THE MAGIC OF 256B INTROS

REPORT: NIPPON NATION 2018

SOLVING LOGICAL PUZZLES

SOLVING NUMERICAL PATTERNS

Genycs Magazine #5 – September 21st, 2018

EDITORIAL

Welcome to a new issue of Genycs Magazine!

As you see, the technical format of this magazine and the way it is presented have changed very much. Let me explain the reasons for this:

While I received plenty of feedback on every issue of the Hugi Magazine released between 1996 and 2014, I have received a comparably little amount of feedback on Genycs Magazine so far. What I have learned from the feedback I received is that there is some interest in the magazine but that the diskmag format is outdated. Moreover, the online edition is not an adequate replacement as it has been generated by automated conversion of the diskmag to HTML tags, and this automated conversion is not perfect.

I have therefore decided that I will publish the issue of Genycs Magazine which you are now reading as a PDF document. This makes it readable online, in the browser, regardless of whether you are using a desktop PC or a mobile device, or offline, after saving it to harddisk.

Note that it has also been suggested to me that I should use Wordpress. However, Genycs Magazine is not supposed to be a blog, it is supposed to be a magazine. Therefore, I do not think that Wordpress would be adequate.

I am curious if the choice of the PDF format has been a wise decision and if the amount of feedback will increase. As I already wrote in the previous issues, it is also not that I wish to create the entire magazine on my own. In other words: Articles written by the readers are welcome. If you want to publish your article in Genycs Magazine, simply send it to me via e-mail. Do not worry about orthography, I will proofread your article anyway. Don't be shy!

This issue features an article about 256b intros, one of the types of demoscene productions I am especially fond of and which has grown to be very interesting lately. Also, there is a report of the Nippon Nation 2018 event, a convention of the Austrian anime scene that was held in Vienna this July. Moreover, I have included two articles which I wrote about subjects related to artificial intelligence already some time ago. To be more precise, they deal with how to program computers to solve numerical patterns and logical puzzles.

Hopefully you will enjoy reading Genycs Magazine #5. I am waiting for your feedback and, perhaps, contributions (articles).

Claus Volko, cdvolko (at) gmail (dot) com

THE MAGIC OF 256B INTROS

The screenshot you can see here was taken from "Brains canyon", a 256-byte intro made by the Hungarian demogroup Abaddon. It was first presented only recently, in September 2018, at the Polish demoparty Riverwash, where it made the first place in the competition.

Of course this is only a static screenshot and thus gives an impression of one moment of the intro. But in order to see what the intro as a whole is like, you have to watch it on YouTube – or download it and run it on your PC using the DOSBox emulator (www.dosbox.com).

256-byte intros are one of the most fascinating things about the PC demoscene for me. This may be due to the fact that I have been fond of size-optimizing code written in x86 Assembler ever since I participated in the Pain Coding Contest, a size-optimizing contest organized by Dake of Calodox in the name of the Pain diskmag, back in 1998 and made the first place. Imagine: I was a more or less unknown 14-year-old dude and won the first place in a competition of a demoscene magazine! Of course this contributed very much to my self confidence (although it still took more than a decade until I fully developed this – I am talking about my self confidence).

I organized similar contests myself, which I called the "Hugi Size Coding Competition Series". The first contest of this series happened in 1998, and there were more than eighty people who participated in it. The last contest was in 2009; it was contest #29. As I was busy studying at university I did not host all of these contests myself; I was helped by an American pal of mine, who used to call himself Sniper. He organized a couple of contests instead of me. But for the majority of the contests, I was responsible.

This picture is a screenshot from "Bump Is Possible", which participated in the 256-byte intro competition at the Czech demoparty Fiasko 1999. It won second place. The creators of this intro called themselves Downtown; they are a Czech demogroup.

I asked a couple of coders of 256-byte intro a few questions as I wanted to learn more about this particular form of art. My first question was: What is special about making 256b intros compared to 4k intros? What do coders especially have to care about?

