A Personal History of APL

Dr. James A. Brown

On the occasion of the 50th anniversary of the APL workspace 1 CLEANSPACE
German Guide Share Europe Working Group, APL Germany and IBM Germany
November 27-29, 2016 IBM Böblingen, Germany

Abstract

If you're looking for a technical paper, this isn't it. In this paper I document my early involvement with APL and what I learned about the early history. I also discuss various events through the years that involve people and the lighter side of technical and customer meetings.

The Early Years

Personal History

This paper contains my personal recollections which are accurate to the best of my ability but could be colored by 50 years of time.

How I came to APL

I first heard of Ken Iverson, the inventor of the original APL notation, shortly after I started work in 1965 at IBM Federal Systems in Owego, New York. (My work history is summarized in Appendix 2.) A colleague (Charlie Stieglitz) was talking in the hall about this person who "reduced the design of the System 360 to a single symbol". I thought this was a bit of an outrageous claim (and it probably was) but I decided to check up on it. I had been interested in symbolic notations for a while and when in college read much of Alfred North Whitehead and Bertrand Russell's "*Principia Mathematica"* [WI1] working out all the proofs. I went out and got Ken's book "*Automatic Data Processing"* [IV1] co-authored with Fred Brooks. Then I got "*A formal description of SYSTEM/360"* [IV2] by Falkoff, Iverson, and Sussenguth – an amazing piece of work which later became known as the "grey manual". I was told that when engineers were trying to fix a problem with the I/O channel, they referred to the formal description not the machine documentation. I thoroughly enjoyed time spent on this document and came to appreciate that symbolic notation really can be used for describing real as well as formal systems.

Being a newcomer in the computer world (note that the term Computer Science did not exist at this time), a friend and I decided to join the ACM (Association for Computing Machinery). One of the meetings was a dinner meeting featuring some unknown speaker and we decided to take our wives to the meeting. I'd always heard that the way to a woman's heart is through technical computer meetings. (I have no corroborating evidence that this is true.) The speaker was a man named Adin Falkoff who had a portable (by some extended meaning of that word – it came in two suitcases) 2741 terminal which he connected to a phone line and dialed into a computer at IBM Research Yorktown Heights and showed us APL.

He carefully and patiently explained to the telephone operator that the phone line would be used to communicate with a computer and the noises on the line were normal. Nevertheless, the session was interrupted several times when the connection was terminated by the operator.

I was completely mesmerized by the presentation. Adin typed in "2 space 3 backspace backspace plus" and pressed return and it typed "5". He called it visual fidelity. I was so impressed. I recall our wives were not as impressed by this as they were apparently able to work out this result in their heads without use of a terminal, phone line and a computer (something that would never occur to a man). A few years ago, I showed this "visual fidelity" concept again at a German APL meeting using the newly open source code of "APL\360" where I typed in "F" "backspace" "L" and displayed the value of the variable "E". [SU1] Adin showed one wonderful thing after another and I left this meeting very excited by this brand new technology.

Imagine, a programming language without any declarations! No Declare statement – No DIMENSION statement. The number of items of data determines the size of the data structure that holds them. Unheard of. If you have zero items of data, you have an empty array. Can you imagine a DIMENSION statement for zero items? See Appendix 4 for a discussion of empty arrays.

IBM Owego had a 2741 terminal with a phone line where I could attach to the APL system at IBM Research in Yorktown Heights. Adin told us how to contact the APL operator using the system command "*)MSG*". The operator of the APL machine at Yorktown was Gerry Barrett and she added an account for me using my IBM employee number and I was off doing my first APL session. (Note that Gerry Barrett ran the APL service at IBM Yorktown and, to my knowledge was never applied to function operands to return a derived function – that's a different kind of operator. Also note that in the fall of 1986, IBM held an "Internal Technical Liaison" ITL meeting for the 20th anniversary of APL and Gerry was presented with an award.) After I had been playing with APL for a while, Gerry sent me another message asking for an account number to which my usage could be charged. "Oh Oh" I thought and quickly signed "*)OFF*". I asked my manager for an account number and he was completely uninterested in letting me "play on company time."

That might have been the end of my time with APL but one of the benefits that IBM offered as part of employment was the ability to take graduate courses at Syracuse University for free. They would fly professors from Syracuse to Endicott, New York and we could take classes one evening a week. One of the teachers, Bill Jones, mentioned how Syracuse University was going to get an APL installation. (I'm pretty sure the only other APL installation outside of Yorktown at this time was at a college in Canada.)

I decided I would go back to school at Syracuse and work on APL. IBM provided grants for employees to get advanced degrees so I applied for a grant. I was denied so I took a leave of absence from my job at IBM Owego to go on campus full time and run their APL service for a salary of $500 per month. (Students give Universities a really cheap work force.) It is, perhaps, interesting that my reading of "*Principia Mathematica"* [WI1] was accepted as fulfilling the mathematical logic requirement for my doctorate. Syracuse also had a foreign language requirement for a PhD and I always thought I would use German but it turns out that, at that time, FORTRAN was one of the accepted languages. I guess it's foreign enough. (By the way, more FORTRAN programmers have died that APL programmers. That has to tell you something!)

I was at Syracuse when Al Rose visited with his famous "portable" 2741 (120 pounds - maybe the same one Adin used the first time I saw him). Sometimes when Al wanted to give a demo of APL, he would be unable to get a phone connection to Yorktown Heights. He solved this problem by noting that the electric typewriter used an acoustic coupler to communicate with the computer. He hooked up a cassette tape recorder to the coupler and recorded the tones from the coupler while he was giving a live demo. Now when he couldn't get a phone line, he played the tape into the coupler and it reproduced the demo session he had recorded in real time – errors and everything. This is memorialized in the song "APL Blossom Time" written by Mike Montalbano [MO3].

Fortunately (in some perverse meaning of that word), Karen and I ran out of money after six months at Syracuse and, in the spring of 1969, I had to get a second job. I applied to IBM Yorktown Research for a summer job. (Note that this application was done by writing words on paper, putting the paper in an envelope with a stamp and sending it though the Post Office. This was called a letter. There's a whole generation of people now who could not imagine such an archaic way to communicate.) Despite this cumbersome way to apply for a job, my application was quickly rejected. Again, fortunately, I had also sent a letter directly to Adin Falkoff of the APL group and they invited me for an interview.

Karen and I showed up at Yorktown Heights on the appointed day and time only to find that the interview had been canceled. Because of the death of President Eisenhower, the lab had been closed the previous day and a scheduled "APL Machine" seminar had been moved one day later. They had tried to reach me to cancel the interview but did not succeed. (Why didn't they send an Email or call me on my cell phone?) But I did see Ken in the flesh for the first time standing outside the auditorium.

I wish I could have attended the APL Machine Seminar. I'm told that central to the discussion was Phil Abram's PhD dissertation *"An APL Machine"* [AB1]. His concepts of drag along and beating made it into the implementation of APL2. It's always felt like a hardware implementation of APL was a natural idea. Even today, NestedComputing and the "Array Operating System" first suggested by Mircea Morosan are being pursued and might influence hardware.

Karen and I came back to the Yorktown lab the next day for the interview and I met Ken Iverson, Adin Falkoff, Larry Breed, Dick Lathwell, and the team for the first time. I got the job and spent the summer at the lab with the APL team. I know now that Ken and Adin sort of took turns managing the group but the focus was always on the work not the management. I've told this story many times before but I had assumed that Adin was my manager. It turns out that Larry Breed was actually my manager but I didn't find out until the exit interview at the end of the summer. (As a strange twist, many years later Larry worked for me at IBM in Palo Alto. I hope he realized it.) Because Larry made an administrative error, my summer job was not terminated and each summer I would go back to Yorktown for a few months and I continued for three years working remotely from the local IBM office in Syracuse during the school year. In those years, among other things, I implemented the extension of "base value" and "representation" (now called "decode" and "encode") to higher rank array arguments and "catenate" on higher rank arrays (it only worked on vectors before).

There was a problem with the definition of "base value" and "representation". Because "representation" only worked on scalars, the implementation treated a one item vector right argument as a scalar:

```
    2 2 2⍴5
1 0 1
    2 2 2⍴,5
1 0 1
```

When extending to higher rank arrays, with the desired shape equation, the result of "representation" of a one element vector would change to be a one-column matrix:

```
    2 2 2⍴5
1 0 1
    2 2 2⍴,5
1
0
1
```

We were running a time sharing service and having a primitive function suddenly get a different answer was a potential problem. To measure the impact of the change, we put counters in the code to count the number of times "representation" was applied to a one-item vector and the number of times it was applied to a scalar. At the end of the day, the counter showed there were many times more of application of "representation" to one-item vectors than to scalars. Way more than expected and a big problem. We speculated that this could be because the shape of a vector was itself a one-item vector. Maybe one-item vectors were more common than we thought. Then, I discovered that another summer employee, Seth Breidbart, overheard us talking about the counters and put two terminals into infinite loops taking the "representation" of one-item vectors. This was going to be my first assignment and I really wanted to do it so I fired up three terminals taking representation of scalars and after a while, my counter surpassed his counter by enough that I was given the go-ahead to do the implementation.

