# OLAIYA VICTOR OMOKUDU

# ENG1406904

# COMPUTER ENGINEERING

# 300L

# COMMERCIAL PROGRAMMING LANGUAGES

# CPE 321

# TITLE:

# COMMERCIAL PROGRAMMING: DESCRIPTION & CLASSIFICATION

**A SHORT DESCRIPTION OF COMMERCIAL PROGRAMMING LANGUAGE**

Commercial programming is the designing and development of softwares or programs for the end user with or without profit as it chief objective. It can be in the form of freeware, which are free to use, shareware which are free but restricts the user and commercial software which requires payment before use.

**COMMERCIAL PROGRAMMING LANGUAGES CLASSFICICATION**

COMPILER ORIENRED LANGUAGE

These are languages typically processed by compilers, they need to be compiled before they csn run. Examples are:

ActionScript

Ada (multi-purpose language)

ALGOL (extremely influential language design – the second high level language compiler)

SMALL Machine Algol Like Language

Ateji PX, an extension of the Java language for parallelism

BASIC (some dialects, including the first version of Dartmouth BASIC)

BCPL

Blue

C (one of the most widely used procedural programming languages)

C++ (One of the most widely used object-oriented (OO) languages specially used in large-scale, highly complex, high-performance software systems)

C# (compiled into intermediate language, which generates a native image at runtime)

Ceylon (compiled into JVM bytecode)

CLIPPER 5.3 (programming Language for DOS-based software)

CLEO (Clear Language for Expressing Orders) used the compiler for the BritishLeo computers

COBOL

Cobra

Common Lisp

Crystal

Curl

D (Attempts a "C++ done right" philosophy)

DASL compiles into Java, JavaScript, JSP, Flex, etc., which are further compiled into a .war file

Delphi (Borland's Object Pascal development system)

DIBOL (Digital Interactive Business Oriented Language)

Dylan

eC

Eiffel (object-oriented language developed by Bertrand Meyer)

Sather

Ubercode

Emacs Lisp

Erlang

F# (compiled into intermediate language, which generates a native image at runtime)

Factor

Forth (professional systems, like VFX and SwiftForth)

Fortran (the first high-level, compiled language, from IBM, John Backus, et al.)

GAUSS

Go

Gosu (compiled into JVM bytecode)

Groovy (compiled into JVM bytecode)

Haskell

Harbour

Java (usually compiled into JVM bytecode although ahead-of-time (AOT) compilers exist that compile to machine code)

JOVIAL

LabVIEW

LANSA

Mercury

Nemerle (compiled into intermediate language bytecode)

Nim

Objective-C

Pascal (most implementations)

Plus

Python (compiled into intermediate Virtual Machine bytecode)

RPG (Report Program Generator)

Rust

Scala (compiled into JVM bytecode)

Scheme (some implementations, e.g. Gambit)

SequenceL – purely functional, automatically parallelizing and race-free

Simula (the first object-oriented language, developed by Ole-Johan Dahl andKristen Nygaard)

Smalltalk generally compiled to platform independent bytecode that runs on a Virtual Machine

Swift

ML

Standard ML (SML)

Alice

OCaml

Turing

Vala (Compiler for the GObject type system)

Visual Basic (Earlier versions compiled directly to a native runtime. Recent .NET versions compile into intermediate language that is just-in-time compiled into a native image at runtime.)

Visual FoxPro

Visual Prolog

X++

X#

XL

Z++

INTERPRETER ORIENTED LANGUAGE

Interpreted languages are programming languages in which programs may be executed from source code form.

Examples include:

Ant

APL

AutoHotkey scripting language

AutoIt scripting language

BASIC (some dialects)

DATABUS (later versions added optional compiling)

DM

Eiffel (via "Melting Ice Technology" in EiffelStudio)

Forth (interactive shell only; otherwise compiled to native or threaded code)

FPr (Virtual machine: Text is compiled to linked lists; linked lists are interpreted)

Game Maker Language

Golo

Groovy

Haskell (GHCi, Hugs, NHC, YHC etc.)

J

Julia (Compiled on the fly to machine code, but a transpiler Julia2C is also available.)

JavaScript

Lisp (Early versions, pre-1962, and some experimental ones; production Lisp systems are compilers, butmany of them still provide an interpreter if needed.)

