Copyright Notice

A programmer's early memories.

Edsger W.Dijkstra

(Text, prepared for the International Research Conference on the History of Computing, June 10 - 15, 1976, Los Alamos.)

Not being a trained historian, I assume myself unable to conduct an objective historical study. Therefore I shall not even pretend objectivity in any sense: I intend to record my from my personal memory what somehow seems relevant or typical to me, leaving it to the professional historian to sort this out, and to select and to ignore in accordance with his professional standards.

In September 1951 I attended as a private person in Cambridge, England, the Summer School on Programme Design for Automatic Digital Computers, the same Summer School that, one year earlier, had been attended by A.van Wijngaarden, then the Head of the Computation Department of the Mathematical Center in Amsterdam. In connection with a letter of recommendation that I needed for a government grant of f200,-- (then $50,--) in support of that trip, I met van Wijngaarden during the summer of '51. Being asked what I had done I showed him two recent discoveries, one being a nomographic technique for finding the one real root of a special class of n-th degree equations that I had encountered, the other being a technique for converting from decimal to octal with a mechanical desk calculator. On the strength of that he offered me a job as a programmer before I had left him; as a consequence of other commitments I had to wait until the end of March '52 before I could accept his offer.

When I started to work at the Mathematical Center I found there B.J. Loopstra and C.S.Scholten, continuously rebuilding a relay machine, called the ARRA. Most of '47, van Wijngaarden had been abroad, studying in the U.S.A. and the U.K.; during a short stay in Amsterdam, however, he had attracted Loopstra and Scholten, who joined the Mathematical Center on the 1st of August '47.

None of us were formally trained as mathematicians. Van Wijngaarden was officially a graduate in mechanical engineering, but in the meantime one with extensive experience in numerical work; Loopstra and Scholten were supposed to study experimental physics at the University of Amsterdam and I still studied theoretical physics in Leiden. Our common background was that we came from two of the then most famous secondary schools of the country, van Wijngaarden and

I from the Gymnasium Erasmianum, and Loopstra and Scholten from the Vossius Gymnasium. I mention this because I firmly believe that that background, which included a solid training in five foreign languages, has had a great influence on the way on which we worked. (Such a binding element was very welcome, for in other cultural aspects we diverged widely: measured to my mildly conservative standards, the other three were at that time radicals on the verge of anarchy, and Scholten, a capable pianist, could even stand the music of certain modern composers!)

Concurrently with the early developments at the Mathematical Center in Amsterdam, in the Hague, and later Delft, W.L.van der Poel developed first the PTERA and then the ZEBRA, but although the distance between Amsterdam and the Hague is negligible, there was surprisingly little contact between those two groups, all the more surprising because in those post-war years of extreme povery and general shortage of everything one could raise the question whether for such a small country two mutually independent efforts at computer development was not a little bit too much. The lacking contact can partly be explained by a difference in background: van der Poel came from a different type of secondary school, he spoke and wrote a different kind of Dutch and communications was, indeed, difficult.

When I joined the Mathematical Center in March '52, the ARRA was believed to approach completion. It was a binary relay computer with a word-length of 30 bits. The fixed-point arithmetic unit that contained the usual two 30-bit registers had been completed first; with a small realy memory it had been driven for some time via uniselectors from a plugboard. In '51 the uniselectors and the relay memory had been replaced by a 1024-word drum. It had an instruction code of 16 instruction, among which two for multiplication and two for division --i.e. both with and without rounding--. The conversion from binary to decimal representation was built in; the single type instruction that caused a number tc be typed was further controlled by a 30-bit code word that specified the layout. Programs were stored with one instruction per word, viz. in the least significant half. The most significant halves of the first 64 words that contained the standard input program were used in such a way as to present the code words for all possible number formats.

The machine, however, was so unreliable as to be practically useless.

It was officially put into operation with all the pomp and circumstance that was deemed necessary; for the demonstration program that it had to execute at that exciting moment, for safety's sake the printing of a table of random numbers had been chosen, and as far as I can remember* that has been the most ambitious program that, in spite of valiant efforts, it has ever executed successfully.