One of the answers I received was that there is no room for the Windows header in 256 bytes, so all 256b intros have to be made for the MS-DOS platform. It also makes the use of general code packers like APACK or UPX impossible; for data compression, in the words of TomCat of Abaddon, "you may write your own special routine if the decompression code is simple and small enough" (see his intro "Life&Life"), and "for saving lots of bytes from the code, you need hardcore Assembler skills and tricks". Seven further explained that in a 256b intro you cannot use 3D acceleration or shaders, and "even high-res modes are very rare – 320x200 with 256 colors is the norm". The fact that you cannot use a general code packer not only has its bad sides according to Seven, as it "gives you the freedom to use any obscure Assembly instruction to save bytes", while in a 4k intro, "you have to feed your compressor (typically Crinkler) with predictable data". As Seven says, "many coding tricks that save uncompressed bytes will result in a bigger compressed size, so they are not worth it in a 4k intro". Moreover, as there is no room for a softsynth, you have to use midi or (typically) no sound at all.

Ham also remarked that "the general opinion at first was that 256 bytes was barely the room for only one 'good' effect". This has been disproven by several multipart intros by now.

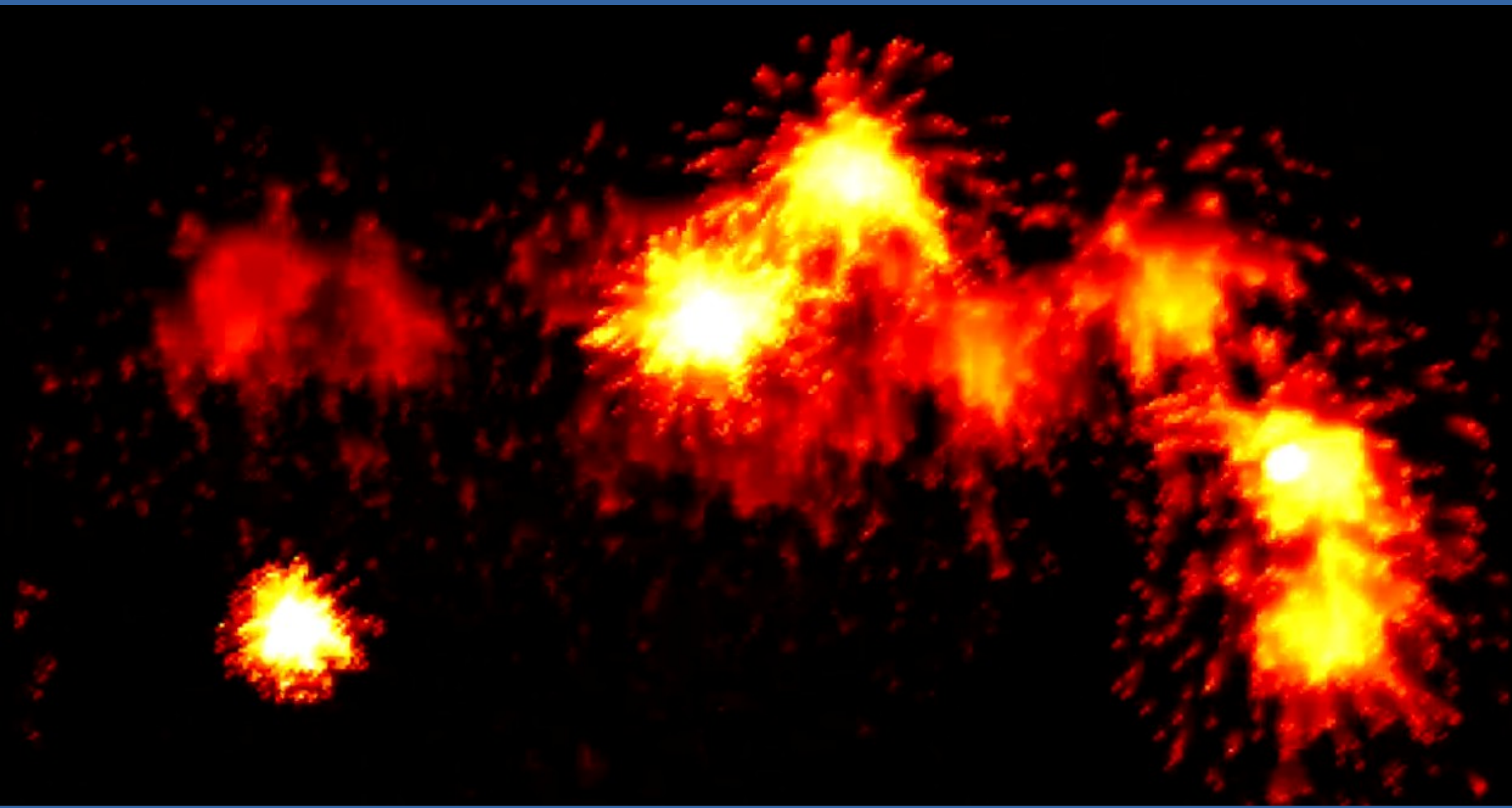
This screenshot is from "The effy++" by Brutal Core Divison – first place at Jumper 1998 (a demoparty in Hungary).