Not many people had the ability to work on computers remotely in those days. I was using a time sharing system called "Time Sharing System" (TSS) and it turns out that for months, my usage was not billed to the APL group. There was a bug in "freemain" the API that released allocated storage. Each time I logged off, TSS crashed as (I assume) my workspace was released. I was logging off so I didn't notice anything unusual. Because session accounting was done on log off, the APL group were never billed for my usage. Eventually the TSS system programmers noticed that in every core dump, my ID was on the system.

In the original APL, an array was either all numeric or all character. I met Trenchard More during a summer at Yorktown and heard discussions of extending APL. I left IBM that summer with the idea that there was no need for arrays to be all numeric or all character. Since I decided to work on extended APL for my PhD Dissertation, I was (quite properly) not permitted access to the proposals that were active at Yorktown in particular the work of Trenchard. My dissertation *"A Generalization of APL"* [BR1] was published in 1971 and contained arrays that could contain some numbers some characters and recursively some arrays. I graduated from Syracuse in the first computer science class with a PhD in Computer Science and moved to IBM at the Philadelphia Scientific Center where the first lines of code for APL2 were written.

Stories from the early years

Much of what I'll say in these sections is not firsthand knowledge but rather things told to me or overheard by me. Someone with more personal knowledge should corroborate.

For some time in the early years, the term "Iverson Notation" was used to describe the symbolic notation invented by Ken Iverson. At some point, the team decided that they needed to give a formal name to the notation. I was told that many names were proposed and most people hated most of them. I wish someone had recorded the names that were suggested but only the final name survives. I was told that it was Adin who suggested taking the initials of Ken's book *"A Programming Language"* [IV3] and APL was born. To this day we still need to point out that the word "Programming" in the title of Ken's book did not refer to "Computer Programming" but to "Mathematical Programming" and "Linear Programming". I wonder how the computing world would be different if they had chosen the name "Java".

The first implementation of Iverson notation was done by Herb Hellerman in a system called "PAT" (Personalized Array Translator System) [HE1] – an interactive (but not time-shared) system on an IBM 1620 not using Greek characters. (I have met Herb Hellerman and I can testify that this picture does not do him justice but it's the best I could find.)

Ken used Hellerman's implementation with students in the local secondary school. This led to his book on elementary functions [IV4].

Not long after this, the IBM Selectric typewriter became available and a type element with the symbols used in Iverson Notation was created.

Creation of the type ball was complicated because opposite sides of the ball had to have characters of the same density so that balance was preserved. I am told that Herb Hellerman called these "Iverson Balls" and this term was used until Ken himself requested that they be called "Iverson Type Elements".

When I first arrived at Syracuse University to start graduate school in earnest, there was not yet a local installation of APL. They had a 2741 and a phone with an acoustic coupler just like they did at IBM Owego. There was a problem with using this terminal because the type ball was often missing. You removed the type ball just by lifting the clip on the top. It became popular for the young female students to steal the type ball and wear it on a necklace around their neck as a love bead. (Remember this was the 1960's.). The solution was to break off the clip on the type ball making it difficult to remove. Garth Foster (who became my Thesis advisor) was the holder of the type ball and you had to reserve time on the terminal.

In 1965, Larry Breed and Phil Abrams built a version of APL in FORTRAN on an IBM 7093 at Stanford University. This was around the time that the "Formal Description of System 360" became available. I had read this paper and I didn't find any errors. But Larry told me that he did find an error in the formal description and when Ken visited Stanford (maybe to see their APL implementation – this I don't know for sure) he and Phil had lunch with Ken. Larry said he waited until Ken had cake in his mouth and then

told him about the error. It was probably this meeting that led Larry to come to IBM Yorktown to work on APL.

The APL implementation that I used from IBM Owego and Syracuse was hosted at IBM Yorktown on an IBM 360 model 50. This machine had a total of 256K of memory. It ran on the DOS operating system (not the DOS from PC days) and when APL was loaded it replaced part of the operating system because there was not room for both APL and the DOS operating system. Users were given 30K workspaces which were swapped from disk to memory and back as users computed. APL is such a compact notation that you can do significant work in a 30K workspace. No one had ever seen anything like it.

When you wanted to attach to the APL service, you called a toll free number and, when you heard a whistling sound, you put the phone into an Acoustic Coupler.

The back of the computer room in Yorktown was a wall of telephones to receive these calls. When the APL service was unavailable, dozens of phones would be incessantly ringing. I was in the machine room with Graham Driscoll (a member of the APL group) during one of these down times. He would go back to the phones and pick them up and make a whistling sound with his mouth and I suspect most people put the phone on their side into the coupler thinking the computer had answered and tried to sign on.

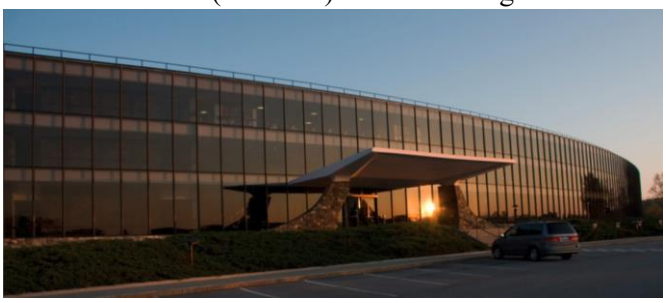
The APL service at Yorktown ran 24 hours a day – something that was not only not normal at that time but at Yorktown not even allowed. At night, the computer ran unattended and that was forbidden. The APL group did it anyway and was continually bothered by management because of it.

At some point, management decided that the machine should be shared with another group in the lab. It was, after all, a valuable resource. I'm told that Adin told Eugene McDonnell to go to the machine room and guard the door. When people from the other group showed up to use their machine time, Eugene physically prevented them from entering the room. This led to a big problem and eventually, the APL team lost the fight for the machine. I'm told that because of this, Ken Iverson resigned from IBM. The team had a going away party for Ken and at the party, someone from IBM management talked to Ken and offered to let him keep the machine. (This might be when Ken was made an IBM Fellow but I am not certain about that.) The APL group had its own machine for the rest of its time at Yorktown and when the group moved to form the Philadelphia Scientific Center, they again had their own computer.

IBM Thomas J. Watson Research Center Yorktown Heights

The building is in the shape of an arc of a circle on a rising hill and has dark glass on the front. Office corridors run from the front to the back so no office has a window - a big advantage if you want to grow mushrooms. There are three floors but the city of Yorktown only allows buildings with two floors. Because of this, the official main entrance is in the back of the building higher on the hill so from the back it is a two floor building. The bottom floor was floor "0". In the USA, the ground floor of a building is normally floor 1. Even at Yorktown, zero-origin indexing was preferred. It makes sense to me that

floors are numbered by their distance from the ground not that distance plus one. Here's a picture of the real main entrance to the lab in the "back" (? "front") of the building:



It was a joy to work at Yorktown heights. The work was important and little else was. At IBM Research there was only one job for technical people "Staff member". Everyone was treated equally. That's probably why I didn't know who my manager was. Every year all the staff members were ranked from first to last and if you ranked in the lower third two years in a row, you had to find another job.

The head of research at that time was a Mathematician (name forgotten) and the one "unequal" thing he insisted was that the mathematicians at the lab had larger chalk boards than the other employees. The APL group held that this was needed because of their cumbersome notation. It is often reported, but not true, that the APL group had small thin chalk boards suitable for writing APL one liners.

There's one memo from management that I wish I had saved. The APL group played Frisbee at lunch time and the black 150 gram Frisbee was registered with the IFC. If it was raining, we sometimes played Frisbee in the hallway. This left some black streaks on the wall on those rare occasions we did not execute a perfect toss. Management sent a memo saying "Only white Frisbees indoors". How cool.

Library 1 CLEANSPACE

The first version of APL had no library system. You signed on, typed in your program, played with it and all was lost when you signed off. If, in the middle of a session, you wanted to start over, you would sign off and back on again and again have an empty workspace. Since there were a limited number of phone lines, you could not always get back on immediately. When System commands and a library system were added in 1966, the first workspace saved was library 1 CLEANSPACE with the now famous timestamp of 1966-11-27 17.53.58. This workspace had no functions or variables and so was formatted as a workspace but otherwise complelely empty. Now you could save your work in your own private library. There was no)CLEAR command. If you wanted to start over with an empty workspace, you loaded 1 CLEANSPACE. Much better than signing off and back on.

