Jon McGrew

Forgotten APL Influences

Each of these conferences typically focus on the future, rather than simply reviewing the past and rehash what we did years ago. I appreciate that. However, because this is the 50th anniversary of APL, my presentation is going to talk about history.

I realized that there are a lot of things from the past for which APL really should get high marks, some of which seem to be forgotten. These are places where APL has really made its mark and made an influence in the world around us, but it may have been forgotten that APL was ever involved with that. We all use instant messaging, word processors, and spreadsheets… but are you aware that these all have links to APL? [1]

Before I get into that, I have a list of *Thank You*'s, and one of the things that I will start with is workspace `1 CLEANSPACE`. I put out an APL newsletter in the 1970s and '80s, internal to IBM, called The APL *Jot Dot Times*, and in the mid-1970s, I wrote an article for it about the history of APL at that time, and one question that I wanted to address was, "*When was APL 'born'?*"

That turned out to be a more complicated question than I had expected, so finding a good answer became kind of a quest for a while. The problem was, should we consider its starting point to be when Ken Iverson first started to come up with ideas for his notation, ***or*** when he started using the material for teaching at Harvard, ***or*** when he sent his seminal book to press ("*A Programming Language*"[1,2,3]), ***or*** when the publisher first made that book available to the public, ***or*** when he joined IBM, ***or*** when his group first started implementing it on the computer, ***or*** when the first product was released, …***or***… ?

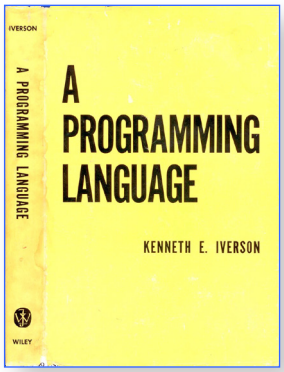


Figure: "A Programming Language" textbook, 1962

There are so many dates that we could have chosen for this—all of them reasonable. And finally, I realized that the first workspace that the developers saved was *still there*, and for the users of our time-sharing systems which we were running there in Kingston, New York—the people who would be receiving my newsletter— I thought that the more important point *to them* is the history of *what they are actually using*, not necessarily the notation that preceded that, so back in the mid-1970s I chose workspace `1 CLEANSPACE` as being a starting point for APL:

1 Given as a multimedia presentation at the APL 50th Anniversary Conference at Böblingen, Germany on 29 November 2016

```
      )LOAD 1 CLEANSPACE
SAVED 1966-11-27 17.53.59 (GMT-5)
```

I have noticed that some people have also pointed to the 1991 issue of the *IBM Systems Journal*[4] which celebrated the 25th Anniversary of APL, saying that this underscores the validity of using 1966 as the starting point for APL. I agree; thank you—that was the issue which Ray Polivka and I created for that event.

Michael S. Montalbano also had some discussion of workspace `1 CLEANSPACE` in his 1982 paper, "*A Personal History of APL.*"[5]

I'm not sure how many other people also started looking at APL's "date of birth" back in the 1970s. But I was pleased to see that this conference observed that as the starting point for APL.

~

So I am going to start with some background by just talking about some of the people involved, with a huge *Thank You* to each of them:

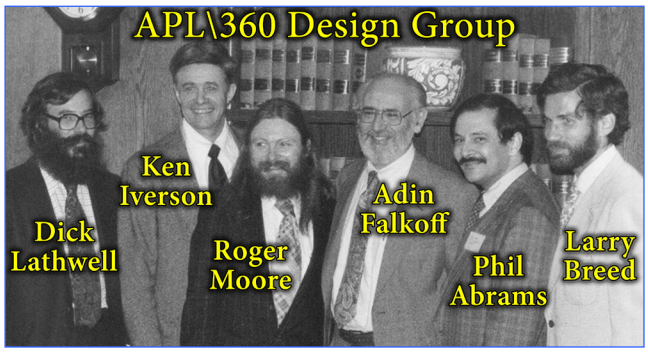


Figure: Original APL\360 Design Group

This photo[6] shows members of the original APL\360 Design Group and probably most of you know the players, but for those of you who don't, Ken Iverson and Adin Falkoff, of course, were the key players who were in charge of designing and creating the APL language, and those who actually did the implementation work were Dick Lathwell, Roger Moore, Phil Abrams, and Larry Breed. They were the original development team for APL\360. This is a picture that was taken at the I. P. Sharp Conference in Toronto in 1982.

…And by the way, just as an aside, does it ever bother you how people sometimes get rather arbitrarily cropped out of pictures?

Figure: Photobombed?! …"Which one is not like the others? Which one doesn't belong?"

A friend showed me this uncropped photo,[7] and… well, okay, so it wasn't actually all *that* arbitrary in this case; I'll have to admit that that's me skulking about in the background, lurking there in the shadows— but I was never part of the APL\360 Design Group. This photo was taken in the conference hospitality suite, and I just happened to be coming in for coffee. I had no idea they were taking a picture there that day, and I certainly had no intention of photobombing anyone. Oops.

Another piece of history that I'll show you is this photo[8] from an ACM meeting in Las Vegas in August of 1968. IBM put quite a large booth together, and notice that there are several IBM 2741 terminals[9,10] set up, each with television cameras pointed at them and monitors above them so that people could see what was being typed on the page; this was back before video display terminals were as commonly available:

Front: Arnold, Sandra Pakin, Conroy, Adin Falkoff; **Back:** Crane, B. Bergquist, T. Wilson, Van Guilderan, Al Rose, Mike Montalbano

Figure: APL at ACM meeting in Las Vegas in 1968

None of this so far has anything to do with the *Forgotten APL Influences*, but because this is a 50th Anniversary Conference, I thought that we ought to observe some of our collective APL backgrounds.

~

My background

Here's a little bit about my own background:

- I am retired from IBM.
- I have been using APL since 1971.
- APL became my career in 1975.
- I worked in the APL support team for a large timesharing system in Kingston, New York. We had users in 17 countries using our machine.
- I then worked on APL projects in the group called *Numerically Intensive Computing*. More about that in a bit.
- I was a member of the APL Development team at IBM during the development of APL2.[11] The main group was in California, and we had part of that same department (under the same manager) in Kingston, New York. (We liked to consider it to be Distributed Intelligence, but some may disagree.)
- Later, I worked in APL development at Morgan Stanley on the Aplus[12,13] project, which is their own in-house version of APL.
- And so that I don't misrepresent myself, let me point out that I was not one of the APL implementers— I didn't work on the internal code. I focused mainly on the language itself, and attended many of the design meetings and tried to argue for what I believed in regarding language design. My main function there was APL programming and APL documentation.

So in putting this together, one problem that I had was, *where do I draw the line* on history discussions? It's hard to decide whether or not to include some of the APL *applications*, because there have been countless thousands of them, some of which are truly notable. I'll list just a few of them here and then not cover them further in the rest of the more detailed presentation:

- Program trading: We all hear about how the Wall Street people have program trading, where the buying and selling is done automatically, and most of that has been done with APL.

- Insurance companies have traditionally used APL very heavily, to the point where, as I visited insurance customers, I was surprised to discover that many of the actuaries thought that APL was the Actuarial Programming Language, because it was so well suited to what they needed that they assumed it was designed for them.

- Automated warehouse: Chuck Norcutt, a coworker in my department, created the code to keep track all of the products in a major IBM warehouse. This project, written entirely in APL, ran a large robotically-automated warehouse in Raleigh, North Carolina.

- IBM mainframe configurator: Ordering a mainframe computer is a very complicated process. There are a lot of options, many of which are mutually exclusive. So for decades, that was all being handled by a complex set of APL functions. I understand that at one point they had a team of 125 people working on rewriting it in a different language— and they failed because it was just too complex.

- Space shuttle: The landing of the NASA space shuttle was being calculated for a while by APL; the trajectories were being computed with APL functions from the IBM Federal Systems Division.

And while any of these topics and many others like them might be interesting (and each could be a presentation of its own), I'll try to focus this discussion on areas where the power of the language or unique features within the language have made it stand out, and places where other languages and processes have emulated what APL did, rather than simply discussing applications, no matter how interesting they are.

A deciding point regarding material to be included is: Does this cause you to say, "I didn't know that APL was involved with *that!*"?

~

I have broken this presentation into two sections, the first being "***Places Where APL was Early to the Game***." By this, I mean that we will discuss approaches or technologies which APL *did not* invent, but in which APL was used to good advantage early on, accomplishing things that other tools didn't or couldn't accomplish in those days. And the second section will be "***Things that APL Pioneered***," for those things that actually *did* originate with APL.

Section 1: Places Where APL was "Early to the Game"

Instant Messaging

Although I am certainly into technology, I was somehow a late adopter when it came to smart phones. My wife and I finally traded in our flip phones for smart phones just two years ago — and at that point, our son laughed at us and thought, well, they're finally getting into the 21st Century. "*So I can actually send you a 'text' now, right? Do you even know what that is? …You're almost starting to get into the modern age.*" And I said, "*Ian, I've been using instant messaging since 1971.*" (A blank stare from our son at this point, basically implying, *That's back when the dinosaurs roamed the earth, isn't it?*)

This is what really got me thinking about all of the things that APLers have used for decades, and which we may have forgotten were APL facilities years ago. That was the basis for starting this paper.

Messages used to be handled by an APL system command and that was, by any other name, instant messaging… or if you prefer, text messaging. Messages were limited to 120 characters. As an example, let's send a note to "Bob," where 265 is the port number that he is on (which I would get from the "*)PORTS*" command):

```
      )MSG 265 HI, BOB. READY FOR LUNCH?
SENT
```

His response might be:

```
265: OKAY. BTW, I ONLY HAVE 20 MIN.  :-(
```

I want to point out also that we used these abbreviations, such as "BTW" for "by the way" —none of this is really new— we used that back in the mid- to late-1970s, and that smiley face, or the frowning face in this case, `:-(` …we used all this back then.

And the fact that we were sending those text messages around the world back then brought up another important aspect of this:

International PTT operations

PTT is "Postal, Telegraph, and Telephone" service, and it is the agency within governments around the world that controls communications. These days, we are very used to the idea of just taking a cable and plugging it into the wall — Ethernet or coax or whatever you have, and talking to the whole world — and it works so smoothly that we don't think about it. But that wasn't always the case. PTT has been a tightly-controlled monopoly in most countries, and you could not simply set up your own communication across borders.

Now, in the mid-1960s, along comes APL with this nifty built-in messaging command, and we want to support users around the world. How do we do this? A company which was instrumental in changing how PTT's handled messaging like this was I. P. Sharp Associates (IPSA) in Toronto, and my hat is off to them for taking on (and winning!) battles at the country level to make sure that they could have a message command in APL and send messages back and forth.[14] This all predated email, of course, so they did groundbreaking work on setting up a worldwide network, which included messaging.

Packet switching

The next part of this is that the I. P. Sharp APL network itself was groundbreaking in several ways. They developed one of the early packet-switched networks.[15,16] This is simply a faster, more efficient, and more reliable means of sending data around the world. Their semi-private network was officially called IPSANET,[17,18] and it became operational in May of 1976. This was implemented by Roger Moore, who was also one of the original APL implementers at IBM. So although APL didn't invent packet switching, it was really one of the very first users of that technology on a grand scale.

Interactivity

Another part of APL's power is its interactivity. To get into that, let me first discuss the *Grace Murray Hopper Award.*[19,20] This award is named for Rear Admiral Grace Hopper[21] and it has been given out since 1971 by ACM (the Association for Computing Machinery). The award goes to a computer professional who makes a single significant technical or service contribution before or at the age of 35. This is a very prestigious award.

Dick Lathwell, Larry Breed, and Roger Moore each received the Grace Murray Hopper Award in 1973.[22] But the thing I want to point out here is that it was *not* given to them for the APL language — there was no discussion in their award that congratulates them for the notation. Instead, it was presented for "*setting new standards in simplicity, efficiency, reliability and response time for interactive systems.*"[23,24] We sometimes forget— everything is so interactive these days, we are used to hopping on a computer and typing something and immediately getting a response or going onto a website and doing the same thing. But it wasn't always that way, and a lot of the push in getting things to be interactive started with APL. So as with other items in this section, APL didn't invent that, but it was very early to the game.