Tea

LPC

Lua

MUMPS (an ANSI standard general purpose language)

Maple

Mathematica

MATLAB

Oriel

Pascal (early implementations)

PCASTL

Perl

PHP

Pikt

PostScript

PROSE

Python

REXX

R

REBOL

Ruby

S-Lang

Standard ML (SML)

Spin

Tcl

TI-BASIC

TorqueScript

thinBasic scripting language

VBScript

Windows PowerShell (Microsoft .NET-based CLI)

Wolfram Language

LITTLE LANGUAGES

Little languages serve a specialized problem domain.

awk – can serve as a prototyping language for C (shares similar syntax)

Comet – used to solve complex combinatorial optimization problems in areas such as resource allocation and scheduling

SQL – has only a few keywords, and not all the constructs needed for a full programming language[1] – many database management systems extend SQL with additional constructs as a stored procedure language

Logic-based languages[edit]

See also: Category:Logic programming languages

Logic-based languages specify a set of attributes that a solution must have, rather than a set of steps to obtain a solution. Examples:

ALF

Alma-0

CLACL (CLAC-Language)

Curry

Fril

Janus

λProlog (a logic programming language featuring polymorphic typing, modular programming, and higher-order programming)

Leda

Oz, and Mozart Programming System cross-platform Oz

Poplog

Prolog (formulates data and the program evaluation mechanism as a special form of mathematical logic called Horn logic and a general proving mechanism called logical resolution)

Mercury (based on Prolog)

Strawberry Prolog (standard Prolog with some extensions)

Visual Prolog (object-oriented Prolog extension)

ROOP

Machine languages[edit]

Machine languages are directly executable by a computer's CPU. They are typically formulated as bit patterns, usually represented in octal or hexadecimal. Each bit pattern causes the circuits in the CPU to execute one of the fundamental operations of the hardware. The activation of specific electrical inputs (e.g., CPU package pins for microprocessors), and logical settings for CPU state values, control the processor's computation. Individual machine languages are specific to a family of processors; machine-language code for one family of processors cannot run directly on processors in another family unless the processors in question have additional hardware to support it (for example, DEC VAX processors included a PDP-11 compatibility mode). They are (essentially) always defined by the CPU developer, not by 3rd parties. The symbolic version, the processor's assembly language, is also defined by the developer, in most cases. Some commonly used machine code instruction sets are:

ARM

Original 32-bit

16-bit Thumb instructions (subset or registers used)

64-bit (major architecture change, more registers)

DEC PDP-6/PDP-10/DECSYSTEM-20

DEC PDP-11 (influenced VAX and M68000)

DEC VAX

DEC Alpha

Intel 8008, 8080 and 8085

Zilog Z80

x86:

16-bit x86, first used in the Intel 8086

Intel 8086 and 8088 (the latter was used in the first and early IBM PC)

Intel 80186

Intel 80286 (the first x86 processor with protected mode, used in the IBM AT)

IA-32, introduced in the 80386

x86-64 The original specification was created by AMD. There are vendor variants, but they're essentially the same:

AMD's AMD64

Intel's Intel 64

IBM System/360 and successors, including z/Architecture

MIPS

Motorola 6800

Motorola 68000 family (CPUs used in early Apple Macintosh and early Sun computers)

MOS Technology 65xx

6502 (CPU for VIC-20, Apple II, and Atari 800)

6510 (CPU for Commodore 64)

Western Design Center 65816/65802 (CPU for Apple IIGS and (variant) Super Nintendo Entertainment System)

National NS320xx

Power Architecture

POWER, first used in the IBM RS/6000

PowerPC – used in Power Macintosh and the technology is used in many older generation game consoles

Sun, Oracle SPARC

MCST Elbrus 2000

Macro languages[edit]

Textual substitution macro languages[edit]

See also: Category:Macro programming languages

Macro languages transform one source code file into another. A "macro" is essentially a short piece of text that expands into a longer one (not too be confused with hygienic macros), possibly with parameter substitution. They are often used to preprocess source code. Preprocessors can also supply facilities like file inclusion.

Macro languages may be restricted to acting on specially labeled code regions (pre-fixed with a # in the case of the C preprocessor). Alternatively, they may not, but in this case it is still often undesirable to (for instance) expand a macro embedded in a string literal, so they still need a rudimentary awareness of syntax. That being the case, they are often still applicable to more than one language. Contrast with source-embeddable languages like PHP, which are fully featured.

cpp (the C preprocessor)

m4 (originally from AT&T, bundled with Unix)

Application macro languages[edit]

Scripting languages such as Tcl and ECMAScript (ActionScript, ECMAScript for XML, JavaScript, JScript) have been embedded into applications. These are sometimes called "macro languages", although in a somewhat different sense to textual-substitution macros like m4.