In late '52 what was euphemistically called "a major revision" was decided. As a matter of fact, a totally new, this time largely electronic, machine was built, that had only the word length and the name ARRA in common with the previous effort. For the purpose of this discussion I shall refer to it as ARRA II. On the 1st of November '52, Loopstra and Scholten were joined by G.A.Blaauw, who came from Howard H.Aiken's Laboratory in Harvard. Whereas the old relay machine had worked with "operation complete signals", Blaauw introduced the clocked machine; besides that he introduced selenium diodes, proper documentation techniques, and plug connections so that faulty units could be replaced. Blaauw's competence, his technical input and his devotion to the job at hand were highly appreciated. It was not surprising, however, that, a few years later, he would leave us again: in the irreverent, godless society we formed, the devout Christian Blaauw did not fit too naturally.

Design, development and construction of ARRA II took 13 months: in December '53 it performed its first programs and for the next two-and-a-half years, until medio '56, it has been in continuous use. It, too, had the familiar two-register, fixed-point serial arithmetic unit, a store of 1024 words, eventually divided over 32 tracks of 32 words each, with two instructions stored in each word. Simple additions were performed in (slightly over) one drum revolution of 20 ms, multiplications and divisions took (slightly over) five drum revolutions. It had a nice symmetric instruction code of 25 instruction, with six instructions involving the A-register only:

(A):= (A) + (n)
(A):= (A) - (n)
(A):= (n)
(A):= - (n)
(n):= (A)
(n):= - (A)

and a similar set for the S-register. Only in multiplications, divisions, and double-register shifts these two registers were treated on different footing.

Conditional jumps were conditional on the sign of the last number written into store, conversion was no longer a special instruction, two fast multiplications by 10 were provided instead.

Its functional description (Report MR12, 1953) was the first report I produced as employee of the Mathematical Center. The first sentence states that the ARRA II is an automatic digital computer. The second sentence is worth quoting. Its English translation is:

> "In the sequel this machine will be described as far as is relevant for the person that uses the machine: we shall describe what the machine does, and not how the machine works." (Underlining as in the original Dutch text.)

and the sequal indeed defines the net effect of each instruction without any reference to the internal mechanisms accomplishing it. In retrospect I think it highly significant and telling that I then felt called to introduce my text in that fashion, underlinings included. It is surprisingly modern when we compare it with the following quotation --alternatively we may conclude that progress is, indeed, very slow!-- from Niklaus Wirth, "Programming languages: what to demand and how to assess them", March 1976(!):

> "Hence, we conclude that the first criterion that any future programming language must satisfy, and that prospective customers must ask for, is a complete definition without reference to compiler or computer." (Underlining, again, as in the original.)

Wirth stresses the profound difference between a language (definition) and its implementation, and urges programmers to distinguish clearly between the two -- in exactly the same way as the functional description of the ARRA II sharply distinguished between the instruction code and its "implementation", i.e. the machine that could obey it.

For me, that second sentence has a deep psychological significance: the nonoperational definitions of the semantics of each single instruction provided the impenetrable interface that I needed between me as an emerging programmer on the one hand, and the machine builders Loopstra, Scholten and Blaauw on the other. And that sentence's inclusion at that predominant place in the report strongly suggests that in that environment at that moment in time, such explicit separation between abstract specification and physical realization was a novel idea, something that is not surprising when we see that 23 years later Wirth has still to argue

the very same point on the next level of language. Another indication of some novelty is that, nearly before the stencil ink was dry, this report was nicknamed "The Appalling Prose". It was written with the thoroughness of a legal document, and my hadrware friends teased me with it, also because I needed a lot of teasing in those days. (But it only needed revision when they changed the machine!)

The machine had a double selection, one for instructions and one for numbers. The motivation for that was two-fold. Firstly, by disconnecting the number selection from the first two tracks, the standard input program could be protected from inadvertent overwriting. Secondly, track selection was the only place in which relays had not been thrown out, and as a result, a track change in one of the selections caused an additional delay of one or two revolutions; by doubling the selection the number of track changes was reduced. This reduction was not as effective as hoped: note that the machine had no B-register and it had, therefore, to change its instructions in store. The replacement by a more reliable and faster electronic track selection was the first major improvement of the machine. A further improvement of its reliability was obtained by replacing the selenium components by germanium diodes.

After about half a year of usage the original standard input program contained in two protected tracks is replaced in '54 by a standard input/output program occupying five tracks. The change is significant. Clearly reliability problems had plagues us. The original input program would assemble words from paper tape and store them on consecutive locations, the address of the first location to be filled being given by a leading control combination on the tape. In the second version this control combination had to be punched twice and the input program checked in addition each time a word had been stored that the address of the store instruction had been correctly increased by one. Besides that, such a piece of paper tape could be read in two modes, either storing or comparing for checking purposes.