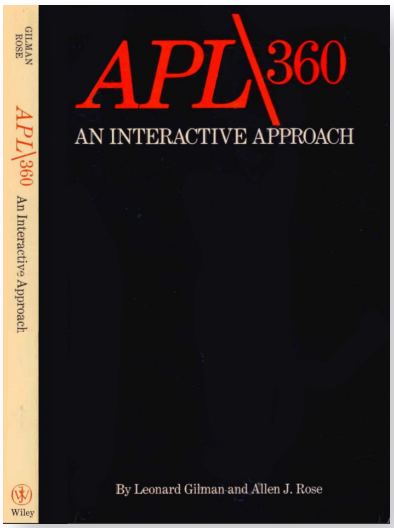


Figure: An early Gilman and Rose textbook

A tribute to the new approach was the Gilman and Rose textbook,[25] one of the early textbooks on APL (1970). It was named "*APL\360: An Interactive Approach*," because interactivity was the new and exciting focus at that time, replacing batch operations, so in many ways like this, APL helped to usher in the new era of timesharing.

APL helped to usher in the era of timesharing

In the mid-1960s, one of the early timesharing users was NASA. APL sounded like a good tool for them, and they were eager to start using it, but at the time, they had no

way to even dial into a computer system, and back then, the phone companies didn't have much expertise in helping people with data questions, and had no modems to provide to customers. So I was told that it was Adin Falkoff who drove 250 miles from IBM Yorktown to the NASA Goddard Center near Washington with ten Bell System 103A2 telephone data set modems so that they could get on-line with APL: [26]

Figure: Phone company-supplied modems, c. 1962

APL provided timesharing in an era when many people hadn't yet heard of the concept. In many areas, APL introduced people to timesharing.

APL public libraries

Years ago, we became used to the idea that the "APL public libraries" offered repositories of a wide variety of programs and information. Of course, many other systems that any of us have worked with have code repositories, and places where you can go to get help information and so forth. But the APL public libraries always went *way* beyond that, offering not only code libraries with routines such as Fast Fourier Transform and Statistical Analysis, but also informational sections such as on-line phone books, ordering information for publications, news, email and more— just a lot of things that are *unrelated* to the computer system you're using. Public Library workspaces were created on a huge range of topics. The Syracuse University Computing Center had once commented that "*the sheer volume of material in this library system was overwhelming*."[27]

Eventually—*much* later—this whole concept and the work that so many people had put into creating these varied facilities ended up evolving into… *websites*.

Election coverage

Election coverage wasn't an APL first; Univac had done this before, but it's another example of a place where APL was early in getting into the game. CBS got election tallying via a dial-up connection into an APL timesharing system.

Figure: Election coverage with APL

I suspect that it was probably still back to that early system in Philadelphia that the APL Design Group used, and one of the stories about this is that Walter Cronkite

was delivering election results at one point, and then he sort of stopped, and there was some confusion— back in those days television news didn't have the flow that it has today, so when a problem came up, they would sort of stop and look around. The problem was that Walter Cronkite had accidentally kicked the modem under the table with his foot and the phone line was disconnected. So as he was trying to fill time, one of the technicians was crawling in under the table in front of him to dial up the APL system again and get it back on the air so that he could get some more election results. …"*And that's the way it is.*"

But even with an occasional glitch like this, the point is that APL was very early in helping to bring this kind of information to people.

Early public computer demonstration

An early goal was just getting to show the public what a computer is all about, and this is a place where APL was involved very early.

Figure: IBM 2741 Selectric Terminal

A computer demonstration was set up at the Franklin Institute Science Museum in Philadelphia, consisting of an IBM 2741 (Selectric typewriter-style terminal)[9,10] connected to the APL Design Group's machine in the middle of Philadelphia — and that's probably the first view that people in Philadelphia had of what a computer was all about. They could actually get onto the 2741 and type in APL expressions, so of course this was all *hands-on*. Probably these days we wouldn't let people use a live connection into a development system, but somehow it all seemed to work back then.

Computer Viruses

Do you remember the old "self-replicating APL expressions"?[28] The challenge was, what is the shortest piece of APL code you can type that will return *itself* — an expression that would return exactly what you typed? It has to have some computing to it; it has to have at least one function, so you can't just type "4", for instance. Here is an example of an expression that returns itself:

```
      22⍴1⌽11⍴'''22⍴1⌽11⍴'''
22⍴1⌽11⍴'''22⍴1⌽11⍴'''
```

Okay, there's no practical point to it, just some fun, and perhaps it's a learning tool, but as a self-replicating expression… well, a variant of that —perhaps a distant cousin of that (…or maybe its evil twin)— is a *virus*. So when did the *first* computer virus get created?

I read through a lot of articles on viruses and one from the BBC talked about what they claimed was the first computer virus,

and it dates it at 1983.[29] Another article struck me as being interesting, which said that in 1982, a student named Rich Skrenta at Mount Lebanon High School in Pittsburgh, Pennsylvania wrote the "Elk Cloner" virus[30]— that caught my attention because that was my high school (…but no, *I* didn't write any viruses). Skrenta was in 9th grade at the time and he created a virus that was originally supposed to be a practical joke, but it got out of hand, and that was what *this* article claimed was the first virus, in 1982.

However, at a meeting of IBM's APL ITL (Interdivisional Technical Liaison) Committee[31] in California in 1976 or 1978, we had discussions of computer viruses — and that was a new term to us at the time. Larry Smith was an IBM executive who had been appointed to study this new phenomenon, and he came to our conference and talked about viruses. He surprised us by pointing out that the first computer virus was written in APL.

Now, I am pleased that I don't have to report that this virus was damaging — it was not. He emphasized that this was a *beneficial* virus. (…A what?) As he described it, this procedure could replicate corrections to other machines and it was therefore described as being analogous to a *flu shot*, delivering a carefully-controlled virus to *protect* you. Obviously, the correction itself had to be correct and he pointed out that when they realized that other viruses written in any language had the potential to do harm, they really started studying this, and he was appointed to be in charge of that study. These discussions precede other virus articles that we read about elsewhere.

Email

Although there are some differing opinions as to when the first email system appeared, credit has at times been given to an MIT in-house system, which was developed in 1965.[32] This was a very limited system, intended for use just within their own campus. The other contender is a system created by Frank Bates III of Mobility Systems, written in 1971.[34] These were both very limited systems, with neither one handling any commercial usage.

A discussion of "Internet in Its Infancy"[32] tells us that "*By the end of 1971 there were 23 computers at 15 different locations connected to the ARPANET … The following year* [1972] *the first true email software was written, and email rapidly became the most popular application on the network.*" This first "*true*" email system was created by Larry Breed, one of the founders of Scientific Time Sharing Corporation (STSC) in Bethesda, Maryland. He created it in 1972, and it was of course implemented in APL. This very successful email system was described as the "APL*PLUS Message Processing System Mailbox," or known to their customers simply as "the Mailbox."[33]

In that same Internet history article, the following statements were made about the STSC mailbox:

> "*In 1976, email was first used to gain political power. Jimmy Carter and Walter Mondale used email during their U. S. presidential campaign to co-ordinate their schedules. They won the election and Carter became a strong supporter of the internet. Each of their emails cost about US$4 to send.*"[32]

White House Press Secretary Jody Powell talked about how they used the STSC Mailbox to help them with Jimmy Carter's successful 1976 U. S. Presidential campaign, saying:

> "*One of the constant problems in a political campaign is caused by the fact that things happen in rapid succession. The candidate tends to be one place, most information —the 'good' information— tends to be another. We of course used the Mailbox system to get that information and have it available when the candidate and travel party needed it. It saved us some time, it saved us some money … I think you'd have to say, all in all, that it worked out pretty well.*"[34]

This same email system was also used by I. P. Sharp Associates (IPSA) in Toronto; these two "friendly-rival" APL timesharing companies shared the same code for their email. Leslie Goldsmith at Sharp rewrote it to incorporate greater security. That new facility was then known to the Sharp APL customers as "`666 BOX`".[34,35]

So, with just one or two very limited, in-house systems preceding these early APL mail systems, I can't *quite* claim that APL actually "invented" email, but in a practical sense, maybe that is the case: APL systems offered their users the first "real," widespread, global email service. And all of this was available to APL users before the term "e-mail" had even been coined.[36]

I had purchased a couple of APL terminals to use at home (…yes, terminals, not PCs). Starting in the mid-1970s, I was a guest user of the Sharp APL system, and had an email account there, and I thought, *this is terrific* —we should have something like this. So although I never saw any of their code, I created an email system[37] at IBM based on the ideas that I got from the I.P. Sharp email system.

Email was still quite rare in these days, so following the Sharp system, mine ended up also being another *one of* the first email systems (but obviously *not* the first)— and of course, all written APL.

Now, since I am not an authority on the STSC or IPSA mail systems, let me tell you about the follow-on system that I created.

In support of our timesharing system, I had a great manager at IBM, Bill Davis, who gave me the freedom to work on the projects that I felt needed to be done, so having seen the IPSA mail system, I decided email would be a good thing to have.

It started as a personal project in 1979, and I released it to the world in the summer of 1981… and by "the world" I mean that on Day One, our users in seventeen countries had email. They were all running on our mainframe computers there in Kingston, New York, so we had people in Japan, Australia, Switzerland, Germany, France, England, Argentina, Brazil —all around the world— logging on to the Kingston system to do their daily work, and now for the first time, also to get their email.

Before I put my email system together, I went to Toronto and met with Leslie Goldsmith to ask him what I should be really focusing on if I create my own email system.

He said the *Number One Priority* needs to be *security*— he said that they have competing companies on their system, all using email, and they have to know that it is secure, that nobody else can see their email. So that was one thing that I made a big point of with my system, to ensure that everything was very secure and that nothing could get misdelivered. There were even some new features added to the APL interpreter itself to ensure the security of the messages.

Now, flash forward to today and look at our current email providers: How secure is email these days? If you were signing up for this conference, you might go on to a website with your credit card number to reserve a hotel— that may be fine. But would you send it by email? I would suggest not.

A lot of different *groups* within IBM used our email system, but I was particularly interested in hearing that IBM Headquarters in Armonk became interested in this system and adopted it as their method of sending out all of the IBM Corporate Communications announcements around the world. So all of the new product announcements and executive promotions and IBM earnings statements, and so forth —everything that went up on the (physical) IBM bulletin boards worldwide— came through my email system.

All of these various APL-based email systems predated CompuServe, just in case anyone still remembers that (limited email in 1989 and full support in 1992), AOL email (1991), and Yahoo (1997). It greatly predated Google (1998) and Google's Gmail (2004), and of course it predated personal computers.

It also predated the commercial Internet (early 1990s). Sharp had their own IPSANET; so how did I send email around the world at IBM? Well, IBM had its own global network, called VNET[38] (mid-1970s), some portions of which still exist, but of course the Internet has taken over for a lot of that now.

An attempt to standardize email formats came in September of 1973 from the Internet Engineering Task Force (IETF),[39] but it was then nearly a decade until SMTP (Simple Mail Transfer Protocol) was introduced as the standardized format for email messages, using the now-familiar "@"-symbol in email addresses (August 1982).[40]

All of this came a decade after many APL users had already been using global email.

I'll emphasize again that of course I realize that I was certainly not the first one to create an email system; I don't claim to be, and I don't even claim to be the first one to create it in APL code. I applaud Larry Breed and the others who got the APL world on-line with email.

The point of this discussion is that APL was very early to the game with email. In fact, it was early enough that with the first release of my system I had to explain to people what "email" was. Some people said to me, "*Of course I know what 'mail' is, but what is the 'e' part? …Does that somehow make it different?*"

Yes… yes, it does….

