Impact Factor: 9.9 ISSN-L: 2544-980X

Recursion and Him in Programming to Use

Tokhirova Sarvinoz Gayratjon Kizi [1]

Abstract: Function to himself himself straight away or some kind of tool through appeal to do to the process recursion is called and such function recursive is called a function. But it's just that definition using right and error free working recursive function make up it 's difficult for recursive of the function main conditions good to know need.

Key words: Recursion, function, Tree, Graph, Heap, QuickSort, MergeSort.

Recursion functional of programming main from the elements is considered Still functional programming about didn't hear if you are about him information looking for by reading to see advice I will give One word with so to speak , now programming field momentum with functional programming paradigm towards going (Go and Scala bright samples).

Any right made up recursion basis two a must organize does

1. **Recursion basis condition**
2. **of the function to himself with the modified argument appeal to do**

Recursive function which one to time come to himself appeal to do to stop need will be Exactly that's it thing recursion basis condition providing gives In our story for example returning if so , to *the sum()* function one how many times appeal did and at the end to the function coming in the array only when one element remains stopped . For this issue leaving only one element in the array basis a must be service does and that's it to the ground when it arrives program to stop need knowing takes Recursive function in making basis condition right to put very important is considered Not yet again we will stop.

Next provided The modified argument is usually the one at the beginning of the issue from the argument a smaller argument is understood (some cases bigger to be can). In our example , every trip to *the sum( )* function appeal when you do in it massive size to one reducing went This thing is very important because one with a different argument repeatedly appeal when done or the argument is invalid when changed function himself infinite times to call right come remains . More on that too again let's talk

Why recursion need

Actually when , *har how recursive processed issue iterative method work possible and of this the opposite is also true* . Of this on top of it *recursive solution each always from memory requires additional space* .

So so what for in it recursion do you need Of course it is enough reasons have :

Recursion almost everyone in the place is used . That is , lo' nda by doing so to speak from him running away get rid of possible no Making an effort to see while expensive fall sure)

Some cases recursive solution much simpler . Especially some of issues iterative the solution too long be leaving can Recursion while the code one how many equal to shortening to give can The majority structures and algorithms without recursion imagination by doing it won't happen . **Tree, Graph, Heap, QuickSort , MergeSort ,** … the list goes on very long continue carry on can Especially complicated structures Tree and In graphs recursion each in step occurs . Software while without them imagination by doing It wo n't be while own in place recursion how much importance set gives

[1] Tashkent University Farg is a mother branch, Muhammad Al- Khorazmi in the name of information technologies

Recursion functional of programming main from the elements is considered Still functional programming about didn't hear if you are about him information looking for by reading to see advice I will give One word with so to speak , now programming field momentum with functional programming paradigm towards going (Go and Scala bright samples).

Another one interesting information , so programming languages there is in them in general recurrence operators no and this about completely to recursion relies on Haskell and Erlang are including Of course , these all of them recursion repetition from the operators completely superiority does not mean Actually , many p cases programmers recursion from using they run away Of this main reasons while :

Recursive in the solution error by doing probability high As we said before , recursion very distracting . Therefore , him in use easily error by doing to put can Recursive the solution mistake to find difficult Such in matters error by doing to put probability high to be with together him found it is also difficult to correct to be can Of this main the reason is that solutions imagination by doing get (fantasy debug) too difficult

Recursive of the algorithm complexity count most of the time very complicated . Even sometimes right the solution of writing is also less being stay can Because , him iterative solution with in comparison his complexity count need will be Recursive in algorithms this most of the time very complicated and well requires mathematics .

```
int fact( int k)
{
if (k==1)
return 1;
otherwise
return k * fact(k-1);
```

In mathematics recursion .

In mathematics recursion functions and numerous of rows identification

methods relatively applied : recursive respectively given function own value another arguments with to call through is determined . From this without there are two options :

Complete (numerical) recursive functions . Such functions finite in the thigh recursive for appeal (call). optional numerous in the argument without recursion considered separately identifiable private of cases to one take to arrive through is given Classic example : negative didn't happen whole of the thigh recursive to be determined :

$$n! = \begin{cases} n \cdot (n-1)!, & n > 0 \\ 1, & n = 0 \end{cases}$$

Here each one next recursive the argument in the call is one to unity becomes smaller . That is , to be determined according to negative didn't happen whole of the thigh factorial calculator function argument is recursive to **n** appeal to do **0!=1** private to the situation when it comes to count stops . So is recursive to be determined despite , for an optional argument this function identification in the field completed function will be

- *Unlimited recursive functions* . Such functions all in cases (none when not some arguments for) to itself appeal in the form of is given An example as infinite numerous rows , infinite continuous fractions and others get can

In mathematics to recursion examples :

- ✓ linear algebraic equations system solve for Gauss -Jordan method

- ✓ recursive is considered
- ✓ negative was not whole of the thigh factorial count
- ✓ Fibonacci the number recurrent relationship using is determined , i.e Fibonacci number the first and second terms are equal to 1 . n- Fibonacci for n>2 number (n-1)- and (n-2)- Fibonacci of the thighs to the assembly line equal to
- ✓ In practice all geometric fractals infinite recursion through is given for example , Serpin's triangle).

REFERENCES

1. Tokhirova Sarvinoz Gayratjon kizi, & Siddiqov Murodali. (2023). NATIONAL ECONOMY AND ITS MACROECONOMIC INDICATORS. Best Journal of Innovation in Science, Research and Development, 152–156. Retrieved from https://www.bjisrd.com/index.php/bjisrd/article/view/980
2. MILLIY IQTISODIYOT VA UNING MAKROIQTISODIY KO‘RSATKICHLARI. (2023). Journal of Technical Research and Development, 1(2), 402-409. https://jtrd.mcdir.me/index.php/jtrd/article/view/81
3. Обухов, В., & Тохирова, С. (2023). МИКРОПРОЦЕССОРНЫЕ СИСТЕМЫ И ИХ ПРОИСХОЖДЕНИЕ. Journal of technical research and development, 1(2), 32-37.
4. Mahmudova, M., & Toxirova, S. (2023, October). MULTISERVISLI TARMOQ XAVFSIZLIGIDA NEYRON TARMOQLARINI O ‘RNI. In Conference on Digital Innovation:" Modern Problems and Solutions".
5. TURANBAYEVNA, K. N., & XUSENOVNA, T. S. (2020). Development of Communicative Didactic Competence of High School Students. International Journal of Innovations in Engineering Research and Technology, 7(12), 45-47.
6. Toxirova, S. (2023). Python dasturida lug’atlar bilan ishlash . Conference on Digital Innovation : "Modern Problems and Solutions". извлечено от https://fer-teach.uz/index.php/codimpas/article/view/1910
7. Toxirova, S. (2023). METHODS OF WORKING WITH SCHOOLCHILDREN WHO CANNOT LEARN. Conference on Digital Innovation : "Modern Problems and Solutions". извлечено от https://fer-teach.uz/index.php/codimpas/article/view/1909