Языки программирования

Кондратенко Д. А.

$28^{12/2022}$

Классификация языков программирования по машинозависимости

- Машинозависимые языки (языки низкого уровня)
- Машинонезависимые языки (языки высокого уровня)

Первые языки низкого уровня

Языки низкого уровня появились вместе с самими компьютерами. Представляли из себя они машинные коды.

Постепенно сложность программ возросла настолько, что машинные коды стали крайне неудобным средством программирования, поэтому были начаты опыты в разработке высокоуровневых языков программирования.

Первые опыты создания трансляторов

В начале 50-х годов для некоторых компьютеров появились программы, переводящие машинные коды из текстового представления в двоичное. Сейчас их именуют *языками ассемблера*.

Первые опыты создания высокоуровневых языков

Первый компилятор («программирующая программа» ПП-1) был создан в 1954 году в Советском Союзе. В 1955 году была создана ПП-2, которая уже была оптимизирующим компилятором.

Первый высокоуровневый язык программирования

За границей первый компилятор высокоуровневого языка был создан в 1957 году в компании IBM. Язык назывался FORTRAN (Formula Translator).

Основные парадигмы программирования

- Императивное программирование
- Процедурное программирование
- Структурное программирование
- Объектно-ориентированное программирование
- Функциональное программирование
- Логическое программирование

Императивное программирование подразумевает представление программы в виде последовательности инструкций.

Впервые реализовано в низкоуровневых языках программирования, в частности, в языках ассемблеров и IBM FORTRAN I. Особенностью таких языков программирования является наличие инструкции безусловного перехода (`GO TO`).

Процедурное программирование подразумевает объединение операторов (команд, инструкций) в подпрограммы или процедуры.

Впервые реализовано в 1958 году в языках FORTRAN II и ALGOL 58.

Структурное программирование подразумевает объединение операторов в основных три типа блоков:

- блок последовательности;
- блок решения (`if`,`switch`);
- блок цикла (`while`,`do-while`,`for`).

В 1958 году в Европе был создан язык программирования ALGOL 58, в котором впервые появились элементы структурного программирования.

В частности, блоки начинались словом begin и заканчивались словом end, были конструкции `if`, `if either-or` (вместо нынешнего `if-then-else`), `for`, `do`, `switch`.

В 1960 году язык ALGOL 58 был доработан и стандартизован как ALGOL 60. В языке вместо конструкции `if either-or` появилась конструкция `if-then-else`, оператор `do while`.

На основе ALGOL 60 в 1970 году Никлаус Вирт создал язык Pascal.

Основные парадигмы программирования

Структурное программирование: пример кода на Algol 60

```
comment from <https://github.com/JvanKatwijk/algol-60-compiler/>
begin
   integer i;
   integer procedure fac (n); value n; integer n;
      fac := if n < 1 then 1 else n * fac (n - 1);

   outstring (1, "Example 1\n");
   for i := 1 step 1 until 10 do
   begin
      outinteger (1, i); space (1); outinteger (1, fac (i)); newline (1)
   end;
end;
```

Введение блоков подобного рода позволило существенно улучшить читаемость кода и возможность его нормальной поддержки. Использование конструкций типа GO TO вместо них превращает код в нечитаемую и неотлаживаемую мешанину. Об этом говорил Эдсгер Дейкстра в своей статье «О вреде оператора `GO TO`».

Эта парадигма в настоящее время реализована практически во всех языках программирования.

Объектно-ориентированное программирование подразумевает концепцию объекта — объединения данных и методов их обработки.

Реализовано разными способами в разных языках программирования, в частности, в C++, C#, Java, Python и других.

В 1967 году был создан язык Simula-67. Этот язык представляет собой расширение ALGOL 60. В нём впервые появилось понятие *класса* как объединения данных и методов их обработки.

```
! from <https://twobithistory.org/2019/01/31/simula.html>
! dogs.sim ;
Begin
    Class Dog;
        Virtual: Procedure bark Is Procedure bark;;
    Begin
        Procedure bark;
        Begin
            OutText("Woof!");
            OutImage;      ! Outputs a newline ;
        End;
    End;
```