Advanced DNA analysis

Getting into more current topics now, techniques for DNA analysis have been done since the mid-1980s by many people, using many different tools and languages. But imagine what could be accomplished if we were to bring in a person who knows both DNA *and* APL.

Charles Brenner is a renowned expert in DNA analysis and in APL. In his words, he "*leverages the nimbleness of APL to identify criminals, fathers, World Trade Center and tsunami victims, and determine race using DNA in a world of fast-changing DNA identification technology*." He speaks of his DNA works as "*an application tailor-made for APL*."

In April 2013 he entered into a competition through NIST (National Institute of Standards and Technology). Over a hundred entrants competed. Five problems were presented, each having difficult situations for DNA analysis.

In his words, "*One of the competitors (supported by years and millions of dollars of government grants) got four problems right and came close on the fifth, viewing it as a three-person mixture, but in fact it was four.*"

Brenner alone correctly analyzed all five exercises, and furthermore correctly diagnosed one suspect as a mixed race person.

Brenner is doing this with Dyalog APL.[41]

He said, "*We APLers try to be modest but it's not always easy. Sometimes there's just not much to be modest about.*"[42-46]

Section 2: Things that APL Pioneered

Uniformity of math notation

Iverson used to point out that when people say that they had a hard time with math in school, the all-too-common comment is that their math teachers must not have been very good. Ken has suggested that this may be misplaced blame; part of the blame should be on the notation itself, which is saddled with different rules for so many different operations:

a)	-5	`-5`
b)	$5!$	`!5`
c)	$\lvert 5 \rvert$	`|5`
d)	$\sum_{i=0}^{\infty} n$	`+/n`
e)	$16 \overline{)711} \quad 44\,\mathrm{R}\,7$	`16|711`
f)	$\frac{3}{5}$ $\quad 3/5 \quad 3\div 5 \quad 5\overline{)3}$	`3÷5`

In conventional math notation, sometimes the function has to be on the left *(a)*; sometimes it's on the right *(b)*. Sometimes it's on both sides *(c)*. Sometimes we have symbols that are scattered around each other *(d)*. Sometimes we have just one symbol, but it goes part way around the numbers and semi-encloses it… and there may not even be an official symbol for the result that we actually want; in this case, the remainder *(e)*. And for *something as simple as division*, why isn't there one standard way to write it? We have several totally different ways of writing it, and all of them are in very common usage *(f)*. Ken thought this could be improved, so he provided a means of making math notation much more uniform, and even apart from programming —just for someone learning math— I think that this

is a wonderful step forward, and I'm hoping that more attention can be put to this in the future, just as part of basic learning in schools.

Introduction of new symbols

It was pointed out that Ken Iverson is the only mathematician in history to put more than two new symbols into common usage—that is, of course, assuming that we consider APL to be "in common usage." It was Donald McIntyre who made that observation, in his paper, "*From Hieroglyphics to APL*".[47,48]

Confucius has told us that "*Signs and symbols rule the world, not words nor laws.*" (Okay, although I have often seen this quoted, I can't actually vouch for its authenticity. …It might be apocryphal, but it *is* food for thought.)

International language

I see APL as being the only truly *international* programming language. (I am considering only serious work-related languages, not toy languages, and I am grouping APL's derivative languages, such as J and K, with APL.)

For an expression that I write for finding unique values (shown below), *I* is unique (…I try to tell people that, by the way…), then if I transfer this code to a user in another country, he should be able to read the code just as easily. A problem with other languages is, because English has been used so commonly for programming languages, if you are in France or Germany and want to do some programming, often you need to learn some English first so that you know the *keywords* in the operations. And if you are in Greece or Russia or Japan or China, for instance, you may have to gain some familiarity with the Latin alphabet, which we use here, in order to use a programming language. Not so with APL.

And sure, although English is widely used for technical work of all kinds, not *all* programming languages use English; there are languages that are localized for use in other countries, of course — but then they are also trapped in *that* language. APL makes it much simpler — the code should always be the same; only the comments and the object names need to be in local languages.

With APL, whether I'm showing it to a Spanish audience or French audience or Japanese audience, we have *no keywords*, so the code is going to be the same; the comments may differ, but the code itself should always be the same. So I maintain that it's still the only truly international language.

You might say, ah, but how about if I get an error message… that's in English, isn't it? What do we do about that? Under APL2, National Language Translation[49] lets you specify what national language you prefer, and that's what is used for error messages:

```
I←((V⍳V)=⍳⍴V)/V    ⍝ Find unique values                (English)
I←((V⍳V)=⍳⍴V)/V    ⍝ Buscar valores únicos             (Spanish)
I←((V⍳V)=⍳⍴V)/V    ⍝ Trouver des valeurs uniques      (French)
I←((V⍳V)=⍳⍴V)/V    ⍝ 一意の値を見つける                (Japanese)
```

Figure: APL coding with different national languages

```
      1 2 3 + 4 5
LENGTH ERROR
      1 2 3+4 5
      ^     ^

      ⎕NLT←'DEUTSCH'

      1 2 3 + 4 5
LAENGENFEHLER
      1 2 3+4 5
      ^     ^
```

System commands also can be entered in your national language and the responses from the system commands of course will come back in that language. It was released in over a dozen languages: Danish, English, Finnish, French, Canadian French, German, Hebrew, Italian, Japanese (double-byte), Katakana (single byte), Norwegian, Portuguese, Spanish, and Swedish— with more added later. And Help text is also in your selected language.

Here's just a bit of trivia: What do you think the very *first* national languages were that APL2 supported? Each of the languages that were released were specified by knowledgeable representatives from each of those countries, so they were done properly (not just done by code developers sitting down with a dictionary and trying and translate things themselves). Those of us in the development group aren't the experts in those various national languages, so *just for testing*, we had to have something else, and therefore the first alternate languages to be implemented were TexMex and Pig Latin… with TexMex being things like, "*WORKSPACE ESAVED*."

Array processing

Some other languages support arrays, but most commonly just for storage or for some selected operations, and I guess I don't need to tell this audience, but having everything work as array operations was a huge pioneering step for APL. No one else has truly caught up yet.

First desktop Personal Computer

In discussing the first personal computer, some definitions need to be established, because the term "personal computer" hadn't even been used yet when the first of the machines that we'll discuss here were created. So let's define a personal computer to be a desktop (or portable) computer that's primarily intended for one person to use, rather than shared use. These machines were commonly used in a home, although small businesses of course also found them to be a great tool.

In defining what a "personal computer" is, IBM's early-1970s *mainframe* offering called VSPC ("Virtual Storage Personal Computing") doesn't help to clarify matters. (As an aside, VSPC did support APL, but this was a mainframe offering, not a personal computer.)

So, what was the *first* desktop PC? The Apple II is commonly given the credit as being the first desktop computer—but *was it?* I started looking into that, and as a starting point, let's look at the Apple *I* computer,[50,51] which was introduced in 1976.

The Apple I wasn't actually a full computer. It was sold as a hand-built circuit board —just the board— at the Homebrew Computer

Club in Silicon Valley, California.[52,53,54] Steve Wozniak built each of these boards himself, by hand.[55] The Apple I lacked a keyboard, monitor, and storage, so in this first example, somebody got his own keyboard and built a wooden case for it and so forth:

Figure: Apple I circuit board in a wooden case

And here's an example of some work done by a person who put their circuit board into a briefcase, to configure it as a portable computer:

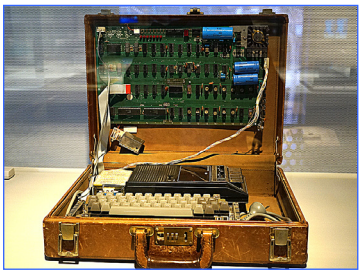


Figure: An Apple I circuit board in a briefcase

But again, this first offering was just a circuit board, not a complete computer. Then, the following year, the Apple II computer came out.[56–59] This was a complete, ready-to-run computer, and it was *said* to be the first desktop personal computer… but again, *was it?*

Figure: Apple II computer

Actually, the first personal desktop computer was the IBM SCAMP machine, built in 1973.[60] "SCAMP" meant "Special Computer APL Machine Portable." This predated other personal computers, and it ran *only* APL. It may look vaguely familiar to some of you:

Figure: The IBM SCAMP machine, open and closed

This was the SCAMP and following are some excerpts[61] from IBM management as they donated the first SCAMP computer to the Smithsonian Institution in Washington, DC, where it remained on display for years. Following are statements made by Paul Friedl, the manager in charge of the development of the SCAMP:

> *"One of the reasons that I am pushing to exhibit SCAMP is that we want everybody to know —contrary to popular opinion— that the micro-computer was not born in January 1975, and it was not a technology that could only be done by teenagers in garages.*

> *"So how did the innovation of SCAMP come about, especially within IBM, which at that time was largely organized around the concept of large centralized computers and timesharing terminals? It happened because IBM had previously created two laboratories in California's Silicon Valley. Both of these labs were given free rein to*

seek and develop new systems and business opportunities for use within several years. These two facilities formed the ideal environment for creating a revolutionary device like SCAMP, the first IBM personal computer. All that was needed was a direction.

*"The direction came in 1972 as Paul Friedl —that's me— a manager in PASC [Palo Alto Scientific Center], conceived the idea for developing SCAMP, a personal, portable IBM computer. To prove feasibility for this idea, I presented IBM executives with an ambitious plan to build SCAMP within six months, and demonstrate it on the desks of IBM executives. At that time, there were no Apple or Microsoft Corporations, and the term "personal computer" had not yet entered into the popular lingo of computing. Since 1972 was a highwater-mark in centralized mainframe computing, when the IBM execs heard my plan, they were stunned, and after a few seconds they spoke out: '**Wouldn't that be something!**' And that phrase became our project motto. …*

"Created by Ken Iverson, APL is a beautiful and extremely powerful programming language which was in use throughout IBM in the 1960s and '70s. APL could do almost anything, but even more importantly, ***APL changed the way you thought.****"*

So there are two myths which the SCAMP developers sought to debunk: The first personal computer was not created by two teenagers in their garage; it was created by IBM. And the first personal computer didn't run BASIC; it ran *only* APL.

PC Magazine called the IBM SCAMP a "*revolutionary*" concept and "*the world's first personal computer.*" And now you might think, this was 1973 and they didn't know what was coming next month or next year, and they didn't really have a perspective on what might be perceived as being revolutionary later on — well, I want to point out, this was a statement they made in their *1983* issue[62] —ten years later— so they had plenty of perspective, and certainly recognized it as being the first personal computer, and indeed, revolutionary.

After the SCAMP prototype, but before the IBM PC, and for that matter also *before* the Apple II was the IBM 5100:[63,64]

Figures: The IBM 5100 Computer

The 5100 could be purchased as a complete system, with an external tape drives and a printer. Below, here's an ad showing a strong man,

holding his breath and grunting while he is trying to hold this 50-pound machine up in the air just long enough for the photo to be taken:

Figure: Advertisement for the IBM 5100

This magazine ad[65] said, "*Now you can have a computer right on your desk.*" That had never been done before. It features "*a typewriter-like keyboard*" (you probably wouldn't sell a lot of machines today by telling people that); "*a 1024-character display screen*" (which is another way of saying that it's all character-based, so there's no graphics); "*integrated tape drive*" (again, not a big selling point today, but certainly was then); and "*16k characters of memory, expandable up to a maximum of 64k characters.*" APL could easily have supported more memory, and the designers wanted to provide that for APL, but the Fortran implementation that they were using only supported 64k, and the planners insisted that they be kept equal. Too bad.

The ad goes on: "*Also available is a communications feature, which allows a 5100 to be used as a terminal.*" Our department got two of these machines when they came out, and they were wonderful as terminals. One of the nice things was that we weren't limited to 134 baud(!) anymore on the 2741 dial-up lines; now we could go to the *high-speed* dial-up lines and run this baby at *300 baud*— pretty amazing… for its day. It was released in three models:

- Model A was APL only
- Model B was Basic only
- Model C was a Combination machine: APL and Basic

And as an aside, just for fun, do a Google search sometime for IBM 5100 and John Titor.[66] He has been discussed on many forums as a time traveler from 2036 who has gone back to 1975 to get an IBM 5100 with APL to solve problems in 2036. And apparently they also want that highly-coveted 370 emulator that let them run APL.SV on the 5100. (…If he is at this meeting, by the way —*as he should be*— please let me know; there are a few things I'd like to discuss with him…).

