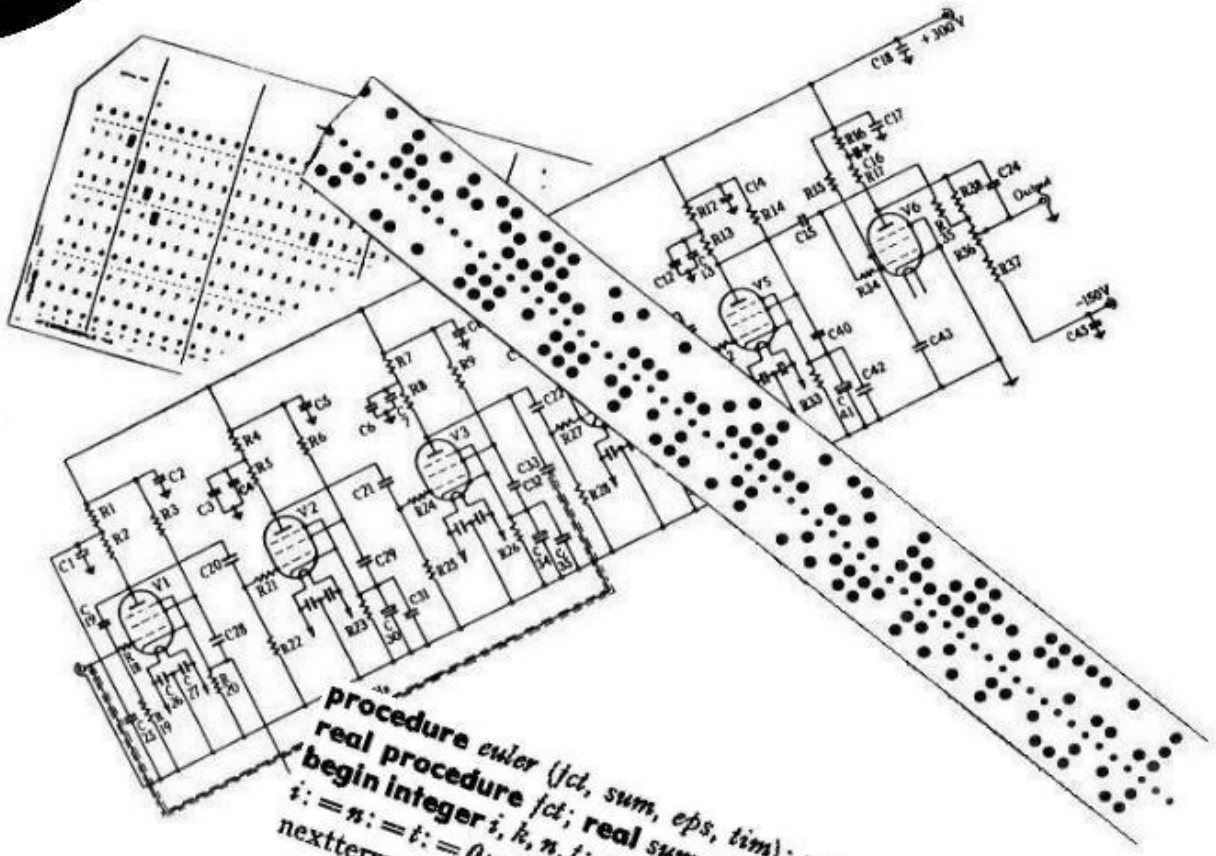
Issue Number 50 **Spring 2010**

bcs The Chartered Institute for IT

sciencemuseum

mosi MUSEUM OF SCIENCE & INDUSTRY



```
procedure euler (fct, sum, eps, tim); value eps, tim; integer tim;
real procedure fct; real sum, eps;
begin integer i, k, n, t; array m[0:15]; real mn, mp, ds;
    i:=n:=t:=0; m[0]:=fct(0); sum:=m[0]/2;
nextterm: i:=i+1; mn:=fct(i);
    for k:=0 step 1 until n do
        begin mp:=(mn+m[k])/2; m[k]:=mn; mn:=mp end means;
    if (abs(mn) < abs(m[n])) ∧ (n < 15) then
        begin ds:=mn/2; n:=n+1; m[n]:=mn end accept
    else ds:=mn;
    sum:=sum+ds;
    if abs(ds) < eps then t:=t+1 else t:=0;
    if t < tim then go to nextterm
end euler
```

RESURRECTION

The Bulletin of the Computer Conservation Society

Computer Conservation Society

Aims and objectives

The Computer Conservation Society (CCS) is a co-operative venture between the British Computer Society (BCS), the Science Museum of London and the Museum of Science and Industry (MOSI) in Manchester.

The CCS was constituted in September 1989 as a Specialist Group of the British Computer Society. It is thus covered by the Royal Charter and charitable status of the BCS.

The aims of the CCS are:

- ◊ To promote the conservation of historic computers and to identify existing computers which may need to be archived in the future,
- ◊ To develop awareness of the importance of historic computers,
- ◊ To develop expertise in the conservation and restoration of historic computers,
- ◊ To represent the interests of Computer Conservation Society members with other bodies,
- ◊ To promote the study of historic computers, their use and the history of the computer industry,
- ◊ To publish information of relevance to these objectives for the information of Computer Conservation Society members and the wider public.

Membership is open to anyone interested in computer conservation and the history of computing.

The CCS is funded and supported by voluntary subscriptions from members, a grant from the BCS, fees from corporate membership, donations, and by the free use of the facilities of both museums. Some charges may be made for publications and attendance at seminars and conferences.

There are a number of active Projects on specific computer restorations and early computer technologies and software. Younger people are especially encouraged to take part in order to achieve skills transfer.

Resurrection

The Bulletin of the

Computer Conservation Society

ISSN 0958-7403

Number 50

Spring 2010

Contents

ALGOL 60 – a Binary Star

Dik Leatherdale

ALGOL 60 was, perhaps, the most influential development in programming languages of all time. Other languages have been more successful in themselves. Certainly there are many whose use has been more widespread. But none have had the influence on thinking, none have demonstrated the great leap forward, none have received so much attention as ALGOL 60. The vast majority of programming languages in the last half century have owed a huge debt to ALGOL 60. The ALGOL 60 Report stands as a colossus of its kind. It was, and still is, a thing of beauty – a work of art.

So, in tribute to those clear-thinking pioneers of 50 years ago, we dedicate this 50th issue of *Resurrection* issue to ALGOL 60.

Not, of course, that ALGOL 60 was without fault. Nor, indeed, did it escape criticism. The lack of any defined input/output facilities was the point most frequently made. That led to the creation of many incompatible I/O systems which, in practice, severely restricted program portability. But for me, and with the benefit of 50 years of hindsight, ALGOL's major shortcoming was that it was a paper tape-orientated language in what was soon to become a punched card world. And reserved word-based dialects such as Burroughs' ALGOL were arguably not "strict" ALGOL.

But the fierceness of the arguments, the multiplicity of "improved" ALGOLs – ALGOL-W, Pascal, CPL etc. – only serve to demonstrate the extent to which the basic concept was accepted as essentially correct.

Today ALGOL is but rarely used. But it is far from forgotten. It lives on in its progeny – C, C++, Delphi, Ada, Java, C# and so many others.

I can surely do no better than to leave the last word to Tony Hoare.

Here is a language so far ahead of its time that it was not only an improvement on its predecessors but also on nearly all its successors.

Society Activity

Changes to the Committee – *David Hartley*

In December 2009 the CCS Committee reviewed its own membership in the context of the new constitution agreed in October. The membership is largely unchanged, and will be subject to the scrutiny of the next AGM which will be held in September or October later this year. We did, however, lose two long service members namely David Anderson of Portsmouth University, who has been a member since 2003, being variously London meetings coordinator and web site editor. Also, David Glover has represented The National Archives since 2005. We thank both of them for their past services.

North West Group

After 17 years Ben Gunn has decided to retire as Secretary. Chairman David Hartley paid tribute to Ben's long service to the Society - *Ben's record is exceptional. The Society is hugely grateful for his dedication and we all wish him well*. Meanwhile Member Gordon Adshead has agreed to take over the running of the Group. We all welcome him and look forward to an equally long and useful association.

