

APL 2010 LPA – Berlin
Array Processing Languages
Learn Parallel Applications



Program
– not yet complete –

Invited Speakers

Dr. James A. Brown (CEO, SmartArrays, Inc.): APL and its Influence on Modern Computing

This paper identifies major features of APL and its array oriented cousins and examines how they have influenced modern day computing facilities. You can find various aspects of APL available (sometimes acknowledged and sometimes not) in many offerings showing that APL has had a profound influence on computing. This is especially true if you include in the picture some of the early APL applications which could be written by people in the professions who were not professional programmers and who could produce significant applications because of the power of the notation.

Yet there are important ideas from APL that have not been exploited or are only beginning to be seen. This leads to the conclusion that APL is today a viable platform for flexible and high performing applications and that APL skills will continue to be valuable in the marketplace far into the future.

Dr. habil. Sven-Bodo Scholz (University of Hertfordshire, UK): Multi-/ Many-Cores: array programming at the heart of the hype!

Array programming, with its roots in data-parallelism, fits very well the programming needs of ongoing multi-/many-core development. However, there are many technical obstacles that need to be overcome to make this vision fly. We have been working on these issues over the last 15 years in the context of SaC, a functional array language (www.sac-home.org).

In this talk, I will highlight the key issues we have encountered on our journey towards compiling high-level array programs into high-performance code. In particular, I will focus on the different requirements and challenges found alongside the road when targeting diverse platforms: My journey will go



APL Germany e.V.



from general purpose hardware, via restrained settings like GPGPUs towards cutting edge, massively parallel systems where concurrency lies at the heart of performance.

Keynote Speakers

Helmut Weber (IBM Deutschland RD GmbH): MultiCore & Hybrid Systems - New Computing Trends?

This presentation will give an overview about recent and current developments in multicore and hybrid systems and then take a closer look at these fundamental trends in computing from multiple different angles: What are the technology drivers, what are the key system aspects and what are the programming paradigms? And last not least - most importantly - what are the killer applications, the business drivers and the competing trends.

The discussion of these areas will outline the most important characteristics of multi-core and hybrid designs, compare them with the more traditional design points and also similar trends in the past. Finally, the summary and conclusion will attempt to establish an outlook to the future of heterogeneous computing.

Papers

Topic: *State of the art of array processing languages*

APL Prototype Functions

The concept of empty arrays while not unique to APL, has been most thoroughly developed there. Nonetheless, the rules for handling empty arrays are inconsistent across the major APL implementations where, for example, the sum of two empty arrays can not only give different results, but also, sometimes (and correctly so) a LENGTH, RANK or DOMAIN error.

This paper attempts to give a comprehensive treatment of prototypes and set down consistent rules for computations with them. Along the way, we introduce the concept of a prototype function – a function associated with each primitive and derived function when working on empty arguments.

Parallel Computation Using PEACH, PRANK and PDOT

One of the challenges currently facing software developers is to take advantage of the parallel hardware that is appearing not only in large data centers, but also on every desktop. In addition to the fine-grained parallelism provided by primitive functions which operate on large arrays without loops, APL language interpreters offer a number of “coarse-grained” parallel constructs, which allow multiple invocations of user-defined functions to be

Sponsored by:

DYALOG LTD

Gold Partner

APL2000 SM
Rapid Application Development

Gold Partner

IBM®
Gold Partner

DPC

Silver Partner

expressed. Three examples are each (`**`), which is available in virtually all modern systems, rank (`ρ`), implemented in SHARP APL and J, and the dot in Dyalog APL which, when placed between an array of objects on the left and an expression on the right, applies the expression to its right to each of the objects on the left.

The paper describes a number of experimental user-defined operators named PEACH, PRANK and PDOT, models of potential extensions to APL interpreters, which allow us to experiment with the performance characteristics of parallel execution using multiple cores in one or more co-operating machines.

Unifying Traditional Functions and D-Fns in APL#

APL systems provide a definition mechanism so that expressions may be collected into non-primitive or “user-defined” functions and operators.