At 50 pounds, the 5100 may seem pretty heavy by today's standards, but remember that it replaced what would previously have been *half a ton* of equipment. Its predecessor was the IBM/1130.[67,68]

And no, that's not on a desk, it *is* the desk:

Figures: Versions of the IBM/1130

The machine on the left is an 1130 with the *expanded memory* installed: There's an additonal three-foot cube bolted onto the left side, giving you an extra *8k* of memory—good for its time. And yes, it ran APL. The first time I saw APL running was on an 1130:

```
      )33
KYM SIGNED ON

                A P L \ 1 1 3 0

      +/⍳100
5050
      (⍳5)∘.+⍳5

    2    3    4    5    6
    3    4    5    6    7
    4    5    6    7    8
    5    6    7    8    9
    6    7    8    9   10
      )OFF
SIGNED OFF
```

Figure: A brief sample APL session on an IBM/1130

It was Larry Breed and Charles Brenner who ported APL to the IBM/1130.[69]

~

There were a few other machines which need to be included in these discussions. A Canadian company called Micro Computer Machines brought out the MCM/70 computer; it was demonstrated in 1973, but not available until the end of 1974.[70–75] It had an odd one-line plasma display, and was further notable in that it ran *only* APL.

Figures: The MCM/70 APL computer

Although it was predated by IBM SCAMP machine, due to the limited availability of the SCAMP, the MCM/70 can make a reasonable claim as being the first commercially-available personal computer.

Although it was meant to be plugged in, the MCM/70 was also unusual in that it had a built-in battery, likely the first PC to offer that. So even if you somehow didn't consider it to be the first PC, it can certainly claim the title of "the first truly portable computer."[76]

This brought about another first: Zbigniew Stachniak, the developer of the MCM/70, told this story:[77]

> *"In August 1973, MCM sent Ted Edwards to the APL Congress in Copenhagen with a prototype of the MCM/70. What was*

unusual about that MCM/70 was that it was mounted in an attaché case and was operating on batteries. Edwards was not only able to board the plane with this unusual device but also reviewed his presentation using that MCM/70 during the flight to Copenhagen. This constitutes another 'first' for MCM: the first portable computer operated during a flight. And, of course, that 'laptop' was running MCM/APL. The MCM/70 story was picked up by the Danish daily Politiken on 1973-09-28."

As an aside, if you haven't read the stories on Roger Hui's "*APL Quotations and Anecdotes*" webpage from Jsoftware,[78] I encourage you to do so. There is a wonderful collection of interesting and informative APL stories there.

The MCM/70 also broke ground in another area: Back in 1975, this was the machine that introduced APL to the Soviet Union. The Computing Center of the Academy of Sciences of the USSR purchased several of the MCM/70 machines.[79–82]

Regarding other machines which have sometimes been called "the first PC," you might hear that the Altair 8800[83,84] was the first PC. It actually came a few months later, in 1975:

And then there was the intriguing Ampere WS-1, an APL laptop produced by Nippon-Shingon in Tokyo.[85–89] Although it isn't a contender for the first PC, it was a very early laptop— probably even predating that *term*, as they called it a "knee-top" machine. This very snazzy-looking *APL-only* laptop was released in 1985, with all of the sexy, curvaceous styling of a sleek sports car:

Figures: Ampere WS-1 APL laptop

And there's a reason for that: The Ampere WS-1 APL laptop was designed by Kumeo Tamura, who also designed the exterior lines of the *iconic* 1970 Datsun 240z sports car:[90,91]

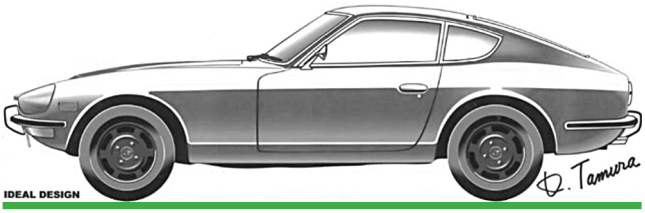


Figure: Design concept for the 1970 Datsun 240z

…So, sure, might as well make use of that design a second time.

Unfortunately, the Ampere WS-1 failed its FCC (Federal Communications Commission) certification for the U. S. marketplace, so it was never able to be sold in the United States.

So, regarding the first desktop PCs, here was the progression:

Timeline:

- IBM SCAMP — APL only . . July 1973
- MCM/70 — APL only . . . . Nov 1974
- Altair 8800 . . . . . Jan 1975
- IBM 5100 — APL and Basic . Sept 1975
- Apple I circuit board . . . . . July 1976
- Commodore PET . . . . . Jan 1977
- Apple II computer . . . . . Apr 1977
- Radio Shack TRS-80 . . . . . Aug 1977
- IBM PC (5150) . . . . . Aug 1981
- Ampere WS-1 — APL only . . Nov 1985

~

All in all, APL played a larger role in the early PCs than you might have expected. So why wasn't APL considered as the default language for PCs? Well, to some extent, it was.

Prior to forming Microsoft, Bill Gates had already known about APL. Gates met with Ian Sharp, Eric Iverson, and Bob Bernecky at IPSA in the very early days of PCs, and talked about using APL.[92] In February of 1976, Bill Gates announced via "*An Open Letter to Hobbyists*" that Micro-Soft (as it was named back then) was working on "*writing 8080 APL and 6800 APL.*"[93] According to Zbigniew Stachniak from MCM, Gates stated that "*Equivalence with the 5100 was my goal.*"[94] Stachniak further stated, "*It was not until 1979 that Microsoft announced its APL-80 interpreter for the Intel 8080 and Zilog Z80 platforms. It was to be out in April 1979 and compatible with IBM's APL.SV software. But in the end the Microsoft APL-80 proved to be vaporware and by the early 1980s, it was Microsoft's BASIC and not APL that was installed on the majority of personal computers.*" So in the end, the Microsoft APL products never saw the light of day.

This is reminiscient of the VHS-versus-Beta format wars. We can argue about which system is better, but there is no doubt that marketing plays a very important role in public acceptance.

Personally, I think that it's a *good* thing that Microsoft *didn't* pursue APL further— because if they had created their own version of APL, I have to think that it might have ended up being vastly altered from what we are accustomed to today, and due to volume, might have become the accepted standard for APL. I am pleased with the APL offerings and continued development that we are seeing these days, from multiple companies.

Early pocket calculators

Going down to the very small end, in about 1972, at a time when pocket calculators were very new, Walt Niehoff at IBM in Endicott, New York worked on creating a handheld APL calculator. (To steal a line from the SCAMP folks, "*Wouldn't that be something!*") I remember seeing the breadboarded prototype for this, which filled several feet of a rack cabinet in his lab in Endicott. But I'm sorry to report that it did not get released as a product.

Just as an aside, the HP-35 calculator,[95] launched in early 1972, was the first pocket calculator to be released with scientific functions, able to replace a slide rule; other calculators of the time had only four functions. I am told that Hewlett Packard designed that calculator using APL\1130.[96]

Supercomputers

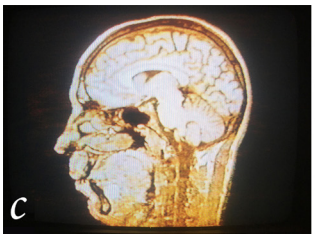
At the other end of the scale are the supercomputers. So *what is* a supercomputer, and what *drives* it? In the mid-1980s, IBM viewed a supercomputer as being their top-end mainframe machines which were also fitted with the IBM Vector Facility;[97,98,99] which turned their largest mainframe into a supercomputer. The Vector Facility was supported only by Fortran and APL, and that made APL very popular with IBM. And it was our *Numerically Intensive Computing* Group in Kingston that created that.

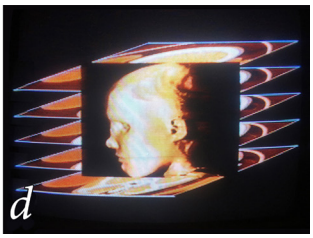
And one of the really beautiful points about using APL with a Vector Facility is that no one's code has to change at all — no tweaks, no changes, and no recompiling… come Monday morning, even old code just runs *faster*. Fortran couldn't offer that.

APL in Cancer Research

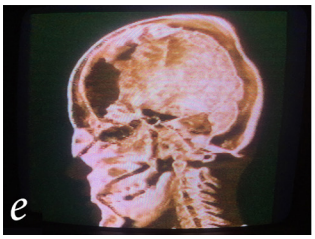
In 1989, Prof. Dr. Hans-Peter Meinzer, at the German cancer-research center DKFZ ("Deutsches Krebsforschungszentrum") in Heidelberg, Germany, analyzed Magnetic Resonance Images (MRI) and Computer Tomography (CT) scans with APL using an IBM 3090 mainframe with the Vector Facility.

Previously, the many images that are created by the MRI (*a*) and CT scans (*b*) had to be viewed individually (*c*), but using the array-processing power of APL, Dr. Meinzer was able to produce highly-detailed 3-D medical images (*d*):





Using sophisticated techniques of anatomic segmentation and semi-transparent raytracing (*e*), Dr. Meinzer was able to construct 3-D rotating views (*f*) of a human head of a patient who had a cancerous tumor:



Figures: MRI and CT scans and viewing of images

A demonstration video of this, called "*Viewing the Invisible*," was shown at the APL90 Conference in Copenhagen. John Mizel, Michael Van Der Meulen, and myself, in IBM's *Numerically-Intensive Computing* group in Kingston, New York collaborated with Dr. Meinzer in creating this video.[100] (I apologize for the low-quality images shown here; Dr. Meinzer's procedure produced very high-quality images; however, *these* images are the result of viewing a quarter-century-old VHS tape on poor equipment.)

The goal was to provide physicians with better visualization techniques, giving them

the means of seeing details which *might otherwise go undetected until the moment of actual surgery*.

Thanks to the array-handling power of APL and the increased processing power of the Vector Facility, Dr. Meinzer's work far surpassed any medical imaging that had ever been done prior to his work.

Complex TV graphics

In the 1980s, all of the major U. S. TV networks suddenly came on the air with fancy, glistening, highly-reflective 3-D floating type and images that fly in from the side with swooshing sounds and so forth.