The output took place via an electric typewriter of which sixteen keys could be operated by means of magnets. The decoding tree selecting the proper magnet was built with relays that retained their position until the next type-instruction was given, and in between, a special "echo-instruction" could read back into the machine the number of the last selected magnet. Short of actually

reading the typed page, this was the most complete check on the output operation that could be envisaged. I designed a very ingenious process --of which I was therefore very proud-- in which the conversions from binary to decimal and the conversion back from the decimal echos to the original binary were merged to the extent that they shared the same multiplication by 10. This rigorous total check on both the conversion and the actual magnet selection became standard practice on ARRA II, and its next three successors and was only abandoned when the lineprinter hardware did not provide anymore for the echo reading. In case of a discrepancy, the machine would type the number again in the same position of the next line.

Today such precautions may seem exaggerated. But in the preface of a 200-page table of interpolation coefficients, published in May '55, van Wijngaarden and I proudly stated:

> "The whole table has been produced by the electronic computer ARRA. The program included complete mathematical checks including the signals sent out to the typewriter. The sheets are reproduced by photo-offset."

I think that our pride was justified --the only error ever found was in the hand-typed preface!-- . This was indeed an achievement with a machine whose major shortcoming was that its not too reliable drum memory was not protected by a parity check. It is, by the way, a sobering thought that practically no computer today could produce those master sheets with the same trustworthiness and printing quality. So much for progress.....

The machine was in continuous use; during the night , often unattended for long periods, it worked on a very time-consuming project of integrating wave equations for research in theoretical chemistry. In order to increase the probability of useful work in spite of an unreliable memory, I rewrote with Scholten that program entirely. The operation "x:= x + 1" was originally coded --in modern notation-- as:

```
A:= x; A:= A + 1; x:= A     ;
```

we replaced it by:

```
repeat A:= x; A:= A + 1; y:= A;
       A:= -x; A:= A + y; A:= A - 1 until A = 0;
repeat A:= y; x:= A; A:= -y; A:= A + x until A = 0
```

I mainly remember that night because, as the hours went by, much to Scholten's amusement I got more and more cross with that program's original author, hurt as I was by all its clumsiness. Scholten and Loopstra, who also acted as maintenance engineers for the machine, were grateful: the transformation was highly effective and from then onwards they hardly received any more calls for service at night.

The first half year of ARRA II's operations, its builders were still busy improving it. Then, in May '54, a contract with Fokker, our national aircraft industry, was signed to deliver to Fokker the FERTA, an improved version of ARRA II. Apart from more elaborate shift instructions it was basically the same machine. Seven months later, in December '54, the FERTA performed its first test computations; it was delivered on the 1st of April next year, less than a year after the contract had been signed.

The Fokker plant, where the FERTA was installed, was not too easily reached. Blaauw and I did the debugging of the hardware during the winter of 54/55. Blaauw, who had been in the U.S.A., was rich enough to have a car, at least some sort of car: in an n-th hand Standard Vanguard he would drive from the Hague, where he lived at the time, to Fokker near Schiphol. I lived as a student in Leiden, and would leave my room at seven o'clock in the morning and cycle towards the highway, where I would hide my bicycle and climb up to the road where Blaauw would pick me up. It was a grim winter, but the combination of an old Canadian army coat and the fact that Blaauw was a punctual man made this possible. One night, at say half past ten, when we wished to return, the old Vanguard, the last car on Fokker's otherwise deserted parking lot, refused to be started. I do remember that it had started snowing, I don't remember how we came home: presumably Blaauw himself fixed another bug that day.

It is sometimes hard to remember how poor we were on those days. When I entered the service of the Mathematical Center at the age of 21 with the best possible credentials, my salary was f 11,-- (about $ 3,--) a day. Three years later, when the FERTA was transferred to Fokker, the successful transaction was concluded by a dinner with the directors of Fokker's Aircraft Industry, I shall never forget the main course: chicken boned by the waiter at our table with knife and fork. And I shall remember the regret with which I saw so much perfect chicken meat being returned to the kitchen: in those days I was not only poor, I was hungry. But, somehow, we did not seem to mind: we worked like mad, we were

never in doubt about the immediate usefulness of our work, lived in a state of continuous excitement and had a tremendous amount of fun.

In retrospect the ARRA II with which I worked for nearly two-and-a-half years was an ideal machine for getting introduced to the subject. We wrote all the usual things like floating point and multilength routines, routines for elementary functions and what not. Its decent order code and its slow but fairly homogeneous store was an incentive to do a nice job. Its limited size and speed saved us from overambitious projects and its unreliability was a sobering reminder.