When)CLEAR, to give you an empty workspace, was added, CLEANSPACE was no longer needed so Adin deleted it by entering ")DROP 1 CLEANSPACE". He had second thoughts and decided CLEANSPACE should be kept for historical reasons and asked Dick Lathwell to add it back. Dick set the clock of the model 50 to the original timestamp with the clock disabled and started APL and entered ")SAVE 1 CLEANSPACE" However, an interval timer interrupt was required to enter the scheduler, so he momentarily enabled the clock. The SAVE happened but the clock advanced one second so the timestamp we see now is one second later than the original. [JS1]

When APL does operations on numbers (like comparisons or conversions) it uses fuzzy operations so, for example, the number 1.99999999999999999 is treated as 2 if it is used as an index. The implementation had a constant called Comparison Tolerance (CT) stored in the bottom of the workspace and set to 1E-13 that was used in these adjustments. There was no way for a user to adjust this constant. However, for some scientific calculations, you do not want any adjustment to numbers so a workspace 1 HEISENBERG was saved with Comparison Tolerance set to zero so there would be no uncertainly in computations. This workspace never received the acclaim of CLEANSPACE. In modern APL systems, users can set any reasonable value of Comparison Tolerance. HEISENBERG was no longer needed and was dropped. It was never recreated. Where's Dick Lathwell when you need him?

Vector Notation

I have a printout from an early APL session (from before I joined the group) and it shows expressions like this:

```
      (2,3,4,5)+(20,30,40,50)
(22,33,44,55)
```

Notice how numeric vectors were written like you might see them in mathematical textbooks you would see at that time. Larry Breed was very proud of the code he wrote in the parser which could recognize the parenthesized expression as a numeric constant and store it internally in the APL workspace as a single token with a vector value. APL programs are executed by an interpreter that at execution time must examine every token so Larry's enhancement meant faster execution because it could scan one token instead of many.

I was told that it was a very painful decision to change the syntax of APL so that the parentheses and commas were not needed.

```
      2 3 4 5+20 30 40 50
22 33 44 55
```

After all, unless you know the rules are otherwise, it sure looks like there is the expression "5+20" in the middle. The decision was made that forming of vectors was more important than the application of operations.

In my opinion, this was a revolutionary step in the development of the language. There was already the idea that the less dense characters (those that used the least ink) should be for the most important uses. Using '.' for an operator uses this principle. What's the least dense character? It's the space character. And now it is being used for the most important role - forming vectors. It's interesting that the least dense character is assigned to the largest key on the keyboard!

When I was designing the APL2 language, without going into details, I decided again that forming of vectors was more important than the application of operations so if A, B, C, D, E, F are any arbitrary arrays,

```
      A B C+D E F
```

means

(A+D)(B+E)(C+F)

This decision (prompted by the work of Trenchard More in his Array Theory [MO2]) led to the most difficult arguments between the originators of APL and me. I always looked for a compromise that would make all parties happy but in this case, it was not possible.

When I moved from APL Development at IBM Santa Teresa in California back to IBM Research in Yorktown Heights, the California group was worried that I'd give up on vector notation. Gene McDonnell said to me before I left "We will know you have caved in when Vector Notation disappears". I didn't cave (painful though that was) and when Gene moved to I.P. Sharpe, he changed his mind about vector notation.

One objection to this vector notation expressed during the decision making process is that more expressions become unreadable without redundant parentheses. I believe this argument has some validity but, for me, other arguments carried the day. This is why, once formal requirements are satisfied, language design is an art not a science.

People have said that APL programs are unreadable. Some have even called APL a write only language. Alan Perlis (famous American computer scientist who was on the ALGOL team) once said this about APL and the APL interpreter:

"APL is like the Bible.
It is not meant merely to be read but to be interpreted."

It was a clever play on words.

In the early days, when the universe of APL people was very small, decisions were made by consensus. I remember taking ideas to Adin. He would pick them apart and tell me how terrible they were (this was Adin's style). I'd take the same ideas to Ken. He would pick them apart and tell be how promising they were (this was Ken's style). As days went on and ideas were discussed, slowly Adin's and Ken's opinions would converge and almost every time their merged opinions were clearly correct. (This balance was lost when Ken left the group years later.) As APL became popular and more people became involved it became more difficult, and eventually impossible, to reach consensus on decisions that had to be made.

The differences of opinion over my "Vector notation", operator definitions and nesting were never resolved and led to two styles of array computing from different companies. You're not right because people agree with you and you're not wrong because people disagree with you. You have to make technical decisions without emotion based on formal arguments. You should ask for advice from a lot of people but not so they can tell you the answer. Rather you ask so that you have the alternatives you need to make a decision.

Another case where I made a compromise was in the 1982 APL2 IUP where, in an attempt to gain agreement, I changed the definition of some operators (notably Outer Product) to a more flat orientation. This seemed reasonable because the older behavior I wanted could be achieved my applying "each" to the operand. This was a terrible decision.

In 1982, IBM and STSC announced their enhanced APL products on the same day. Here's an article from ComputerWorld:

July 12, 1982

COMPUTERWORLD

THE NEWSWEEKLY FOR THE COMPUTER COMMUNITY

Weekly Newspaper Second-class postage paid at Framingham, Mass. and additional mailing offices

IBM Follows STSC's Lead, Brings Out APL2

By Lois Paul
CW Staff

WHITE PLAINS, N.Y. — Following hard on the heels of STSC, Inc.'s recent announcement of a nested-arrays extension to its APL*Plus system (CW, June 28), IBM has unveiled

The extensions to the APL operators are said to include user-defined operators, use of defined functions with operators, new operators designed to improve the usability of general arrays and extensions to existing operators to eliminate previ-

Changes to VS APL workspaces and auxiliary processors may be required for operation under APL2, the vendor noted. In addition, the APL-based information center products are not supported for use with APL2.

Crick said that the key to the rele-

STSC's had the definitions of the operators from my PhD thesis and IBM's did not. Thanks to Bob Smith, who visited IBM Santa Teresa sometime after he left STSC, the decision was reversed before the release of the official Program Product two years later. Bob Smith is also responsible for some of the nomenclature. I called the extended arrays "General Arrays". Bob called them "Nested Arrays" and I felt this name was more descriptive and adopted it.

I'm very proud of the final definition of the APL2 language and its correctness is due to a myriad of interested and talented people.

You can find a chronology of APL Systems and influences at http://www.sigapl.org/APLChronology.php.

APL Blossom Time

The song "APL Blossom Time" was written by Mike Montalbano using the pseudonym J.C.L Guest. Mike used a pen name for many of his writings. He wrote several articles for "Datamation", a computer magazine that was published in print form in the United States between 1957 and 1998 and still continues on the web [Wikipedia]. These articles were viewed as criticism of IBM management and I was told by Mike that for many years, IBM was trying to find the person who wrote those articles. He probably didn't need to use his fake name for "APL Blossom Time" but he did.

The lyrics are a very accurate representation of the early years of APL. A discussion of the writing of the song is included in Mike's "A Personal History of APL" [MO1]. I discussed earlier in this paper seeing Al Rose "fed it a tape when he couldn't get a phone line" at Syracuse.

At APL 81 in San Francisco, Larry Breed, John Bunda, Diana Dloughy, Al O'Hara , Rob Skinner and I performed this song at the banquet with 1100 people singing along. Someone unknown to me took this picture of us performing the song:

A YouTube video that contains the lyrics and the audio of this performance is viewable at [MO4]:

https://youtu.be/g4xvjfr297E

The quality of the recording is not bad considering that the recording was made by Bob Armstrong sitting in the audience using a hand held cassette recorder. We had the lyrics on foils (transparencies that could be projected by an overhead projector) and, in the introduction to the song, I say how we had a musical interlude between verses so the foils could be swapped. I still have the original foils.

Sometime after APL 81, I made a recording of the song on a multitrack recorder at my home with vocals by me and Mike Wheatley, guitar by me and John Bunda, Bass by me, and drums by Brian Duff. A YouTube video that contains some historical pictures and this audio can be found at [MO3]:

https://youtu.be/0zSauxkMxPo

As of November 2016 this video already has over 60 views and is on its way to becoming a viral video. There is (at least) one glaring error in the video. See if you can spot it. I would like to claim that it is not an error but rather copyright protection. At one point in APL history, a company stole the source code for VS APL and claimed it as their own. Doug Aiton, a member of the APL team, was charged with showing that the code was stolen. This was done by cataloging the bugs in the product over the years. By identifying which bugs were fixed and which were not, Doug was able to pinpoint the date on which the code was stolen within a couple of weeks. Should someone steal my video of APL Blossom Time, I can prove it by pointing out the errors. However, it's not clear what intellectual property is being protected. Mike Montalbono gave us permission to use his words. The music was written in 1936 by Jimmy Driftwood for a song called "The Battle of New Orleans". I assume the music in the public domain even though there was a popular rendition of the song by Johnny Horton which went to number 1 on the Billboard Hot 100 in 1959 [Wikipedia].