My second question was: "How much do the FPU, MMX and other advanced codesets help in creating 256b intros (to achieve the size limit)?"

HellMood replied that the FPU is important for 2D and 3D rotation, while MMX and other advanced technologies are unimportant. TomCat wrote an even longer answer to this question. Let me quote it entirely: "Code with integers is always smaller but some effects need floating point calculations and then the FPU is a must. I don't know about a 256b intro with MMX, but Kuemmel has shown with the TrueColor version of Codegrinder that the SSE could be an option despite of the size of instructions. The PentiumPro advanced instructions (FCOMI, FCMOVB, CMOVB etc.) really could help to achieve the size limit and hopefully more and more DOSBox variants will support them soon."

Basically this confirmed what I had already guessed. The FPU is definitely useful in size-optimizing because of its built-in instructions for calculating the sine, cosine and other trigonometric functions.

It is interesting to see in this context that the number of 256b intros showing 3D effects is growing. While in the 1990s most 256b intros displayed 2D effects, it now seems to be a requirement for winning a compo that the intro must be 3D.



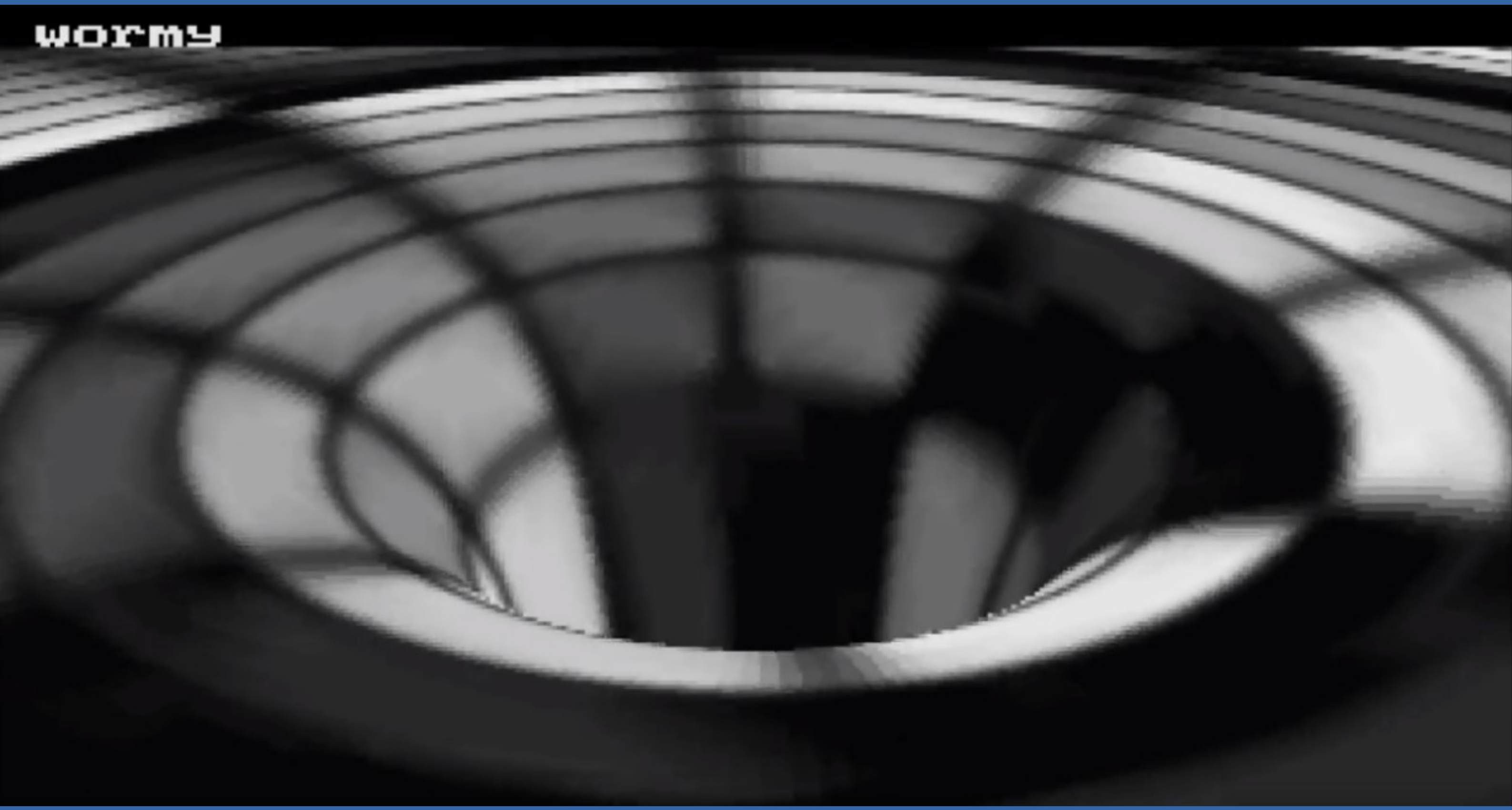
This is a screenshot of "Fireworks", coded by Picard, then a member of Hydrogen, and presented at the Hungarian demoparty "Scenest vs Rage 1998".