The traditional APL defined function, even when extended with control-structures, is procedural in nature, and does nothing to discourage looping, destructive assignment or the use of non-result-returning and niladic “functions”.

In 1996, Dyalog introduced a purer functional definition style, now referred to as a “D-Function”, which was designed to fit better with the functional programming paradigm.

This paper details an attempt, in Dyalog’s APL# project, to combine both the traditional “Trad-Fn” and functional “D-Fn” definition styles into a unified whole, which supports both the procedural and functional modes of programming.

A Regular Expression System Operator

In so-called “scripting languages” (Perl, Ruby, Awk and Tcl, to name a few), the ability to search text using “regular expressions” is a cornerstone for the power and flexibility that these languages deliver. Although APL is (currently) mostly used to process numeric data, APL has most of the characteristics of a good scripting language, and many current and future APL applications could benefit from the availability of regular expression support tightly integrated with the language.

This paper will discuss the design decisions which led ultimately to Dyalog’s first “system operator”, which integrates regular expressions as implemented by the popular PCRE engine with Dyalog APL. APL users can search text using “Perl Compatible Regular Expressions”, and make modifications either by using a simple transformation syntax (similar to that used by the Unix utility “sed”), or by using an APL function as an operand to the system operator, to express the transformation.

APL2XML

Did you ever wonder why XML is such a success, or whether object-oriented programming in APL is worth all the fuss? This talk provides answers to both questions. APL2XML stands for a couple of classes that allows the creation of XML files excepted as input files by a third-party software, HelpAndManual, which is designed to create CHM or HLP help files, PDF documents, e-books or RTF documents from those XML files. HelpAndManual is by far the best software available for creating any kind of help files. With APL2XML one can automatically extract information from well-documented classes (just an example), create XML files and then create the appropriate kind of output with the help of HelpAndManual. APL2XML will become available as part of APLAPL (APL Application Programming Library) which is an Open Source project.

Hash Arrays as Dyalog APL Objects

This paper presents a method to store arbitrary key / value parsing a hash array. Other than storing information of arrays in data areas, containing space for each possible element of data, hash arrays only store information of data available. Therefore, memory usage and access time may be saved in cases where only a few elements of large arrays contain data. The implementation is done as an APL object in Dyalog APL. The APL object covers the main functionality as the implementation in perl.

User defined hash functions may be provided for the internal operation of the APL object to optimize data storage and data retrieval. With respect to storing and retrieving data for sparse arrays powerful member functions are provided with the APL object. Hence, massive data storage and retrieval is possible for processing the objects data including sub areas like planes from a cube.

The talk covers the description of the concepts and the implementation.

Benefits and Limitations of Array-style Programs for Parallel Execution

We recently implemented a parallelizing APL compiler on an Intel architecture 4-core desktop computer running Linux. The compiler generates C code mixed with OpenMP directives ready for parallel execution. We measured the runtime and speedups on several programs written in classical APL-style, i.e. array-oriented, and observed quite good speedups. From this observation the benefit of array-oriented programs for parallel execution is apparent as the parallel speedup comes at no extra cost to an APL user, i.e. there is no variable declarations, no data-layout declarations, and no need to modify program for parallelism. However, there are limitations to this approach to automatic parallelism as array-style programs can easily incur a large amount of unnecessary computations and data movement; moreover, there are many

computation intensive applications which are either of irregular structure or iterative in nature which will require new compiler work and data structures.

Web-enable Your APL-based Application System

Why Web-enable?

- Benefits to the Customer
- Benefits to the Developer/Maintainer/Vendor

Overview of web-enabling an APL+Win Application

- Re-purpose the application system into a multi-tiered architecture
- Use GUI construction technology suitable for web-clients
- Web server(s) to house the APL+Win-based application system business rules and calculations
- Application monetization options

Samples available on the APL2000 Forum

Consider the .Net alternative

Visual APL integrated with Visual Studio: Fully-mannaged .Net Solutions

VisualAPL is a .Net programming language which emits IL (intermediate language) that complies with the Microsoft .Net CLR (common language runtime) producing fully-managed .Net assemblies.