The Future for me

After 31 years, I retired from IBM (as of this writing 20 years ago) in 1996 because of a headcount cut that would have caused others in the team to be let go if I didn't leave. I've continued to work to move the array programming paradigm represented by APL into mainstream computing. Together with James Wheeler, SmartArrays Inc. produced an add-in to C++, C#, and JAVA that gives these languages array computing capabilities similar to and beyond what APL provides. With Mircea Morosan, Morten

Kromberg and Gitte Christensen, NestedComputing has a possibility of moving array computing deeper into operating systems and even into the hardware. Array representation of data and array processing may play a part in the emergence of a new generation of computers that have flat address spaces.

Vignettes and Anecdotes

APL Advertising

IBMs APL2 supported the IBM 3090 Vector Facility – hardware that did array arithmetic. I think this support was wonderful because any APL program ever written could potentially use the vector facility without the need for recompilation. Every other programming language required recompilation and special processing. This made the vector facility attractive to APL customers and because of this I was moved into IBM marketing for a couple of years. The marketing manager of the marketing group was Howard Richmond who knew how to get things done. He is the one who arranged for the free trial version of APL2, called TryAPL2, to be given away as a demo version.

Howard authorized me to do advertising for a series of APL2 seminars he wanted us to hold. He told me to get a radio commercial in the cities where the seminar would be held. We only had the budget to run a single ad in each market. I decided to not buy radio commercials for two reasons. One – when a radio commercial is over, it's over and gone forever. If someone missed it there wasn't going to be another chance to hear it. Two – I felt a radio commercial would be dangerous to the APL community. I could imagine an APLer driving down the highway when the radio starts playing an APL advertisement and being so surprised, amazed and astounded that he drives off a bridge and is killed. I couldn't be responsible for this.

Instead, I took out a newspaper advertisement in each city. A newspaper is less volatile than radio and someone might see the ad a day or two later than it ran. Someone could also refer it to a friend – "Hey, don't you use APL. Have you seen this?"

Here's the ad from the New York Times:

THURSDAY, SEPTEMBER 7, 1989

Business Day

The New York Times

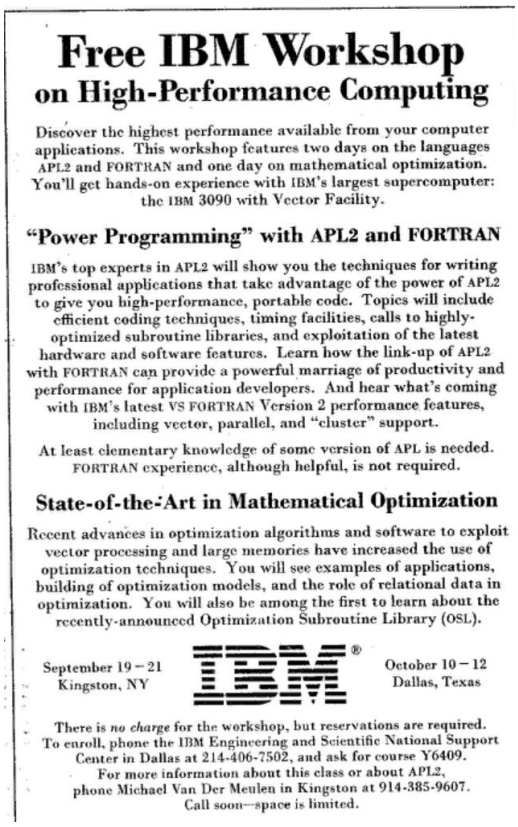
BUSINESS Digest

THURSDAY, SEPTEMBER 7, 1989

A Small Lobby's Large

By ANDREW POLLACK

SAN JOSE, Calif. — The Semiconductor Industry Association has



Free IBM Workshop
on High-Performance Computing

Discover the highest performance available from your computer applications. This workshop features two days on the languages APL2 and FORTRAN and one day on mathematical optimization. You'll get hands-on experience with IBM's largest supercomputer: the IBM 3090 with Vector Facility.

"Power Programming" with APL2 and FORTRAN

IBM's top experts in APL2 will show you the techniques for writing professional applications that take advantage of the power of APL2 to give you high-performance, portable code. Topics will include efficient coding techniques, timing facilities, calls to highly-optimized subroutine libraries, and exploitation of the latest hardware and software features. Learn how the link-up of APL2 with FORTRAN can provide a powerful marriage of productivity and performance for application developers. And hear what's coming with IBM's latest VS FORTRAN Version 2 performance features, including vector, parallel, and "cluster" support.

At least elementary knowledge of some version of APL is needed. FORTRAN experience, although helpful, is not required.

State-of-the-Art in Mathematical Optimization

Recent advances in optimization algorithms and software to exploit vector processing and large memories have increased the use of optimization techniques. You will see examples of applications, building of optimization models, and the role of relational data in optimization. You will also be among the first to learn about the recently-announced Optimization Subroutine Library (OSL).

September 19 – 21
Kingston, NY

IBM®

October 10 – 12
Dallas, Texas

There is *no charge* for the workshop, but reservations are required. To enroll, phone the IBM Engineering and Scientific National Support Center in Dallas at 214-406-7502, and ask for course Y6409. For more information about this class or about APL2, phone Michael Van Der Meulen in Kingston at 914-385-9607. Call soon—space is limited.

The most successful seminar, in terms of attendance, was in Denver, Colorado. We had a newspaper ad in the Rocky Mountain Sentinel. The New York Times ad was in the business section of the paper. The Denver ad was on the obituary page – the page where they list people who had recently died. This might seem like bad placement but apparently people look at the obituary page every day and if they don't see their name, they go to work. (I got this information from the Internet so it must be true.) One interesting thing I noticed on the obituary page (and verified this by looking in other papers) is that apparently people die in alphabetical order. I wouldn't have guessed that.

An Intentional Bug

During the time that Larry Breed worked for me, I noticed that whenever he was writing a defined function just to try something out or show someone something, he used the function name "FOO". This might be a shortened version of the word "foofaraw" meaning "excessive worrying about something unimportant". Since "F00" is a valid hexadecimal number, it is found all over the IBM implementations of APL where uninitialized storage (that would commonly be set to all zeros) was set to "00F00F00". This made it easy to tell if the implementation referenced uninitialized storage.

As a sort of joke, I put code in the APL2 symbol table support that made any name that contained the letters "FOO" illegal. It was great fun watching Larry try to show APL2 to someone. He couldn't get past trying to use the name "FOO" and for a while didn't understand what the problem was.

This would not be notable except that we were near a release of the product and I forgot about the code and APL2 was shipped to customers with FOO disabled. IBM kept track of the number of defects in products so it was very painful to have to issue a fix for this problem – but we did it.

Digital improves everything

My first encounter with a digital computer was in 1958 at Gannon College in Erie, Pennsylvania. The institution had two digital computers. An LGP-30 (with a magnetic drum memory and typewriter and paper tape for I/O) and a newly delivered IBM 1620.


The LGP-30 was so slow that there was a one line display and you could watch the bits march across the accumulator. I used the LGP-30 to print my graduate school applications with an illegal character on the paper tape so it would stop printing for me to manually type in the variable information. The IBM 1620 was often called “CADET”, meaning "Can't Add, Doesn't Even Try", because it used a table in memory for addition instead of a circuit. You could patch this table and do arithmetic in other number bases. The electrical engineers used an analog computer which they claimed was much better for their purposes. You could compute 2 times 2 and get 3.9999999999999… to as many digits precision you might want. It was clear to me from the beginning that digital was better.

This is an excerpt from a letter I wrote to Jon McGrew on the occasion of his 70th Birthday. Many years ago, Jon made an APL2 clock for me. You can see it over my right shoulder in this photograph of me in my office at IBM taken in 1982 around the announcement of the APL2 IUP. Above it is a needlepoint of a computer made by my wife Karen and to my left is a 3270 Terminal (Remember those? Compare them with your Smart Phone! What will the next 30 years bring?)

Jon’s APL2 clock is one of my prized possessions. As you can see, it is in the shape of an apple and each of the 12 numbers on the face is the digit 2. Clearly such an analog timekeeping device can be improved. I made a digital version which I think is vastly superior. Here’s a picture:

Even though this works perfectly well and is clearly an improvement, some people object to it because any time you look at it the time is 2:22. I point out that the digital version is exactly correct twice a day (remember that the US does not use a 24 hour clock). This was never true of Jon's clock. It was always slightly off.

Don't eat the cookies

One of my customers was the US Food and Drug Administration (FDA) which is charged with making sure food, cosmetics, medicine and almost any product is safe and useful. I was using the newly available APL2 to SmartArrays translator to convert one of their APL2 applications to C++. Down the hall from the office where I was working on the first day was a breakroom where they had some cookies, crackers and beverages sitting out. "Breakroom" was not a familiar concept to me. It's a room where you can go when you take a break from your work. I was surprised that I never saw anyone in the breakroom consuming any of the snacks or drinks. That afternoon, my host, Larry Dusold, took me on a tour of the building. They had a very impressive computing facility and high-tech testing equipment. I was again surprised (and maybe a little horrified) when the "breakroom" was part of the tour. It was not a breakroom – it was a testing room and the foods and drinks were being tested for bacteria. I was very glad that I did not sneak in and grab a cookie.