Picard was in fact one of the regular participants in the Hugi Size Coding Competitions. He regularly achieved top scores. This was what eventually made the Hungarian demogroup Exceed aware of him. Picard then joined Exceed and among other things, he made the 64k intro "Heaven Seven" and the demo "Spot". Especially the former is considered a true classic of the demoscene. I am somewhat proud that I, in a way, contributed to the making of this demo, as I was organizing the Hugi Compos and thus made Exceed meet Picard.

My third question was: "What are the greatest technical and aesthetic achievements in 256b intros so far?" Ham replied: "After all these years, tube is still awesome." Indeed, tube by baze of 3sc is one of the most outstanding 256b intros ever made. The Slovak coder baze released it at the Slovak demoparty Syndeecate 2001, and it is still among the all-time top 10 productions available at the popular demoscene website www.pouet.net.

HellMood has a similar opinion: "tube and puls had it all, progress nowadays is doing 3D and music in at most 64 bytes." The 256b intro puls was made by the Czech coder Řrřola and presented at the demoparty Riverwash 2009, where it made first place. You can find a download link at the website pouet.net as well.

And here comes TomCat: "Look at Řrřola's intro pyrit at Demobit and now at Function. This RayTracing quality is a great achievement. I was also happy at my Seashell intro (HiRes TrueColor with Z buffer). And also don't forget Digimind's ducks. Another new thing is the real music (not just sound) synced with visual (see TCTRONIC intro)."



This screenshot is from the intro Wormy, which was presented at Xenium 1999.

My final question was: "What resources would you recommend for learning to create 256b intros?" Most of the coders answered that the website www.sizecoding.org is a good starting point. TomCat also recommends taking a look at source code of "asm gurus" like Řrřola, frag and others. Besides, there are videos of seminars (e.g. those delivered by Řrřola at Demobit 2017 and 2018) available at YouTube. For ham, you mainly need "an Assembler, good knowledge of your platform, and a lot of patience". Xtr1m recommends disassembling a 256b intro "if the source is not provided".

Let me finally quote a statement which HellMood wanted to make in addition to the answers to these question: "Oh well, and Protected Modes, HiRes (up to 1280x1024), TrueColor, ARE a thing. People slowly adapt it. As well as real music. Snippets on sizecoding.org suggest that there is MUCH more possible in 256 bytes than expected. We have fractals in 16 bytes, bass line in 8 bytes, interactivity (paint) in 16 bytes, Fake 3D in 32 bytes, Raycasting in <64 bytes. Custom colors in <16 bytes. The -documented- toolkit is there, it's up to the people to creatively assemble it. Literally."