Exhibition to mark the Turing Centenary – *Simon Lavington*

The CCS is working with the Science Museum on an idea for a temporary exhibition in 2012 to mark the centenary of the birth of Alan Turing. The project is in its early stages but the plan is for a small public outreach exhibition, lasting from June 2012 to June 2013, in the Maths and Computing Gallery at the Science Museum. The exhibition has received initial approval from the Science Museum Executive Group with an indicative budget of between £50,000 and £60,000 and the possibility of additional external sponsorship. All this will require further approval and sign-off from the Executive Group, Director and Trustees as the project develops.

The exhibition could be themed round four chronological phases of Alan Turing's involvement with the theory and practice of digital computation. Briefly, these could be:

1. Theory at Cambridge and Princeton: the Universal Turing Machine.
2. Bletchley Park and cryptanalysis: a vital interlude.
3. NPL and the Pilot ACE: hardware and architecture.
4. The Manchester years: software and applications.

At its meeting on 18th February the CCS Committee set up a small working party under Simon Lavington to progress the project. Technical advice is being sought from a number of researchers whose work has rendered them expert in various aspects of Alan Turing's work towards the development of digital computers and their application. There is clearly a need to identify relevant artefacts. A meeting has been held with the Bletchley Park Trust and areas of collaboration have been identified.

The timescale for the project is likely to be as follows.

Phase 1 (to be completed by end-November 2010): the CCS and others will finalise the intellectual content of the exhibition, which may include an indication of main themes, list of desired artefacts, simulators, models, etc.

Phase 2 (to start December 2010 and be completed before end-June 2011): the Museum's team will organise some form of audience evaluation, to mould the intellectual content into a presentation suitable for public outreach.

Phase 3 (to be completed in time for the opening in June 2012): the Museum's exhibition team undertakes the implementation of the whole project.

Anyone who has ideas to contribute is invited to contact Simon Lavington at *lavis@essex.ac.uk.*

Elliott 803 Project – *Peter Onion*

Not much to report this time. The 803 itself is continuing to work well. The only problem has been that over Christmas the tape reader seemed to become less reliable and eventually became unusable. After much cleaning and adjustment it was still unreliable, so the decision was made to try and set up one of the several spare readers. The one chosen was in good condition mechanically and initial tests with a tape loop showed all five data channels were producing correct output signals. However the sprocket-hole channel was producing a very low amplitude output pulse. The fault was quickly diagnosed and the cause identified as the output transistor's load resistor which had become disconnected from the PCB track. The "new" reader has been performing well for the last few weeks.

A Calcomp 565 plotter has been retrieved from the TNMOC store and work will begin soon to restore this to operation. The 565 was supplied by Elliotts as an option for the 803. Sadly our 803 doesn't have the optional instruction decoding circuits installed in the processor. Nor

do we have the original interface unit to connect the plotter to the processor. However, a successful test has been conducted with an alternative scheme that requires minimal connections to be made to the processor. This will allow the plotter to respond correctly when the 803 executes the appropriate output instructions.

Elliott 401 Project – *Chris Burton*

The moratorium on accessing the machine continues. Remediation of asbestos-containing materials will take place by Museum conservation staff after similar work to the Pegasus. An alternative to the "firebar" resistors in the power supply will be sought. Procedures, roles and methods of working will have to be worked out with the Museum management before we will be allowed to operate the machine again.

Pegasus Project – *Len Hewitt*

After the discovery of asbestos in the Pegasus fuses and an alternator connection box very little happened as far as the Pegasus Project is concerned. In December the official asbestos reports were released and arrangements were made by the Science Museum for meetings to discuss the return of Pegasus to active service.

There were a number of items concerning Health and Safety, asbestos, and Project practices to be discussed. The good news from the CCS point of view is that it is being discussed and the Museum is leading the way forward. Chris Burton and Peter Holland attended the two meetings and I attended one for some time via a telephone conference call.

The asbestos problem with the alternator gasket has been dealt with by sealing. The asbestos-bearing fuses will be replaced with modern asbestos-free versions of otherwise identical type.

Various procedures have been put forward to the Museum on how the repairs should be completed. Before they can take place the Museum has a number of Health and Safety issues that have been considered and will be implemented Until that is done repairs to the damaged wiring cannot take place. From our photographs and circuit investigation and discovery of three blown fuses, we are hoping that a replacement of burnt wires and a damaged tag block should be all that is necessary. There may be other components damaged in the PSU but we are hopeful that the blown fuses should have protected the rest of the circuitry.

Harwell Dekatron Computer Project – *Tony Frazer*

There was considerable damage to the wiring on racks 4 and 5 (used for the Dekatron stores). The send/receive guide pulse and power connections run the length of the racks to all stores and across to the extension on rack 5. On both racks, these wires span exposed areas between the bays like the strings of a harp. Many had thus been broken off and others had suffered weakening near the solder joints. All such wiring on rack 5 has been replaced with single strand PVC insulated plate wire using the same insulation colour scheme. Fortunately, the laced loom running up the centre U-points appeared undamaged and has been left in place. Similar repairs will be necessary to rack 4. Spare U-points have been located on equipment in D-block which will be used to replace at least two broken units on rack 4.

In addition, 38 wires linking racks 4 and 5 which were hard-wired between the two racks had been cut at some point in order to separate racks 4 and 5. This wiring has been replaced on the rack 5 side and two Jones connectors have been attached using existing threaded holes on rack 4.

Three of the five store units have been cleaned and repopulated with tested Dekatrons - one store has been populated with GC10B dekatrons (much brighter than the original GC10As). This has almost exhausted our supply of spare GC10Bs, so we will be looking out for more of these. Three more stores and the rack will be fully populated. We have sufficient GC10Bs to populate the accumulator and one option may be to populate half the stores in each of two store boxes with GC10Bs to facilitate demonstration of the machine to visitors (i.e. bright dekatrons in both send and receive stores). I have also modified a Russian OG-4 Dekatron (which appear to be readily available) as a possible substitute by changing the pinout using a spare octal base as a hard-wired adaptor.

Eddie continues to clean and adjust relays - of particular note are relays where spring sets have been deliberately kinked rather than adjusting the travel to make contact. Eddie has completed work on the arithmetic rack and is now working through the store units on rack 5.

Johan has made a start on straightening the casing and internal frame of one of the stores (with all dekatrons removed!). It had evidently been impacted causing crumpling of the interior metalwork and possible damage to components. Similar repairs were necessary on two of the other stores.

ICT 1300 Project – *Rod Brown*

Work continues to build the high speed interface. This is to allow us to capture the data from the magnetic tape data transfer unit, to complete the software recovery aims of this project.

The project was very pleased to be invited to present its story and the history of the 1300 series to the CCS on the 15th of October 2009. This allowed us to continue the communication about the 1301 resurrection project, which is why the website at *www.ict1301.co.uk* was created.

All of this activity is running against a backdrop of growing doubt about the longer term future of the project. There is pressure from Ashford Borough Council to impose and collect non-domestic rates from the building which houses the project instead of being included in the domestic rates of the farm house. This would place the running costs beyond our current budget!

We are fighting this issue as decisions which are being made at a national level by the Valuation Office Agency (an executive agency of H.M. Revenue and Customs) could affect not only the 1301 project, but set a precedent for any other restored or conserved computers held by any organisation or individual in UK within garages, sheds, barns etc.

The BCS Legal Department have provided us with a supportive statement which has been included in the letter to the Minister responsible for the Valuation Office Agency.

Three representatives from the Valuation Office Agency visited the project on 17th February. The decision they reach will determine the final outcome and be used as a precedent for any other non-domestic equipment on which other Councils may seek to levy Rates.

The BBC has visited Flossie the ICT 1301 in its home in deepest Kent and interviewed its custodians. The result was broadcast on 18th February.

Bombe Rebuild Project - *John Harper*

In *Resurrection 49* I said that we had completed all planned construction. With hindsight this statement has turned out to be premature. Soon after *Resurrection 49* went to press the team decided that they wanted the extra three German surface fleet wheels in the Typex. This, as many will recall, is the British Typex set up to operate as a German three wheel Enigma machine. This construction is now well under way and when complete will allow us the decrypt any three wheel message and demonstrate a complete Intercept to Ultra process for any combination of eight wheels.