Free Samples

Often customers like to give little gifts to visitors and while IBMers cannot accept gifts for personal gain when representing IBM, they can accept small courtesy gifts. For example, on one visit to Karsten's Mfg (Manufacturer of Ping Golf Clubs) in Phoenix, Arizona, they gave me a small metal candlestick holder that they had milled on a numerically controlled machine run by an APL application. This is not the actual one I was given but it's close:

Karsten's designed and manufactured their golf clubs using APL.

In September and October 1995, I was on the longest business trip of my career visiting 13 European cities. IBM management was concerned that I might visit customers who have little potential for future business so they wanted some indication that the customers I proposed seeing this trip were real prospects. This was resolved by asking the customers to pay for the visits. This was offered through a German IBM Business Partner, Dittrich & Partner Consulting (DPC), who have supported me repeatedly throughout my IBM days and after.

One customer I visited was Mercedes Benz in Karlsruhe, Germany. I had hopes that a nice courtesy gift would be a new car.

Those hopes were dashed when I realized that it was the Mercedes Benz truck assembly plant. A truck would have been a nice gift but with all luggage I carried for this long trip, I really didn't need a truck as well. There is also the problem that they didn't offer one.

However, another customer proved more fruitful. Banque de France in Paris is the agency that, among many other missions, designs the French paper currency. It's a very difficult job to design currency that is pleasing to see yet secure. Copy machines are so good these days that a top quality color copier can make a passable copy of paper money. Banque de France designed all their money with APL2. I was given a sample of the currency that was designed with APL and I still have it:

I was so impressed that I asked if I could have a few hundred more samples but they declined. A few hundred of these would have been more desirable than a few hundred candlestick holders.

Follow the money

SHARE is an IBM user group formed in 1955 and SEAS is SHARE Europe. These user groups host periodic technical conferences. (SHARE denies that the name is an acronym for "Society to help avoid redundant effort".) Sometime in the early 1980's, exact date, organization and location forgotten, there was a SEAS meeting somewhere in the Netherlands. During this meeting I did some fundraising for APL development using the Apple Bank pictured here.

You put a coin on the red button and a worm comes out of the bank and grabs the coin and deposits it in the bank. You can see this in action on the YouTube video at this address;

https://youtu.be/gtPkal19D8I

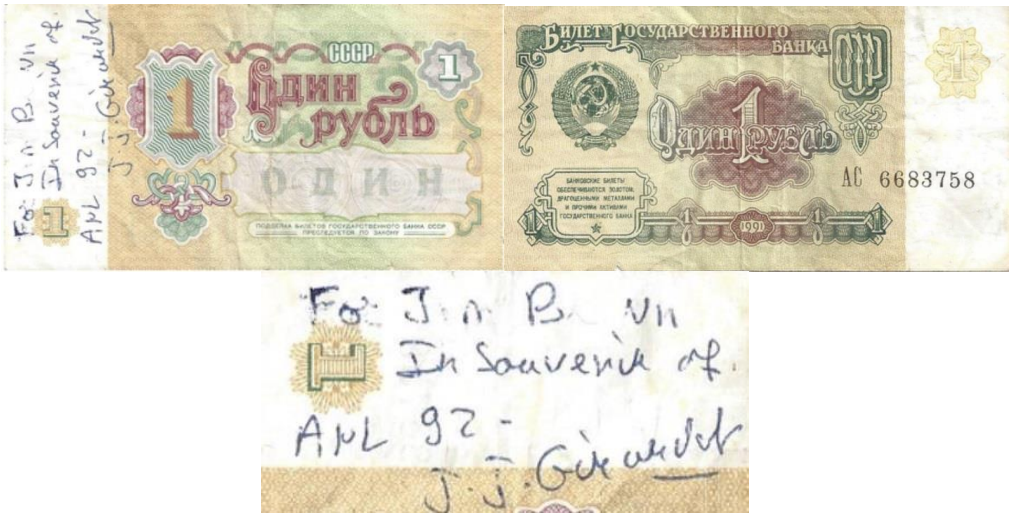
Here is the total contribution made at this meeting. I still have everything that was donated proving what a secure investment this was.

There are two 1 Gilder coins and an assortment of 10 25 and 50 cents plus one coin labeled “Filtrum Hazerswoude (runduk)”. I don’t know what this last one is worth and I suspect it is some kind of metro or Amusement Park token that someone wanted to get rid of. I don’t think it is money because there is no denomination on it and I was told that the only coin in the world that doesn’t show its denomination is the US Dime (worth 10 cents).

Signing the money

One of the most memorable APL conferences was APL92 in St. Petersburg Russia. (Karen and I had attended their practice meeting in 1990 in the same city – Leningrad.) One of the social events was a visit to Kronshtadt (also spelled Kronštadt) naval base. We were the first group of non-governmental people to be permitted to visit this secret base. It is located on Kotlin Island, 30 kilometers west of St. Petersburg. Its location was so secret that of the group of boats that took us to the island, only the lead boat knew the directions or destination. As we walked from the dock to the meeting hall, naval personnel came out of their dwellings to take pictures of us. They’d never seen something like this before.

I don’t remember what started it but at the reception given at the naval base I asked Jean-Jacques Girardot, from France, to sign some money. I told him that when he became famous, I could sell his signature on eBay for a lot of money. (I didn’t really say eBay because this was 1992 and eBay was not founded until 1995 but you get the idea.) Here is pictures of the note with Giradot’s signature:



I don't know what country's money this is. It cannot be Russian Rubles because I don't think it is legal to take Russian money out of the country and yet I have these.

This bill was signed by Alexander (Sasha) Skomorokhov:

Sasha is a nuclear engineer from Obninsk, Russia which could explain the currency he is using. Obninsk used to be so secret that its location was not on Russian maps. Sasha has become a dear friend.

Suddenly it became popular for people to sign money for me. Many of the signatures are not readable but here's pictures of them and the signatures:

I would, of course, never suggest that people just give me money but I was too polite to say no. Again note the security of this investment as I still have every bill given to me. To this day I would still not suggest that people just give me money but again, I am too polite to say no.

Evening Seminar

APLers are never ones to party when we could have technical discussions. At the Leningrad conference in 1990 and again in St. Petersburg in 1992, most evenings we attended "Evening Seminar". These gatherings are remarkable when you consider the number of different native languages that are represented by the attendees. Yet the discussions were always in English. The following quiz might have been due to Erkki Juvonen, from Finland, but I'm not certain about this:

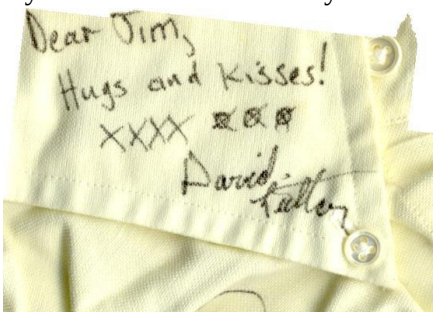
Question: What do you call a person who speaks three languages?
Answer: Tri-Lingual

Question: What do you call a person who speaks two languages?
Answer: Bi-Lingual

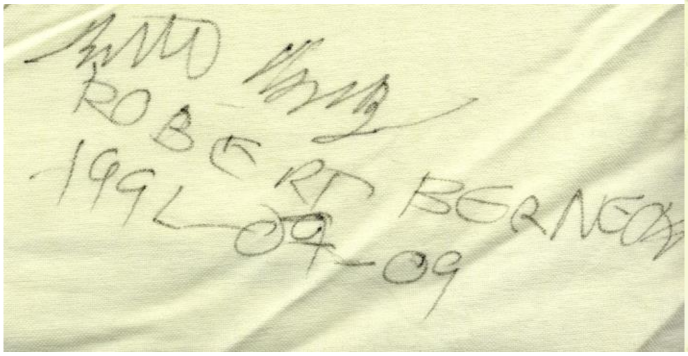
Question: What do you call a person who speaks one language?
Answer: American

Evening seminar always started with a vodka welcome toast followed by a vodka toast to the creators of APL followed by a vodka toast to … you get the picture. We never did get past the toasts to the technical agenda. At the beginning of the week we had high quality vodka. By the end of the week, you had to drink fast or the vodka would eat through the bottom of the Styrofoam cup.