As a matter of fact, I also tried twice to create an intro sized between 128 and 256 bytes myself. The first result I called "Indian Summer" (released at 0a000h 2008), the other "Indian Winter" (released at Demobit 2018). I wrote about the latter in Genycs #1.

I am looking forward to what's happening in the 256b scene, and maybe I will get to coding yet another intro myself as well!

Claus Volko, cdvolko (at) gmail (dot) com

REPORT: NIPPON NATION 2018

I have been somewhat active in the computer demoscene as a teenager, editing a diskmag (Hugi Magazine; it issued from 1996 to 2014 on a more or less regular basis) and occasionally participating with own works in the 256b intro and game development competitions. I was almost the only Austrian participating in the demoscene in those years. There was a small scene in Vienna but not much anywhere else. Even the people in Vienna were more or less inactive.

I only got to know by chance that while the demoscene is pretty dormant in Austria, there is another scene which is far more active. Moreover, while I always used to be one of the youngest involved in the demoscene, this other scene matches my own age group much better; as a matter of fact, I would even belong to the oldest people in that scene. Most are younger, some even far younger than me.

This scene is the anime scene. A scene which mostly focuses on Japanese popular culture – anime, manga, video games.

The anime scene has several meetings on a regular basis here in Austria. One of the best known meetings, or "conventions", as they call it, is the AniNite. It takes place every August either in Vienna or a suburb of Vienna. Another one is the Nippon Nation, which happens every July. The event venue is the Museumsquartier close to the center of Vienna.

The Museumsquartier is located near the Ringstraße, which is a circular road encapsulating the innermost city ("Innere Stadt"), the first district of Vienna where most of the historical buildings are located. The name comes from the fact that it is close to two of the largest museums of Vienna, the Naturhistorisches Museum and the Kunsthistorisches Museum (museums of natural history and art history).

The venue can be easily reached with the Viennese subway. There is even a station of the U2 subway line called "Museumsquartier".

As you can see in the photo, there was quite a crowd waiting for being let into the venue a couple of minutes before the doors officially opened. The anime scene in Austria is pretty big; far bigger than the demoscene.

In the anime scene it is common that people dress up in fancy ways. Some put on costumes that resemble anime or videogame characters. As you can see in this picture, there was a guy who had a knapsack that looked like a controller of the Nintendo Entertainment System (NES) console, one of the best selling video gaming consoles of the 1980s.

One of the activities of the anime scene is the so-called cosplay contests. In these contests, people dress up in such a way as I have just described it. Then the audience votes which costumes are the most impressive ones. The most impressive costumes get prizes.

Of course, there was also a cosplay contest at Nippon Nation 2018. But to be honest, I did not stay at the event venue for a long time and therefore was not able to experience this contest myself. However, what I did was that I tried to explore the whole venue and get an overview of what was happening at the event.

Basically, after the entrance hall you could either go downstairs to a kind of theatre hall where special performances happened, or you could go to a large hall where a lot of vendors were selling stuff such as dolls or comic books. Moreover, there was a fairly big space in that hall where people were playing video games.

In the room before the theatre hall, downstairs, there were a few tables which were rented by the "Kasu" club. This club focuses on traditional Asian board gaming. One of the most popular games among members of this club is Mahjongg, which the people you see in this photo are playing. It is quite different from the computer game with the same name (which is sometimes also called Shanghai).

The club "Kasu" has a room in an apartment in the seventh district of Vienna where its members meet on Fridays and Sundays. You can take a look at the official website www.kasu.at to find out when the next meeting will happen. Usually they do not charge money from first-time visitors. Only if you regularly attend their meetings, you will have to pay.

As a matter of fact I was cordially invited by some members of Kasu when I met them at Nippon Nation 2018. A couple of weeks later I went to the club meeting indeed and had a nice time playing a logical puzzle game. Most of the members of Kasu seem to be about the same age as me (I was born in 1983).

As already mentioned, a part of the large hall was dedicated to video gaming. This was also what most of my time spent at Nippon Nation 2018 was consumed by: I enjoyed very much playing a game that was like an egoshooter but instead of shooting other players, you had to colorize your surroundings.

There were very young people playing video games as well as other people my age. Video gaming is a phenomenon that connects the generations, except for the really old ones – but that will change with time.