These days, we are so used to seeing fancy 3-D logos and images that we scarcely notice them anymore, but back in the early 1980s, they were stunning. And they all seemed to arrive at the same time. I remember thinking back then, how is it that all of the television networks have come up with that *simultaneously?*

Figures: Early TV logos and images made with APL

All of these logos appeared at about the same time because they were all created by the same person: It was Judson Rosebush,[101] using APL and Fortran. His company was Digital Effects, Inc.[102] He was creating the graphics by actually dialing into the STSC APL system in Bethesda from New York City, and doing it all via dial-up timesharing (at 1200 baud). I was told that one month his timesharing bill hit a quarter of a million dollars, and at that point he reportedly decided that it may be time to get their own mainframe.[103]

By the way, regarding the spinning globe that was used for the NBC Nightly News intro, a little-known fun-fact is that their first pass of the animation inadvertently had the globe spinning in the wrong direction, so it had to be reworked.[103]

In addition to creating the network logos with APL, they also created a lot of TV commercials for companies around the world using APL-generated graphics. Digital Effects became so closely associated with this work that its name became a noun for this type of work.[102] Additionally, Digital Effects did some work on movies; in *Xanadu*, complex scene transitions were created by Judson Rosebush with APL.[104]

Judson Rosebush's company, Digital Effects, was also one of four companies that collaborated to create the movie *Tron* in 1982.[105] This movie used APL heavily for creating the visual effects. But although *Tron* was a groundbreaking film, it was never presented with an Oscar for technical achievement or special effects. Why? … Because at the time, the Academy felt that

Tron had "cheated" by using computers to create the scenes.[106]

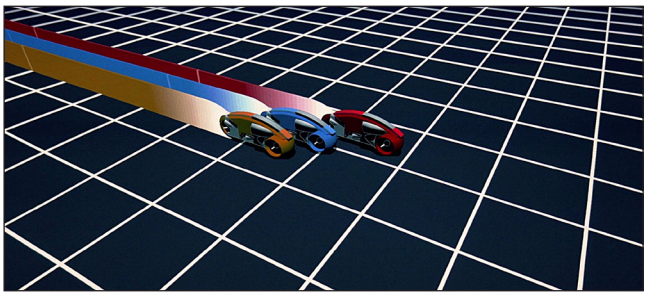


Figure: Graphics from the TRON movie

Robust word processing

Part of my job has always been to write documentation, so word processing has always been important to me. Finding the tools to do that has been difficult at times. It's so simple now — we sit down at any PC, and we have Microsoft Word or whatever other tool you want. These days, I have Adobe InDesign on my machine; that's a *wonderful* tool for text processing and page layout. But before Microsoft Word came out in 1983… before WordPerfect appeared, back in 1979… and even before WordStar and TEX appeared in 1978 — how did anyone do it? Well, back then there was [fanfare] *The APL Text Machine.*[107]

The APL Text Machine was originally called APL text (pronounced "appletext"). The version that was first used in Yorktown Heights was designed by Adin Falkoff, and Mordecai (Morty) Zryl was the APL programmer. Later enhancements were added by Elizabeth Llanso. Maintenance was handled by Don Orth starting in 1974.

The APL Text Machine was used to produce the first APL\360 manual in late 1966 or early 1967. Agnes Carlin used it to produce several of Ken Iverson's books, e.g., *Elementary Algebra*.

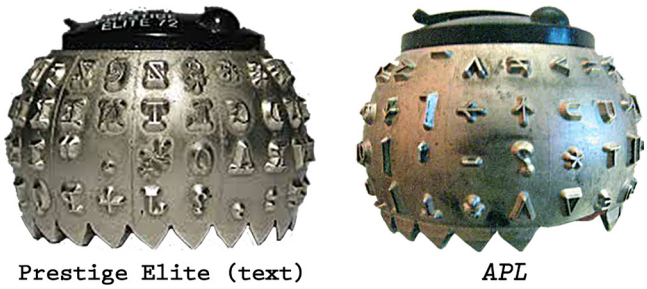
The APL Text Machine was started in the late 1960s, and formalized by the 1970s. It was put out into the APL Public Library at that point, so that any of our users could use it. It really solved a lot of problems for us. With the text machine we could —*for the first time*— have the printer "automatically" switch fonts back and forth. When we were writing APL documentation, we *had* to have that. For other people who were writing purely text documents, perhaps this wasn't as necessary and you could get by with lesser tools, but we needed something robust.

By the way, I put the word "automatically" in quotes when I spoke of font switching, because of course we were using the IBM Selectric typing element[108,109,110] (or "type ball") back then, and so it wasn't *really* automatic. We had to manually switch the type ball but the point is that *we could do it now* — the APL Text Machine finally provided the commands for controlling it.

Our IBM 3211 system printer was set up with an "APLFULL" print train, which contained the APL font plus an upright caps and lowercase text font, so the APL Text Machine could print on that device without any manual intervention, but the print quality wasn't good enough for publications; it was really just meant for daily reports. For publications, I chose the Selectric terminal with a high-quality single-pass Mylar ribbon.

As an aside, I went onto the Web to get a couple of pictures of Selectric type balls

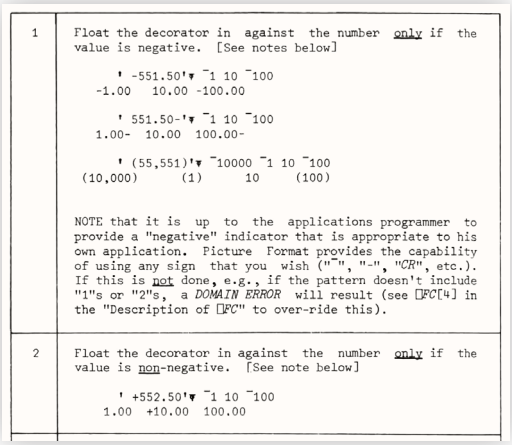
for the presentation, and I thought that it was pretty funny that the picture of the APL type ball was *of course* missing a couple of teeth. That seemed to always be the case; remember that? It would then misprint some of the characters because the printer couldn't rotate it properly:



Prestige Elite (text) APL

Figure: Selectric type balls

These days, we don't even think about the complexities involved in changing fonts; we just switch fonts back and forth within Word and then print it on pretty much any printer. But it wasn't always that easy.



1 Float the decorator in against the number only if the value is negative. [See notes below]

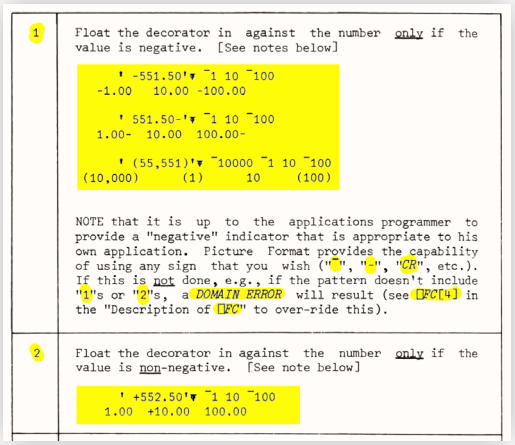
```
      ' -551.50'▼ ¯1 10 ¯100
   -1.00   10.00 -100.00

      ' 551.50-'▼ ¯1 10 ¯100
    1.00-  10.00  100.00-

      ' (55,551)'▼ ¯10000 ¯1 10 ¯100
 (10,000)      (1)      10      (100)
```

NOTE that it is up to the applications programmer to provide a "negative" indicator that is appropriate to his own application. Picture Format provides the capability of using any sign that you wish ("¯", "-", "CR", etc.). If this is not done, e.g., if the pattern doesn't include "1"s or "2"s, a *DOMAIN ERROR* will result (see ⎕FC[4] in the "Description of ⎕FC" to over-ride this).

2 Float the decorator in against the number only if the value is non-negative. [See note below]

```
      ' +552.50'▼ ¯1 10 ¯100
    1.00  +10.00  100.00
```

Here's a portion of a page from a newsletter that I put together back in 1980. This was a Reference Issue of *Jot Dot Times*[111] that showed how to use, in this case, Picture Format. It doesn't look like it would be that hard to type this up, *except* that we realize that we need to switch fonts back and forth — and not just within blocks of text, but buried within paragraphs, too, so manual (physical) cut-and-paste was just far too involved to consider:



1 Float the decorator in against the number only if the value is negative. [See notes below]

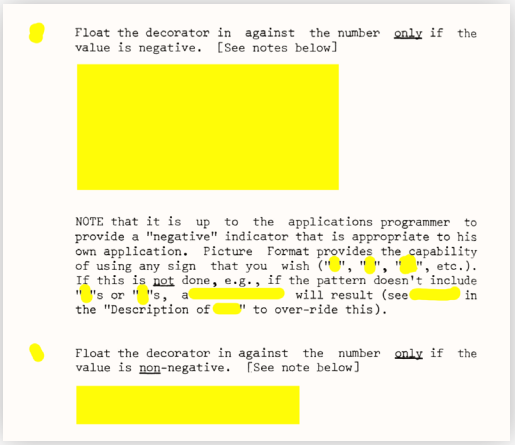
```
      ' -551.50'▼ ¯1 10 ¯100
   -1.00   10.00 -100.00

      ' 551.50-'▼ ¯1 10 ¯100
    1.00-  10.00  100.00-

      ' (55,551)'▼ ¯10000 ¯1 10 ¯100
 (10,000)      (1)      10      (100)
```

NOTE that it is up to the applications programmer to provide a "negative" indicator that is appropriate to his own application. Picture Format provides the capability of using any sign that you wish ("¯", "-", "CR", etc.). If this is not done, e.g., if the pattern doesn't include "1"s or "2"s, a *DOMAIN ERROR* will result (see ⎕FC[4] in the "Description of ⎕FC" to over-ride this).

2 Float the decorator in against the number only if the value is non-negative. [See note below]

```
      ' +552.50'▼ ¯1 10 ¯100
    1.00  +10.00  100.00
```

The highlighted part is in the APL font and the rest is in a text font. The APL Text Machine let me enter codes to indicate all of the formatting that I wanted, including font switching. But having it stop and make me change type balls back and forth, perhaps many times in each paragraph, would be far too tedious, so it had an easier approach: For each *page*, it would print *just the text portion* first…

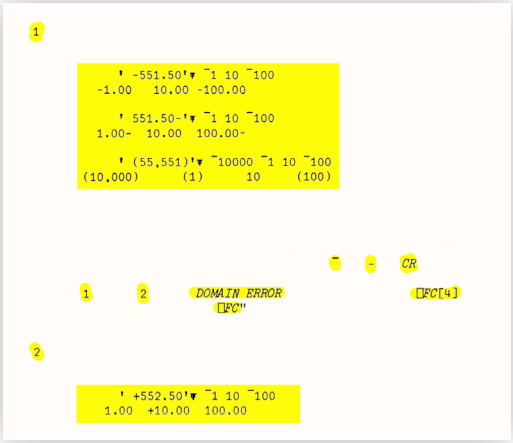


Float the decorator in against the number only if the value is negative. [See notes below]

NOTE that it is up to the applications programmer to provide a "negative" indicator that is appropriate to his own application. Picture Format provides the capability of using any sign that you wish ("¯", "-", "CR", etc.). If this is not done, e.g., if the pattern doesn't include "1"s or "2"s, a will result (see in the "Description of " to over-ride this).

Float the decorator in against the number only if the value is non-negative. [See note below]

…and I would then roll the page back up to the top, change the type ball, and press Enter, and it would print *just the APL portion* of that page:



1

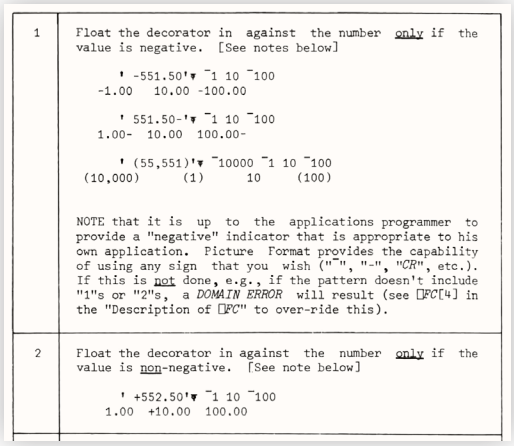
```
      ' -551.50'▼ ¯1 10 ¯100
   -1.00   10.00 -100.00

      ' 551.50-'▼ ¯1 10 ¯100
    1.00-  10.00  100.00-

      ' (55,551)'▼ ¯10000 ¯1 10 ¯100
(10,000)      (1)      10     (100)
```

¯ - CR

1 2 DOMAIN ERROR ⎕FC[4]
⎕FC"

2

```
      ' +552.50'▼ ¯1 10 ¯100
    1.00  +10.00  100.00
```



1 Float the decorator in against the number only if the value is negative. [See notes below]

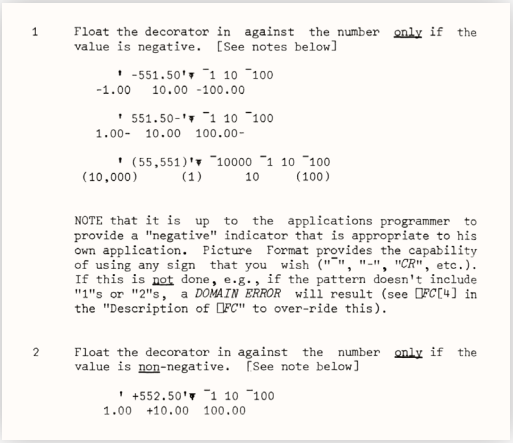
```
      ' -551.50'▼ ¯1 10 ¯100
   -1.00   10.00 -100.00

      ' 551.50-'▼ ¯1 10 ¯100
    1.00-  10.00  100.00-

      ' (55,551)'▼ ¯10000 ¯1 10 ¯100
(10,000)      (1)      10     (100)
```

NOTE that it is up to the applications programmer to provide a "negative" indicator that is appropriate to his own application. Picture Format provides the capability of using any sign that you wish ("¯", "-", "CR", etc.). If this is not done, e.g., if the pattern doesn't include "1"s or "2"s, a *DOMAIN ERROR* will result (see ⎕FC[4] in the "Description of ⎕FC" to over-ride this).