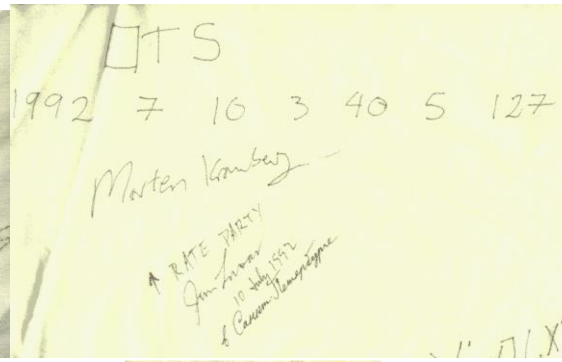
I'm not exactly sure how this started but at some point after Karen had gone back to our room to sleep, the discussion turned to the money that had been signed and given to me. I am sure that David Liebtag was the first to do this – he signed my brand new short sleeve yellow shirt. Here's what he wrote:



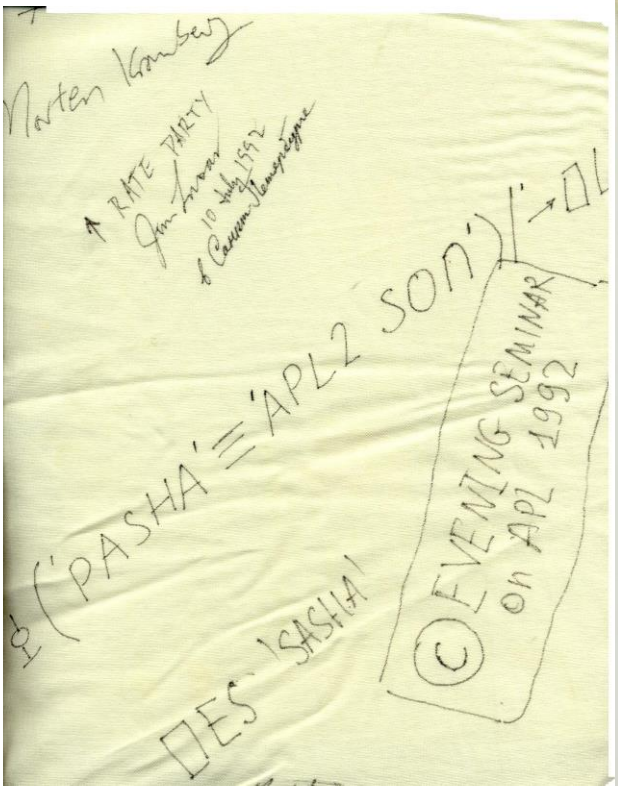
(The first signer was actually Khris Garner but I think David deserves the blame.) Now, since the shirt was ruined anyway, I had everyone else at the seminar sign it. Here's the rest of the signatures:



ROBERT BERNECKY
1991-07-09

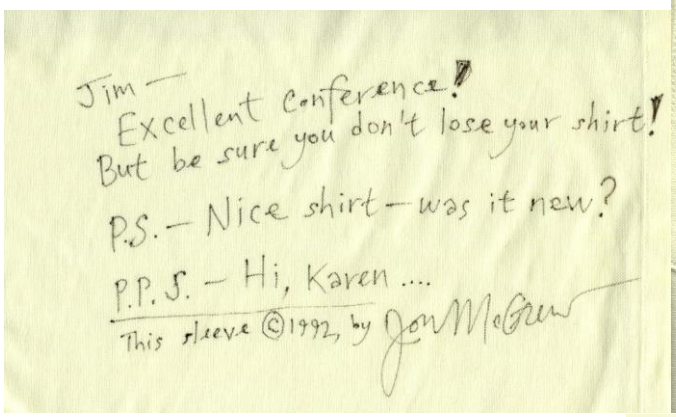


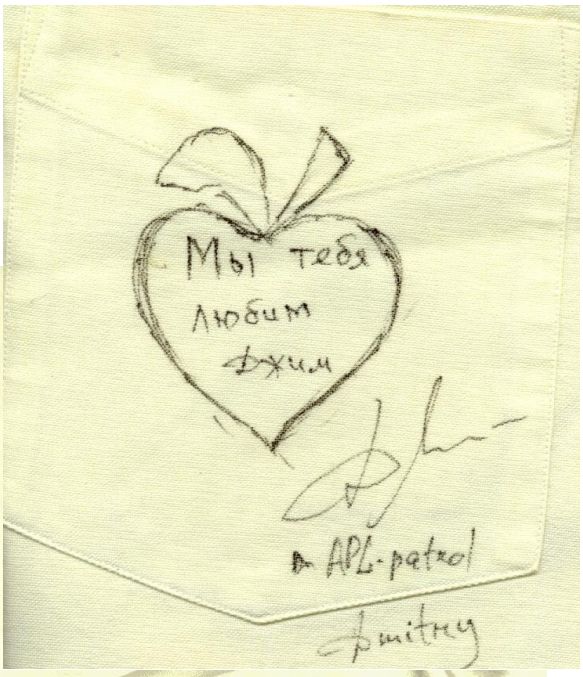
⎕TS
1992 7 10 3 40 5 127
Morten Kromberg
↑ RATE PARTY
10 July 1992

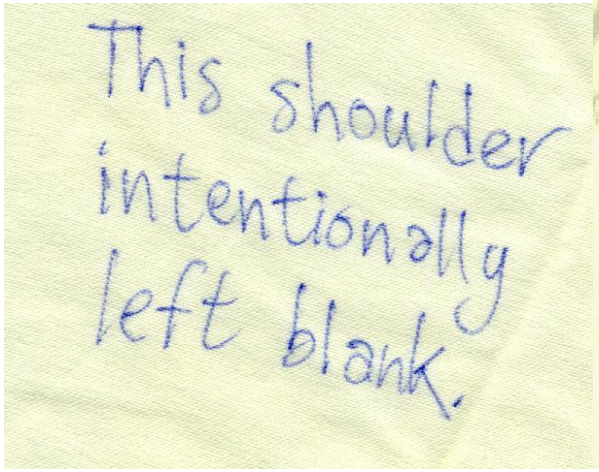


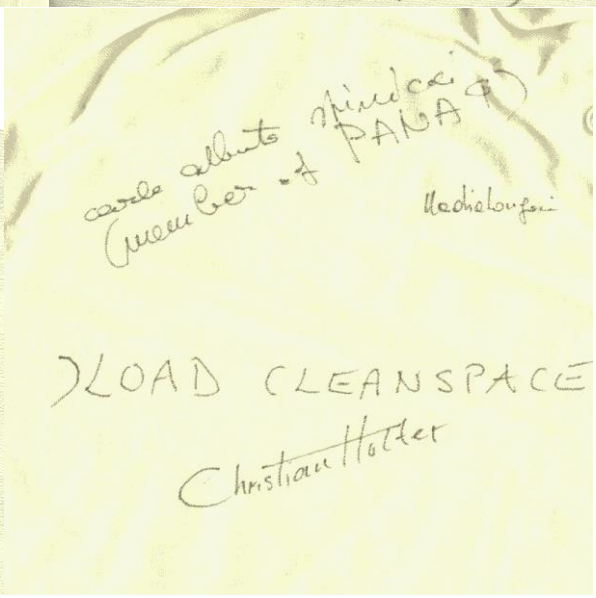
Morten Kromberg
↑ RATE PARTY
10 July 1992
('PASHA'≡'APL2 SON')
⎕ES 'SASHA'
© EVENING SEMINAR on APL 1992

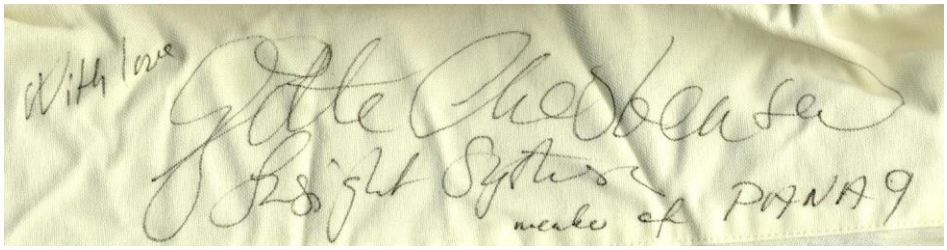
St. Petersburg experience!!
3:40am Kris Garner
Lynne C. Shaw
7/10/92 04:17:00



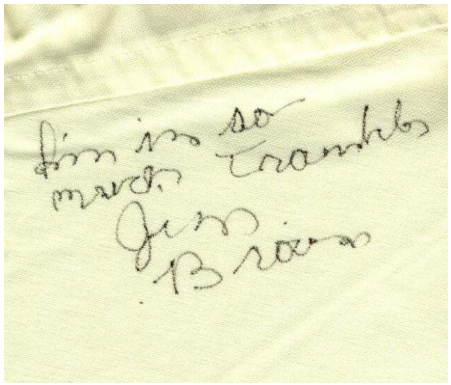








Lastly, I signed it:



When I got back to my room, I folded the shirt and put it in my briefcase. I didn't want it washed with the other laundry. In the morning, Karen asked "Where was my new shirt?" I gave the obvious answer "In my briefcase". This did not seem to be an adequate answer.

Conclusion

Warren Buffett once said "Always associate yourself with people who are better than you." I certainly did this when I began to associate with APL in 1967 and join the group in 1969 and I've continued to do this whenever the opportunity was available. It was and is an honor and a pleasure to work with APL and the talented people who are drawn to APL.