All in all I enjoyed my time spent at Nippon Nation 2018 very much. It was worth the entrance fee. I sometimes regret having discovered the anime scene only when I was already a university student, as the anime scene would have brought me opportunities which the demoscene did not offer, such as real-life, face-to-face meetings. But: It's never too late! I enjoy attending anime conventions ever since I attended my first AniNite, which was in the year 2003, if I remember it correctly.

I am already looking forward to next year's Nippon Nation!

Claus Volko, cdvolko (at) gmail (dot) com

SOLVING LOGICAL PUZZLES

This article is about how you can solve tasks from the PM Logic Trainer with the help of a computer.

First of all: Yes, you can program computers to solve these tasks! It is true that intelligence is not specifically human. At the TU Vienna one learns that intelligent behaviour is probably only a question of calculation and therefore one can easily program the computer in such a way that it shows intelligent behaviour, which one might have only expected from humans so far.

The only difficulty in solving logic trainer tasks with a computer is translating the clues into a language that the computer can understand. The processing of natural language has not yet progressed that far. But: Once translated, the computer can easily solve the logic trainer tasks! Let's take a look at a typical logic trainer task: Three girls, three fruits, three sweets. In these tasks it is always true that exactly one fruit is eaten by exactly one girl, exactly one sweet is preferred by exactly one girl, and exactly one fruit is associated with exactly one sweet. In order to facilitate the solution, the PM Logic Trainer usually provides a table such as the following:

	Zitrone	Apfel	Birne	Bonbon	Zucker	Lolli
Anna						
Berta						
Claudia						
Bonbon						
Zucker						
Lolli						

Suppose the first note is that Anna likes to eat lemons. This means that we put a plus in this table in the cell connecting Anna and the lemons. But since only Anna eats the lemon and not the other girls and Anna eats the lemon and not the other fruits, a plus in the cell "Anna - Lemon" means that we can put a minus in the same box in the cells which are in the same column and in the same row. The rule that a plus automatically implies a minus in the same row and the same column is also known by a computer programmed accordingly. Internally, the computer will work with a data structure that is very similar to this table. It will know: If I put a plus somewhere, then I can put a minus somewhere else. This method is called constraint propagation. It means: A constraint, i.e. a restriction of the possibilities where I can set a plus or a minus, also has effects on other fields; I propagate this constraint further. "Constraint Propagation" is the most important method with which a computer program can solve logic problems.

	Zitrone	Apfel	Birne	Bonbon	Zucker	Lolli
Anna	+	-	-			
Berta	-					
Claudia	-					
Bonbon						
Zucker						
Lolli						

If a further hint now reads that Anna does not like sweets, then this means in the first place that I have to set a minus in the field "Anna - Sweets", but it also has the effect that I have to set a minus in the field "Sweets - Lemons", because Anna is positively connected with the lemon. This means that everything that applies to Anna also applies to the lemon, and vice versa. The computer simply has to be programmed so that every time a minus is set, it checks whether there is a plus in the same row or in the same column; if this is the case, then it has to "propagate" this constraint.

The tasks in the PM Logic Trainer can be solved without exception by constraint propagation. In principle, the computer proceeds in exactly the same way as a trained human puzzle solver would.

Apart from that, there is another possibility for the computer to solve logic problems: namely by systematically testing all possibilities and checking whether they fulfill the constraints. This method is called "backtracking". It is slow, but always leads to the goal, even if "Constraint Propagation" is not enough (which can happen for example in Sudoku puzzles).

I hope the article was interesting and you could learn from it.

Claus Volko, cdvolko (at) gmail (dot) com

The original version of this article was published in MATHEMATIQ #25, from 2013, in German.

SOLVING NUMERICAL PATTERNS

In common intelligence tests, series of numbers frequently occur that have to be completed. For example 3, 5, 8, 12 – what is the next number? If it's a sign of intelligence that humans can solve these tasks, the question arises as to whether this is a specifically human property. Or can computers also solve number series?The answer is: yes, they can, if they have been programmed accordingly. In the following article I would like to show you how it works.