Training of Bombe operators is now our major activity. Recruitment has got off to a good start but we still need more people interested in operating our Bombe at weekends. One thing that had not been anticipated is that many volunteers are only able to be at Bletchley about once a month. Therefore we need more volunteers and need more time from existing team members for training. Having said that, the recruits we have so far are of high calibre and they are learning faster than we had expected. This will reduce the training time for all involved. Training documentation has been produced and is being refined as we go along. This too will all help when it comes to training later volunteers.

One thing that is satisfying is that the machine is standing up well to new potential operators. We thought that mistakes might be made by trainees and that the Bombe might suffer but this concern is so far unfounded. The machine continues to do all that is asked of it and virtually all the problems encountered can be put down to human error.

We now have a new activity. Letchworth - First Garden City Heritage Museum is staging a year-long exhibition entitled *Letchworth Garden City and the Second World War*[1] from 20th March 2010 – 19th March 2011. The British Tabulating Machine Company and the Bombe design and manufacture will figure prominently. As part of this the Bletchley Park Trust is loaning display material and our team is involved in lending one of the Props Bombes produced specifically for filming of *Enigma*. Many will recall that this was a full length film featuring Kate Winslet and other well known actors. The Props Bombe we have chosen is slightly worse for wear and needs a good tidying up before it moves to Letchworth. This is already underway. When it arrives we have to refit parts left off during transit.

Our website is still at *www.jharper.demon.co.uk/bombe1.htm*

[1] *www.gardencitymuseum.org/exhibitions/letchworth_garden_city_and_second_world_war*

News Round-Up

It is our sad duty to report the death at the age of 91 of Andrew Booth, one of the UK's computer pioneers. Starting in 1945 at Birkbeck College London, he built a series of small, low cost machines. His (then) revolutionary ambition was to build a machine cheap enough that every university could afford one! This, as it turned out, satisfied a need which nobody else working in the field had identified, prefiguring the economic rationale of minicomputers by more than a decade. The APEC was adopted by the British Tabulating Machine company as the basis of its earliest computers, the HEC series, later enhanced and renamed as the ICT 1200 which became the most popular British computer in the late 1950s.

Andrew Colin, writing in *Resurrection 5*, recalled Booth's development of computers at Birkbeck. Former CCS Chairman Roger Johnson's *Computer Science at Birkbeck College - A Short History*[2] also contains a great deal of interesting material. Tilly Blyth and Roger Johnson took part in a tribute to Booth on BBC Radio 4's *Last Word* programme on 22nd January.

101010101

The recession having hit newspaper advertising revenues, *Technology Guardian* ceased publication in December after some 27 years. One of the last articles to be published was Jack Schofield's excellent coverage of the visit of Sir Maurice Wilkes to The National Museum of Computing at Bletchley Park, organised by our ever-energetic chairman, David Hartley. See
www.guardian.co.uk/technology/2009/nov/18/maurice-wilkes-computing-witch

Happily Technology Guardian lives on in an online edition which is hidden away at *www.guardian.co.uk/technology*. Liberated from its weekly publication cycle, articles are now published as they become available. Schofield has promised to try to add an entry to his popular *Ask Jack* column every day. Recommended.

101010101

The National Museum of Computing and the National Physical Laboratory have combined to create a new gallery at Bletchley Park celebrating the creation of the Internet and recalling the invention of

[2] *www.dcs.bbk.ac.uk/~rgj/Computer%20Science%20at%20Birkbeck%20College%20comp*

packet switching by Donald Davies. See Martin Campbell-Kelly's profile of Davies in *Resurrection 44*.

101010101

The UK's first ever Vintage Computer Festival will be held in The National Museum of Computing at Bletchley Park on the 19th and 20th June. Enthusiasts and collectors from across Europe are expected to enjoy the many exhibitions, demonstrations, flea market and auction and a number of special talks and presentations. CCS secretary Kevin Murrell describes the event as "The ultimate geek version of the Antiques Roadshow!"

101010101

Under Tilly Blyth's direction the *Oral History of British Science* programme for National Life Stories at the British Library has been conducting interviews with many of Britain's surviving computer pioneers. These life story interviews differ from others as they tend to focus on early experiences and education, as well as the significant advances during a career. Interviews have started with Geoff Tootill and Raymond Bird, and have been agreed with many more significant pioneers. Unless interviewees request they remain confidential, the interviews will be made available to the public through the Sound Archive at the British Library. More information about the programme is available at *www.bl.uk/historyofscience*.

101010101

The BBC, in conjunction with the British Museum, is currently broadcasting a series of 100 programmes on Radio 4 which tell the stories of 100 objects. As you might expect, this ambitious series is more about antiquities than anything in living memory although Colossus is mentioned on the series website. But a spinoff is, we hear, being developed which will deal with 20th century technology. We have no news of broadcast dates as yet.

101010101

The CCS has renewed for another year its bulk subscription to the I.E.E.E. *Annals of the History of Computing*. This allows CCS members to obtain a year's copies at the discounted price of £30 - a figure which has not changed despite the financial upheavals of the past year. Members who are not already subscribers can join the scheme by contacting Hamish Carmichael.

Pioneer Profile: Michael Woodger

David Yates

Michael Woodger

Anyone working on the development of digital computers as early as 1946, as Mike Woodger was, clearly has impeccable credentials as one of the true pioneers of computing. At first an assistant to Alan Turing, no little distinction in itself, he went on to play important parts in the development and application of the Pilot ACE and ACE computers at the National Physical Laboratory, and later in the methodology, design and documentation of programming languages, especially ALGOL 60 and Ada.

Mike was born on 28th March 1923 in Epsom, Surrey, England, the eldest of four children. His father Joseph Henry Woodger was emeritus professor of biology in the University of London, and one of his particular ambitions was the construction of a foundation of mathematical logic for biology, based on Whitehead and Russell's work on the logical basis of mathematics. He taught the young Mike symbolic logic as soon as he was able to follow it. This unusual early grounding in rigorous scientific thinking, and particularly in formal methods, evidently planted a seed which was to grow and bear fruit a hundredfold.

After graduating in mathematics from University College London in 1943 and working on military applications for the remainder of World War II, Mike joined the newly-established Mathematics Division of the National Physical Laboratory at Teddington in south-west London in May 1946. After a short period of training in numerical analysis, the main subject of the Division's work, he was introduced to Turing's plans for the ACE computer and von Neumann's plans for EDVAC. He then assisted Turing with detailed logical design and experimental hardware,

the latter including an abortive attempt to make an acoustic delay line store using the air column in a drainpipe. He remembers Turing as being kind, shy, and encouraging, and how they shared a common interest in logic. Eventually, after Turing's departure in 1947 and NPL's belated recruitment of staff with experience of pulse electronics, the development of a computer to be known as the Pilot ACE was undertaken in earnest. Mike's main task was writing standard procedures for mathematical operations, which were successfully tested and put to use when Pilot ACE became operational in 1950. He also played a vital part in the application of Pilot ACE to practical problems. In an (unpublished) interview recorded in 1980 he says "*I ended up developing an automatic system for handling matrices of reasonable size in the store . . . . . and this was very, very successful*". Indeed it was; his system allowed a whole program of matrix manipulations to be specified for the machine by someone not familiar with the machine language. It included space-saving and error checking facilities, and it was rewritten for the DEUCE and in this form widely used well into the 1960s. This work was important in two ways: it made the machines available to more users more efficiently and so greatly extended the range of applications to which they were put; and also, from a historical perspective, it proved to be one of the earliest practical steps on the road to modern high-level computer languages.

In similar vein, the development of the ALGOL 60 language is mentioned in the same interview, but Mike, modest chap that he is, does not stress the importance of his own contribution, which lay particularly in organising and capturing, in a precise written form, the ideas of Peter Naur and his other fellow members on the International ALGOL Committee. Mike is one of the joint authors of the ALGOL 60 report, a milestone in the history of programming languages.