Metaprogramming languages[edit]

Metaprogramming is writing of programs that write or manipulate other programs (or themselves) as their data or that do part of the work that is otherwise done at run time during compile time. In many cases, this allows programmers to get more done in the same amount of time as they would take to write all the code manually.

C++

Curl

D

eC

Elixir

Forth

Groovy

Haskell

Julia

Lisp

Lua

Maude system

Mathematica

MetaOCaml

Nemerle

Perl

Python

REBOL

Ruby

Rust

Scheme

SequenceL

Smalltalk

Wolfram Language

XL (concept programming)

Multiparadigm languages[edit]

Main article: Comparison of multi-paradigm programming languages

Multiparadigm languages support more than one programming paradigm. They allow a program to use more than one programming style. The goal is to allow programmers to use the best tool for a job, admitting that no one paradigm solves all problems in the easiest or most efficient way.

Ada (concurrent, distributed, generic (template metaprogramming), imperative, object-oriented (class-based))

ALF (functional, logic)

Alma-0 (constraint, imperative, logic)

APL (functional, imperative)

BETA (functional, imperative, object-oriented (class-based))

C++ (generic, imperative, object-oriented (class-based), functional)

C# (generic, imperative, object-oriented (class-based), functional, declarative)

Ceylon (generic, imperative, object-oriented (class-based), functional, declarative)

ChucK (imperative, object-oriented, time-based, concurrent, on-the-fly)

Cobra (generic, imperative, object-oriented (class-based), functional, contractual)

Common Lisp (functional, imperative, object-oriented (class-based), aspect-oriented (user may add further paradigms, e.g., logic))

Curl (functional, imperative, object-oriented (class-based), metaprogramming)

Curry (concurrent, functional, logic)

D (generic, imperative, functional, object-oriented (class-based), metaprogramming)

Delphi (generic, imperative, object-oriented (class-based), metaprogramming)

Dylan (functional, object-oriented (class-based))

eC (generic, imperative, object-oriented (class-based))

ECMAScript (functional, imperative, object-oriented (prototype-based))

ActionScript

ECMAScript for XML

JavaScript

JScript

Eiffel (imperative, object-oriented (class-based), generic, functional (agents), concurrent (SCOOP))

F# (functional, generic, object-oriented (class-based), language-oriented)

Fantom (functional, object-oriented (class-based))

FPr (function-level, object-oriented (class-based))

Go (functional, object-oriented (class-based), imperative, procedural),

Groovy (functional, object-oriented (class-based),imperative,procedural)

Harbour

Hop

J (functional, imperative, object-oriented (class-based))

Julia (imperative, multiple dispatch ("object-oriented"), functional, metaprogramming)

LabVIEW (dataflow, visual)

Lava (object-oriented (class-based), visual)

Leda (functional, imperative, logic, object-oriented (class-based))

Lua (functional, imperative, object-oriented (prototype-based))

Mercury (functional, logical, object-oriented)

Metaobject protocols (object-oriented (class-based, prototype-based))

Nemerle (functional, object-oriented (class-based), imperative, metaprogramming)

Objective-C (imperative, object-oriented (class-based), reflective)

OCaml (functional, imperative, object-oriented (class-based))

Oz (functional (evaluation: eager, lazy), logic, constraint, imperative, object-oriented (class-based), concurrent, distributed), and Mozart Programming System cross-platform Oz

Object Pascal (imperative, object-oriented (class-based))

Perl (imperative, functional (can't be purely functional), object-oriented, class-oriented, aspect-oriented (through modules))

PHP (imperative, object-oriented)

Pike

Poplog (functional, imperative, logic)

Prograph (dataflow, object-oriented (class-based), visual)

Python (functional, compiled, interpreted, object-oriented (class-based), imperative, metaprogramming, extension, impure, interactive mode, iterative, reflective, scripting)

R (array, interpreted, impure, interactive mode, list-based, object-oriented prototype-based, scripting)

Racket (functional, imperative, object-oriented (class-based) and can be extended by the user)

REBOL (functional, imperative, object-oriented (prototype-based), metaprogramming (dialected))

ROOP (imperative, logic, object-oriented (class-based), rule-based)

Ruby (imperative, functional, object-oriented (class-based), metaprogramming)

Rust (concurrent, functional, imperative, object-oriented)

Scala (functional, object-oriented)

Seed7 (imperative, object-oriented, generic)

SISAL (concurrent, dataflow, functional)

Spreadsheets (functional, visual)

Tcl (functional, imperative, object-oriented (class-based))

Tea (functional, imperative, object-oriented (class-based))