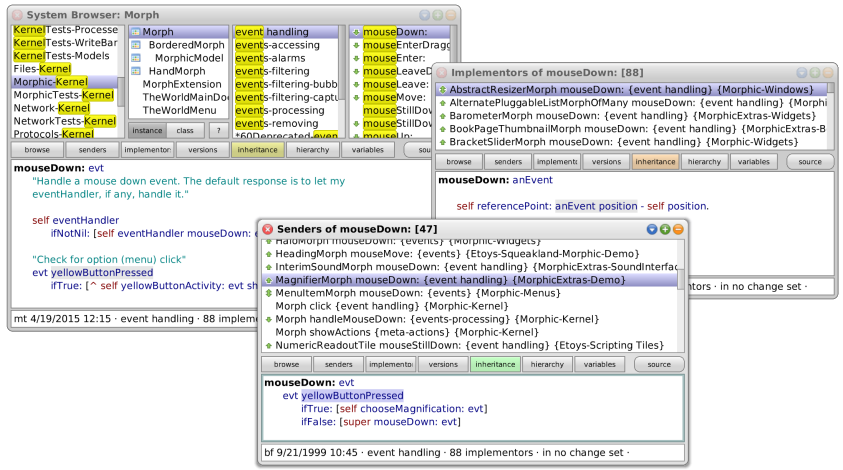
```
    Dog Class Chihuahua;  ! Chihuahua is "prefixed" by Dog ;
    Begin
        Procedure bark;
        Begin
            OutText("Yap yap yap yap yap yap");
            OutImage;
        End;
    End;

    Ref (Dog) d;
    d :- new Chihuahua;   ! :- is the reference assignment operator ;
    d.bark;
End;
```

С 1969 по 1980 гг. Алан Кэй разрабатывал язык Smalltalk, который является чистым объектно-ориентированным языком. В нём всё является объектом, в том числе числа, и этим объектам посылаются сообщения. За счёт этого синтаксис серьёзно упрощается.

В этом языке впервые была полностью реализована инкапсуляция.

Реализации языка Smalltalk поставляются вместе с виртуальной машиной, предоставляющей графическую среду выполнения. Исходные коды её можно изменять во время выполнения, и они все написаны на Smalltalk.



```
"from <https://github.com/jwerle/smalltalk-examples/>"

Object subclass: #Person.
Person comment:
  'I am a primitive form of a person'.
Person instanceVariableNames:
  'name gender age'.

"Person statics"
Person class extend [
  new [|p|
    p := super new.
    p init.

    ^p
   ]
]
```

Основные парадигмы программирования

Объектно-ориентированное программирование. Пример кода на Smalltalk

```
"Person prototype"
Person extend [
  |name gender age|
  init [ <category: 'initialization'>
   name   := nil.
   gender := nil.
   age    := nil.
  ]

  friendlyName [
    ^name
  ]

  printOn: stream [
    <category: 'printing'>
    stream nextPutAll:
      '-named: ' . name printOn: stream.
    stream nextPutAll:
      ' -age: ' . age printOn: stream.
    stream nextPutAll:
      ' -gender: ' . gender printOn: stream.
  ]
```

```
  name [
    ^name
  ]
]

Person subclass: #Joe.

Joe extend [
  init [
    name    := 'joe'.
    gender  := 'male'.
    age     := 22.
  ]
]

joe := Joe new.


joe printNl
```

Функциональное программирование подразумевает представление алгоритма в виде последовательности функций, которые по цепочке передают результат из выхода одной функции на вход другой.

Первый функциональный язык программирования разработан в 1958 году. Называется он Lisp (от List Processor).

Программа на Lisp представляет собой список, в который помещаются символы или другие списки. Список помещён в круглые скобки.

```
(defun replace-all (string part replacement &key (test #'char=))
"Returns a new string in which all the occurences of the part
is replaced with replacement."
    (with-output-to-string (out)
      (loop with part-length = (length part)
            for old-pos = 0 then (+ pos part-length)
            for pos = (search part string
                              :start2 old-pos
                              :test test)
            do (write-string string out
                             :start old-pos
                             :end (or pos (length string)))
            when pos do (write-string replacement out)
            while pos)))
```

В 1973 году был создан функциональный язык программирования ML (Meta Language), в котором впервые был реализован вывод типов на основе т. н. *алгоритма Хиндли-Милнера*.

Этот язык программирования породил огромное множество различных производных языков программирования, в том числе Haskell, OCaml и F#.

```
(* This code was taken from <https://cs.lmu.edu/~ray/notes/introml/>
 * A little program that writes the command line arguments from last
 * to first, to standard output. It is longer than necessary because
 * it is a teaching example.
 *)

(* Returns the reversal of a list. Warning: wheel reinvention. *)

fun reverse [] = []
  | reverse (h :: t) = reverse t @ [h];

fun concat_space s = s ^ " ";

(* Prints each command line arg, suffixed with a space. *)

val _ =
  let
    val args = CommandLine.arguments()
  in
    map (print o concat_space) (reverse args);
    print "\n"
  end;
```

Логическое программирование подразумевает представление программы в виде последовательности логических умозаключений и правил их вывода.

Впервые логическое программирование было реализовано в языке Planner в 1969 году. Основан он был на языке Lisp.