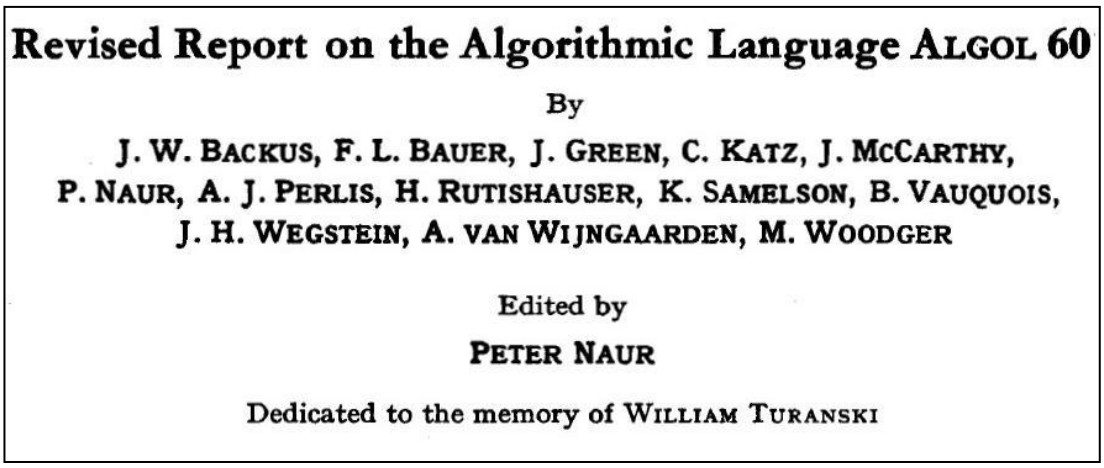
At the time the interview was held in July 1980, Mike had started his second period as chairman of IFIP Working Group 2.3, the international forum for discussion of programming methodology. In this work he had stressed the idea that, since a large program is best constructed as a set of modules in such a way that modules at one level use the services of modules at the next lower level and provide a service to modules at the next higher level, the design of programming languages should reflect these same structural concepts. Each level should have its own terminology and not be cluttered with either text or concepts appropriate

to other levels. His ideas, which have stood the test of time very well, are explained in more detail in ref. [3].

His next contribution, acting as consultant to a consortium led by CII - Honeywell Bull, was to the design of the language at first called Green, one of the four contenders short-listed in a competition to establish a standard language for use in the US Department of Defense. The language was renamed Ada after its success in the competition in 1979. Mike continued his contribution via Alsys S.A., a French company developing Ada documentation, both before and after his retirement from NPL in 1983. Jean Ichbiah, the head of this company, described him fittingly as "*an artist of technical writing*". His part-time work for Alsys ended in 1991.

Fortunately for those interested in the history of computing, Mike has throughout his career kept documents and records systematically. When he retired from NPL the historical part of this considerable collection was moved to the library of the Science Museum in South Kensington, London, so that it could be made available to those with appropriate specialist interests.

The computing world has been fortunate that Mike saw the vacancy notice for mathematicians at NPL in 1946. It led him to a career which was to prove ideally matched to his talents: clarity of thought, ability to handle complex abstract concepts and meticulous attention to detail, qualities he has continued to show to the present day.



Revised Report on the Algorithmic Language ALGOL 60

By

J. W. BACKUS, F. L. BAUER, J. GREEN, C. KATZ, J. MCCARTHY,
P. NAUR, A. J. PERLIS, H. RUTISHAUSER, K. SAMELSON, B. VAUQUOIS,
J. H. WEGSTEIN, A. VAN WIJNGAARDEN, M. WOODGER

Edited by

PETER NAUR

Dedicated to the memory of WILLIAM TURANSKI

As published in Communications of the ACM

[3] M Woodger. On semantic levels in programming. In: C V Freiman (ed.), *Information Processing 71: Proc. IFIP Congress 71*, held in Ljubljana, August 1971, North-Holland, 1972, pp.402-407.

Reminiscences of Whetstone Algol

Brian Randell

The co-author of *ALGOL 60 Implementation*, always known as *Randell and Russell*, muses on the origins of Whetstone ALGOL for the KDF9 and its influence. Whetstone ALGOL was inspired by a week-long school at Brighton Technical College in 1961 at which Edsger Dijkstra and Peter Naur seem to have cast a spell over everybody present (see also Tony Hoare in *Resurrection 48*).

I was hired from college in 1957 by English Electric to program nuclear reactor codes on DEUCE. This was at their Atomic Power Division in Whetstone, just outside Leicester. However, I first spent the summer with IBM at their main London offices, which were in Wigmore Street, behind Selfridges Department Store.

At IBM I programmed and operated a 650 computer, at the time IBM's only computer in the UK. The 650 was proudly installed behind large plate glass windows, where it attracted great attention from the passers-by. After the sleek lines of the 650, with its simple and straightforward programming facilities, and its glossy printed documentation, the DEUCE's workmanlike cabinetry, its binary-level programming, and its duplicated and stapled typewritten manuals, came as rather a shock to me when I arrived at Whetstone in the autumn of 1957.

The DEUCE, English Electric's first computer, was largely based on Alan Turing's 1945, i.e. post-Bletchley Park, plans at NPL for the ACE computer. It therefore differed greatly from just about all its contemporaries, both British and American. The ancestry of these other machines could all be traced back more or less directly to the work of von Neumann and his colleagues on the design for EDVAC at the Moore School of the University of Pennsylvania. In particular, DEUCE's design emphasised computation speed and hardware economy. The assumption was that programmers would be willing and able to take full advantage of the machine's very low-level form of programming. This involved working in binary, and directly controlling the timing of bringing operands to function units and of initiating the operation of these units – in effect a form of what later came to be known as microprogramming.

The effect was that, in the hands of an expert programmer, DEUCE could considerably outperform other computers of basically similar hardware speed.

I and a fellow maths graduate from Imperial College, Mike Kelly, who joined Whetstone at the same time as me, soon came to regard DEUCE with affection, even as a giant toy. So we did not confine our programming to nuclear reactor codes, but instead, on our own initiative, wrote numerous demonstration programs, and started investigating how we could get DEUCE to help with some of the more detailed clerical aspects of its own programming.

DEUCE was in fact not only difficult to program, it was also very difficult to compile for. Most of the then-existing aids to programming DEUCE were interpreters rather than compilers. The effectiveness of these interpreters depended on the cost of the interpretation loop relative to the value of the individual interpreted operations. Thus NPL's interpreter controlling execution of a set of (efficiently hand-coded) linear algebra routines was very cost effective, and being also convenient was heavily used. At the other end of the spectrum was Alphacode, an interpreter that had been developed for a simple language for floating point scalar arithmetic. This was normally used only for small one-off calculations since programs written in it were typically orders of magnitude slower than the equivalent hand-optimised machine code.

The fact that DEUCE, like all the earliest machines, did not have any built-in floating-point arithmetic was very significant. There were subroutines for the various basic floating-point arithmetic operations. But in the interests of efficiency many calculations were laboriously programmed using fixed-point arithmetic. Indeed we programmers regarded the use of floating point almost as cheating, and the coming of hardware floating point as more of a challenge than an opportunity. Against this background Mike and I came up with a compromise solution that took advantage of some peculiar DEUCE hardware features in order to produce a very fast interpreter, albeit just for fixed-point computations. (As I recall it, the main loop of our interpreter had only five instructions, whereas more conventional interpreters typically employed loops involving 50 to 100 instructions.) Despite management opposition – and indeed threats of dismissal – we pressed on with our ideas, and mainly in our own time developed a system which we called EASICODE.

EASICODE was so successful, particularly among the scientists and engineers who were the main computer users at Whetstone, that by the time the KDF9 came onto the horizon, I was in charge of a small

"Automatic Programming Department". Lawford Russell had joined me, replacing Mike Kelly who had left for pastures new at IBM (where he put his great coding skills to developing the microcode for what became the System 360/40). Moreover, there was full management support for us to exercise our skills and enthusiasms on this exciting new computer.

One important aspect of our environment was that we were working in very close proximity to the users of our programs, and had extensive experience of operating our own and other people's programs on the DEUCE. We therefore had developed strong (self-defensive) views on the need for programs to be robust in the face of input, operator and hardware errors, and for them to provide meaningful feedback to users whenever possible. (Little did I know that such experiences and views would strongly influence most of my subsequent career, not just our plans for KDF9.)

KDF9 was a floating point machine, with a set of 16 high speed registers, organised as a small arithmetic stack or pushdown store, together with another set acting as a pushdown store for subroutine return addresses. So it was very different from DEUCE, and indeed all other computers. (Superficially it had similarities with the contemporary Burroughs B5000, but in fact it presented a very different and harder challenge to compiler writers than the B5000.) Indeed, as with DEUCE, carefully hand-coded KDF9 programs could be very efficient, but the problem of compiling comparably good code from a high-level language was far from straightforward.