2 Float the decorator in against the number only if the value is non-negative. [See note below]

```
      ' +552.50'▼ ¯1 10 ¯100
    1.00  +10.00  100.00
```

At this point you would *hopefully* end up with the two parts lined up. Yes, there were indeed some spoiled pages, but this is a photo of a page from the actual newsletter that I published, so they certainly could be made to line up:



1 Float the decorator in against the number only if the value is negative. [See notes below]

```
      ' -551.50'▼ ¯1 10 ¯100
   -1.00   10.00 -100.00

      ' 551.50-'▼ ¯1 10 ¯100
    1.00-  10.00  100.00-

      ' (55,551)'▼ ¯10000 ¯1 10 ¯100
(10,000)      (1)      10     (100)
```

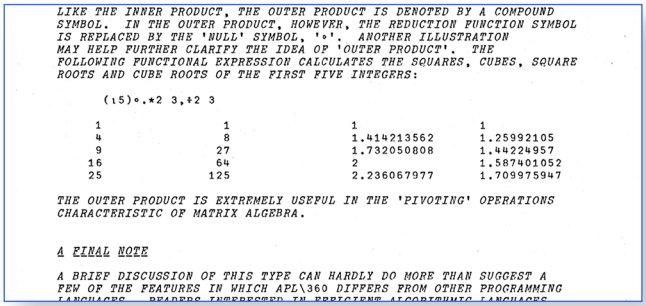
NOTE that it is up to the applications programmer to provide a "negative" indicator that is appropriate to his own application. Picture Format provides the capability of using any sign that you wish ("¯", "-", "CR", etc.). If this is not done, e.g., if the pattern doesn't include "1"s or "2"s, a *DOMAIN ERROR* will result (see ⎕FC[4] in the "Description of ⎕FC" to over-ride this).

2 Float the decorator in against the number only if the value is non-negative. [See note below]

```
      ' +552.50'▼ ¯1 10 ¯100
    1.00  +10.00  100.00
```

So, we're done now, right? Well, no, not quite. Unlike more modern products like MS Word, the APL Text Machine didn't have any provision for drawing the lines around tables… *not* because that couldn't be programmed into it, but because back then, we didn't have any *printers* that could draw the lines—everything was character-based. So the lines had to be drawn in manually, very carefully, using a technical-illustrator's pen (sometimes producing, of course, some more spoiled pages):

It was quite a task to create APL manuals years ago, but thanks to the APL Text Machine, we finally had the tools to do it. But there were APL manuals before the APL Text Machine was available, so you may ask, how did *they* do it? The answer is: poorly… very poorly:



LIKE THE INNER PRODUCT, THE OUTER PRODUCT IS DENOTED BY A COMPOUND SYMBOL. IN THE OUTER PRODUCT, HOWEVER, THE REDUCTION FUNCTION SYMBOL IS REPLACED BY THE 'NULL' SYMBOL, '∘'. ANOTHER ILLUSTRATION MAY HELP FURTHER CLARIFY THE IDEA OF 'OUTER PRODUCT'. THE FOLLOWING FUNCTIONAL EXPRESSION CALCULATES THE SQUARES, CUBES, SQUARE ROOTS AND CUBE ROOTS OF THE FIRST FIVE INTEGERS:

```
      (⍳5)∘.*2 3,*2 3

 1    1    1          1
 4    8    1.414213562  1.25992105
 9   27    1.732050808  1.44224957
16   64    2            1.587401052
25  125    2.236067977  1.709975947
```

THE OUTER PRODUCT IS EXTREMELY USEFUL IN THE 'PIVOTING' OPERATIONS CHARACTERISTIC OF MATRIX ALGEBRA.

A FINAL NOTE

A BRIEF DISCUSSION OF THIS TYPE CAN HARDLY DO MORE THAN SUGGEST A FEW OF THE FEATURES IN WHICH APL\360 DIFFERS FROM OTHER PROGRAMMING

This is an APL manual from 1960s,[112] *prior* to the APL Text Machine. Headings were just done with underscored letters and the text was all caps, because the APL type ball didn't have lowercase. Remember that we needed to have text and APL symbols intermixed, and prior to the APL Text Machine, the tools for doing that just didn't exist.

Commands within text

One of the things that came out of this formatting work is the introduction of *readable commands within text*. The APL Text Machine let you type your text and at the beginning of the line, if you put in a delimiter

character, like a slash, what came after that was a command rather than text. We are very used to that now with a lot of facilities we use, but where did this come from? Well, IBM Script was released in 1970, and it had readable commands within text like that. TEX was released in 1978, and it is certainly based on this concept (and by the way, I was told that Donald Knuth modeled TEX in APL[113]). SGML (Standard General Markup Language) was formalized in 1986, and its cousin, HTML (HyperText Markup Language) was released in 1993. All of these facilities used this approach of embedding commands within text. But before any of these were available, we had the APL Text Machine.

A patent claim was presented to IBM in the 1980s claiming ownership of the concept of embedding readable commands within text. That claim found its way to Adin Falkoff who then contacted me, and we were able to show that SGML and TEX and others were predated by the APL Text Machine.

On-line documents in court

From 1969 to 1982, the United States Government pursued a major lawsuit against IBM regarding the unbundling of software and services[114,115]; this 13-year antitrust lawsuit was eventually dropped in 1982. But while it was in process, it was a huge "you bet your company" undertaking, where the IBM legal team knew that the entire future of the IBM Corporation was dependent upon the evidence that they could present in court about how IBM was conducting its business.

With so much documentation that could be called up in court, they wanted to have *on-line* access to documents in court, so that any of perhaps millions of memos and other documents could be called up instantly. These days, of course, any large case would do that, but back then, it was a brand-new concept to bring a terminal into the courtroom to call up documents pertinent to the case.

They were doing all this with APL, and in fact, were doing it on our Kingston APL timesharing system. Chuck Norcutt and I got involved to assist with it. One of problems was that the code had been written by a summer student and it was too slow —that old story— and so of course, we said, "Okay; let's look at the code." *Gasps* from the legal team! You can't look at the code — it's *Registered IBM Confidential*, and it's very, very sensitive, so absolutely not — *no one* can look at the code. Well then, the big challenge became, *how do you speed up code when you're not allowed to look at it?* Is that even possible?

Surprisingly, the answer to that is *yes*. It turned out that an outgrowth of that court case was an APL application that Chuck Norcutt worked on in my department (with a little help from me), called "Pareto." Vilfredo Pareto (1848–1923)[116] was an Italian economist, and one of his findings was that around 80% of the result of almost anything is generally based on about 20% of its components. Applying his precepts to code, you turn on the timer and then just run your application normally; its timing analysis will show you the 20% of the code that's taking most of the time, and then you can fix *just that portion* — the rest of it isn't nearly as consequential. That timing facility later became the APL *Application Performance Analysis*[117,118] tool that is now *built-in* to the APL2 interpreter.

If APL is so good, where are the derivative languages?

Morgan Stanley's Aplus[12] (or "A+") was derived from other APL systems— some people call Aplus a derivative language, and some people simply consider it to be another APL dialect, just as Sharp APL and IBM APL2 differ somewhat, but both are APL dialects.

Ken Iverson's J language[119,120] (1990) was a direct outgrowth of APL, and it answers the question, *if you had to do it all over again, what would you change?* He couldn't simply choose to change the APL language because of the impact to the installed user base. J was a brand-new language, to get around that issue. And now K[121,122] (1993) is a further derivative, and Kx[123] (1993) grew from that.

Additionally, ALGOL 68 was greatly influenced by APL, and even used the APL type ball for coding. Matlab and Mathematica are also systems which derived from APL.

Detouring for just a moment, perhaps you have seen "*Zippy the Pinhead*" comics in the newspapers. Shown below is the "*Symbol-Minded*" strip[124] where Zippy is trying to figure out what "J" is doing.

And now let's talk about one more derivative of APL…

Spreadsheets

My final topic here is on spreadsheets: Where do spreadsheets come from? If you look up spreadsheets, for instance in Wikipedia,[125] you'll find some discussion of IBM's *Financial Planning and Control System*[126] from 1976. It was used in 30 countries around the world, and is considered to be an early spreadsheet. It was written in APL, but the users didn't see APL — the underlying language was hidden from them.

The next step in this is credited to Dan Bricklin.[127,128] He was an APL user[129] and he reportedly admired the way that APL could display a matrix of numbers on the screen—but wouldn't it be convenient if you could run the cursor up and just overtype a value that you wanted to change, rather

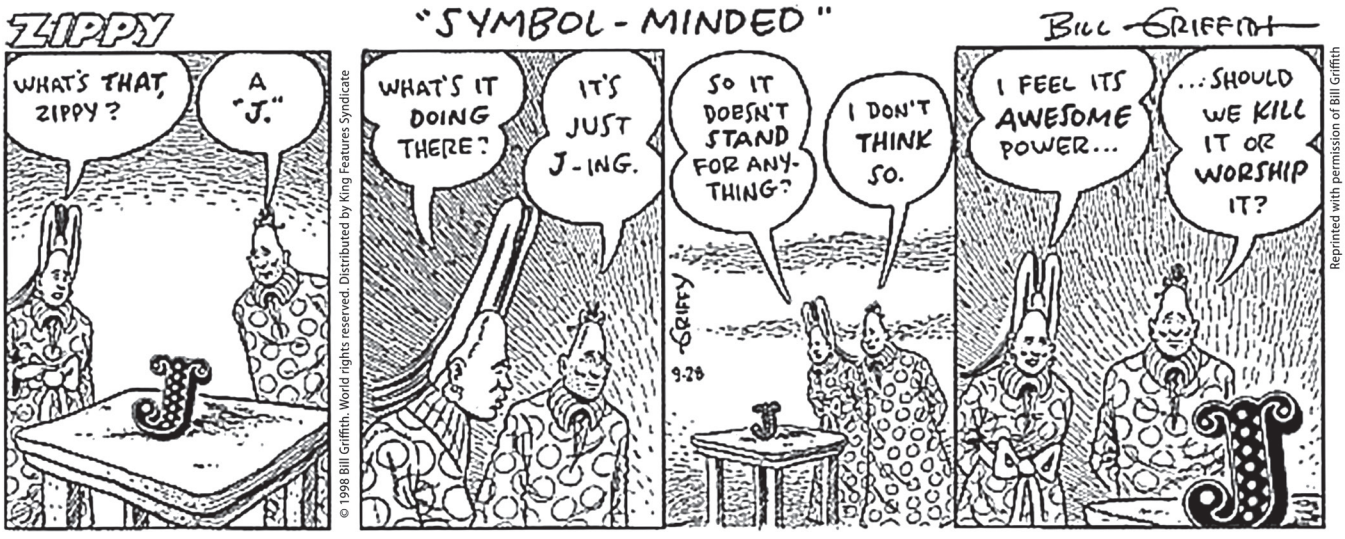


Figure: 'Zippy the Pinhead' contemplates J [124]

than having to index into an array. And of course you would then want to have it recalculated automatically. So while he was a student at the Harvard Business School, he developed VisiCalc[130,131] to solve some of what he saw as limitations in APL. This was in 1979, and that one application is widely credited with fueling the rapid growth of the personal computer industry. From that came Lotus 123 and Excel and others. He was presented with the Grace Murray Hopper Award in 1981[22] for VisiCalc. I will point out that spreadsheets are therefore a direct outgrowth of APL.