Windows PowerShell (functional, imperative, pipeline, object-oriented (class-based))

Wolfram Language

XL (concept programming approach)

Numerical analysis[edit]

AIMMS

AMPL

Analytica

GAUSS

GAMS

Julia

Klerer-May System

Mathematica

MATLAB

PROSE

Seneca – an Oberon variant

Wolfram Language

Non-English-based languages[edit]

Main article: non-English-based programming languages

ARLOGO – Arabic

Chinese BASIC – Chinese

Fjölnir – Icelandic

Language Symbolique d'Enseignement – French

Lexico – Spanish

Rapira – Russian

Object-oriented class-based languages

Class-based Object-oriented programming languages support objects defined by their class. Class definitions include member data. Message passing is a key concept (if not the key concept) in Object-oriented languages.

Polymorphic functions parameterized by the class of some of their arguments are typically called methods. In languages with single dispatch, classes typically also include method definitions. In languages with multiple dispatch, methods are defined by generic functions. There are exceptions where single dispatch methods are generic functions (e.g. Bigloo's object system). Examples includes:

Common Lisp

Cecil

Dylan

Julia

ActionScript 3.0

Actor

Ada 95 and Ada 2005 (multi-purpose language)

BETA

Blue

C++

C#

Ceylon

Oxygene (formerly known as Chrome)

ChucK

Cobra

ColdFusion

Curl

D

DASL

Delphi

E

GNU E

eC

Eiffel

Sather

Ubercode

F-Script

Fortran 2003

Fortress

FPr

Gambas

Game Maker Language

Harbour

J

Java

Processing

Groovy

Join Java

Tea

X10

LabVIEW

Lava

Lua

Modula-2 (data abstraction, information hiding, strong typing, full modularity)

Modula-3 (added more object-oriented features to Modula-2)

Nemerle

IBM NetRexx

Oberon-2 (full object-orientation equivalence in an original, strongly typed, Wirthian manner)

Object Pascal

Object REXX

Objective-C (a superset of C adding a Smalltalk derived object model and message passing syntax)

OCaml

Oz, Mozart Programming System

Perl 5

PHP

Pike

Prograph

Python (interpretive language, optionally object-oriented)

Realbasic

Ruby

Scala

Simula (the first object-oriented language, developed by Ole-Johan Dahl and Kristen Nygaard)

Smalltalk (pure object-orientation, developed at Xerox PARC)

Bistro

F-Script

Little Smalltalk

Squeak

Scratch

IBM VisualAge

VisualWorks

SPIN

SuperCollider

Transcript (programmer does not get to pick the objects)

VBScript (Microsoft Office 'macro scripting' language)

Visual DataFlex

Visual FoxPro

Visual Prolog

X++

XOTcl

Object-oriented prototype-based languages

Prototype-based languages are object-oriented languages where the distinction between classes and instances has been removed:

ABCL/1

ABCL/R

ABCL/R2

ABCL/c plus

Agora

Cecil

ECMAScript

ActionScript

ECMAScript for XML

JavaScript (first named Mocha, then LiveScript)

JScript

Etoys in Squeak

Io

Lisaac

Lua

MOO

NewtonScript

Obliq

R

REBOL

Self (the first prototype-based language, derived from Smalltalk)

TADS

Off-side rule languages

Off-side rule languages are those where blocks are formed, indicated, by their indentation.

ISWIM, the abstract language that introduced the rule

ABC, Python's parent

Python

Cobra

Boo

Genie

Miranda, Haskell's parent

Orwell

Haskell

Curry

Elixir

F#

Nim

Occam

SPIN

XL

Procedural languages

Procedural programming languages are based on the concept of the unit and scope (the data viewing range of an executable code statement). A procedural program is composed of one or more units or modules, either user coded or provided in a code library; each module is composed of one or more procedures, also called a function, routine, subroutine, or method, depending on the language. Examples of procedural languages include:

Ada (multi-purpose language)

ALGOL (extremely influential language design – the second high level language compiler)

SMALL Machine Algol Like Language

Alma-0

BASIC (BASICs are innocent of most modularity in (especially) versions before about 1990)

BCPL

BLISS

Blue

C

C++ (C with objects plus much else, such as, generics through STL)

C# (similar to Java/C++)

Ceylon

ChucK (C/Java-like syntax, with new syntax elements for time and parallelism)

COBOL

Cobra

ColdFusion

Combined Programming Language (CPL)

Curl

D

DASL (partly declarative, partly imperative)

eC

ECMAScript

ActionScript

ECMAScript for XML

JavaScript (first named Mocha, then LiveScript)

JScript

Eiffel

Fortran (better modularity in later Standards)

F

GAUSS

Go

Harbour

HyperTalk

Java

Groovy

Join Java

Tea

JOVIAL

Julia

LANSA

Lasso

Modula-2 (fundamentally based on modules)

Mathematica

MATLAB

MUMPS (More modular in its first release than a language of the time should have been; The standard has become still more modular since then.)