Let us start with very short series of numbers, where only two numbers are given. 1, 2 - what is the next number in the series? There are two possibilities: 3 or 4. For the time being, we would like to restrict ourselves to additive series of numbers; in this case, we would like to have the 3 correct. If the number series is generally a, b, then how do you get to c? One calculates c = b + (b - a) = 2b - a. That would be a general solution formula for additive number series that consist of only two numbers. If we now have three have numbers, a, b, c, how do we then calculate the fourth number d? Example: 1, 2, 3. Well, in this case there are two possibilities: d = c + (c - b) = 2c - b or d = c + (b - a). Both lead to the same result. However, this is only true because the difference c - b is equal to b - a. What if they were different? Example: 1, 2, 4. As we assume that it is an additive series of numbers, the result is 7 (2 - 1 = 1, 4 - 2 = 2, so we have to calculate 4 + 3 = 7). Symbolically we have calculated: d = c + (c - b) + ((c - b) - (b - a)) = 3(c - (b) + a. This is the general formula for additive number series consisting of three numbers This allows us to solve any series of numbers of this type.

For number series consisting of four or more numbers, you can use analogous formulas to make the whole thing more descriptive. To make the whole thing a little more vivid, we use the following notation: b' = b - a; c' = c - b; etc., further: c'' = c' - b', d'' = d' - c' etc. For a number series which consists of four numbers then results as a solution formula: e = d + d' + d'' + d'''.

If you want to program a computer so that it is able to solve additive number series automatically, there are two possibilities: Either you calculate simple formulas for number series from two, three, four,... Numbers. Then it is sufficient to insert them into these formulas. This makes the calculation very efficient. Or you program the computer so that it is able to solve series of numbers consisting of any number of numbers. In this case you store the last number (e.g. f), then calculate b' to f' and add f' to the result, then calculate b'' to f'' and so on. Since you can overwrite the numbers of the current generation (once you have buffered f, you can overwrite f with f', then e with e', etc.), there is no additional memory requirement, but the calculation takes longer. The number of subtractions that have to be done is (if n is the number of given numbers) n(n - 1)/2, in the language of computer scientists $O(n^2)$. This is not exactly efficient, but this approach allows to work with arbitrarily long series of numbers.

However, this second approach is also helpful if we want to supplement our program so that it can also cope with series of numbers containing multiplicative elements. If you treat 1, 2, 4 as a multiplicative number series, the solution is not 7, but 8. This is easily achieved by proceeding exactly as before, but not calculating differences, but quotients, and not adding the result, but multiplying it. So solving purely multiplicative number series is not a problem either. But how to solve 1, 3, 7 or 1, 4, 10? First you should try the multiplicative approach. But since 7 cannot be divided by 3 without a remainder (and 10 cannot be divided by 4), you have to proceed additively. The resulting differences are 2, 4 or 3, 6. Now we can proceed multiplicatively. So the solution formula for the first row is d = 7 + 4 * 2 = 15 and for the second row d = 10 + 6 * 2 = 22. By first trying the multiplicative approach at each step and, if this does not work, proceeding additively, we can solve such mixed series of numbers. But of course point calculation before line calculation is valid. We cannot, as with purely additive or purely multiplicative number series, simply add the difference c' to c, but we must first multiply the difference c' by the quotient "c". This can be achieved by working with a primitive stack. You store in the variable r1 the value of c between, then in r2 the value of the difference c'. If in the next step the additive approach would take place again, one could add the value of r2 to r1 and store "c" in r2. But since we now proceed multiplicatively, r2 must be multiplied by "c" and only in the next step r2 must be added to r1.

So you can see that some of what man considered to be specifically human talents can also be accomplished by computers - but there is still a need for a clever man who is able to program the computer accordingly.

Claus Volko, cdvolko (at) gmail (dot) com

The original version of this article was published in MATHEMATIQ #29, from 2014, in German.

LEGAL STUFF

Official website of Genycs Magazine:
http://www.genycsclub.net/

RELATED LINKS

Hugi Magazine:
http://www.hugi.scene.org/

Hugi Size Coding Competition Series:
http://www.hugi.scene.org/compo/

MATHEMATIQ:
http://www.hugi.scene.org/adok/mensa/mathsig/

Claus Volko:
http://www.cdvolko.net/
http://www.hugi.scene.org/adok/cdvolko/

Claus Volko, cdvolko (at) gmail (dot) com