Association for Computing Machinery
Advancing Computing as a Science & Profession
A. M. Turing Award

1977 – John Backus

Citation

For profound, influential, and lasting contributions to the design of practical high-level programming systems, notably through his work on FORTRAN, and for seminal publication of formal procedures for the specification of programming languages.

Biographical Information

John Backus (born December 3, 1924) is an American computer scientist, notable as the inventor of the first high-level programming language (FORTRAN), the Backus-Naur form (BNF, the almost universally used notation to define formal language syntax), and the concept of Function-level programming. He received the 1977 ACM Turing Award for these seminal achievements. Backus' Turing citation read as follows:

> *For profound, influential, and lasting contributions to the design of practical high-level programming systems, notably through his work on FORTRAN,*

and for seminal publication of formal procedures for the specification of programming languages. Backus was born in Philadelphia, Pennsylvania but grew up in Wilmington, Delaware. He studied at the Hill School in Pottstown, Pennsylvania, and was apparently not a diligent student. After entering the University of Virginia to study chemistry, and failing that, he then joined the US Army and began medical training, which he also dropped out of after nine months.

After moving to New York City he initially took training as a radio technician and discovered an interest in mathematics. He graduated from Columbia University with a degree in the topic in 1949, and joined IBM in 1950. During his first three years, he worked on the SSEC; his first major project was to write a program to calculate positions of the Moon.

The difficulties of programming were acute, and in 1954 Backus assembled a team to define and develop Fortran for the IBM 704 computer. Whilst arguably not the first high-level programming language, it was the first to achieve wide use. During the latter part of the 1950's Backus served on the international committes which developed ALGOL 58 and the very influential ALGOL 60, which quickly became the de facto worldwide standard for publishing algorithms.

He later worked on a "function-level" programming language known as FP which was described in his Turing Award lecture "Can Programming be Liberated from the von Neumann Style?" Sometimes viewed as Backus's apology for creating Fortran, this paper did less to garner interest in his own FP than to spark research into functional programming in general. FP was strongly inspired by

Kenneth E. Iverson's APL, even using a non-standard character set. Backus spent the latter part of his career developing FL (from "Function Level"), a successor to FP. FL was an internal IBM research project and development of the language essentially stopped when the project was finished (only a few papers documenting it remain), but many of the language's innovative, arguably important ideas have now been implemented in Iverson's J and Herrera's NGL programming languages.

Backus was named an IBM Fellow in 1987, and was awarded a Draper Prize in 1993. He retired in 1991.

Turing Lecture

Additional Links

John Backus

[Home] [ACM Awards] [A. M. Turing Award]

March 20, 2007

John W. Backus, 82, Fortran Developer, Dies

By STEVE LOHR

John W. Backus, who assembled and led the I.B.M. team that created Fortran, the first widely used programming language, which helped open the door to modern computing, died on Saturday at his home in Ashland, Ore. He was 82.

His daughter Karen Backus announced the death, saying the family did not know the cause, other than age.

Fortran, released in 1957, was “the turning point” in computer software, much as the microprocessor was a giant step forward in hardware, according to J.A.N. Lee, a leading computer historian.

Fortran changed the terms of communication between humans and computers, moving up a level to a language that was more comprehensible by humans. So Fortran, in computing vernacular, is considered the first successful higher-level language.

Mr. Backus and his youthful team, then all in their 20s and 30s, devised a programming language that resembled a combination of English shorthand and algebra. Fortran, short for Formula Translator, was very similar to the algebraic formulas that scientists and engineers used in their daily work. With some training, they were no longer dependent on a programming priesthood to translate their science and engineering problems into a language a computer would understand.

In an interview several years ago, Ken Thompson, who developed the Unix operating system at Bell Labs in 1969, observed that “95 percent of the people who programmed in the early years would never have done it without Fortran.”

He added: “It was a massive step.”

Fortran was also extremely efficient, running as fast as programs painstakingly hand-coded by the programming elite, who worked in arcane machine languages. This was a feat considered impossible before Fortran. It was achieved by the masterful design of the Fortran compiler, a program that captures the human intent of a program and recasts it in a way that a computer can process.