Acknowledgements

So many people contributed to the design, implementation, support and marketing of APL and APL2 that it is impossible to give even a partial list. Nevertheless, here's a few citations with apologies to those not mentioned.

Of course Ken Iverson and Adin Falkoff must be mentioned. Even though Adin and I had serious disagreements on the language definition, he was always a good manager and furthered my career. Many people from the early APL group need to be mentioned. Larry Breed showed me that the work is important and, as I said, I did not know he was my manager. Eugene McDonnell showed me how to use mathematics in the design of notation. Trenchard More never lost his cool and presented arguments with equations not emotion. He had a tremendous influence on the final definition of the APL2 language. His array theory became the theoretical basis for the arrays in APL2. Dick Lathwell showed me a certain indifference to the chain of command. When I left Syracuse and came back to IBM at the IBM Philadelphia Scientific Center, Dick was my manager. He was a great manager and I believe we did significant and valuable work together, He was also fun to be around. He advised me to take positions strongly and then, if nobody objects, you've won. If someone takes a strong counter-position and it has some merit, back off. (This absolutely the right thing to do. You don't think so? Nevermind.) Dick was also very influential in my personal life.

Garth Foster, my thesis advisor at Syracuse University, was a guiding influence throughout the creation of the first version of what became APL2 as represented in my thesis *"A Generalization of APL"* [BR1]. I avoided many potential trouble areas because of his close attention. It was very pleasing to me that Garth

used my dissertation as study material for some classes for several years after I graduated. Garth has continued to host APL implementer's conferences at the Syracuse Minnowbrook Conference Center.

As mentioned before, Bob Smith, when he visited the Santa Teresa Lab, forced me to face the fact that simplicity and elegance are more important than consensus. He continues to do exciting work with his Nested Arrays Research System NARS2000 (http://www.nars2000.org/) which is in the public domain.

My wife, Karen, (best friend for over 55 years) has been through it all with me. She's been to so many of my presentations that she could probably give some of them. At APL81, in Washington, D.C., she overheard someone pointing her out by saying "There's the other Mrs. Nested Arrays". We believe this was Bob Smith's wife, Mary, who was called "Mrs. Nested Arrays" by his STSC collogues.

Finally, I must thank Axel Güth for being my mentor for technical, business and marketing matters. This began when we first met and continues to this day. We still have long phone calls to discuss technical and business matters. He remains my only colleague who has a separate mail folder in my Email client:

There is no significance that his folder comes before the "Humor" folder. It is an accident of alphabetical order.

References

[AB1] Abrams, P.; "An APL Machine"; PhD Dissertation Stanford University; 1970

[BR1] Brown, J.; "A Generalization of APL"; PhD Dissertation Syracuse University School of Computer Science; 1971

[HE1] Hellerman, H.; "Experimental personalized array translator system"; Communications of the ACM; Volume 7 Issue 7, July 1964; Pages 433-438

[IV1] Iverson, K,& Brooks, P; "Automatic Data Processing"; John Wiley and Sons; 1963

[IV2] Iverson, K., Falkoff, A., Sussenguth E.; "A Formal Description of SYSTEM/360"; IBM Systems Journal; Vol. 3 No. 3, 1984

[IV3] Iverson, K.; "A Programming Language"; John Wiley & Sons, Inc.; 1982

[IV4] Iverson, K." Elementary Functions: An Algorithmic Treatment"; Science Research Associates;1966

[JS1] Various; "APL Quotations and Anecdotes"; http://www.jsoftware.com/papers/APLQA.htm

[MO1] Montalbano, M.; "A Personal History of APL"; http://ed-thelen.org/comp-hist/APL-hist.html

[MO2] More, T; “Notes on the Development of a theory of Arrays”; IBM Philadelphia Scientific Center Tech, Report No. 320-3016, May 1973

[MO3] Montalbano, M. (Guest, J.C.L); “APL Blossom Time”; YouTube video https://youtu.be/0zSauxkMxPo

[MO4] Montalbano, M. (Guest, J.C.L); “APL Blossom Time Lyrics”; YouTube video https://youtu.be/g4xvjfr297E

[PO1] Post, Ed.; “Real Programmers Don't Use Pascal”; Letter to the editor, Datamation July1983

[SU1] Sushtek, L, “The APL Programming Language Source Code”, http://www.computerhistory.org/atchm/the-apl-programming-language-source-code/

[WI1] Whitehead, A.N., Russell, B; “Principia Mathematica”; Cambridge at the University Press,1910.

Appendix 1: APL Blossom Time Lyrics

APL Blossom Time

J. C. L. Guest (Michael S. Montalbano)

Back in the old days, in 1962,
A feller named Ken Iverson decided what to do.
He gathered all the papers he'd been writing fer a spell
And he put them in a little book and called it APL.

Well...
    He got him a jot and he got him a ravel
    And he revved his compression up as high as she could go
    And he did some reduction and he did some expansion
    And he sheltered all his numbers with a ceiling and a flo'

Now Sussenguth and Falkoff, they thought it would be fine
To use the new notation to describe the product line.
They got with Dr. Iverson and went behind the scenes
And wrote a clear description of a batch of new machines.

Well...
    They wrote down dots and they wrote down squiggles
    And they wrote down symbols that they didn't even know
    And they wrote down questions when they didn't know the answer
    And they made the Systems Journal in nineteen sixty-fo'

Now writing dots and squiggles is a mighty pleasant task
But it doesn't answer questions that a lot of people ask.
Ken needed an interpreter for folks who couldn't read
So he hiked to Californ-i-a to talk to Larry Breed.

    Oh, he got Larry Breed and he got Phil Abrams
    And they started coding Fortran just as fast as they could go
    And they punched up cards and ran them through the reader
    In Stanford, Palo Alto, on the seventy ninety oh.

Well a Fortran batch interpreter's a mighty awesome thing
But while it hums a pretty tune it doesn't really sing.
The thing that we all had to have to make our lives sublime
Was an interactive program that would let us share the time.

    Oh, they got Roger Moore and they got Dick Lathwell,
    And they got Gene McDonnell with his carets and his sticks,
    And you should've heard the uproar in the Hudson River valley
    When they saved the first CLEANSPACE in 1966.

Well, when A1 Rose saw this he took a little ride
In a big station wagon with a type ball by his side.
He did a lot of teaching and he had a lot of fun
With an old, bent, beat-up 2741.

    Oh, it typed out stars and it typed out circles
    And it twisted and it wiggled just like a living thing.
    Al fed it a tape when he couldn't get a phone line
    And it purred like a tiger with its trainer in the ring.

Now, there's much more to the story, but I just don't have the time
(And I doubt you have the patience) for an even longer rhyme.
So I'm ending this first chapter of the tale I hope to tell

Of how Iverson's notation blossomed into APL.

So..

Keep writing nands when you're not writing neithers,
And point with an arrow to the place you want to be,
But don't forget to bless those early APL sources
Who preserved the little seedling that became an APL tree.

Dedicated to the pioneers of APL with respect and
affection by J. C. L. Guest

Appendix 2: Work History

1965-1968 IBM Owego, Owego New York

1968-1971 Syracuse University, Syracuse, New York

1969-1971 IBM Research Yorktown Heights, New York

1971-1974 IBM Philadelphia Scientific Center, Philadelphia, PA

1974-1974 University of Pennsylvania Moore School of Engineering, Philadelphia, PA Professor

1974-1978 IBM Palo Alto, Palo Alto, Ca.

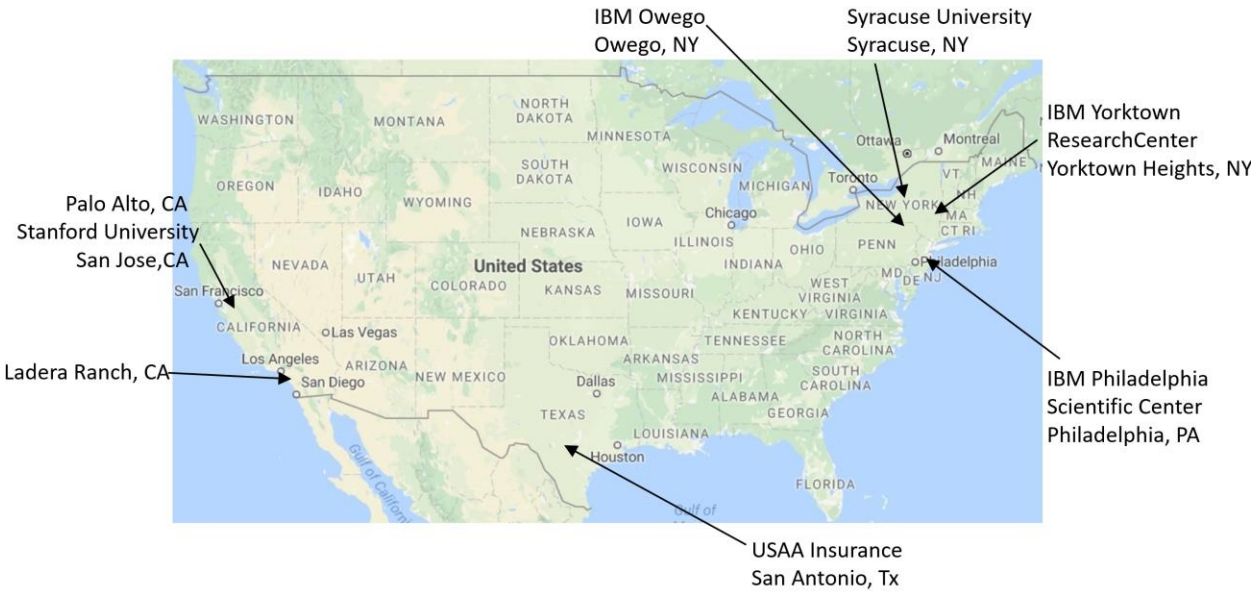
1978-1981 IBM Research Yorktown Heights, New York