VisualAPL incorporates several unique features including APLNext LAE – Lightweight Array-processing Engine, Cielo Explorer – Interactive session for C# and VisualAPL within Visual Studio, APLNext Scripting Engine – Create, Edit, Save and Compile APL source code scripts, APLNext Data Serialization – Create, Edit, Save and Use XML-format representations of APL arrays.

VisualAPL is integrated with Microsoft Visual Studio. VisualAPL supports both dynamic and strong data typing.

VisualAPL fully inter-operates with any other .Net language including object orientation, exception handling, debugging, index origin, object localization, non-proprietary Unicode-format source code, separation of source code, data and state, etc.

APL+Win Has Some Interesting Interfaces

APL+Win has been, and will continue to be, enhanced numerous times to interface with popular 'external' objects. These APL+Win interfaces include COM/ActiveX support, Microsoft .Net assemblies, Microsoft .Net

ADO databases, email, XML-format text, zip-format archives, native files, APL component files, the Windows command prompt, Unicode-based information, source code repositories, pre-.Net dll's and assembly language.

Parallel Programming Theory and Examples Towards Sketching a Taxonomy for Problem Estimation

A consideration of some general ideas about parallelizing applications, focused by means of some specific examples making use of multi-core chips, leading toward a sketch of a taxonomy of parallel problems. This taxonomy aims to classify parallel solutions along a few important dimensions to help size them and guide their formulation appropriately.

We will briefly look at the history of ideas in which parallelism has been framed and some of the early attempts to implement parallel processing. This will lead us to consider some of the dominant trends in parallelism today. We will focus on one of these trends - multi-core chips - by showing some specific examples of how one might take advantage of this to achieve significant performance improvements with readily available hardware. We will also note some of the general drawbacks of parallel implementations.

Finally, we will summarize major considerations for parallel processing problem decomposition to begin to establish a taxonomy general enough to apply to a wide range of problems yet sufficiently specific to usefully guide an approach to a specific parallelization task.

Hashing for Tolerant Index-Of

We consider the problem of $x \approx y$ where x and y are real (64-bit floats) or complex (64-bit floats of the real and imaginary parts), with tolerant comparison (nonzero ϵ). We show how to use hashing to solve the problem.

Hashing algorithms for real and complex arguments have been implemented in Dyalog v13.0. APL models and test cases are provided. Benchmarks demonstrate the improvement over the previous implementation.

XCUBE: A Personal Network (PN)

Development Environment on the Workstation and Runtime Environment on the HOST? It works!

A short overview will be given over the hardware/software configuration at Allianz including DB2 program database.

The APL application development process and environment on both platforms will be explained, in particular the Allianz production release authorization.

The whole process will be live demonstrated on the Allianz mainframe envi-

Topic: *Challenging (production type) applications solved with array processing languages*

Building reference systems in German health insurance

The presentation starts with a brief overview of the complicated and very tightly regulated German health insurance system. The business model of the latter is based on (huge) benefit reserves, which are the single most important item in the balance sheet. Similarly the annual compulsory control of the premiums, is the most important process. Simple but effective reference system for checking the regularly resulting premium recalculation as well as the benefit reserves and exporting the results to Excel are presented. Some technical details on managing the necessary data in big arrays are given. The comparison concepts and the use of the divers results are explained.

Improving Violinists' Intonation

Pure intonation calls for a much larger gamut than that provided by a standard piano. In this paper the implications for string players are studied.

Singers are free to intonate a melody according to their musical ear and personal taste. Pianists depend on fixed preparations of their instrument including its musical temperament. A violinist finds himself in an intermediate position. After tuning his four strings he can intonate his music as he likes. Harmonic tone systems and violin strings are analyzed to help violinists to a better intonation. In a first step, the mathematical properties of known harmonic tone systems are revisited using the Euler space of intervals. The interval spectrum for a suitable subspace is derived and visualized in elaborate diagrams. Audio samples of intervals and scales will also be presented. In a second step the physical properties of a stretched and oscillating string are summarized. By combining interval spectra with string properties virtual fingerboard frets can be calculated. Results are visualized using APL2 and AP207.