Nemerle

Oberon and Oberon-2 (improved, smaller, faster, safer follow-ons for Modula-2)

Component Pascal

Lagoona

Seneca

Occam

Oriel

Pascal (successor to ALGOL 60, predecessor of Modula-2)

Free Pascal (FPC)

Object Pascal (Delphi)

PCASTL

Perl

Pike

PL/C

PL/I (large general purpose language, originally for IBM mainframes)

Plus

PROSE

Python

R

Rapira

RPG (available only in IBM's System i midrange computers)

Rust

S-Lang

VBScript

Visual Basic

Visual FoxPro

Wolfram Language

X++

X#

XL

Reflective languages

Reflective languages let programs examine and possibly modify their high level structure at runtime. This is most common in high-level virtual machine programming languages like Smalltalk, and less common in lower-level programming languages like C. Languages and platforms supporting reflection:

Befunge

C#

Ceylon

Charm

ChucK

Cobra

Component Pascal BlackBox Component Builder

Curl

Delphi

eC

ECMAScript

ActionScript

ECMAScript for XML

JavaScript

JScript

Eiffel

Forth

Harbour

Java

Java virtual machine

Groovy

Join Java

X10

Julia

Lisp

Clojure

Common Lisp

Dylan

Logo

Scheme

Lua

Maude system

.NET Framework Common Language Runtime

Oberon-2 – ETH Oberon System

Objective-C

PCASTL

Perl

PHP

Pico

Poplog

POP-11

Prolog

Python

REBOL

Ruby

Smalltalk (pure object-orientation, originally from Xerox PARC)

Bistro

F-Script

Little Smalltalk

Self

Squeak

IBM VisualAge

VisualWorks

Snobol

Tcl

Wolfram Language

XOTcl

X++

XL

Scripting languages

Scripting language has two seemingly different, but in fact similar meanings. In a traditional sense, scripting languages are designed to automate frequently used tasks that usually involve calling or passing commands to external programs. Many complex application programs provide built-in languages that let users automate tasks. Those that are interpretive are often called scripting languages.

Recently, many applications have built-in traditional scripting languages, such as Perl or Visual Basic, but there are quite a few native scripting languages still in use. Many scripting languages are compiled to bytecode and then this (usually) platform-independent bytecode is run through a virtual machine (compare to Java virtual machine).

AppleScript

AWK

BeanShell

Bash

Ch (Embeddable C/C++ interpreter)

CLIST

ColdFusion

ECMAScript

ActionScript

ECMAScript for XML

JavaScript (first named Mocha, then LiveScript)

JScript

CMS EXEC

EXEC 2

F-Script

Falcon

Game Maker Language (GML)

ICI

Io

JASS

Golo

Groovy

Join Java

Julia (still, compiled on the fly to machine code)

Lasso

Lua

MAXScript

MEL

Oriel

Perl

PHP (intended for Web servers)

Pikt

Python

R

REBOL

REXX

Ruby

Smalltalk

S-Lang

sed

Tea

Tcl

TorqueScript

Transcript

VBScript

WebDNA, dedicated to database-driven websites

Windows PowerShell (Microsoft .NET-based CLI)

Winbatch

Many shell command languages such as the Unix shell or DCL on VMS have powerful scripting abilities.

Synchronous languages

Synchronous programming languages are optimized for programming reactive systems, systems that are often interrupted and must respond quickly. Many such systems are also called realtime systems, and are used often in embedded systems. Examples:

Argus

Averest

Esterel

Lustre

SyncCharts

Syntax handling languages:

These languages assist with generating lexical analyzers and parsers for Context-free grammars.

ANTLR

Coco/R (EBNF with semantics)

GNU bison (FSF's version of Yacc)

GNU Flex (FSF's version of Lex)

glex/gyacc (GoboSoft compiler compiler to Eiffel)

lex (Lexical Analysis, from Bell Labs)

M4

yacc (yet another compiler compiler, from Bell Labs)

JavaCC

Visual languages

Visual programming languages let users specify programs in a two-(or more)-dimensional way, instead of as one-dimensional text strings, via graphic layouts of various types.