Faced with this challenge, and a market where IBM's scientific computers already had an effective optimising compiler for FORTRAN, English Electric's Computer Division at Kidsgrove was planning a very ambitious full optimising compiler for the recently defined ALGOL 60. It was evident to us that this compiler would not be ready until some considerable time after the first KDF9s were due to become operational. And our view was that, though this compiler might produce fast code, the compiler itself was likely to be far from fast. We therefore decided to concentrate on the early provision of a programming facility that would be suitable for repeated use during program development.

Then Lawford Russell and I, and members of the Kidsgrove team, attended a workshop at the Brighton College of Technology in 1961, organised by their Automatic Programming Information Centre. At this meeting Edsger Dijkstra described the then very new ALGOL 60 compiler for the Electrologica X1 computer that he and his colleague Jaap A. Zonneveld had implemented. At this stage Lawford and I had not decided what programming language we should support, and were showing dangerous signs of wanting to create our own. Luckily, one of the Kidsgrove group suggested that we should approach Dijkstra to find out if he would support our basing our work on his compiler. He readily agreed, and – with some difficulty – we got English Electric to send us to Amsterdam for a week to confer with him.

Brighton Technical College

Our week of discussions with Dijkstra was spent not just on learning how the X1 compiler worked, but also on the design of a high speed translator/interpreter of full ALGOL 60 for the KDF9. These discussions we documented in a lengthy report. For the next few years Dijkstra used our report to defend himself from the numerous further requests he was getting from people who wanted to visit him and find out about the X1 compiler. (Having lost my own copy of this report many years ago, it was only recently that I found another one after a long search through numerous libraries and archives, and made it available on the Internet. I have also only just been kindly alerted – by Doaitse Swierstra – to the existence of a Report by F.E.J. Kruseman Aretz published in 2003 by the Centrum voor Wiskunde en Informatica, successor to the Mathematical Centre. This report contains the full assembly code text of the X1 Compiler together with that of an equivalent Pascal version.)

The X1 compiler was I believe the world's first ALGOL compiler, certainly the first to cope with all the major new challenges of ALGOL 60, such as block structuring, call by name parameters, recursion, etc. (Indeed I recall that some of the German members of the ALGOL Committee, who

were struggling to implement the new language that they had voted for, campaigned for the removal of some of these facilities from the language.)

There was at this time very little published literature on the subject of ALGOL compilation. I recall there being just one book on compilers, that by Halstead on compiling NELIAC, a dialect of ALGOL 58 language. (ALGOL 58 introduced a number of the features that were extended and generalised in ALGOL 60 – for example it allowed nesting of procedures, with corresponding scoping of identifiers, but it did not have block structuring.) There had, in addition, been a recent growth in publications on formal syntax and syntax analysis (in large part spurred by the ALGOL 60 Report's use of what became known as BNF), and on arithmetic expression translation. But we relied almost entirely on information and advice that we received from Dijkstra.

We set to, and started the detailed design of what became known as the Whetstone ALGOL Translator. We never used the term "WALGOL" ourselves, and avoided the word "compiler" because we were converting ALGOL 60 into an intermediate language, of the form that later became known as P-code, rather than into machine instructions. In effect our intermediate language defined the order code of the machine that we would have liked to have had as our target – an order code that was, I learned some years later, one of the sources of inspiration for the Burroughs 6500 computer, the successor to their B5000.

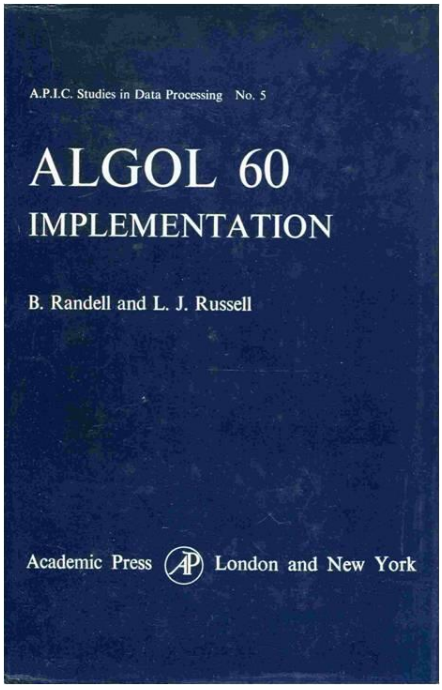
There was considerable, in general friendly, rivalry between Whetstone and Kidsgrove, but also – thanks largely to Fraser Duncan, to whom the Kidsgrove team reported – good cooperation in ensuring the compatibility or our respective systems. In any arguments over such issues we used the ALGOL 60 Report as a neutral referee, so to speak, and thus both projects stuck pretty closely to the letter of the Report.

We at Whetstone placed considerable emphasis on easing the program development task. One consequence was that we were dissatisfied by Dijkstra's strategy of having his compiler signal the first error that it found and then refuse to carry on any further – rather we worked hard on developing a strategy whereby our translator would process the entirety of even a very error-ridden program. It did this in such a way as to have a very good chance of producing an accurate list of many of the errors contained in the program, without getting confused and reporting lots of alleged errors in perfectly good sections of program text. I still recall that when Dijkstra wrote to acknowledge the copy of the report that we had sent him describing this scheme he for the first time

addressed me as “Brian”. In all our extensive previous correspondence he had always addressed me as “Mr Randell”. I felt as though I’d had just been awarded a Masters degree!

We chose to invent a flow diagram notation to use for the detailed design and documentation of our translator and interpreter. (The idea of documenting our design in ALGOL and using bootstrapping as way a means of implementing it, an approach used for NELIAC, was too strange and unfamiliar an idea for us to contemplate at all seriously.) We then hand-coded the flow diagrams in the KDF9’s assembly language (“usercode”) and used a painfully slow emulator of the KDF9 that had been written for DEUCE to start the process of checking them.

But then the delivery date for the first KDF9 started slipping – just a few weeks at a time, but over and over again. These delays were very disruptive, but in fact we ended up putting them to very good use. Someone suggested that we publish the details of Whetstone ALGOL in book form. We sought Dijkstra’s views on this. He was very supportive, and gave me some very good advice, that I’ve since passed on to many graduate students. This was to the effect that a straightforward description of our system would be of interest to only a very limited readership. However, an account that documented all the possibilities we had considered for each design decision that we had taken, explained why we had made our particular design choices (admitting where necessary that a decision had been arbitrary), and reviewed the merits of these decisions in the light of our subsequent experience, would be of much greater value.



So we set out to write such a book, while we awaited the arrival of the KDF9. We worked on the book very intensively, since we had no idea how long the delays would continue, and we thought (in fact wrongly) that others elsewhere, in particular Peter Naur in Copenhagen and Gerrit van der Mey in at the Dr. Neher Laboratorium in the Netherlands, were already busy preparing books on their ALGOL efforts. We completed the book in nine months, mainly in our spare time.

Our inexperience as authors resulted in our providing the publishers, Academic Press, with a much more polished manuscript than they were used to

receiving. We were mindful of all the horror stories that Dijkstra had told us of errors that had been introduced into some of his publications by the manual typesetting process – indeed he urged us to insist on direct photographic reproduction of our typescript. We did not follow this advice, but instead had several typewriters modified so as to have the full ALGOL 60 character set, even including the small subscript "10" character. And we also had all numeric digits use an italic font – so that there could be no confusion between the letter "O" and the digit "*0*", or between the letter "l" and the digit "*1*". These precautions paid off handsomely.

Somewhere during this process, reacting to an announcement of yet another two-week delay, for fun we started to see what we could produce during this period in the way of a little ALGOL -like system for DEUCE, with one-character identifiers (associated with specific words in a 32-word mercury delay line memory). A small succession of further delays, and consequent extensions of our little toy system, ended up with our having to – our own surprise – produced a useful system. This implemented block structure, simple arithmetic expressions, assignment and for statements, procedures, simple parameters, and input/output. Quite a few people started using it in anger for small one-off calculations. But once the first KDF9 became operational at Kidsgrove we abandoned this diversion.