CPAM – Array Structured Product Data at Volvo Cars

The Cars from Volvo are extremely diversified products which changes frequently. The paper will show how we managed to describe this complexity in compact array form in Dyalog APL. This allows us to supply users with flexible information. The array data is published in a Master Server which gives instant product information through (web) services to 20 other systems. In addition to this 800 personal users can look at and analyze product data in many ways. The information is used for order configuration, pricing, production planning, order validation, technical information, presentations and creation of Bill Of Material.

Automatic determination of weight for railway waggons

Dynamic Wheel Force Measurement

For five years we have been developing an APL-program to obtain the weight of waggons from a moving train. This method is called dynamic wheel force measurement, DWFM. DWFM is a quick and effective way to dynamically measure the weight of moving trains. It makes time-consuming and expensive static weighing of fully detached waggons on costly static balance scales obsolete.

DWFM is a fully integrated system consisting of strain gauges, a signal converter box and a laptop. It can easily be moved and applied to any railway track and therefore is highly flexible in use.

The strain gauges are installed at specific positions on the side of the rail track. The strain gauges are supplied with voltage from a power source. The elastic deformation of the track steel when a train wheel is passing is indicated by the change in electrical resistance of the strain gauge circuit.

The measurement is amplified and transmitted via TCPIP interface to a computer harddisc txt-file. In correlation to the amount of waggons/locomotives, the size of this file can vary from 1 to 10 MBytes. An APL2-software reads the data sequentially via the associated processor 12 (files as arrays) from the external txt-file from both channels which are caused by the left and right track. After controlling the data the calculation of statistical parameters (min, max, mean, median etc.) is carried out.

Automatic detection of waggons

The physical equation $d = v \cdot t$ relates to distance, velocity and time. As the speed of the train is $v = constant$, the relation of d and t is directly proportional, $d \sim t$. Therefore, the distance between two axis can be determined with the time. With the help of expert knowledge, the waggon can be distinguished by type (2-axis, 3-axis, 4-axis, 6-axis etc).

Results are written in a database

The processed data is prepared and the results are stored in a database. The relevant output is shown directly via dot.com in an excel-sheet.

Java as an interface for APL

The control of the software is done by a user friendly Java-program.

Applying APL inner products to Sudoku paths

Paths are an important tool for solving sudokus by "constraint propagation", i.e. without trial and error. In "APL2 IN DEPTH" (Thomson/Polivka) the inner product \cdot and is used to extending connectivity matrices up to closure. If in a boolean matrix element (row,column) is set to 1, this means that point "row" is connected with point "column". In this way, sudoku paths of type X (only connecting cells containing one specific candidate), can be found. For paths of type Y (connecting pair cells, also called golden chains or generalized xy-wings), to know that some cell is connected with some other cell is not enough: it is necessary to concretely know all the paths. These can be found

by an inner product “`.,join`”, where “`join`” is an appropriately defined APL function.

Topic: *Theory and practice in array processing language design and implementation*

0: Array Processing Language

I will release my 0 (a.k.a. NIL) APL (Array Parallel Language) that will automatically use multiple cores, processors, computers over the network.

1. 0 will be released in ALPHA form in this quarter.
2. 0 has been over 20 years in development.
3. 0 will be open source with a GNU license so I hope it will benefit the world.

We do plan to have our product for sale at APL 2010 but it may be only beta. I will send you a full product description, submit a paper, and reserve a booth but we won't be ready until June according to our plan.

Damage Resistant Component Files Using Journaling and Other Techniques

Dyalog component files are maintained by the interpreter using indices and other high-level structures within an underlying “native” file, which is often shared by many users on multi-user operating systems and networks. Each component file update (e.g. `⊞FAPPEND` or `⊞FREPLACE`) requires a number of updates to different parts of this underlying file; once started, and until the process is complete, there are windows during which the file may be in an inconsistent state. In some environments, sufficiently abrupt termination of the application making the update, or unrecoverable network failures, can cause “file damage”.