Concluding remarks

APL's mark extends far beyond just its notation. APL has had some major influences on many portions of the computer industry, and in many cases, we don't even recognize features as being related to APL anymore, but APL's influences are all around us, even in unexpected places. ▪

Author: Jon McGrew, IBM (retired); Kingston, New York; phone: +1-845-338-5558, email: McGrew@TypeMatters.com

Jon McGrew has specialized in working with various flavors of the APL programming language throughout most of his career, since 1971. His focus has been on the design and development of the language itself (as opposed to the underlying implementations), and on applications. In 1981, McGrew developed one of the early widespread email systems, with service to seventeen countries; it was, of course, implemented entirely in APL.

He has spent his career working as both an APL programmer and a technical writer, and is the author of widely-used programming texts ("*An Introduction to APL2*,"[132–135] and the IBM APL periodical "*Jot Dot Times*"[37,111,136]). For many years, McGrew hosted international APL conferences twice each year around the world, as the Chairman of the ITL (Interdivisional Technical Liaison) Committee on the APL language.[31] He received the IBM *President's Award* for service to the IBM APL community. He was Team Leader for APL applications and customer support within IBM. Later, he was an APL applications writer within IBM's *Numerically Intensive Computing* group, providing support for APL on supercomputers. He then joined IBM's *APL Language Development* group under IBM's *Scientific Languages*, where he did applications programming for APL products. On the side, he volunteered for twelve years as the Production Editor for SIGAPL's *"APL Quote Quad"* under ACM (the Association for Computing Machinery), the programming community's major professional society.

Later, McGrew worked in the Aplus Development and Support group[13] at Morgan Stanley Dean Witter, supporting the Aplus language[12] and developing and teaching classes for their in-house version of APL.

McGrew is the 2001 recipient of the industry-wide Iverson Award ("*The Kenneth E. Iverson Award for Outstanding Contribution to the Development and Application of APL*"),[137] presented at the international APL conference, held at Yale University.

Cited references, further information, and notes

[1] *A Programming Language*, by Kenneth E. Iverson, 1962: John Wiley & Sons, Inc.; ISBN-10: 0471430145; ISBN-10: 0471430145; Computer History Museum: http://www.softwarepreservation.org/projects/apl/Books/APROGRAMMING%20LANGUAGE/view

[2] *A Programming Language*, by Kenneth E. Iverson, 1962: John Wiley & Sons, Inc., ISBN:0-471430-14-5: http://www.jsoftware.com/papers/APL.htm

[3] *A Programming Language*, by Kenneth E. Iverson, 1962: John Wiley & Sons, Inc.; ISBN-10: 0471430145; ISBN-10: 0471430145; Amazon: https://www.amazon.com/Programming-Language-Kenneth-Iverson/dp/0471430145

[4] *IBM Systems Journal*, Volume 30, Issue 4, December 1991 (Ray Polivka, guest editor; Jon McGrew, typography and production): http://ieeexplore.ieee.org/xpl/tocresult.jsp?isnumber=5387434

[5] *A Personal History of APL*, Michael S. Montalbano: http://ed-thelen.org/comp-hist/APL-hist.html#PersonalHistory

[6] APL Design Group photo is from the Computer History Museum in California, http://www.computerhistory.org/atchm/wp-content/uploads/2012/10/iverson_team.jpg

[7] APL Design Group photo, outtake version: http://lathwellproductions.ca/wordpress/2010/04/21/unsung-jedi-warrior/

[8] ACM Conference photo is from the collection of Raymond P. Polivka

[9] IBM 2741 terminal: https://en.wikipedia.org/wiki/IBM_2741

[10] IBM 2741 terminal: http://www.textfiles.com/bitsavers/pdf/ibm/27xx/GA24-3415-3_2741_Data_Terminal_Aug72.pdf

[11] IBM APL2: http://www-03.ibm.com/software/products/en/apl2

[12] Morgan Stanley's Aplus language: http://www.aplusdev.org

[13] Morgan Stanley's Aplus development and support team: http://www.aplusdev.org/Develop/devTeam.html

[14] *I P Sharp Associates and the Telephone Monopolies*, Ian P. Sharp: http://rogerdmoore.ca/INF/EIPSPTTa.html

[15] Packet switching: https://en.wikipedia.org/wiki/Packet_switching

[16] Packet-Switching History, Roger D. Moore: http://rogerdmoore.ca/PS/

[17] IPSANET: https://en.wikipedia.org/wiki/IPSANET

[18] IPSANET Documents, Roger D. Moore: http://rogerdmoore.ca/INF/

[19] Grace Murray Hopper Award: https://en.wikipedia.org/wiki/Grace_Murray_Hopper_Award

[20] Grace Murray Hopper Award, ACM: http://awards.acm.org/hopper

[21] RADM Grace Hopper: https://en.wikipedia.org/wiki/Grace_Hopper

[22] Grace Murray Hopper Award Recipients: https://en.wikipedia.org/wiki/Grace_Murray_Hopper_Award#Recipients

[23] Grace Murray Hopper Award text: https://en.wikipedia.org/wiki/Richard_H._Lathwell

[24] Grace Murray Hopper Award annoucement letter for Richard Lathwell, from ACM Awards Committee Chairman, American Institute of Physics, May 31, 1973: https://aprogramminglanguage.files.wordpress.com/2010/02/hopperdad.jpg

[25] *APL\360: An Interactive Approach*, John Wiley and Sons, Inc., Leonard Gilman and Allen J. Rose, 1970-01-01, ISBN-13:9780471300205, 335 pgs: http://www.softwarepreservation.org/projects/apl/Books/GillmanAndRose

[26] Early time-sharing systems and APL, Eugene McDonnell, *The Socio-Technical Beginnings of APL*, APL Quote-Quad, Volume 10, Number 2, 1979-12: http://www.jsoftware.com/papers/eem/socio1.htm#early

[27] *Management of APL public libraries*, Marguerite A. Boisvert, Systems Analyst, APL '79 Proceedings of the international conference on APL: part 1, pp 381–384: http://dl.acm.org/citation.cfm?doid=800136.804491

[28] Self-replicating APL expressions: http://gopher.quux.org:70/Archives/usenet-a-news/NET.lang.apl/82.04.26_uwvax.343_net.lang.apl.txt

[29] First computer virus, per BBC: http://news.bbc.co.uk/2/hi/technology/8366703.stm

[30] Elk Cloner virus: https://en.wikipedia.org/wiki/Elk_Cloner

[31] Proceedings of the IBM ITL (Interdivisional Technical Liaison) Committee on the APL language, Computer History Museum, http://www.computerhistory.org/collections/catalog/102734222

[32] *Internet in its infancy*, ActewAGL Retail, ABN 46 221 314841, https://web.archive.org/web/20110227151622/http://www.actewagl.com.au/Education/communications/Internet/historyOfTheInternet/InternetOnItsInfancy.aspx

[33] Mailbox, *The STSC Story: It's About Time*: https://www.youtube.com/watch?v=BSkxr6rQU0Y

[34] 666 BOX, *APL Quotations and Anecdotes*, Roger Hui: http://www.jsoftware.com/papers/APLQA.htm#666box

[35] 666 BOX was the I. P. Sharp mail system: https://en.wikipedia.org/wiki/I._P._Sharp_Associates

[36] *Will You Love Electronic Mail or Hate It?* Computer Decisions, Volume 11, Hayden Publishing Company, December 1979, pg 47

[37] *The APL Jot Dot Times*: An IBM in-house newsletter created by Jon McGrew; "Introducing the APL Mailbox," Summer 1981, 64 pages

[38] IBM VNET: https://en.wikipedia.org/wiki/IBM_VNET

[39] *Standardizing Network Mail Headers*, IETF Network Working Group, RFC 561, September 1973: https://tools.ietf.org/html/rfc561

[40] *Simple Mail Transfer Protocol (SMTP)*, IETF Network Working Group, RFC 821, August 1982: https://tools.ietf.org/html/rfc821

[41] Dyalog APL: https://www.dyalog.com/

[42] *There's DNA Everywhere*, SIGPLAN Chapter on Array Programming Languages, Charles Brenner: http://www.sigapl.org/CharlesBrennerDNATalkIntro.php

[43] Charles Brenner, DNA Identification Technology and APL: http://dna-view.com/DNAtechID.htm

[44] *There's DNA Everywhere— an Opportunity for APL*, Charles Brenner: http://video.dyalog.com/Dyalog14/?v=oXlP3r6PzeE

[45] Charles Brenner (mathematician): https://en.wikipedia.org/wiki/Charles_Brenner_(mathematician)

[46] Charles Brenner, *A Conversation With Charles Brenner; A Math Sleuth Whose Secret Weapon Is Statistics*, NY Times, Science section, By Claudia Dreifus, Aug. 8, 2000: http://www.nytimes.com/2000/08/08/science/conversatipon-with-charles-brenner-math-sleuth-whose-secret-weapon-statistics.html

[47] *Language as an Intellectual Tool: From Hieroglyphics to APL*, Donald B. McIntyre, IBM Systems Journal, Volume 30, Issue 4, December 1991, pp 554–581: http://ieeexplore.ieee.org/document/5387447/?arnumber=5387447

[48] *Language as an Intellectual Tool: From Hieroglyphics to APL*, Donald B. McIntyre (1991), IBM Systems Journal. 30 (4): 554–581: http://domino.research.ibm.com/tchjr/journalindex.nsf/e90fc5d047e64ebf85256bc80066919c/9c834f5a16efa82085256bfa00685c72!OpenDocument

[49] National Language Translation: APL2 Programming: Language Reference, SH21-1061-01, February 1994, pg 314; http://publibfp.boulder.ibm.com/epubs/pdf/h2110611.pdf

[50] Apple I computer: http://www.computerhistory.org/revolution/personal-computers/17/312/2312

[51] Apple I computer: https://en.wikipedia.org/wiki/Apple_I

[52] Homebrew Computer Club: https://en.wikipedia.org/wiki/Homebrew_Computer_Club

[53] Homebrew Computer Club: http://www.computerhistory.org/revolution/personal-computers/17/312

[54] Homebrew Computer Club: http://www.oldcomputers.net/applei.html

[55] *Steve Wozniak Debunks One of Apple's Biggest Myths*: https://www.youtube.com/watch?v=pJif4i9NRdI

[56] Apple II computer: http://www.computerhistory.org/revolution/personal-computers/17/300

[57] Apple II computer: https://en.wikipedia.org/wiki/Apple_II

[58] Apple II computer: https://en.wikipedia.org/wiki/Apple_II_series

[59] Apple II computer: http://oldcomputers.net/appleii.html

[60] SCAMP: http://www-03.ibm.com/ibm/history/exhibits/pc/pc_1.html

[61] SCAMP presentation to the Smithsonian Institute, Paul Friedl, IBM Corporation *SCAMP - The 1st IBM personal computer; the missing link in the PC's past!* : https://www.youtube.com/watch?v=L_BWL7vCaa0

[62] *World's first PC*: PC Magazine, Vol. 2, No. 6, November 1983, Ziff-Davis Publishing, "SCAMP: The Missing Link in the PC's Past?", pp 191–197: https://books.google.com/books?id=q8fwTt09_MEC&pg=PA196&lpg=PA196&dq=PC+Magazine,+November+1983+scamp

[63] IBM 5100: http://www-03.ibm.com/ibm/history/exhibits/pc/pc_2.html

[64] IBM 5100: https://en.wikipedia.org/wiki/IBM_5100

[65] IBM 5100 ad: http://bluefaqs.com/2009/09/35-vintage-tech-ads/

[66] John Titor and the 5100: https://en.wikipedia.org/wiki/John_Titor

[67] IBM 1130 computer: https://www-03.ibm.com/ibm/history/exhibits/1130/1130_intro.html