Analytica

Blockly

CODE

DRAKON

Fabrik

G (used in LabVIEW)

Lava

Limnor

Max

NXT-G

Pict programming language

Prograph

Pure Data

Quartz Composer

Scratch (written in and based on Squeak, a version of Smalltalk)

Snap!

Simulink

Spreadsheets

Subtext

ToonTalk

VEE

VisSim

vvvv

EICASLAB

NOTE: Some dataflow programming languages are also visual languages.

Wirth languages

Computer scientist Niklaus Wirth designed and implemented several influential languages.

ALGOL W

Euler

Modula

Modula-2 (and Modula 3, etc. variants)

Obliq Modula 3 variant

Oberon (Oberon, Oberon-07, and Oberon-2)

Component Pascal

Lagoona

Oberon-2

Pascal

Object Pascal ("umbrella" name for Delphi, Free Pascal, Oxygene and others)

XML-based languages

These are languages based on or that operate on XML.

Ant

Cω

ECMAScript for XML

MXML

LZX

XAML

XPath

XQuery

XProc

XSLT

Concurrent languages

Message passing languages provide language constructs for concurrency. The predominant paradigm for concurrency in mainstream languages such as Java is shared memory concurrency based on monitors. Concurrent languages that make use of message passing have generally been inspired by CSP or the π-calculus, but have had little commercial success, except for Ada and Erlang. Ada is a multipurpose language and concurrent programming is only one option available.

Ada (multi-purpose language)

Alef – concurrent language with threads and message passing, used for systems programming in early versions of Plan 9 from Bell Labs

Ateji PX an extension of the Java language for parallelism

ChucK – domain specific programming language for audio, precise control over concurrency and timing

Cilk – a concurrent C

Cω – C Omega, a research language extending C#, uses asynchronous communication

Clojure – a dialect of Lisp for the Java virtual machine

Chapel

Co-array Fortran

Concurrent Pascal (by Brinch-Hansen)

Curry

E – uses promises, ensures deadlocks cannot occur

Eiffel (through the SCOOP mechanism, Simple Concurrent Object-Oriented Computation)

Erlang – uses asynchronous message passing with nothing shared

Go

Java

Join Java – concurrent language based on Java

X10

Julia

Join-calculus

Joule – dataflow language, communicates by message passing

Limbo – relative of Alef, used for systems programming in Inferno (operating system)

MultiLisp – Scheme variant extended to support parallelism

occam – influenced heavily by Communicating Sequential Processes (CSP)

occam-π – a modern variant of occam, which incorporates ideas from Milner's π-calculus

Orc

Oz – multiparadigm language, supports shared-state and message-passing concurrency, and futures, and Mozart Programming System cross-platform Oz

Pict – essentially an executable implementation of Milner's π-calculus

Rust – actor-based

SALSA – actor language with token-passing, join, and first-class continuations for distributed computing over the Internet

Scala – implements Erlang-style actors on the JVM

SequenceL – purely functional, automatically parallelizing and race-free

SR – research language

Unified Parallel C

XProc – XML processing language, enabling concurrency

Curly-bracket languages

The curly-bracket or curly-brace programming languages have a syntax that defines statement blocks using the curly bracket or brace characters { }. This syntax originated with BCPL (1966), and was popularized by C (1972). Many curly-bracket languages descend from or are strongly influenced by C. Examples of curly-bracket languages include:

ABCL/c+

Alef

Limbo

Go

AutoHotkey

AWK

B

bc

BCPL

C – developed circa 1970 at Bell Labs

C++

C#

Ceylon

ChucK – audio programming language

Cilk – concurrent C for multithreaded parallel programming

COFFEE

Cyclone – a safer C variant

D

Dart

DASL – based on Java

E

eC

ECMAScript

ActionScript

ECMAScript for XML

JavaScript

JScript

TypeScript

Fantom

GML (Game Maker Language)

GLSL

ICI

Java

Golo

Groovy

Join Java

Kotlin

Processing

Tea

X10

Xtend

LPC

MSL

MEL

Nemerle – combines C# and ML features, provides syntax extension abilities

PCASTL

Perl

PHP

Pico

Pike

R

Rust

S-Lang

Scala

sed

SuperCollider

Swift

UnrealScript

Windows PowerShell (Microsoft .NET-based CLI)

Yorick

Dataflow languages

Dataflow programming languages rely on a (usually visual) representation of the flow of data to specify the program. Frequently used for reacting to discrete events or for processing streams of data. Examples of dataflow languages include:

Hartmann pipelines

G (used in LabVIEW)