Dyalog has addressed this problem in two stages, using file journaling, checksums and other techniques - resulting in component files which are protected not only from damage caused by interrupting an application during an update, but also those which are caused by loss of file caches following a catastrophic event such as a power failure or operating system crash during an update.

Supporting APL keyboards on Linux

All of the GUI interfaces for Linux are built on X-windows. X-windows is a client-server design. The screen-keyboard-mouse side is the server. Programs running on any machine use the services provided to receive input and draw output.

There is an extension to X-windows servers called `xkb` which provides additional keyboard support.

This paper describes how Dyalog have used this feature to implement an APL keyboard that overlays the original keyboard. Thanks to the widespread adoption of Unicode for encoding text – including APL symbols - the APL character set can now be entered into almost all applications, whether the underlying keyboard is US, Russian – or any other language.

APL# - An APL for Microsoft.Net, Mono, SilverLight and MoonLight

Microsoft.Net is a software platform which is designed to be “language agnostic”: it contains a “Common Language Runtime” component which provides an “application virtual machine” with memory management, exception handling and other services which significantly simplify the task of implementing programming languages. The shared type system allows data to be easily exchanged between different languages, and as a result, the application stack often consists of functions implemented using a variety of programming languages. The tight inter-operation with other languages more or less forces us to abandon some of the most central dogma of “classic” APL interpreters:

- The notion that there are only two data types: numbers and characters,
- That arguments are always passed “by value”, and
- That user-defined names are global by default, and local variables are visible to all sub-functions

The paper discusses the design of APL# (pronounced APL Sharp), a new dialect of APL designed with object oriented / language-agnostic platforms in mind, using Microsoft.NET as the initial target platform. Since complete “upwards compatibility” is not achievable, the project presents an opportunity to “tidy up” a little. The goal has been to produce a language which is as powerful a “Tool of Thought” as classic APL and APL2, at the same time as it should feel significantly more acceptable to a “software engineer”.

APLNext Supervisor

The APLNextSupervisor is a productivity platform for executing APL+Win functions in a multi-threaded manner. When the computer hardware on which APL+Win has been installed has multiple processors or multiple ‘cores’, installing the APLNextSupervisor and configuring the APL+Win application system to use it means that programmer-designated operations associated with APL+Win functions can be executed concurrently.

Concurrent execution of an APL+Win function means that the processing performed by that function occurs asynchronously with respect to other processes of the APL+Win application system. Thus the ‘main thread’ of an APL+Win application system is not ‘blocked’ while an APL+Win function of that application system is running concurrently because it is running in a separate ‘thread’.

Application systems which can identify sections of the processing algorithm that can be asynchronously executed can usually benefit from multi-threading.

Typically such application systems perform repeated, in-memory processing of a collection of similar data elements, such as those associated with time series, populations, stochastic processes, simulations, etc.

APL+Win Performance Enhancements

Discuss enhancements that allow APL+Win to run many kinds of calculation intensive applications in less than $\frac{1}{2}$ the time they took with previous versions of APL+Win. Will also discuss WS size increases of 1-2 GB providing about 2.7 GB on Win32 system (when 4-Gigabyte-Tuning is enabled) and about 3.7 GB on Win64 operating systems. APL+Win achieves this size increase while remaining a 32-bit application by splitting the WS into segments. This allows it to increase size while continuing to run with existing 32-bit components that would have to be replaced in a Win64 version of the system.

APL+Win Language Enhancements

Discuss enhancements to control structures, error handling and debugging in APL+Win including:

1. Inline Control Sequences such as (Cond1 :AND Cond2...), (Cond1 :OR Cond2...) and (Cond :THEN True :ELSE False) that are progressively evaluated similar to C-style (Cond1&&Cond2...), (Cond1||Cond2...) and (Cond?True:False);
2. addition of :FINALLY, :CATCH pattern-matching, :TRYALL, .THROW, etc.;
3. debugging, validation, and testing tools :DEBUG, :ASSERT, :TEST, usable version of .WATCHPOINTS, etc.
4. and diagnostic tools for dealing with errors and crashes in development contexts and deployed applications.