1981-1997 IBM Palo Alto and IBM Santa Teresa, San Jose, California

1999- present SmartArrays Inc, Ladera Ranch, California

2004-2007 USAA, San Antonio, Texas

2010- present NestedComputing Corp, Ladera Ranch, California



Appendix 3: Cast of Characters

These are people mentioned in the “Early Years” portion of this paper with a few words about what they did and what influence they had one me. Apologies to the innumerable people who influenced my work who did not get a mention.

Ken Iverson	Creator of the original APL and was the moral leader of the APL community for the rest of his life.
Adin Falkoff	Adin's talk at an ACM meeting was my first encounter with APL. He was my manager at IBM several times over the years as I moved back and forth from the east to the west to the east to the west coast of the US.
Edward Sussenguth	Collaborator on "The Formal Description of System 360". To my knowledge, he had no further influence on the APL world. I never met him.
Gerry Barrett	Operated the IBM 360 Model 50 on which the first APL service ran.
Bill Jones	Professor at Syracuse University who taught at IBM Endicott. Responsible for me going to Syracuse.
Al Rose	First marketer of APL. Famous for his portable 2741 and the ability to give an APL demo even when he could not attach to the APL service.
Mike Montalbano	Proponent of APL from the early days. Author of the lyrics of the song "APL Blossom Time". Member of the APL Development group in Northern California.
Phil Abrams	Wrote an early APL system with Larry Breed at Stanford University. His PhD Thesis "An APL Machine" contained execution schemes which I incorporated in the implementation of APL2.
Larry Breed	My first APL manager (unbeknownst to me) and implementer of what became APL\360. Moved to STSC (Scientific Time Sharing) to work on their service. Later came back to IBM and worked for me in Palo Alto.
Dick Lathwell	Implementer of many important aspects of IBM APL (including shared Variables). Was my manager at the IBM Philadelphia Scientific Center and became a close personal friend.
Seth Breidbart	Another summer employee at IBM Research in Yorktown heights. He was known for a dry sense of humor. If you said "The sun is hot today" he'd say "Yes. Several million degrees". He has continued to produce interesting work both in the APL world and other disciplines.
Trenchard More	Creator of “Array Theory”. His theory and his style of work had a tremendous influence on APL2 and on me personally. He is a true computer scientist who applies rigor to everything he does.
Herb Hellerman	Author of the first "Iverson Notation" computer implementation PAT (Personalized Array Translator System).
Garth Foster	My PhD thesis advisor at Syracuse University. Instrumental in the spread of APL and creator of the APL Quote Quad - the APL periodical. He still runs APL Technical meetings at the Syracuse University Minnowbrook Conference Center.
Graham Driscoll	Member of the early APL group at IBM Yorktown.
Eugene McDonnell	Member of the early APL group at IBM Yorktown. Gene was responsible for many of the numerical primitives in the APL language notable the circle functions and computing on complex numbers.
Alan Perlis	A computer scientist who was the first person to receive the Turing Award. He was on the team that developed ALGOL. As a professor at Yale University, he was interested in APL and wrote many papers including “Should APL and Lisp be combined”. The answer was NO!
Bob Smith	Creator of the nested array system for STSC (Scientific Time Sharing). He had a tremendous influence on me and the IBM language. He continues to produce leading edge work in the array world with his NARS200 open source implementation. He is still a close friend.

Appendix 4: Empty Arrays (and the Great Empty Array Joke Contest)

After vector notation, the next biggest controversy is APL2 empty arrays with prototypes. Like Vector Notation, again, one faction of APL providers did not accept the notion.

One might think that if an array is empty, it doesn't have anything in it. But that was never true in APL. To describe this situation, Gene McDonnell came up with this joke – the first official empty array joke:

> Man in Restaurant: I'd like to have strawberries without cream.
> Waiter: We haven't any cream.
> Man: I want my strawberries without cream!
> Waiter: We don't have any cream!
> Man: Do you have Milk?
> Waiter: Yes, we have milk.
> Man: Then I'll have strawberries without milk.

I used this joke in talks about APL2 attempting to describe the reasoning behind prototypes. Here's an outline of that attempt.

APL has a function "take" where the left argument says how many items to take from the list given as right argument. Here are two examples:

```
      3↑ 10 20 30 40 50
10 20 30
      3↑'ABCDE'
ABC
```

You can take more items than there are and the result is filled with typical items:

```
      5↑ 10 20 30
10 20 30 0 0
      5↑'ABC'
ABC    ←—— There's two blanks at the end. See them? (Take my word for it)
```

You can take more items than there are from an empty vector. Here's an example with an empty numeric and an empty character vector:

```
      5↑⍳0
0 0 0 0 0
      5↑''
        ←—— There's five blanks at the end. See them? (Take my word for it)
```

All this is in the original APL without my data structure extensions. What does it mean? There is more to an array that just shape and values – there is also type and it's really noticeable with empty arrays. My

argument is that an empty numeric vector is a vector without cream and an empty character vector is a vector without milk.

Here's overtake of nested vectors:

```
      5↑(1 2 3)(10 20 30)
1 2 3  10 20 30  0 0 0  0 0 0  0 0 0
      5↑'AB' 'CD'
AB CD
```

← Three items at the end are nested two-item vectors of blanks See them?

Zero take of these arguments produce empty arrays of three item vectors. This leads to another empty array joke:

> Question: How many empty arrays does it take to fill a workspace?
> Answer: One, if it's big enough

(You have to admit – empty array jokes are hilarious!)

Here's another from Gene McDonnell:

> Customer: What price are your pork chops?
> Butcher: $5.98 per pound.
> Customer: That's outrageous! Mr. Schmidt, down the street, charges only $3.98 per pound.
> Butcher: Why don't you buy them from Mr. Schmidt, then?
> Customer: He's all out of them today.
> Butcher: When I'm all out of chops, I charge only $2.98 for them.

I don't have information on the source of these jokes:

> Customer: My beer glass is empty. I'll have another.
> Bartender: Why do you want another empty beer glass?

For a number of years, I was the official joke editor for the APL Quote Quad – the APL periodical from the Association of Computing Machinery. I had hoped to list here all the jokes which were printed but I was unable to locate all of them. The jokes were numbered and I give the original number if known along with the name of the submitter when known.

Here's some from the APL Quote Quad Vol 13 No,2; December 1982:

#11

> A women gets on a bus with three sets of twins.
> Driver: Gosh, lady, do you always get twins?
> Woman: Not always – hundreds of times we don't get anything at all.
> Donna Haas

#12

Patient: Doctor, have you got a cure for complete loss of voice?
Doctor: Good morning, can I help you?
Rodger Bagnell

#13

Question: How many Marxists does it take to screw in a proletarian light bulb?
Answer: None (of course) because a proletarian light bulb contains the seeds of its own revolution.
George V.E. Otto

#14

Archaeologist: I have made an astounding discovery! On the basis of my investigation of the pyramid of Cheops, I find that the ancient Egyptians knew all about wireless radio.
Reporter: That's astounding! How did you determine that?
Archaeologist: In all my investigations, I never found any wire.
Unknown

These stories led to the creation of "The Great Empty Array Joke Contest". There is a cash reward of 100 to the person who submits the best empty array joke. (The currency to be used for the prize will be determined at a later date. Originally the prize was limited to $10.00 but I'm feeling generous.) Best of all, there is no time limit for the submission of the best. Here's the joke that I deem to be best so far:

Man in Bar: Didn't I meet you in Zanzibar last year?
Woman: I've never been to Zanzibar
Man: Neither have I. It must have been two other people.

I like this because it's an empty array whose prototype is a two-item vector. The contest has been going on for 30 years but it's still not too late to submit your joke. Do it now before you forget.

During these years, I received many submissions (by paper mail) and I was astounded to see what some people thought was funny. A large number of people submitted an envelope with nothing in it. An empty submission. Even more people didn't even do that much. Jon McGrew claims to be the first in this last group. You need to see him for proof of this claim.

Many thanks to Jon McGrew, Curtis Jones and Lucille Boone for digging through archives to find some of the original stories.