Meanwhile various colleagues, at Whetstone and elsewhere, started to implement our flow diagrams in order to provide full ALGOL 60 on various other machines, even including DEUCE. In so doing they provided valuable feedback on both the diagrams, and the early drafts of our book. (I recall that the chapter on blocks and display, at the time very novel and difficult to understand concepts, was rewritten at least five times. It ended up as perhaps the clearest part of the book!) As a result, when we at last got access to a more-or-less working KDF9, our implementation of Whetstone ALGOL went very smoothly. (For a while, if we ever had unexpected results when running our system on the first KDF9, it was our habit to try to reproduce them on Kidsgrove's DEUCE emulator for KDF9. Any time the results differed we could be sure that the machine and/or the emulator were at fault, and we would gleefully hand both back to their developers while they found out which needed to be mended.)

Our system was designed to work on a minimum configuration KDF9, with 8k word memory, and to read its input directly from paper tape. Given its intended use as a program development system, we put considerable effort into ensuring that the translator operated in a single

pass. And Lawford did such a good job of overlapping translation with input that the translator kept the paper tape reader operating at full speed, and there was no perceptible delay after the end of tape was reached before the translated program started running. Thus users could regard our system as essentially cost-free. (In contrast, to our great scorn, KDF9's assembler seemed to do most of its work only after reading had finished.)

Another ALGOL compiler activity of which we were directly aware was that by Tony Hoare at Elliott Brothers. His compiler, for the Elliott 803, was in fact I believe operational before any of the compilers based on our design. However, it was for a somewhat limited version of ALGOL. Thus when Tony published his famous QuickSort algorithm in the Communications of the ACM , he included a version that avoided the use of recursion, since this was not supported by his compiler. I'm afraid that, as soon as our system was operational, we rather mischievously sent in a certification of the original recursive version of QuickSort, whose publication in effect signalled the completion of Whetstone ALGOL.

I have already alluded to the ALGOL compiler designed by Gerrit Van der Mey – probably the most impressive of the early ALGOL compiler efforts. This was not just because it was the most complete implementation of the language of which I was aware – it even handled dynamic own arrays and various other unfortunate little complications of the language that just about everybody else avoided tackling. Rather it was because van der Mey, who I met just once when I was taken to his house by Fraser Duncan, was both totally blind and deaf. Yet despite these handicaps, he had almost singlehandedly designed his compiler, though colleagues at the Dr. Neher Laboratorium helped with such debugging as was needed. The one other ALGOL Compiler I can recall knowing about in those days was an amazing effort by Brian Higman, for an incredibly small machine, an experimental process control computer being developed by G.E.C. whose total memory capacity was 512 20-bit words. But elsewhere in the UK the emphasis was on Autocode rather than ALGOL, but this was a subject we had very little contact with.

By 1964 the Whetstone System was operational, our book had been published, more compiler projects had been undertaken based on it, and I had been head-hunted by IBM to join the T.J. Watson Research Center in Yorktown Heights. I had agreed to join IBM on condition that I was not asked to continue working on ALGOL compilers – I wanted a change. They agreed, and moreover agreed to my accepting the invitation that I'd had a long time earlier to join the IFIP Working Group 2.1 on ALGOL, an invitation that English Electric had not let me accept because of the costs

of the foreign travel involved. (The Working Group was formed in 1962 as a successor to the original ALGOL Committee.) So I attended the 1964 meeting of the Committee, and the associated IFIP Working Conference on "Formal Language Description Languages", held in Baden, near Vienna. (This conference attempted, not altogether successfully, to bring together people interested in language theory and ones interested in actual programming languages and their compilation.) And thus, over the next few years, I found myself in the midst of the growing controversy over plans for a successor to ALGOL 60, culminating in being one of the small group of committee members who in 1968 wrote a Minority Report on ALGOL 68, and resigned en masse from the Committee. But that is another story.

One final anecdote: when I returned to the UK in 1969, to the Chair of Computing Science at Newcastle University, I found that Whetstone ALGOL was still the mainstay of their first-year programming courses, and so was being used very extensively on their soon-to-be-replaced KDF9. I took care to make it clear that I wanted no part in dealing with any problems that the University might have with the system, only to be told: "We wouldn't let you touch it even if you wanted to!" Similarly, I also can claim no credit at all for the recent splendid effort by David Holdsworth and colleagues that has succeeded in getting the Whetstone ALGOL system working again, starting from a lineprinter listing of its code, on an emulator of the KDF9.

Editor's note: This is a transcript of a talk given by the author at the Science Museum on 14 January 2010. Brian Randell is Emeritus Professor of Computer Science at the University of Newcastle and was recently made a fellow of the ACM for his outstanding contribution to computer history. He can be contacted at brian.randell@ncl.ac.uk.

CCS Website Information

The Society has its own website, which is located at *www.computerconservationsociety.org*. It contains news items, details of forthcoming events, and also electronic copies of all past issues of *Resurrection*, in both HTML and PDF formats, which can be downloaded for printing. We also have an FTP site at *ftp.cs.man.ac.uk/pub/CCS-Archive*, where there is other material for downloading including simulators for historic machines. Please note that the latter URL is case-sensitive.

On the Occasion of the 50th Anniversary of the ALGOL 60 Report

Tony Hoare

> *"Here is a language [ALGOL 60] so far ahead of its time that it was not only an improvement on its predecessors but also on nearly all its successors. Of particular interest are its introduction of all the main program-structuring concepts, and the simplicity and clarity of its description, rarely equalled and never surpassed."*

The quotation is from the appendix to the published version of a keynote talk that I gave at the ACM SIGPLAN Conference in Boston, October 1973. The talk put forward the view that "*the primary purpose of a programming language is to help the programmer in the practice of his art… it should give him the greatest assistance in the most difficult aspects of his art, namely program design, documentation, and debugging".* The relevant criteria I gave for judging quality of a language included "*simplicity*, *security*[4], and *readability".* Most of my talk described how several of ALGOL 60's rivals and successor languages pursued alternative criteria.

"…I do not wish to deny that there are many other desirable properties of a programming language: for example, machine independence, stability of specification, use of familiar notations, a large and useful library, existing popularity, or sponsorship by a rich and powerful organisation." Subsequent history has shown that these desirable properties have been dominant in the choice of programming language by the vast majority of programmers; it was the failure of the ALGOL family of languages to meet these criteria that contributed to its eclipse.

Even so, the ideas and the ideals of the ALGOL family of programming languages live on. They are most apparent in newer language designs such as Java and C#, and in the continuing stream of features that are being added to those languages. Even beyond just language design the ALGOL influence stretches to the design of

4 Security was defined as the property that the effect of running a program (even an incorrect one) is entirely predictable from the specification of the high level language, and does not require knowledge of the underlying machine or operating system or translator.

Application Program Interfaces for general-purpose and special-purpose library packages. It extends also to design patterns, which are widely copied (with adaptation) into programs written in existing languages. A design pattern is an embodiment of a software architecture, or an implementation of an abstract programming concept. Effectively, the user of a design pattern is coding by hand a new structure or feature of a more abstract programming language.

There remains the objection that extensions to existing languages only build further complex superstructure on foundations that are already too complicated, and inherently insecure. So let us not abandon research to discover a language which combines the simplicity and security of ALGOL 60 with the full range of modern program-structuring principles used in libraries and design patterns: these include resource management, control of responsiveness, object orientation, distribution and concurrency. Such a language would play the same fundamental and unifying role in Computer Science as the framework of natural law plays in other branches of science. Even if the language never achieves wide usage, everyone who understands and appreciates it would thereafter be a better programmer, no matter what language they use for coding their designs. Because that, in essence, has been the really enduring achievement of ALGOL 60 itself.

Editor's note: This is a paper submitted by the author to the seminar on ALGOL 60 at the Science Museum on 14 January 2010. Professor Sir Tony Hoare is Emeritus Professor of Computer Science at Oxford University and is now a principal researcher at Microsoft Research in Cambridge. He can be contacted at thoare@microsoft.com.

Contact details

Readers wishing to contact the Editor may do so by email to *dik@leatherdale.net*, or by post to 124 Stanley Road, Teddington, TW11 8TX. Queries about all other CCS matters should be addressed to the Secretary, Kevin Murrell, at *kevin.murrell@tnmoc.org*, or by post to 25 Comet Close, Ash Vale, Aldershot, Hants GU12 5SG.

An ALGOL 60 Spiritual User

Luca Cardelli

Today most respectable programming languages inherit from ALGOL 60. Here we consider a small selection of them and show what ideas have been copied to best effect.