I must mention a demonstration program I wrote for it at the occasion of the International Mathematical Congress that was held in Amsterdam in 1954. It faked the ability "to learn to type a digit": after each digit typed the environment had to signal whether the machine had typed "the desired digit" or not. If, for instance, a "3" had been chosen as desired digit, it would eventually type almost always a three. It was then possible to withhold the "appreciation": gradually the threes would be replaced by other digits. It was a simple one-page program concocted within an hour; it generated pseudo-random numbers and had a few thresholds determining the speeds of "learning" and "forgetting" and the frequency of "errors". During its exhibition I got alarmed by the enthusiasm it evoked among the impressed visitors who were mislead to believe that we were seriously simulating learning processes. Eventually I tried to explain that we were only faking, but I was not very successful, because their enthusiasm had carried our visitors too far away: it was a frightening experience....

* * *

By the end of June '56, fifteen months after the delivery of the FERTA, Loopstra and Scholten had completed their next machine, the ARMAC, a new machine for which I had written again the functional description, designed the program notation and written the basic communication programs. For the third time the entire road from conception to a fully documented and working machine had been covered in less than one-and-a-half year. Clearly we were getting experienced!

In view of our previous equipment the ARMAC was an exciting monstrum.It was exciting because on certain computations it could be 50 times as fast as the ARRA II, which was instantaneously dismantled when the ARMAC was put into operation. Personally I remember the awe with which we looked at it when it per-

formed its first production program, a computation of transversational movements of railway carriages. Another reason for remembering that occasion was that the program had been written by M.C.Debets, bound to become mrs.Dijkstra a year later. The word-length of the ARMAC was 34 bits, two instructions per word and an instruction code of 30 instructions. Its main store consisted of a drum of 112 tracks of 32 words. The first 512 words of store were intended to be realized by ferrite cores; eventually this was only effectuated for the first 32, the so-called "fast page". Besides the fast page the machine was equipped with a 32-word ferrite core buffer: as soon as control switched to a new drum track, the computation came to a grinding halt for the duration of a revolution of the drum (which rotated at a speed of 75 revolutions per second), during which the new track was copied into the high-speed buffer, and from then onwards instruction fetch was again instantaneous. The device --a one-page virtual memory avant la lettre-- was extremely effective. It was also very tricky: the machine had still no B-register and, therefore, had to modify instructions in store; to save time it had special instructions modifying only the copy in the buffer. Needless to say, a full description of that wonderful feature, complete with warnings for all the pitfalls occupied several pages in the manual. It was a time when all standard routines were elaborately polished until they fitted exactly on a single 32-word page.

The ARMAC had a parity check on the memory and, as far as speed and reliability were concerned, it was a great improvement over its predecessor. The fact that its programmer had to be highly conscious of the existance of the individual tracks was a step backwards; the high penalty of buffer refilling furthermore made many more advanced programming techniques, although logically possible, efficiency-wise too unattractive to be considered seriously. Autocoders and compilers were plainly out of question. In retrospect this has been a blessing: it has forced us to skip the stage of infancy of what was then called "automatic programming", and when we got our next machine, the EL-X1, we could approach the problem of compiling ALGOL 60 without the burden of misleading experiences.

As far as programming was concerned, the ARMAC period was one of consolidation of our practice of running the Computation Department. The manual, for instance, states as one of the strict rules of the house that each program should

be "restartable", i.e. without the need for reading it in another time: no more saving of a few instructions by means of initializing some locations during program input.

* * *

I recall from that time one failure on my side which is indicative for the progress of the art of programming that would materialize during the next decades. Let in N successive storage locations be given a permutation of the numbers from 1 through N , which is not the alphabetically last one; it is then asked to write a program that will transform that permutation into its immediate alphabetical successor. I remember that I tried to solve this problem for more than two hours and then gave up! In the early seventies, when I needed a simple example for an introductory programming course, I suddenly remembered that problem and solved it without pencil and paper in twenty minutes, and --what is more-- I also did not need more than twenty minutes to explain my solution next morning to my novice audience! I myself was very impressed by this palpable proof of progress and tried to drive home that message by faithfully reporting my failure of some sixteen years earlier. (They were not impressed: they concluded that if I had not been able to solve that problem, I could not have been a very bright student.) But I remember my failure so well, that I still know how I failed. I was still thinking in terms of individual instructions, some of them being jumps: I programmed without sequencing discipline and had not the syntactical grasp provided by the repetition clause. I did not even know what a loop was, I only knew that as a result of jumping around, instructions could be executed many times. To add to the confusion I was mentally coding for a machine that, for lack of a B-register, had to modify its own instructions in store, thereby blurring the difference between the constant program and the variables. In view of our primitive way of thinking about the programming task it is in a sense a marvel that in spite of it we still designed so many nontrivial programs.

