Good morning ladies and gentlemen!

It’s a great pleasure to see you all here today!

Let me first introduce myself, I’m Velikodnyij Vladimir. I’m a second year student of Computer Science Faculty of Voronezh State University.

Today I’ll talk about why computers won’t make thrmselves smarter.

This presentation will be interesting for those of you, who wants to work in IT sphere.

I have divided my presentation into 3 main parts:

First of all, I’d tell about what might self-recursive self improvement looks like for human being.

Then I'll tell you about how comuters can improve themselves.

Finally, I’d say some words about why techological explosion more usefull than intelligent one.

It’ll take about 6 minuts to cover all main points of my presentation.

After my presentation I’ll be happy to answer the questions you may have, so let’s get started.

In the first part of my presentation, I’ll tell how humans can improve themselves. What might recursive self-improvement look like for human beings? For the sake of convenience, we’ll describe human intelligence in terms of I.Q. For humans, cursive self-improvement would look like this: Once there’s a person with an I.Q. of, say, 300, one of the problems this person can solve is how to convert a person with an I.Q. of 300 into a person with an I.Q. of 350. And then a person with an I.Q. of 350 will be able to solve the more difficult problem of converting a person with an I.Q. of 350 into a person with an I.Q. of 400. And so forth.

Do we have any reason to think that this is the way intelligence works? I don’t believe that we do. For example, there are plenty of people who have I.Q.s of 130, and there’s a smaller number of people who have I.Q.s of 160. None of them have been able to increase the intelligence of someone with an I.Q. of 70 to 100, which is implied to be an easier task. None of them can even increase the intelligence of animals, whose intelligence is considered to be too low to be measured by I.Q. tests. If increasing someone’s I.Q. were an activity like solving a set of math puzzles, we ought to see successful examples of it at the low end, where the problems are easier to solve. But we don’t see strong evidence of that happening.

Maybe it’s because we’re currently too far from the necessary threshold; maybe an I.Q. of 300 is the minimum needed to increase anyone’s intelligence at all. But, even if that were true, we still don’t have good reason to believe that endless recursive self-improvement is likely. For example, it’s entirely possible that the best that a person with an I.Q. of 300 can do is increase another person’s I.Q. to 200. That would allow one person with an I.Q. of 300 to grant everyone around them an I.Q. of 200. But that would still leave us at a plateau; there would be no recursive self-improvement and no intelligence explosion. “If the human brain were so simple that we could understand it, we would be so simple that we couldn’t.”

That’s all I want to tell you about how humans can improve themselves. Now let’s move on to next part of my presentation.

In the second part of my presentation, I’d like to tell you how computers can make themselves smarter.

Let’s imagine that we have an A.I. program that is just as intelligent and capable as the average human computer programmer. Now suppose that we increase its computer’s speed a hundred times and let the program run for a year. That’d be the equivalent of locking an average human being in a room for a hundred years, with nothing to do except work on an assigned programming task. Many human beings would consider this a hellish prison sentence, but, for the purposes of this scenario, let’s imagine that the A.I. doesn’t feel the same way. We’ll assume that the A.I. has all the desirable properties of a human being but doesn’t possess any of the other properties that would act as obstacles in this scenario, such as a need for novelty or a desire to make one’s own choices. (It’s not clear to me that this is a reasonable assumption, but we can leave that question for another time.)

When you’re developing software, you typically use a program known as a compiler. The compiler takes the source code you’ve written, in a language such as C, and translates it into an executable program: a file consisting of machine code that the computer understands. Suppose you’re not happy with the C compiler you’re using—call it CompilerZero. CompilerZero takes a long time to process your source code, and the programs it generates take a long time to run. You’re confident that you can do better, so you write a new C compiler, one that generates more efficient machine code; this new one is known as an optimizing compiler.

You’ve written your optimizing compiler in C, so you can use CompilerZero to translate your source code into an executable program. Call this program CompilerOne. Thanks to your ingenuity, CompilerOne now generates programs that run more quickly. But CompilerOne itself still takes a long time to run, because it’s a product of CompilerZero. What can you do?

You can use CompilerOne to compile itself. You feed CompilerOne its own source code, and it generates a new executable file consisting of more efficient machine code. Call this CompilerTwo. CompilerTwo also generates programs that run very quickly, but it has the added advantage of running very quickly itself. Congratulations—you have written a self-improving computer program.

But this is as far as it goes. If you feed the same source code into CompilerTwo, all it does is generate another copy of CompilerTwo. It cannot create a CompilerThree and initiate an escalating series of ever-better compilers. If you want a compiler that generates programs that run insanely fast, you will have to look elsewhere to get it.

The technique of having a compiler compile itself is known as bootstrapping, and it’s been employed since the nineteen-sixties. Optimizing compilers have come a long way since then, so the differences between a CompilerZero and a CompilerTwo can be much bigger than they used to be, but all of that progress was achieved by human programmers rather than by compilers improving themselves. And, although compilers are very different from artificial-intelligence programs, they offer a useful precedent for thinking about the idea of an intelligence explosion, because they are computer programs that generate other computer programs, and because when they do so optimization is often a priority.

That’s all I want to tell you about how computers can make themselves smarter. Now let’s move on to next part of my presentation.

In the third part of my presentation, I’d like to tell you about why techological explosion is so usefull.

Obviously, none of this proves that an intelligence explosion is impossible. There is one context in which I think recursive self-improvement is a meaningful concept, and it’s when we consider the capabilities of human civilization as a whole. Note that this is different from individual intelligence. There’s no reason to believe that humans born ten thousand years ago were any less intelligent than humans born today; they had exactly the same ability to learn as we do. But, nowadays, we have ten thousand years of technological advances at our disposal, and those technologies aren’t just physical—they’re also cognitive.

The ability of humans to build on one another’s work is precisely why I don’t believe that running a human-equivalent A.I. program for a hundred years in isolation is a good way to produce major breakthroughs. An individual working in complete isolation can come up with a breakthrough but is unlikely to do so repeatedly; you’re better off having a lot of people drawing inspiration from one another. They don’t have to be directly collaborating; any field of research will simply do better when it has many people working in it. The rate of innovation is increasing and will continue to do so even without any machine able to design its successor. Some might call this phenomenon an intelligence explosion, but I think it’s more accurate to call it a technological explosion that includes cognitive technologies along with physical ones. Computer hardware and software are the latest cognitive technologies, and they are powerful aids to innovation, but they can’t generate a technological explosion by themselves. You need people to do that, and the more the better. Giving better hardware and software to one smart individual is helpful, but the real benefits come when everyone has them. Our current technological explosion is a result of billions of people using those cognitive tools. We’re a long way off from being able to create a single human-equivalent A.I., let alone billions of them. For the foreseeable future, the ongoing technological explosion will be driven by humans using previously invented tools to invent new ones; there won’t be a “last invention that man need ever make.” That’s all I want to tell you about why techological explosion is so usefull.

Before I stop, let’s go through the main parts of my presentation once again.

In conclusio I’d say that we needn’t worry about a superhumanly intelligent A.I. destroying civilization, we shouldn’t look forward to a superhumanly intelligent A.I. saving us in spite of ourselves. For better or worse, the fate of our species will depend on human decision-making.

Now, I’ll be happy to answer all you questions.

If you don’t have any further questions, thank you for your attantion.