WIKIPEDIA

IDL (programming language)

IDL, short for **Interactive Data Language**, is a programming language used for data analysis. It is popular in particular areas of science, such as astronomy, atmospheric physics and medical imaging. IDL shares a common syntax with PV-Wave and originated from the same codebase, though the languages have subsequently diverged in detail. There are also free or costless implementations, such as GNU Data Language (GDL) and Fawlty Language (http://www.flxpert.hu/fl/) (FL).

IDL (Interactive Data Language)

Paradigm	vector-oriented programming
Designed by	David Stern
Developer	David Stern & ITT Visual Information Solutions (ITT VIS)
First appeared	1977
Stable release	IDL 8.8 / August 2020
Typing discipline	Dynamic
Website	www.harrisgeospatial.com/Software-Technology/IDL (https://www.harrisgeospatial.com/Software-Technology/IDL)
Major implementations	
IDL, GNU Data Language, Fawlty Language	

Contents

Overview

IDL is vectorized, numerical, and interactive, and is commonly used for interactive processing of large amounts of data (including image processing). The syntax includes many constructs from Fortran and some from C.

IDL originated from early VMS Fortran, and its syntax still shows its heritage:

```
x = findgen(100)/10
y = sin(x)/x
plot,x,y
```

The `findgen` function in the above example returns a one-dimensional array of floating point numbers, with values equal to a series of integers starting at 0.

Note that the operation in the second line applies in a vectorized manner to the whole 100-element array created in the first line, analogous to the way general-purpose array programming languages (such as APL, J or K) would do it. This example contains a division by zero; IDL will report an arithmetic overflow, and store a NaN value in the corresponding element of the `y` array (the first one), but the other array elements will be finite. The NaN is excluded from the visualization generated by the `plot` command.

As with most other array programming languages, IDL is very fast at doing vector operations (sometimes as fast as a well-coded custom loop in Fortran or C) but quite slow if elements need processing individually. Hence part of the art of using IDL (or any other array programming language, for that matter) for numerically heavy computations is to make use of the built-in vector operations.

History

The predecessor versions of IDL were developed in the 1970s at the Laboratory for Atmospheric and Space Physics (LASP) at the University of Colorado at Boulder. At LASP, David Stern was involved in efforts to allow scientists to test hypotheses without employing programmers to write or modify individual applications. The first program in the evolutionary chain to IDL that Stern developed was named Rufus; it was a simple vector-oriented calculator that ran on the PDP-12. It accepted two-letter codes that specified an arithmetic operation, the input registers to serve as operands, and the destination register. A version of Rufus developed on the PDP-8 was the Mars Mariner Spectrum Editor (MMED). MMED was used by LASP scientists to interpret data from Mariner 7 and Mariner 9. Later, Stern wrote a program named SOL, which also ran on the PDP-8. Unlike its predecessors, it was a true programming language with a FORTRAN-like syntax. SOL was an array-oriented language with some primitive graphics capabilities.[1]

Stern left LASP to found Research Systems Inc. (RSI) in 1977. The first RSI product was IDL for the PDP-11.[1] In this release, the graphics supported by IDL were primarily Tektronix terminals and raster graphics displays. RSI sold its first IDL licenses to NASA's Goddard Space Flight Center and Ball Aerospace & Technologies Corp. in 1979. Two years later RSI released an initial VAX/VMS version of IDL, which was written in VAX MACRO and FORTRAN. It took advantage of the VAX virtual memory and 32-bit address space.[1] The National Center for Atmospheric Research (NCAR), the University of Michigan, the University of Colorado, and the Naval Research Laboratory started to use IDL with this version.

In 1987 RSI shifted development work of IDL to the Unix environment, which required a complete re-write of the code in C rather than a port of the existing version of VAX IDL. [1] Stern and Ali Bahrami rewrote IDL for Unix on the Sun 3, taking advantage of the re-write to extend and improve the language. Subsequently, IDL was further expanded and ported to several variants of Unix, VMS, Linux, Microsoft Windows (1992), and Mac OS (1994).

Widgets were added to IDL in 1992, providing event-driven programming with graphical user interfaces. In 1997 ION (IDL On the Net), a web server-based system, was commercially released. The first version of ENVI, an application for remote sensing multispectral and hyperspectral image analysis written in IDL, was released in 1994. ENVI was created, developed and owned by Better Solutions Consulting, LLC, until it was purchased from BSC in October 2000 by Eastman Kodak coincident with their purchase of RSI. RSI sold, marketed and supported ENVI under the terms of a license agreement with BSC, LLC from 1994 through October 2000. New object and pointer types, and limited object-oriented programming capabilities, were added to IDL in 1997.

IDL has been applied widely in space science, for example in solar physics. The European Space Agency used IDL to process almost all of the pictures of Halley's Comet taken by the Giotto spacecraft. The team repairing the Hubble Space Telescope used IDL to help them diagnose anomalies in the main mirror. In 1995, astronauts on board a Space Shuttle used IDL loaded on a laptop to study ultraviolet radiation. Currently, amongst other applications, IDL is being used for most of the analysis of the SECCHI part of the STEREO mission at NRL, USA, and at the Rutherford Appleton Laboratory, UK.

RSI became a wholly owned subsidiary of ITT Industries in March 2004. As of 15 May 2006, RSI began doing business as ITT Visual Information Solutions. Effective 31 October 2011, as a result of restructuring, that company became Exelis Visual Information Solutions. As of 2015, IDL is now owned and maintained by Harris Geospatial Solutions.