Luca Cardelli

ALGOL 60 was one of the most talked-about programming languages when I was an undergraduate in Pisa in the early 70's, and it clearly had a great influence on my studies. I should make it clear right away, though, that a language with a grand total of three types (real, integer, Boolean) had no conceivable practical use for me! The avant-garde languages of the time were Simula 67 and ALGOL 68. I carefully read the grid-arranged chapters of the ALGOL 68 book in both orders, and the object-ridden Simula 67 book a few times. Although more attracted to ALGOL 68, I ended up using Simula 67 as my main programming language because it was available on my IBM mainframe. Considering the alternatives (Fortran, COBOL, PL/1, and Lisp 1.5) there was really little choice. Simula 67 had ALGOL 60 as a subset, but I was attracted to it mostly by its rich type system, garbage collection, and string processing, none of which were features of ALGOL 60. ALGOL block structures were put to new uses, as objects, inventing object-oriented programming over 10 years before it became widely popular. There is no doubt that, through Simula, I was raised in the algorithmic programming style of ALGOL 60, and on the Structured Programming approach of Dijkstra and Hoare. To the point that later in life I could never bring myself to read assembly language algorithms (i.e. Knuth's): that must surely count as a negative influence of ALGOL 60!

The other main influence on my education was l-calculus, and through it Lisp 1.5. I used Lisp extensively too, but I never enjoyed it because, compared to Simula it had no types, and its supposedly

advanced interactive environment was actually quite bizarre. The main connection between l-calculus and ALGOL 60 goes through Landin: I was familiar with that work too as an undergraduate. ALGOL 60 also deeply influenced my studies of denotational semantics, because some of its more challenging applications were in the semantics of imperative languages (typically ALGOL – like) through Wadsworth's continuations.

After my graduation from Pisa, the influence of ALGOL 60 on me stopped for a while: I moved to Edinburgh and started studying various schools of formal semantics, and functional programming. I found the compilation of functional languages to be an interesting topic, and decided to implement an ML compiler. Ah, but implement it in what language? Surely not in error-prone Lisp, and ML itself was not up to the task yet. I had no longer access to Simula 67, but we had Pascal on the department's VAX. Hence Pascal became the most ALGOL-like language I ever actually used: it had no objects or garbage collection, but was perfect for the systems programming tasks of the times, including implementing a compiler. A bit later, my first paper on ML type-checking was published with an appendix consisting of a Modula-2 version of my Pascal code for the Hindley-Milner algorithm: that was because ALGOL still had such a large influence that the reviewers rejected the "obscure" ML code that I had first written for that appendix.

The purpose of the discussion so far is to set the context of my involvement in Modula-3, because that is the only (possibly) ALGOL – like language that I helped design. Functional programming, the main subject of my research, was not up to the task of writing real systems programs. During my first job at Bell Labs, in the original Unix group, I used C extensively. By the time I moved to Digital Equipment Corporation I was glad to return to a more ALGOL – like language. Modula-2+ was a Xerox-PARC inspired extension of Modula-2 that was widely used at DEC Systems Research Centre: it had a strong list of innovations, but was in need of a redesign. The general problem to be solved then (and now) was to carry out systems programming in a well-structured and type-safe language. My main contributions to Modula-3 were in reformulating the type system of Modula-2+, using techniques coming largely from functional programming. An explicit goal of the Modula-3 committee as a whole was to be inspired by both Simula 67 and ALGOL 68 (and Modula-2 of course). For about 10 years, between Simula 67 and Java, Modula-3 became my favourite language, and my favourite working language of all times. I still regard its module system as unsurpassed, and I say this as a user rather than a designer: it made it possible to cleanly structure the large programs I was writing in ways that

I have not found in other languages, including modern or functional languages.

Which brings us back to ALGOL 60. The original spirit of ALGOL 60 was the effort to structure *algorithms* so that they can be better understood, in contrast, for example to structuring functions, or equations, or formulas, or machine architectures. That is what has always driven me, as a language user and designer. ALGOL 60 did not have much of a type system, but its direct successors ALGOL 68 and Simula 67 made radical and divergent contribution to structuring algorithms by types. It did not have Simula's classes or Modula's modules, but both evolved from program blocks as independent disciplines of programming. Algol 60 lacked all the most important features, and yet its spirit inspired them all. ALGOL 60 is my favourite language that I have never used, except in spirit.

Editor's note: This is a paper submitted by the author to the seminar on ALGOL 60 at the Science Museum on 14 January 2010. Luca Cardelli is a Principal Researcher at Microsoft Research Cambridge. He heads the Programming Principles and Tools Group and his main interests are in type theory and operational semantics, mostly for applications to language design and distributed computing. He can be contacted at luca@microsoft.com.

In the absence of their respective authors, both the paper above and that of Tony Hoare were read at the Science Museum by Cliff Jones of Newcastle University. The meeting was held jointly with the Advanced Programming Group of the BCS with speakers organised by the APG's John Florentin. The CCS is grateful to everybody who contributed to a fascinating afternoon.

Cliff Jones

The Atlas ALGOL System

Bob Hopgood

To overcome the variation between dialects of ALGOL 60, the Chilton Atlas team implemented an ingenious and versatile pre-processor for the language.

I joined the programming group at Harwell in 1959 and spent much of the next three years writing quantum chemistry programs in Fortran, initially for the IBM 704 at Aldermaston and later the 7090. Source code consisted of about five 2,000-card trays and the relocatable binary about 500 cards. A typical development run was the relocatable binary card set plus the source of any revised subroutines. Initially this was sent by car to Aldermaston or plane to Risley but later an IBM 1401 link was established. In 1962 the UKAEA had 10 times the computer power of the UK academic community as a whole. In 1963 Aldermaston replaced the 7090 with an IBM 7030 (Stretch) and the 7090 moved to Risley. Unfortunately, the Fortran compiler delivered by IBM was very poor and Alick Glennie at Aldermaston assembled a team to produce three Fortran compilers in succession (S1, S2 and S3) for the 7030. S1 was a simple fast-compilation system, S2 had basic-block optimisation and S3 included global optimisation. I worked on the lexical/syntax analysis parts of S2 for the first half of 1963 before joining the new Atlas Computer Laboratory (ACL) at Chilton in September 1963.

In 1961 work started on a Fortran compilation system for the Ferranti Atlas called Hartran. This was written in Fortran and ran initially on the 7090 before being moved to Atlas.

The role of ACL was to provide additional computer facilities for the universities, Harwell and the Rutherford High Energy Laboratory. It was clear early on that the interest in ALGOL in the universities was sufficiently high that an ALGOL Compiler was needed on the Chilton Atlas that could handle programs originating from the Elliott and English Electric families as well as from Ferranti machines. Alex Bell had joined ACL from Manchester. Alex and I were given the responsibility of producing an ALGOL system on Atlas that would be of use to the UK academic community. Initially, the idea was to write yet another ALGOL compiler but after discussions with Robin Kerr (Ferranti, later ICT) and John Clegg (Manchester University) we decided to use the ALGOL compiler that they were writing for Atlas using the Brooker-Morris Compiler-Compiler system and replace the first stage by a flexible pre-processor that would allow the different ALGOL dialects in use to be input and run with their own input/output system.

The two computers for which ALGOL programs were most frequently written were the English Electric KDF9 and the Elliott 803. In each case both the input/output facilities provided and the punched form of the program were completely different from the Atlas ALGOL conventions. The 803 used five-hole paper tape, the KDF9 eight hole while Atlas used seven hole! Luckily the tape reader on Atlas could read all three forms of tape. Reading University had an Elliott 803 and we were much indebted to Leonard Dresel who allowed us to come over and run tests on a Friday afternoon. In return, when the pre-processor was up and running in late 1965, we were able to run an eight hour shift of Elliott 803 ALGOL programs through Atlas in about five minutes. To get round the lack of separate compilation facilities in the ALGOL language, the Atlas system wrapped each ALGOL program inside an outer block which had its own variable declarations and a set of procedures. This allowed the individual I/O libraries, the ALGOL library and the GROATS graphics system to be selectively added to any ALGOL program.