Mask and Mesh Revisited

Succinct - A new APL dialect

Succinct is a new APL-like general purpose programming language, which features advanced programming concepts such as closures, generators, lightweight threads, co-routines as well as fundamental integration of regular expressions for string handling and distributed/network programming. Succinct encourages functional and object oriented design with hierarchical name spaces, objects, inheritance and delegates. Functional composition is augmented by allowing tacit programming. Traditional control structures of C/Java are present so that sequential algorithms can be easily transcribed and as necessary improved into array expressions. Succinct can be used in a graphical workbench or as Unix-type scripts. Succinct uses only ASCII and is thus similar to J or K. It is suited for cross platform applications and runs on

Windows, Linux and Mac OS X. The small size of the Succinct interpreter and its reasonably fast performance make it a competitive candidate for applications where Perl/Python/Ruby are used. The tight integration of string processing and regular expressions on the language level allow for tasks of system administration and server-side processing in addition to the standard mathematical prototyping capabilities.

Short Communications

Topic: *Array processing languages in computer sciences*

APL Wiki

The software the APL wiki is based on (MoinMoin) comes with an impressive number of features, but many of these features do not exactly advertise themselves. This talk provides background information regarding the APL wiki as well as tips and tricks how to get the best out of the wiki. Note that it is mostly about using the APL wiki; only a small part of the information provided addresses the needs of contributors.

Topic: *Challenging (production type) applications solved with array processing languages*

How to use an APL+Win application in a .NET environment

We want to introduce a way to use an APL+Win application in a .NET environment. .NET is so-called middleware. Middleware is a concept in modern software architecture and of growing importance. It offers better possibilities of interoperability of single applications, improves ways of scaling systems and last but not least lowers time and costs of development. Two important examples of middleware are Java EE and .NET.

Microsoft's .NET and the IDE Visual Studio 2008 support COM/ActiveX applications. APL+Win is not a .NET language, but the ActiveX Server for APL+Win is a feasible way to use an APL+Win application in a .NET environment. To exchange data between an APL+Win application and the .NET environment XML, a simple standard with wide adoption, is a good choice.

Topic: *Theory and practice in array processing language design and implementation*

APL Prototype Functions

The concept of empty arrays while not unique to APL, has been most thoroughly developed there. Nonetheless, the rules for handling empty arrays are inconsistent across the major APL implementations where, for example,

the sum of two empty arrays can not only give different results, but also, sometimes (and correctly so) a LENGTH, RANK or DOMAIN error.

This paper attempts to give a comprehensive treatment of prototypes and set down consistent rules for computations with them. Along the way, we introduce the concept of a prototype function – a function associated with each primitive and derived function when working on empty arguments.

Tutorials

Dan Baronet (Dyalog Ltd.): User Commands in Dyalog APL

User Commands (UCMDs) are similar to system commands, except that they are written in APL. They provide a way to make tools and utilities available at all times – without requiring them to be copied into the active workspace before use. The user command processor also provides a mechanism for on line help, and encourages developers of UCMDs to provide consistent behavior across different commands.

Dyalog introduced User Commands with version 12.1. In the Dyalog implementation, the source for a UCMD is a single Unicode text file, which means that UCMDs can very easily be shared. As soon as the text file defining a UCMD is copied into the UCMD folder, the user command can be called from the APL session – or invoked under program control. It is Dyalog's hope that the introduction of user commands will lead to much more widespread sharing of development tools in the APL community.

This tutorial will show how to create, debug and modify user commands and how to manage them. It will show how to deal with arguments, options and results. Several examples will be shown, from the very simple to the more complex.

Workshops

Brian Becker (Blue Dolphin Solutions): APL and Web Services

APL programmers have long built highly functional applications and utilities to perform various types of analysis, query databases, and a myriad of other tasks. Sharing the results of these efforts with others, particularly those outside of APL realm, has often been cumbersome and sometimes problematic. Conversely, incorporating the results of functionality developed outside of APL has proven to be similarly challenging. Enter the Stand Alone Web Service (SAWS) framework.