[68] IBM 1130 computer: https://en.wikipedia.org/wiki/IBM_1130

[69] *How We Got to APL\1130*, Larry Breed, Vector (British APL Association), 22 (3),August 2006, ISSN 0955-1433: http://vector.org.uk/art10001190

[70] MCM/70, *How Toronto invented the PC, then forgot about it*: http://spacing.ca/toronto/2015/04/15/toronto-invented-pc-forgot/

[71] *The MCM/70 Microcomputer*, by Zbigniew Stachniak (developer): http://www.xnumber.com/xnumber/MCM_70_microcomputer.htm

[72] *Inventing the PC: The MCM/70 Story*, book, by Zbigniew Stachniak (developer): https://www.amazon.com/Inventing-PC-MCM-70-Story/dp/0773538526

[73] *Inventing the PC: The MCM/70 Story*, review, David C. Brock: https://muse.jhu.edu/article/476817

[74] *The Making of the MCM/70 Microcomputer*, by Zbigniew Stachniak, IEEE Annals of the History of Computing, April–June 2003, pp. 62–75: https://ia801300.us.archive.org/10/items/MCM01203059/MCM-01203059.pdf

[75] MCM/70, *Core 4.1*, Computer History Museum, September 2003, pp 6–12: http://s3data.computerhistory.org/core/core-2003.pdf

[76] *Introduction of the MCM/70, the First Truly Portable Computer & the First Truly Usable Microcomputer System*, Jeremy Norman, History of Information: http://www.historyofinformation.com/expanded.php?id=4806

[77] MCM/70 used on a plane, Zbigniew Stachniak, from *APL Quotations and Anecdotes* by Roger Hui: http://www.jsoftware.com/papers/APLQA.htm#MCM1973

[78] *APL Quotations and Anecdotes*, Jsoftware, compiled and edited by Roger Hui: http://www.jsoftware.com/papers/APLQA.htm

[79] *МСМ/70 и другие: из истории персональных компьютеров* ("MCM/70 and others: from the history of personal computers"), 08.09.2013: http://itc.ua/articles/mcm-70-i-drugie-iz-istorii-personalnyih-kompyuterov/

[80] APL in Russia, *Russia: a future great APL power?*, Erkki Juvonen, ACM SIGAPL APL Quote Quad - Russian focus issue, Volume 22 Issue 2, Dec. 1991, pp 1–2: http://dl.acm.org/citation.cfm?doid=130647.130649&CFID=797566137&CFTOKEN=80570900

[81] APL in Russia, *Nuclear power plant diagnostics in APL*, Alexander O. Skomorokhov, Institute of Physics and Power Engineering, 1 Bondarenko Square, Obninsk, Kaluga Region 249020, USSR, APL '91 Proceedings of the international conference on APL '91, pp 289–300: http://dl.acm.org/citation.cfm?id=114087&CFID=797566137&CFTOKEN=80570900

[82] APL in Russia, *The SovAPL award for excellence in APL*, Alexander Skomorokhov, Chairman of SovAPL, The Russian Chapter of ACM SIGAPL, ACM SIGAPL APL Quote Quad, Volume 31 Issue 1, 09/01/2000, pp 29–30: http://dl.acm.org/citation.cfm?id=570511&CFID=797566137&CFTOKEN=80570900

[83] Altair 8800, Popular Electronics, Jan 2, 1975, pp 33–38: http://www.americanradiohistory.com/Archive-Poptronics/70s/1975/Poptronics-1975-01.pdf

[84] Altair 8800, Setup and Users Manual, vintage computer information, published by Briel Computers, July 2010: http://www.hackersinformation.com/uploads/1/9/1/6/19169525/micromanual.pdf

[85] Ampere WS-1: https://fi.wikipedia.org/wiki/Ampere_WS-1 (Finnish)

[86] *The Amazing Ampere WS-1* (1985), Norbert Landsteiner: https://plus.google.com/+NorbertLandsteiner1/posts/VJYZrHZg7Zy

[87] Ampere WS-1, Classic Tech, Vintage computers and related technology, Ampere Inc. (Tokyo, Japan): https://classictech.wordpress.com/computer-companies/ampere-inc-tokyo-japan/

[88] Ampere WS-1: http://www.old-computers.com/museum/computer.asp?st=1&c=66

[89] Ampere WS-1, *The Computer Chronicles— Japanese PCs (1984)*: https://www.youtube.com/watch?v=rbh1XP4kCT4

[90] *The Guy Who Designed the Datsun 240Z Also Designed This Fascinating Laptop*, Jalopnik: http://jalopnik.com/5929191/the-guy-who-designed-the-datsun-240z-also-designed-this-fascinating-laptop

[91] *Datsun 240Z Exterior Designer "Kumeo Tamura"*: https://www.youtube.com/watch?v=ju7RfHhTI1c

[92] Bill Gates meeting at IPSA, *APL Quotations and Anecdotes*: http://www.jsoftware.com/papers/APLQA.htm#Bill_Gates

[93] *An Open Letter to Hobbyists*, Homebrew Computer Club Newsletter, Vol 2, Issue 1, February 3, 1976, pg 2, by Bill Gates; DigiBarn Computer Museum: http://www.digibarn.com/collections/newsletters/homebrew/V2_01/homebrew_V2_01_p2.jpg

[94] *APL: Good for the Brain*, article by Bill Gates, Electronics Today International Magazine (Canada), Vol. 3 No. 3, March 1979, by Steve Braidwood (editor), March 1979

[95] Hewlett-Packard HP-35 scientific calculator: https://en.wikipedia.org/wiki/HP-35

[96] Hewlett-Packard HP-35 scientific calculator designed with APL\1130: Personal email discussion with Richard Lathwell, 2017-01-04

[97] IBM Vector Facility: http://www-03.ibm.com/ibm/history/exhibits/mainframe/mainframe_FS9000.html

[98] *IBM Vector Facility: Large Systems Technical Support*, IBM 3090 Processor Complex: Planning and Installation Reference, GG66-3090-01, November 1987, http://chiclassiccomp.org/docs/content/computing/IBM/Mainframe/Hardware/System/GG66-3090-01_3090ProcComplexPlanningInstallationRef_Nov87.pdf

[99] IBM Vector Facility: https://en.wikipedia.org/wiki/IBM_3090#Vector_facility

[100] Prof. Dr. Meinzer, DKFZ: Descriptions taken fromvideotape liner notes created by Jon McGrew; video program created by John M. Mizel, editing by Jon McGrew and Mike Van Der Meulen; IBM Numerically-Intensive Computing, February 1990. The video was created for the IBM NYC exhibit entitled, "100 Years of Computing."

[101] Judson Rosebush, Wikipedia: https://en.wikipedia.org/wiki/Judson_Rosebush

[102] Digital Effects, Inc., Wikipedia: https://en.wikipedia.org/wiki/Digital_Effects_(studio)

[103] Digital Effects, Inc., from comments made at the presentation by Judson Rosebush at the I. P. Sharp Conference, Toronto, 1982

[104] Xanadu movie trailer, 1980: https://www.youtube.com/watch?v=WNcUv1q2JAs&t=23s

[105] Tron movie, 1982, *Film Opening at Flynn's Arcade*: https://www.youtube.com/watch?v=P7LclSGFOFg&t=12s

[106] Tron movie, 1982, *The Making of Tron*, Director's comments: https://www.youtube.com/watch?v=pr2LvJUI6ZY&t=1s

[107] APL Text Machine history: Personal email discussion with Richard Lathwell, 2017-01-04

[108] Selectric typing element: http://www-03.ibm.com/ibm/history/exhibits/vintage/vintage_4506VV2122.html

[109] Selectric typing element: https://en.wikipedia.org/wiki/IBM_Selectric_typewriter

[110] Selectric typing element: http://www.computerhistory.org/collections/catalog/102662795

[111] *The APL Jot Dot Times*: An IBM in-house newsletter created by Jon McGrew; "Special Reference Issue," Fall 1980, 98 pages, Computer History Museum: http://www.softwarepreservation.org/projects/apl/Papers/APLJotDotTimes

[112] Pre-APL Text Machine manual, *APL\360 Language and Time Sharing System*, 1968: http://bitsavers.trailing-edge.com/pdf/ibm/apl/APL_360_Features.pdf

[113] Donald Knuth usage of APL: Personal email discussion with Richard Lathwell, 2017-01-04

[114] IBM antitrust lawsuit; Wikipedia, History of IBM: https://en.wikipedia.org/wiki/History_of_IBM#1969:_Antitrust.2C_the_Unbundling_of_software_and_services

[115] IBM antitrust lawsuit; New York Times, February 15, 1981, U.S. vs. I.B.M.: http://www.nytimes.com/1981/02/15/business/us-vsibm.html

[116] Vilfredo Pareto: https://en.wikipedia.org/wiki/Vilfredo_Pareto

[117] APL2 timing facility, *TIME— Application Performance Analysis*: APL2 User's Guide, IBM, SC18-7021-23, 1994/2017, pp 287–288; http://publibfp.boulder.ibm.com/epubs/pdf/c187021n.pdf

[118] APL2 timing facility, *TIME Workspace*: APL2 User's Guide, IBM, SC18-7021-22, 1994/2017, pp 547–549; http://publibfp.boulder.ibm.com/epubs/pdf/c187021n.pdf

[119] J language: http://www.jsoftware.com/

[120] J language: https://en.wikipedia.org/wiki/J_(programming_language)

[121] K language, Vector, Vol. 10 No. 1, July 1993, pp 74–79: http://archive.vector.org.uk/art10010830

[122] K language, Wikipedia: https://en.wikipedia.org/wiki/K_(programming_language)

[123] Kx Systems, 1993: https://kx.com/

[124] *Zippy the Pinhead*, © 1998, Bill Griffith; published on September 28, 1998; Used here through the permission of the artist: https://en.wikipedia.org/wiki/Zippy_the_Pinhead

[125] Spreadsheet, Wikipedia: https://en.wikipedia.org/wiki/Spreadsheet

[126] Spreadsheet; *IBM Financial Planning and Control System (FPCS)*, Wikipedia: https://en.wikipedia.org/wiki/Spreadsheet#IBM_Financial_Planning_and_Control_System

[127] Dan Bricklin's website: http://bricklin.com

[128] Dan Bricklin, Wikipedia: https://en.wikipedia.org/wiki/Dan_Bricklin

[129] Dan Bricklin on APL; *Bricklin on Technology*, Dan Bricklin, John Wiley and Sons, Inc., 2009, ISBN-13: 978-0470402375, pp 370, 401–404, 423; http://www.barnesandnoble.com/w/bricklin-on-technology-daniel-bricklin/1102658682?ean=9780470402375

[130] VisiCalc background, from Dan Bricklin's website: http://bricklin.com/visicalc.htm

[131] VisiCalc: https://en.wikipedia.org/wiki/VisiCalc

[132] *An Introduction to APL2* (Installed User Program), IBM, Jon McGrew, June 1982, SB21-3039-0

[133] *An Introduction to APL2*, APL2 Version 1 Release 2 (Program Product), IBM, Jon McGrew, December 1985, SH20-9229-1, Computer History Museum: http://www.computerhistory.org/collections/catalog/102679862

[134] *An Introduction to APL2*, APL2 Version 1 Release 2 (Program Product), IBM, Jon McGrew, December 1985, SH20-9229-1, Bitsavers: https://archive.org/details/bitsavers_ibmaplSH20toAPL2Dec85_12392164

[135] *An Introduction to APL2*, APL2 Version 2 Release 1 (Program Product), IBM, Jon McGrew, March 1992, SH21-1073-00: http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss?CTY=US&FNC=SRX&PBL=SH21-1073-00

[136] *The APL Jot Dot Times*: An IBM in-house newsletter created by Jon McGrew; "Special Security Issue," Spring 1985, 234 pages, Computer History Museum: http://www.softwarepreservation.org/projects/apl/Brochures/APLJotDotTimes

[137] Iverson Award: https://en.wikipedia.org/wiki/Iverson_Award

~