The pre-processor eventually had over 15 different dialects of ALGOL that it could handle including both French and Danish dialects. KDF9 ALGOL programs could run with Elliott I/O and vice versa if needed. We introduced a card-based dialect and attempted to overcome the sheer size of the source programs by having a pseudo-binary format that encoded each ALGOL basic symbol, identifier and number as a single column on a card together with a symbol table. This made the size manageable but made editing difficult. So we added an editor to allow the pseudo-binary cards to be changed. Even so, running large ALGOL programs was always more difficult that running Fortran programs. Despite all our efforts, the ALGOL usage on Atlas was never more than about 10% of the workload compared with 65% for Fortran and the remainder spread across machine code, Mercury and Atlas Autocodes etc. Each week, Atlas read a million cards and 30 miles of paper tape!

The elegance of the ALGOL language never overcame its basic weaknesses: no defined markup, no standard I/O, no defined libraries, no sub-compilation and no relocatable binary loader. Despite all our efforts, the bulk of the scientific community stayed with Fortran despite its language deficiencies. Fortran COMMON and EQUIVALENCE statements together with sub-compilation and relocatable binary made it the more attractive option.

Editor's note: Bob Hopgood is Emeritus Professor of Computer Science at Oxford Brookes University. He can be contacted at bhopgood@brookesncl.ac.uk.

ALGOL 60 - Questions of Naming

Dik Leatherdale

Take pity upon your poor editor!

Contributions to this, the special ALGOL 60 edition of *Resurrection*, arrive from hither and thither and yet no sign of any agreement about how to write "ALGOL". Some write "Algol", others "ALGOL". It's a short form of "Algorithmic Language" of course, but shouldn't that be "AlgoL"? But nobody writes "AlgoL". So we refer, of course, to *Wikipedia* which tells us that *The name of the family is sometimes given in mixed case (Algol 60) and sometimes in all uppercase (ALGOL 68). For simplicity, this article uses ALGOL.* – a ruling *Wikipedia* then proceeds to ignore with what we used to call "gay abandon".

The report was originally published in the *Communications of the ACM* and in the *Computer Journal*. As you will see on page 13, *Comm. ACM* uses "A**LGOL**" which looks rather elegant. Other sources, however, do not. In the end I decided to do whatever the *Computer Journal* did. Our esteemed chairman looked it up for me. So "ALGOL" it is then.

But the original name (of ALGOL 58) was the *International Algebraic Language – IAL* for short. In the hands of Jules Schwartz, this grew into the once-popular *JOVIAL – Jule's Own Version of the International Algebraic Language*, reminding us of Donald Knuth's wise words –

> *The most important thing in the programming language is the name. A language will not succeed without a good name. I have recently invented a very good name and now I am looking for a suitable language*

And then again, there is another Algol - the first example to be discovered of a "binary star" - two or more stars orbiting around one another which appear at first sight to be a single celestial object. Once again, we are indebted to *Wikipedia* to discover that: *The name Algol derives from Arabic* رأس الغول *ra's al-ghūl : head (ra's) of the ogre (al-ghūl) ("the ghoul") which was given from its position in the constellation Perseus.*

So there you have it – a programming language, a bon mot and a binary star.

So take pity upon your poor editor. This is what he does all day!

Forthcoming Events

London Seminar Programme

15 Apr 2010	Software Development at Hursley: The First 35 Years and Beyond	Geoffrey Sharman & Andy Krasun
20 May 2010	Pegasus @ 50	Chris Burton, Len Hewitt & others
16 Sep 2010	Research Machines	John Leighfield
28 Oct 2010	Computers & Communications	John Naughton & Bill Thomson
18 Nov 2010	Konrad Zuse and the Origins of the Computer	Horst Zuse

London meetings take place in the Director's Suite of the Science Museum, starting at 14:30. The Director's Suite entrance is in Exhibition Road, next to the exit from the tunnel from South Kensington Station, on the left as you come up the steps. Queries about London meetings should be addressed to Roger Johnson at *r.johnson@bcs.org.uk*, or by post to Roger at Birkbeck College, Malet Street, London WC1E 7HX.

Manchester Seminar Programme

Meetings are currently suspended while refurbishment proceeds at the Museum. It is expected that resumption will be in the autumn.

North West Group meetings take place in the Conference Room at MOSI – the Museum of Science and Industry in Manchester – usually starting at 17:30; tea is served from 17:00. Queries about Manchester meetings should go to Gordon Adshead at *gordon@adshead.com.*

Details are subject to change. Members wishing to attend any meeting are advised to check the events page on the Society website at *www.computerconservationsociety.org* for final details which will be published in advance of each event. Details will also be published on the BCS website (in the BCS events calendar) and in the Events Diary columns of *Computing* and *Computer Weekly*.

Museums

MOSI : Demonstrations of the replica Small-Scale Experimental Machine at the Museum of Science and Industry in Manchester have been suspended due to development work in the museum. Resumption is likely to be late next summer.

Bletchley Park : daily. Guided tours and exhibitions, price £10.00, or £8.00 for concessions (children under 12, free). Exhibition of wartime code-breaking equipment and procedures, including the replica Bombe and replica Colossus, plus tours of the wartime buildings. Go to *www.bletchleypark.org.uk* to check details of times and special events.

The National Museum of Computing : Thursday and Saturdays from 13:00. Entry to the Museum is included in the admission price for Bletchley Park. The Museum covers the development of computing from the wartime Colossus computer to the present day and from ICL mainframes to hand-held computers. See *www.tnmoc.org* for more details.

Science Museum :. Pegasus "in steam" days have been suspended for the time being. Please refer to the society website for updates.

North West Group contact details

Chairman Tom Hinchliffe: Tel: 01663 765040.
Email: *tom.h@dsl.pipex.com*
Secretary Gordon Adshead Tel: 01625 549770.
Email: *gordon@adshead.com*

Committee of the Society

Chairman: **Dr David Hartley FBCS CEng**: david.hartley@clare.cam.ac.uk
Secretary & Leader DEC Project: **Kevin Murrell**: kevin.murrell@tnmoc.org
Treasurer: **Dan Hayton**: daniel@newcomen.demon.co.uk
Chairman, North West Group: **Tom Hinchliffe:** tom.h@dsl.pipex.com
Editor, Resurrection: **Dik Leatherdale MBCS**: dik@leatherdale.net
Website Editor: **Alan Thomson**: alan.thomson@bcs.org
Archivist: **Hamish Carmichael FBCS**: hamishc@globalnet.co.uk
Meetings Secretary: **Dr Roger Johnson FBCS**: r.johnson@bcs.org.uk
Digital Archivist & Leader, Our Computer Heritage Project:
Professor Simon Lavington FBCS FIEE CEng: lavis@essex.ac.uk
Science Museum representative: **Dr Tilly Blyth**: tilly.blyth@nmsi.ac.uk
MOSI representative: **Catherine Rushmore**: c.rushmore@mosi.org.uk
Codes and Ciphers Heritage Trust representative: **Pete Chilvers**:
pete@pchilvers.plus.com
Leader Colossus Project: **Tony Sale Hon FBCS**: tsale@qufaro.demon.co.uk
Leader, Elliott 401 Project: **Chris Burton CEng FIEE FBCS**:
cpb@envex.demon.co.uk
Leader, Bombe Rebuild Project: **John Harper Hon FBCS CEng MIEE**:
bombe@jharper.demon.co.uk
Leader, Elliott 803 Project: **Peter Onion**: peter.onion@btinternet.com
Leader, Pegasus Project: **Len Hewitt MBCS**: leonard.hewitt@ntlworld.com
Leader, Software Conservation Project: **Dr Dave Holdsworth CEng Hon FBCS**:
ecldh@leeds.ac.uk
Leader, 1301 Project: **Rod Brown**: sayhi-torod@shedlanz.co.uk
Leader, Harwell Dekatron Computer Project: **Tony Frazer**: tony.frazer@tnmoc.org
Professor Martin Campbell-Kelly: m.campbell-kelly@warwick.ac.uk
Peter Holland: p.holland@talktalk.net
Dr Doron Swade CEng FBCS MBE: doron.swade@blueyonder.co.uk

Point of Contact

Readers who have general queries to put to the Society should address them to the Secretary (see page 24 for postal address). Members who move house should notify Kevin Murrell of their new address to ensure that they continue to receive copies of *Resurrection*. Those who are also members of the BCS should note that the CCS membership is different from the BCS list and is therefore maintained separately.

Resurrection is the bulletin of the Computer Conservation Society. Copies of the current issue are available from the Secretary for £5.00 each.

Editor – Dik Leatherdale Typesetting – Dik Leatherdale
Cover design -- Tony Sale
Printed by the British Computer Society