Lucid

Max

Oz

Prograph

Pure Data

Reaktor

StreamBase StreamSQL EventFlow

VEE

VHDL

VisSim

WebMethods Flow

Data-oriented languages

Data-oriented languages provide powerful ways of searching and manipulating the relations that have been described as entity relationship tables which map one set of things into other sets. Examples of data-oriented languages include:

Clarion

Clipper

dBase a relational database access language

MUMPS (an ANSI standard general purpose language with specializations for database work)

Caché (similar to MUMPS)

SPARQL

SQL

Tutorial D

Visual FoxPro – a native RDBMS engine, object-oriented, RAD

WebDNA

WebQL

Wolfram Language

Data-structured languages

Data-structured languages are those where logic is structured in ways similar to their data. Such languages are generally well suited to reflection and introspection. There are three main types:

Array-based

List-based

Stack-based

Assembly languages that statically link data inline with instructions can also be considered data-structured, in the most primitive way.

Array language

Array programming (also known as vector or multidimensional) languages generalize operations on scalars to apply transparently to vectors, matrices, and higher-dimensional arrays.

A+

Ada

Analytica

APL

Chapel

Fortran

Freemat

GAUSS

J

Julia

K

MATLAB

Octave

Q

R

S

S-Lang

SequenceL

X10

ZPL

IDL

Wolfram Language

List-based languages

List-based languages are a type of data-structured language that are based upon the list data structure.

Lisp

Arc

Clojure

Common Lisp

Dylan

Emacs Lisp

Racket

Scheme

Logo

FPr

Joy

R

Tcl

Tea

TRAC

Stack-based languages

Stack-based languages are a type of data-structured language that are based upon the stack data structure.

Beatnik

colorForth

Factor

Forth

Joy (all functions work on parameter stacks instead of named parameters)

Piet

Poplog via its implementation language POP-11

PostScript

RPL

Declarative languages

Declarative languages describe a problem rather than defining a solution. Declarative programming stands in contrast to imperative programming via imperative programming languages, where serial orders (imperatives) are given to a computer. In addition to the examples given just below, all (pure) functional and logic-based programming languages are also declarative. In fact, "functional" and "logical" constitute the usual subcategories of the declarative category.

Analytica

Ant (partially declarative languages, partially imperative programming)

DASL (partially declarative languages, partially imperative programming)

Lustre

Mercury

MetaPost

Modelica

Prolog

Oz

SequenceL – purely functional, automatically parallelizing and race-free

SPARQL

SQL

Wolfram Language

xBase

XSL Transformations

Embeddable languages

Source embeddable languages embed small pieces of executable code inside a piece of free-form text, often a web page.

Client-side embedded languages are limited by the abilities of the browser or intended client. They aim to provide dynamism to web pages without the need to recontact the server.

Server-side embedded languages are much more flexible, since almost any language can be built into a server. The aim of having fragments of server-side code embedded in a web page is to generate additional markup dynamically; the code itself disappears when the page is served, to be replaced by its output. They are classified into two main parts, examples are:

Server side:

PHP

VBScript

SMX – dedicated to web pages

WebDNA – dedicated to database-driven websites

The above examples are particularly dedicated to this purpose. A large number of other languages, such as Erlang, Scala, Perl and Ruby can be adapted (for instance, by being made into Apache modules).

Client side:

ActionScript

Java

JavaScript

ECMAScript

JScript

VBScript (Windows only)

Educational languages: These are languages developed primarily for the purpose of teaching and learning of programming.

Alice

Blockly

Blue

COMAL

Elan

Logo

KTurtle

Modula-2

Pascal

Scheme

Scratch

Snap!

Turing

Wolfram Language

Esoteric languages

An esoteric programming language is a programming language designed as a test of the boundaries of computer programming language design, as a proof of concept, or as a joke.

Beatnik

Befunge

Brainfuck

Chef

INTERCAL

LOLCODE

Malbolge

Piet

Shakespeare

Whitespace

Extension languages

Extension programming languages are languages embedded into another program and used to harness its features in extension scripts.

Ateji PX – an extension of the Java language for parallelism

AutoLISP (specific to AutoCAD)

BeanShell

CAL

C/AL(C/SIDE)

Guile

Emacs Lisp

JavaScript and some dialects (e.g. JScript)

Lua – e.g. embedded in many games

OpenCL – an extension of C and C++ to use the GPU and parallel extensions of the CPU

OptimJ – an extension of the Java programming language with language support for writing optimization models and powerful abstractions for bulk data processing

Perl

Pike

Python (embedded in Maya, Blender and other 3-D animation packages)

REXX

Ruby (Google SketchUp)

S-Lang

SQL

Squirrel

Tcl

Vim script

VBA

Windows PowerShell

Fourth-generation languages

Fourth-generation programming languages are high-level languages built around database systems. They are generally used in commercial environments.