Язык чисто логического программирования — Пролог — появился в 1972 году.

```
// from <https://athena.ecs.csus.edu/~mei/logicp/prolog/>
studies(charlie, csc135).
studies(olivia, csc135).
studies(jack, csc131).
studies(arthur, csc134).


teaches(kirke, csc135).
teaches(collins, csc131).
teaches(collins, csc171).
teaches(juniper, csc134).

professor(X, Y) :- teaches(X, C), studies(Y, C).

?- studies(charlie, What).
?- professor(kirke, Students).
```

Современные языки программирования

В настощее время используются самые разные языки программирования, в самых различных сферах. В основном, это языки высокого уровня.

Современные языки программирования

В сфере системного и прикладного программирования в настоящее время широко используются языки Си и C++. Однако для чисто прикладных нужд придуманы и другие языки: Rust, Go, Java, C#, Python и многие другие.

В представлении не нуждается.

Широко используется в системном и прикладном программировании. На нём написано практически всё.

Создан Бьёрном Страуструпом в 1984 году, в 1998 году - впервые стандартизован. За эти годы успел обрасти большим количеством новых языковых возможностей: в нём появилась обработка исключений, шаблоны и многое другое. Новые возможности продолжают добавляться.

Широко используется в прикладном программировании. Для системного программирования используется, в основном, общее подмножество с Си.

В Си управление памятью ручное, что может являться причиной многих уязвимостей, связанных с переполнением буфера, поэтому создаются языки, которые пытаются решить эту проблему с помощью автоматического управления памятью.

Иногда говорят, что Си должен вообще выйти из употребления, отправиться на свалку истории. На роль таких «убийц» Си выдвигаются языки Rust и Go.

Mozilla создала язык Rust. Язык был более-менее закреплён в 2018 году. Предъявляет повышенные требования к безопасности. Многие ошибки отлавливаются ещё на этапе компиляции, в частности, с использованием механизма *проверки заёмов указателей* или *borrow checker*: компилятор запрещает использовать две ссылки на один и тот же объект.

Недостатки:

- У языка Rust отсутствует формальная спецификация, что не позволяет ни понять, что делает исходный компилятор, ни реализовать новый;
- Компилятор `rustc` сильно зависит от LLVM, что влияет на переносимость компилятора на микроконтроллеры.

Оба эти фактора приводят к невозможности широкого распространения Rust.

Google создала язык Go. Он больше подходит для прикладного программирования, так как, в отличие от Rust, использует сборщик мусора. Операционную систему написать будет крайне затруднительно.

Как бы то ни было, язык Си закопать не получится. Особенно это касается программирования различных микроконтроллеров.

Существует виртуальная машина Java (JVM), в машинные коды которой компилируются некоторые языки программирования, в основном, Java и Kotlin. Использование отдельной виртуальной машины позволяет запускать единожды скомпилированный код на разных платформах.

Язык Java создан Джеймсом Гослингом в 1995 году. Применяется во многих сферах, в том числе, в программировании для Android. На Java написаны, в частности, среды разработки Eclipse, Netbeans и большая часть IntelliJ IDEA.

Язык Kotlin создан в 2011 году Андреем Бреславом. Более безопасен, чем Java, во многом из-за того, что в нём основные типы не могут принимать значения NULL. Это позволяет отлавливать многие ошибки ещё на этапе компиляции.

В настоящее время используется для программирования на Android, однако сфера его применения им не ограничивается.

Компания Microsoft создала платформу .NET Framework, которая использует собственную виртуальную машину, и несколько языков программирования для неё.

Чаще всего используется язык C#. Это объектно-ориентированный язык программирования. Применяется преимущественно для программирования под ОС Windows, тем не менее, возможно писать программы для .NET на других платформах.

Язык Python создан Гвидо ван Россумом в 1992 году. Является интерпретируемым языком программирования.

Подходит для самых различных нужд, в частности, может быть использован в математических расчётах и построениях. Используется также для нейронных сетей. С использованием Django также пишутся сайты.

- Fortran (для математических расчётов);
- PHP (для динамических сайтов);
- Javascript (для браузерных скриптов);
- Perl (для обработки текстов);
- R (для статистической обработки данных);

и многие другие языки.

В настоящее время на языках низкого уровня пишутся программы, в основном, для микроконтроллеров, а также высокопроизводительные вставки, где требуется увеличение скорости или уменьшение потребления памяти, а может, и то, и другое.

Контактная информация

E-mail: dan.kondratenko2013@ya.ru
danilakondratenko95@gmail.com

Codeberg: https://codeberg.org/danila-kondr
Github: https://github.com/danil-kondr2016

Эта презентация создана с использованием LaTeX 2ε.
(`\documentclass{beamer}`)