In the Fortran project, Mr. Backus tackled two fundamental problems in computing — how to make programming easier for humans, and how to structure the underlying code

to make that possible. Mr. Backus continued to work on those challenges for much of his career, and he encouraged others as well.

“His contribution was immense, and it influenced the work of many, including me,” Frances Allen, a retired research fellow at I.B.M., said yesterday.

Mr. Backus was a bit of a maverick even as a teenager. He grew up in an affluent family in Wilmington, Del., the son of a stockbroker. He had a complicated, difficult relationship with his family, and he was a wayward student.

In a series of interviews in 2000 and 2001 in San Francisco, where he lived at the time, Mr. Backus recalled that his family had sent him to an exclusive private high school, the Hill School in Pennsylvania.

“The delight of that place was all the rules you could break,” he recalled.

After flunking out of the University of Virginia, Mr. Backus was drafted in 1943. But his scores on Army aptitude tests were so high that he was dispatched on government-financed programs to three universities, with his studies ranging from engineering to medicine.

After the war, Mr. Backus found his footing as a student at Columbia University and pursued an interest in mathematics, receiving his master’s degree in 1950. Shortly before he graduated, Mr. Backus wandered by the I.B.M. headquarters on Madison Avenue in New York, where one of its room-size electronic calculators was on display.

When a tour guide inquired, Mr. Backus mentioned that he was a graduate student in math; he was whisked upstairs and asked a series of questions Mr. Backus described as math “brain teasers.” It was an informal oral exam, with no recorded score.

He was hired on the spot. As what? “As a programmer,” Mr. Backus replied, shrugging. “That was the way it was done in those days.”

Back then, there was no field of computer science, no courses or schools. The first written reference to “software” as a computer term, as something distinct from hardware, did not come until 1958.

In 1953, frustrated by his experience of “hand-to-hand combat with the machine,” Mr. Backus was eager to somehow simplify programming. He wrote a brief note to his superior, asking to be allowed to head a research project with that goal. “I figured there had to be a better way,” he said.

Mr. Backus got approval and began hiring, one by one, until the team reached 10. It was an eclectic bunch that included a crystallographer, a cryptographer, a chess wizard, an employee on loan from United Aircraft, a researcher from the Massachusetts Institute of Technology and a young woman who joined the project straight out of Vassar College.

“They took anyone who seemed to have an aptitude for problem-solving skills — bridge players, chess players, even women,” Lois Haibt, the Vassar graduate, recalled in an interview in 2000.

Mr. Backus, colleagues said, managed the research team with a light hand. The hours were long but informal. Snowball fights relieved lengthy days of work in winter. I.B.M. had a system of rigid yearly performance reviews, which Mr. Backus deemed ill-suited for his programmers, so he ignored it. “We were the hackers of those days,” Richard Goldberg, a member of the Fortran team, recalled in an interview in 2000.

After Fortran, Mr. Backus developed, with Peter Naur, a Danish computer scientist, a notation for describing the structure of programming languages, much like grammar for natural languages. It became known as Backus-Naur form.

Later, Mr. Backus worked for years with a group at I.B.M. in an area called functional programming. The notion, Mr. Backus said, was to develop a system of programming that would focus more on describing the problem a person wanted the computer to solve and less on giving the computer step-by-step instructions.

“That field owes a lot to John Backus and his early efforts to promote it,” said Alex Aiken, a former researcher at I.B.M. who is now a professor at Stanford University.

In addition to his daughter Karen, of New York, Mr. Backus is survived by another daughter, Paula Backus, of Ashland, Ore.; and a brother, Cecil Backus, of Easton, Md.

His second wife, Barbara Stannard, died in 2004. His first marriage, to Marjorie Jamison, ended in divorce.

It was Mr. Backus who set the tone for the Fortran team. Yet if the style was informal, the work was intense, a four-year venture with no guarantee of success and many small setbacks along the way.

Innovation, Mr. Backus said, was a constant process of trial and error.

“You need the willingness to fail all the time,” he said. “You have to generate many ideas and then you have to work very hard only to discover that they don’t work. And you keep doing that over and over until you find one that does work.”