SAWS has been developed for Dyalog APL version 12.1 and addresses needs of both Web Service Requestors (Clients) and Web Service Providers (Services). For the Requestor, SAWS makes it easy to retrieve the results of Web Services developed by others and use them in a natural "APL-like" manner.

For the Provider SAWS enables an APL programmer to easily make the functionality they have developed available via Web Services without having to become an expert in all of the standards and protocols necessary to develop and deploy Web Services.

This workshop will give attendees hands-on experience building and using Web Services from Dyalog APL. A non-commercial license version of Dyalog APL 12.1 will be supplied for those who need it.

Richard Smith (Dyalog Ltd.): Using Regular Expressions from APL

A new system function `⎕RX` adds support for Regular Expressions to version 13.0 of Dyalog APL. The workshop will provide an introduction to regular expressions and how they can be used from APL.

John Daintree (Dyalog Ltd.): Using the Microsoft.Net Framework

Dyalog integrates comfortably with the Microsoft.Net Framework. This course will give an overview of, and show how you can take advantage of, the features included in the Framework itself, and in Visual Studio, Microsoft's cross-language development platform. John will show you how to find and understand documentation of the framework class libraries, and he will introduce you to some of the most useful classes. We will explore how to use the VS Form Designer to build forms which use APL code, and write APL classes which can be used from C# and VB.Net. The course will very briefly show how to call APL code from Microsoft Internet Information Services (IIS).

Morton Kromberg (Dyalog Ltd.): Windows Presentation Foundation

Windows Presentation Foundation is a graphical subsystem for rendering user interfaces, originally developed for desktop applications built using the Microsoft.Net platform – but now also available for web and mobile applications under Windows, Linux, the Apple Macintosh – and mobile operating systems. The workshop will demonstrate how WPF can be used from Dyalog APL and APL# to create desktop and web applications.

John Scholes (Dyalog Ltd.): Introduction to D-Functions

Since their introduction in 1996, Dyalog’s direct-definition functions (D-fns) have grown from an experimental toy to a notation used to implement large pieces of commercial software. D-fns are not only useful for expressing idioms, but also as a tool of thought for any problem that can benefit from a functional approach (and some would say that covers almost everything). Owing to their functional nature, D-fns also have greater potential for internal optimization, including compilation – and have been selected as the foundation for the new function syntax in the APL# dialect.

The workshop will start with an easy introduction; discuss where the use of D-fns is appropriate; and finish with some fireworks.

Alan Graham (Black Net Box): 0 in 1 hour

Open Forum

Dick Bowman, Chris Hogan, John Jacob, Phil Last: APL in 2020

We are currently promoting discussions at comp.lang.apl on-line usenet group under the banner “APL in 2020” on the future directions of APL as seen by current users.

These discussions are being collated on the web-site aplin2020.org and will be presented to the conference to introduce a number of open forums where a panel of experienced APLers will lead and encourage the conference participants to develop and synthesize some consensus on what form (or forms) APL should take looking forward to 2020.

After the conference we will collate all the material gathered and publish it along with a summary at aplin2020.org - to leave a time capsule of the interests and views of the APL development community as it stood in 2010.

Selling Tutorials

Gitte Christensen (Dyalog Ltd.): APL - why, when and where

In this presentation we look at the current trends in software development and how the rest of the software world is doing. Some areas where APL is particularly competitive are identified and a list of Selling Points for APL is presented.

Paul Grosvenor: Making Money with APL

To make APL or indeed any other technology pay we must be very clear and positive about what it is we are selling, where we are selling it and why our client needs it. If that wasn't enough, and in the current climate, our customers are very sensitive to both cost and risk and these two hurdles must be addressed and overcome quickly and decisively.

In this part of the presentation we will explore a few simple techniques to bring your proposal to the top of the list. It all comes down to confidence, preparation and rapport with a little bit of persistence and luck.

With some audience participation I aim to get you thinking 'out of the box' and demonstrate how you can make your proposal stand out from the crowd, to be so intriguing in fact, that your prospective client will be calling you.

I'll even show you how to tell someone your name in a way that stands a better chance of them remembering it.