Features

As a computer language, IDL:

- is dynamically typed.
- has separate namespaces for variables, functions and procedures, but no namespace hierarchy.
- was originally single threaded but now has many multi-threaded functions and procedures.
- has all function arguments passed by reference; but see "problems", below.
- has named parameters called keywords which are passed by reference.
- provides named parameter inheritance in nested routine calls, by reference or value.
- does not require variables to be predeclared.
- provides COMMON block declarations and system variables to share global values among routines.
- provides a basic form of object-oriented programming, somewhat similar to Smalltalk, along with operator overloading.
- implements a persistent, global heap of pointer and object variables, using reference counting for garbage collection.
- compiles to an interpreted, stack-based intermediate p-code (à la Java Virtual Machine).
- provides a simple and efficient index slice syntax to extract data from large arrays.
- provides various integer sizes, as well as single and double precision floating point real and complex numbers.
- provides composite data types such as character strings, homogeneous-type arrays, lists, hash tables, and simple (non-hierarchical) record structures of mixed data types.

Problems

Some of these features, which make IDL very simple to use interactively, also cause difficulties when building large programs. The single namespace is particularly problematic; for example, language updates that include new built-in functions have on occasion invalidated large scientific libraries.[2]

Arrays are passed by reference, and this mechanism is an advertised feature of the language to pass data back out of a subroutine – in contrast, array slices are copied before being passed, so that data modifications do not flow back into array ranges (after the subroutine exits), violating the principle of least surprise.

Many historical irregularities survive from the early heritage of the language, requiring individual workarounds by the programmer. As an example:

- Array indexing and subroutine entry can both be carried out with exactly the same syntax (parentheses); this ambiguity, coupled with the single namespace for all variables and subroutines, can cause code to stop working when newly defined subroutines or language extensions conflict with local variable names. IDL programmers can avoid many of these problems by using square brackets for array indexing, thereby avoiding conflicts with function names which use parentheses.

The preceding issue can be alleviated using this compiler option:

```
COMPILE_OPT STRICTARR
```

- ITT Visual Information Solutions (ITT VIS), the developers of IDL, have taken explicit steps to prevent bytecode compatibility with other environments. Files containing compiled routines use a binary tagged-data-structure format that has not been officially published but has been investigated and documented by users[3] but also contain the following notice as ASCII text embedded within each saved file: "IDL Save/Restore files embody unpublished proprietary information about the IDL program. Reverse engineering of this file is therefore forbidden under the terms of the IDL End User License Agreement (IDL EULA). All IDL users are required to read and agree to the terms of the IDL EULA at the time that they install IDL. Software that reads or writes files in the IDL Save/Restore format must have a license from ITT Visual Information Solutions explicitly granting the right to do so. In this case, the license will be included with the software for your inspection. Please report software that does not have such a license to ITT Visual Information Solutions..." As of February 2010, the statement has not been tested in a court of law.

Also, that provision of the IDL EULA has no effect in Australia, as a result of sections 47D (http://www.austlii.edu.au/au/legis/cth/consol_act/ca1968133/s47d.html) and 47H (http://www.austlii.edu.au/au/legis/cth/consol_act/ca1968133/s47h.html) of that country's Copyright Act.

Examples

The following graphics were created with IDL (source code included):

- Image of random data plus trend, with best-fit line and different smoothings
- Plots of delta-o-18 against age and depth (from EPICA and Vostok)
- coyote IDL gallery (http://www.idlcoyote.com/gallery/index.html) examples of IDL imaging

See also

- List of numerical-analysis software
- ENVI – an image processing software package built in IDL
- IRAF – a free, graphical data reduction environment produced by NOAO
- MATLAB – a technical computing environment providing similar capabilities to IDL
- NumPy – an extension for Python that gives it array math capabilities similar to those of IDL
- Perl Data Language (PDL) – An extension to Perl that gives it array math capabilities similar to those of IDL
- Scilab - a high-level, numerically oriented programming language designed for Scientific computing and interfaces
- Solarsoft – library for solar data analysis and spacecraft operation activities written predominately in IDL
- GDL – GNU Data Language, a free implementation similar to IDL.
- Fawlty Language (http://www.flxpert.hu/fl/) – Fawlty Language is an IDL8 (Interactive Data Language) compatible compiler.

References

1. Schienle, Mike (1991-01-19). "IDL FAQ" (http://www.faculty.virginia.edu/rwoclass/astr511/IDLresources/idl-faq-ivsoft-v4.html). Retrieved 8 February 2019.

2. Fanning, David. "Program Naming Conflicts in IDL 8" (https://web.archive.org/web/20140306211336/http://www.idlcoyote.com/ng_tips/idl8_name_conflicts.html). Archived from the original (https://www.idlcoyote.com/ng_tips/idl8_name_conflicts.html) on 6 March 2014. Retrieved 30 September 2014.
3. Markwardt, Craig (2011-12-21). "Unofficial Format Specification of the IDL "SAVE" File" (http://www.physics.wisc.edu/~craigm/idl/savefmt/). Retrieved 2013-02-13.

External links

- IDL home page (http://www.harrisgeospatial.com/SoftwareandTechnology/IDL.aspx)
- Coyote's Guide to IDL Programming (http://www.idlcoyote.com/)
- The IDL Astronomy User's Library at NASA Goddard (http://idlastro.gsfc.nasa.gov/)
- Fawlty Language home page (http://www.flxpert.hu/fl/)