объединения языков программирования

Как видно из приведенной схемы, существует общая для всех современных языков семантическая зона, в которую входят конструкции, принадлежащие всем (или большинству) языков программирования. Таким образом, семантику каждого языка программирования можно условно поделить на «область пересечения» (общие для всех языков конструкции) и «область объединения» (специфические для данного языка конструкции). Создание входного языка для многоязыкового компилятора можно произвести двумя различными способами:

1. Использовать только общие конструкции (область пересечения), отбросив все «особенные» конструкции языков, как необязательные. Это приведет к усечению всех участвующих в работе языков программирования.
2. Использовать все имеющиеся в языках конструкции (область объединения). В этом случае каждый из языков должен быть дополнен конструкциями, имеющимися в других языках программирования. Этот подход чреват чрезмерным расширением семантической базы.

Разумеется, в чистом виде ни один из этих подходов применяться не должен, но, тем не менее, более правильным представляется первый вариант, поскольку в «области пересечения» содержится исторически наработанный необходимый минимум семантических конструкций.

Стремительное развитие компьютерной индустрии не может не поставить перед создателями «средств производства» программ (компиляторов) новые задачи. Компиляторы должны стать более «адекватными» эпохе визуального программирования и Интернета. Последнее время высока популярность WWW-программирования. Языки WWW-программирования обладают рядом свойств, которые позволяют использовать их на платформе, специализированной для работы в качестве сервера. Чаще всего это интерпретаторы (такие, как Perl, PHP), позволяющие использовать их на стороне сервера, или языки, поддерживаемые клиентом (браузеры) – HTML, XML, Java, JavaScript, или специальные модули (plug-in), расширяющие клиента – Flash.

Унификация языков программирования и создание общепринятой семантической базы – необходимые условия продолжения прогресса в этой области программного обеспечения, и, в конечном итоге, всей компьютерной индустрии.

Литература

1. Андреев, А. Эволюция современных языков программирования / А. Андреев // Мир ПК. – 2001. – № 3. – (http://www.osp.ru/pcworld/2001/03 /056.htm)
2. Информатика: учебник / под ред. проф. Н.В. Макаровой. – М.: Финансы и статистика, 1997. – 768 с.
3. Кауфман, В. Языки программирования. Концепции и принципы. / В. Кауфман. – М.: Радио и связь, 1999. – 231 с.
4. Коцюбинский, А.О. Хрестоматия работы на компьютере: практ. пособ. / А.О. Коцюбинский, С.В. Грошев.– М.: Изд-во «ТРИУМФ», 2001. – 640 с.

5. Турбо Паскаль 7.0. – К.: Издат. группа BHV, 1998. – 448 с.
6. Гедранович, В.В. Основы информатики и вычислительной техники: учеб.-метод. комплекс: в 2 ч. / В.В. Гедранович, Ю.В. Змеева. – Минск: Изд-во МИУ, 2006. – Ч. 2. – 162 с.

КОНТРОЛЬНЫЙ ТЕСТ

1. Языки программирования условно разделены на классы: машинные, машинно-ориентированные, алгоритмические, процедурно-ориентированные, проблемно-ориентированные. По какому признаку классифицированы языки?

- синтаксис образования конструкций языков программирования
- принадлежность к одному из оформившихся к настоящему времени стилей программирования
- уровень языка программирования, характеризующий степень его близости к машинному языку

2. Языки программирования условно разделены на классы: процедурные, функциональные, логические, объектно-ориентированные. По какому признаку классифицированы языки?

- синтаксис образования конструкций языков программирования
- принадлежность к одному из оформившихся к настоящему времени стилей программирования
- уровень языка программирования, характеризующий степень его близости к машинному языку

3. К языкам высокого уровня не относятся

- Ada
- язык Ассемблера
- двоичный язык
- C/C++

4. Примером функциональных языков программирования может служить

- LISP
- PROLOG
- DELPHI
- PASCAL

5. Примером процедурных языков программирования может служить

- LISP
- PROLOG
- DELPHI
- PASCAL

6. Примером логических языков программирования может служить

- LISP
- PROLOG
- DELPHI
- PASCAL

7. Программа, на основе которой машина преобразует вводимые в нее команды на машинный язык, называется

- переводчиком
- транслятором
- системным администратором
- редактором связей

8. Транслятор, который переводит каждую команду программы с одновременным её выполнением и, если обнаруживает ошибку, сообщает о ней и прекращает выполнение программы, называется

- компилятором
- редактором связей
- системным администратором
- интерпретатором

9. Транслятор, который переводит всю программу целиком и в конце работы выдаёт список ошибок, если они обнаружены, называется

- компилятором
- редактором связей
- системным администратором
- интерпретатором

10. К языкам высокого уровня относятся

- Ada
- язык Ассемблера
- двоичный язык
- C/C++