ABAP

CorVision

CSC's GraphTalk

Easytrieve report generator (now CA-Easytrieve Plus)

FOCUS

IBM Informix-4GL / Aubit-4GL

LANSA

LINC 4GL

MAPPER (Unisys/Sperry) – now part of BIS

MARK-IV (Sterling/Informatics) now VISION:BUILDER of CA

Progress 4GL

SAS

Transcript (not based on a database; still, the goal is to work at a higher level of abstraction than 3GLs)

Ubercode (VHLL, or Very High Level Language)

Uniface

Visual DataFlex

Visual FoxPro

xBase

Functional languages

Functional programming languages define programs and subroutines as mathematical functions. Many so-called functional languages are "impure", containing imperative features. Many functional languages are tied to mathematical calculation tools. Functional languages include:

Pure

Agda

Charity

Clean

Coq (Gallina)

Curry

Elm

Frege

Haskell

Hope

Joy

Mercury

Miranda

Idris

SequenceL

Impure

APL

ATS

CAL

C

C++ (since C++11)

C#

Ceylon

D

Dart

Curl

ECMAScript

ActionScript

ECMAScript for XML

JavaScript

JScript

Elm

Erlang

Elixir

LFE

F#

FPr

Groovy

Hop

J

Java (since version 8)

Julia

Lisp

Clojure

Common Lisp

Dylan

Emacs Lisp

LFE

Little b

Logo

Scheme

Racket (formerly PLT Scheme)

Tea

Mathematica

ML

Standard ML (SML)

Alice

OCaml

Nemerle

Opal

OPS5

Poplog

Python

Q (equational programming language)

Q (programming language from Kx Systems)

R

REBOL

Ruby

REFAL

Rust

Scala

Spreadsheets

Wolfram Language

Hardware description languages

In electronics, a Hardware description language or HDL is a specialized computer language used to describe the structure, design and operation of electronic circuits, and most commonly, digital logic circuits. The two most widely used and well-supported HDL varieties used in industry are Verilog and VHDL. More Examples on this type of programming language include:

HDLs for analog circuit design

Verilog-AMS (Verilog for Analog and Mixed-Signal)

VHDL-AMS (VHDL with Analog/Mixed-Signal extension)

HDLs for digital circuit design

Advanced Boolean Expression Language (ABEL)

Altera Hardware Description Language (AHDL)

Bluespec

Chisel

Confluence

ELLA

ESys.net

Handel-C

HHDL

Impulse C

JHDL

Lava

Lola

M

MyHDL

PALASM

Ruby (hardware description language)

SystemC

SystemVerilog

Verilog

VHDL (VHSIC HDL, Very High Speed Integrated Circuit Hardware Description Language)

Imperative languages

Imperative programming languages may be multi-paradigm and appear in other classifications. Examples are:

Ada

ALGOL

BASIC

Blue

C

C++

C#

Ceylon

COBOL

D

eC

ECMAScript (JavaScript)

FORTRAN

GAUSS

Go

Groovy

Java

Julia

Lua

MATLAB

Machine language

Modula-2, Modula-3

MUMPS

Nim

Oberon

Object Pascal

OCaml

Pascal

Perl

PHP

PROSE

Python

Ruby

Rust

Wolfram Language

Interactive mode languages

Interactive mode languages act as a kind of shell: expressions or statements can be entered one at a time, and the result of their evaluation is seen immediately. The interactive mode is also known as a REPL (read–eval–print loop). Examples include:

APL

BASIC (some dialects)

Clojure

Common Lisp

Dart (with Observatory or Dartium's developer tools)

Erlang

F#

Forth

FPr

Fril

GAUSS

Groovy

Haskell (with the GHCi or Hugs interpreter)

IDL

J

JavaScript (using command line tools like Node.js or Rhino or the developer tools built into web browsers like Firefox or Chrome)

Julia

Lua

MUMPS (an ANSI standard general purpose language)

Maple

Mathematica (Wolfram language)

MATLAB

ML

Perl

Pike

PostScript

Python

PROSE

R

REBOL

REXX

Ruby (with IRB)

Scala

Scheme

Smalltalk (anywhere in a Smalltalk environment)

S-Lang (with the S-Lang shell, slsh)

Tcl (with the Tcl shell, tclsh)

Windows PowerShell (Microsoft .NET-based CLI)

Visual FoxPro