The original English text of an interview with the Russian news organization 3DNEWS:

1. You're probably asked this question a thousand times, and yet - what features of C + + allowed it to move out to other programming languages, and to this day remain a living classics?
    - C++ had (and has) C's simple and efficient model of hardware and the ability to compose objects simply and without space overheads. On top of that, it provided (and provides) the programmer the ability to define flexible abstractions without overhead compared to alternatives. The constructors and destructors were novelties that made a huge difference. Compared to K&R C, C++ provided major improvements in type safety. Compared to Simula, C++ offered significant flexibility and efficiency.

      Importantly, C++ didn't stop growing in 1985 after its first commercial release. Over the years, it has slowly become a far more effective and flexible tool for system development. Its specific strengths (e.g. for building software infrastructure and resource constrained applications) build directly on my original aims and techniques for C++: a simple, direct and efficient mapping to hardware plus very flexible (zero-overhead) abstraction mechanisms.

2. Do you believe sometime in the fact that programming languages, as such, will wither away, and the computer will understand normal human English?
    - I doubt it. I doubt it would be desirable. When I specify an action to be done by a computer, I want far more economy of expression and a far more precise definition of meaning than I can have in informal English: "a = b+c" is both terser and better defined than "add c to b and put the result into a." The languages of programming are far closer to highly technical and specialized shorthands used by human professionals, such as mathematicians and medical doctors, than to everyday speech. In fact, I would not be at all surprised if programming became more formalized and moved further from ordinary human speech.

      Naturally, I'm taking about software development here. Our ordinary interactions with computers are likely to be more like human-to-human communication. For example, "give me a hot dog on French bread, with mustard, no ketchup, and by the way some raw onions" could be perfectly understandable to a computerized snack-food stand. However, the speech recognition program "understanding" that request and the embedded system that acts on it would be programmed in a highly specialized technical language.

3. One of the major challenges currently facing software developers is to ensure effective support for multi-core processors (notably, of course, it comes to customer applications, because on the servers everything seems to be fine). A lot of applications still use only one CPU-core. Is there a way to resolve this disharmony?
    - We need better high-level concurrency models to make programming concurrent systems easier. C++0x provides a start by providing a type-safe and standard set of facilities for traditional threads-and-locks style concurrent programming. That (and the lock-free programming support also provided by C++0x) is not ideal for concurrent applications, just better than what most languages provide today. However, it allows more advanced, simpler, and more specialized models to be provided as libraries implemented on top of the standard facilities.

We need more work on simplifying the specification of concurrent systems – even after about 50 years of research.

4. In Russia, every second student-programmer invents its own programming language. Do you believe in the success of brilliant singles, and whether there is market demand for an entirely new language?
    - Designing a new language and implementing it is (relatively) easy. The difficult part is convincing serious developers of interesting systems to use a new language. To be useful, a new language has to be a better solution to a problem than existing (and known) tools and languages. I recommend against designing a new language unless you are thoroughly familiar with a problem area and are sure that you can provide a better solution in the form of a language. Often a tool or a library in an existing language is a more effective solution – this is one reason many languages die young: the older language quickly catch up.

      That said, I like to see new languages and I know that far better languages than the ones we have today are possible. It would be very sad if new languages stopped coming. Designing a new language is one of the significant outlets for new ideas and approaches to understanding problems and their solution. I don't think there is ever "a market" for a new language, but just occasionally a new one escapes the hazards of conservatism and market pressures towards use of the established languages. That's good and exciting!

      I think that an individual designer is important because a group cannot sustain a focus, a coherent set of aims for a language. Groups get lost in details. However, a language cannot be designed in isolation; a designer needs feedback and inspiration that only a lively technical community can offer. I was lucky to work at AT&T Bell Labs when C++ was born – the technical environment there was so diverse and demanding that it gave me the tough challenges C++ needed to survive. I owe a lot to my colleagues there.

5. What do you think - why immigrants from Russia demonstrate such efficiency and success in the development of software and services, and why they are usually particularly successful outside the native country?
    - Maybe the ones who go abroad are the ones who have the most initiative, the most curiosity, and feel the greatest pressure to do something new? Maybe they are the ones most willing to take risks and having moved they are often under more severe pressure to succeed?

      I suspect that it's a more general immigrant phenomenon than simply a Russian one (but then I'm an immigrant myself).

6. Which of digital technology or devices made the greatest impression on you in the last ten years?
    - I love photography from both a geek and an aesthetic perspective and the developments in digital imaging are simply astounding. Obviously, there is also a lot of software in those wonderful cameras.

      Also, I couldn't function professionally or keep in contact with my friends without the amazing communications infrastructure that connects me to the whole world through my laptop.

7. For what reason you come to Russia and what are your plans for the visit?
    - Partly, I come because I have never been to Russia. Partly, I come for the opportunity to talk to Russian software developers. Unfortunately, I couldn't find enough time to be a real tourist – October is in the middle of the semester and I can't be away from my students for too long. The conference people will look after me well and make sure that I see some of Moscow's tourist sites, such as the Red Square and the Bolshoi. Between that and my talks, I expect to be rather tired by the time I head back to the US, but I'm looking forward to my visit to Moscow.