* * *

Three months after the completion of the ARMAC, in September 1956 the specifications for a next machine, the EL-X1, were drafted. Its detailed design was ready by the end of the year, and another year later, at the end of '57, the prototype X1 performed its first computations. Again in less than eighteen months. The EL-X1 was really exciting. It was a fully transistorized machine that would

have ferrite cores as its main memory, and it was ten times as fast as the ARMAC.

This time the design and production of the standard input/output program was a totally new challenge, and it was so for two reasons. The one reason was that the hardware required the standard input/output program to be wired in. I designed the program as carefully as I could, paid equal care to the ARMAC program that was used to derive from my hand-punched version the punched paper tapes that were going to control the semi-automatic wiring machine. It was the first time in my life that a considerable investment was made in the hard-wiring of a 25.000 bit program that I had been unable to test. I am grateful for the experience, as it taught me that the flawless production of such a program is not a superhuman task. The other reason was that, following a suggestion from A.W.Dek, the X1 was equipped with a real-time interrupt signalling the completion of I/O-commands. When this was first suggested it frightened the wits out of me as soon as I realized that it would make the machine a nondeterministic one without reproducible behaviour, and I managed to delay by my fear the final decision to incorporate the interrupt for a few months. Eventually I was flattered out of my resistance, and this double inability to test my program before it went into "production" gave a new flavour to the task.

I remember my first contribution: when trying to design the status saving and status restoring protocol in such a way that I could prove its correctness, I discovered that with the proposed means for enabling and disabling the interrupt it was impossible to do so. After Loopstra and Scholten had checked my argument, that part of the instruction code was changed in accordance with my recommendations.

After that programming task had been completed in early 1958, I did not program for a while. I was absorbed by writing my thesis on that last effort, translating my thesis from Dutch into English and, besides that, the running of the Computation Department took more of my time, due to temporary absence of its director who had had a serious road accident. Preliminary discussions, based on ALGOL 58 and eventually leading to ALGOL 60 were started. Besides that, it would last until March '60, until the Mathematical Center would get its copy of the EL-X1, the first copies being granted to more impatient customers.

On the 28th of October '59 I defended my Thesis, with A.van Wijngaarden acting as my Promotor. Four days later, on the 1st of November we started the discussion on how to implement ALGOL 60, of which the final definition reached us in January 1960. Harry D.Huskey had just spent a few sabbatical months at the Mathematical Center, working on an algebraic compiler, but his style of working differed so radically from mine that, personally, I could not even use his work as a source of inspiration; the somewhat painful discussion with my boss, when I had to transmit to him that disappointing message, is remembered as one of the rare occasions at which I banged with my fist on the table. During my '59 summer holiday in Paterswolde I had given my first thoughts to the question how to implement recursion, in the early months of '60 we discovered how to do, in combination with that, justice to the scope rules of ALGOL 60. After the interface with the run time system had been decided, J.A.Zonneveld and I wrote the compiler, while Marlene Römgens and Fiek Christen wrote under my supervision the routines of the run time system. In March the machine arrived and in August our implementation was operational. Our first testcase had been something like "begin real a; a:= 7 end" ; our fourth testcase had been a recursive summation procedure, via the call-by-name mechanism and Jensen's device called to perform a double summation. The combination of no prior experience in compiler writing and a new machine without established ways of usage greatly assisted us in approaching the problem of implementing ALGOL 60 with a fresh mind.

* * *

Our implementation of ALGOL 60 marks the end of the period of growth that I intended to cover, it also marked the beginning of a new one. Finally I was beginning to consider myself as a professional programmer, and, thanks to the beard I grew during the ALGOL 60 project, I was even beginning to look like one. It marked the beginning of the period during which the programming activity would begin to evolve from a craft to a scientific discipline. In December '60 I bought my first car, a Volkswagen I could barely afford. I had a wife, one-and-a-half child, a telephone, a piano and a car: the son, whose wanderings had sometimes worried his parents, showed at last the symptoms of a respectable citizen.

NUENEN, 24th May 1976
The Netherlands

prof.dr.Edsger W.Dijkstra
Burroughs Research Fellow