Task №1 (Sum of array)

It is necessary to create an array *a* with the size of *N* elements. It is assumed that the launch of the executable file will take place using *p* processes. One of the *p* processes (which will be the "main" process) fills the array elements as follows: *a[i] = i*, where *0 <= i < N* – index of the array. After filling the array, the same "main" process:

splits the array *a[N]* into *p* parts and sends the elements of each of the *(p-1)* parts to the rest *(p-1)* processes (the "main" process works with one of the parts of the array);

Further, each of the processes, that received their volume of elements must find the sum of these elements and send it back to the "main" process.

The "main" process receives all the sums from the worker processes and, adding them, gets the sum *S* of all elements of the array *a[N]*.

The task:

1) Display in a column the amounts calculated by each of the worker processes, indicating its number.

2) Display the sum *S* of all elements of the array, calculated by adding all the sums received by the "main" process from the worker processes.

3) Display the sum S_0 of all array elements, calculated by the "main" process sequentially. Compare it with the sum *S*.

4) On the one coordinate plane, draw 3 graphs of the speedup *S* on number of processes *p* dependence, where *p = 1,2,4,8,16* for N = 1000, N = 10^4 and N = 10^5.

Notes:

1) Think about splitting the array into parts when *N* is not evenly divisible by *p* (you have a remainder).

2) Use the following programming feature: if you do not refer to any of the processes through the *if statement*, but write the code in the general part, then this piece of code will be executed by all processes in the same way.

3) If a variable is declared in general part of the program, then it has the same name in different processes, but it can take different values in different processes.