MINISTRY OF SCIENCE AND HIGHER EDUCATION
OF THE RUSSIAN FEDERATION
Federal State Autonomous Educational Institution
higher education "NORTH CAUCASUS FEDERAL UNIVERSITY"

APPROVE
Acting Director of the Institute,
Candidate of Technical Sciences,
Associate Professor A.A. Porohnia

ASSESSMENT TOOLS FUND FOR DISCIPLINE

Parallel computing
name of the discipline

Direction of training	09.04.03 Applied Informatics Knowledge Management
Focus (profile) Year of commencement of studies	2026
Form of study	full-time
Implemented in a semester	1

Introduction

1. Purpose: FOS is intended for an objective assessment of the level formation of competencies.

2. FOS is an appendix to the program of the discipline "Parallel computing" / Parallel computing»

3. Developer Amirokov S.R., associate professor.

4. An examination of the FOS was carried out. Members of the expert group:

Chairman Azarov I.V., Acting Director of the Department

Members of the commission:

Orlova A.Yu., Associate Professor

Orlinskaya O.G., Associate Professor

Representative of the employer organization Cherevko S.A., General Director of KRAFTKOD LLC (*Full name, position*)

Expert opinion: FOS for the discipline allows to assess the level of formation of competencies. Proceed to testing.

5. The validity period of the FOS is determined by the period of implementation of the educational program.

1. Description of the criteria for assessing competence in various stages of their formation, description of assessment scales

Levels formation and competencies, indicator(s)	Descriptors			
	Minimum level not achieved (Unsatisfactory But) 2 points	Minimum level (satisfactory) O) 3 points	Intermediate level (Fine) 4 points	High level (Great) 5 points
UK-6				
ID-1 UK-6. reaches result by way self-development.	With difficulty reaches result by self-development	Reaches result by self-development is not in in full	Overall it reaches result by self-development	reaches result by self-development
ID-2 UK-6. uses methods goal setting.	Uses with difficulty methods goal setting	Uses little methods goal setting	Generally uses methods goal setting	uses methods goal setting
OPK-2				
ID-1 OPK-2. develop original algorithms For solutions professional x tasks;	With difficulty is developing original algorithms for solutions professional tasks	Weak is developing original algorithms for solutions professional x tasks	Mostly is developing original algorithms for solutions professional x tasks	is developing original algorithms for solutions professional x tasks
ID-2 OPK-2. develop software means For solutions professional x tasks	With difficulty is developing software means for solutions professional tasks	Weak is developing software means for solutions professional x tasks	Mostly is developing software means for solutions professional x tasks	is developing software means for solutions professional x tasks
PC-3				
ID-1 PC-3. develop fundamental And interdisciplinary knowledge	Cannot develop fundamental and interdisciplinary e knowledge	Allows serious mistakes V developme nt fundamental And interdisciplinary knowledge	Mostly accurate develops fundamental And interdisciplinary knowledge	develops fundamental And interdisciplinary knowledge
ID-2 PC-3. Develop methods scientific research	Develops with difficulty scientific methods research	Allows serious mistakes V developme nt methods scientific research	Mostly accurate develops methods scientific research	develops methods scientific research
PC-10				
ID-1 PC-10. deploy, informational systems, th eir components An informational services	With difficulty deploys, informational systems, their dcomponents and informational services	Weak deploys, informational systems, their components and informational services	Generally deploys, informational systems, their components and informational services	deploys, informational systems, their components and informational services

ID-2 PC-10 install informational systems, their components An informational services	With difficulty installs informational systems, their dcomponents and informational services	Weak installs informational systems, their components and informational services	Generally installs informational systems, their components and informational services	installs informational systems, their components and informational services

The assessment of the level of development of competence in the discipline is carried out on the basis of the “Regulations on the current monitoring of academic performance and midterm assessment of students in higher education programs - bachelor's degree programs, specialist programs, master's degree programs - at the federal state autonomous educational institution of higher education "North Caucasus Federal University" in the current version.

ASSESSMENT TOOLS FOR CHECKING THE LEVEL OF COMPETENCY DEVELOPMENT

Nom er task nia	Correct answer	Contents of the question	Competence
		Full-time education, semester 1	
1.	A parallel program within the framework of MPI is understood as set simultaneously running processes. Processes can be run on different processors, but several processes can be located on one processor (in this case, their execution is carried out in time-sharing mode). In the extreme case, one processor can be used to run a parallel program - as a rule, this method is used for the initial verification of the correctness of a parallel program.	The concept of a parallel program	
2.	The processes of a parallel program are combined into groups. Another important concept of MPI, describing a set of processes, is the concept of a communicator. In MPI, a communicator is a specially created service object that combines a group of processes and a number of additional parameters (context) used when performing data transfer operations. Pairwise data transfer operations are performed only for processes belonging to the same communicator. Collective operations are applied simultaneously to all processes.	The concept of communicators	

	one communicator. As a result, the indication of the used communicator is mandatory for data transfer operations in MPI.		
3.	The first MPI function called must be: int MPI_Init(int *argc, char ***argv), where argc — pointer to the number of command line parameters, argv — command line parameters used to initialize the execution of the MPI program. The functioWnednesday parameters are the number of arguments in the command line and the address of the pointer to the array of characters of the text of the command line itself.	Initialization of MPI programs	
4.	One of the most common methods of classifying computers is the Flynn taxonomy, which focuses on the analysis of the architecture of computing systems. methods interactions sequences (streams) of performed commands and processed data.	What is the basis of Flynn's classification?	
5.	1.3	What characterizes concurrency in the context of parallel computing? (1) ensuring the minimum execution time of one program (2) Primacy of throughput (3) there is no need to ensure maximum isolation of processes from each other (4) ensuring that resources are distributed as evenly as possible between processes	
6.	3.4	In what situations can true parallelism of computation be realized?	

		(1) calculations are performed on a computer with a single-core processor in a multitasking OS (2) calculations are performed on a computer with a single- core processor in a single-task OS (3) the calculations are performed on a multiprocessor device (4) a processor that supports physical vectorization is used for calculations	
7.	3	Which frommodes calculations supports classical consistent computer background Neumann? (1) processing multiple instructions and a single data element at a time (2) processing a single instruction and multiple data streams at any given time (3) processing a single instruction and a single data element at a time (4) processing multiple instructions and multiple data streams at any given time	
8.	1	A symmetric multiprocessor is characterized by (1) uniform access to the memory of all processor units (2) access of each processing device to a separate part of physically distributed memory (3) non-uniform access to the memory of all processing devices (4) uniform access of some processor devices to some common memory	
9.	1.3	What factors prevent obtaining results with the expected accuracy when parallelizing arithmetic calculations? (1) the machine operation of adding numbers does not have the property of commutativity (the order of adding two numbers is important)	

		(2) parallelized algorithms are implemented on a single-core processor (3) the machine operation of multiplying numbers does not have the property of associativity (the order of multiplication of three numbers is important) (4) the machine operation of multiplying numbers has the property of commutativity (the order of multiplying two numbers is not important)	
10.	1,2,3	Which of the proposed strategies for parallelizing the algorithm for finding the arithmetic mean of a sequence of 1000 numbers are correct? (1) the sequence is divided into 4 parts, the elements in each part are summed and divided by 1000 on a separate processor unit, the resulting values are added on one processor unit and divided by 4 (2) the sequence is divided into 4 equal parts, the elements in each part are summed up, the resulting values are added up and divided by 1000 on one processor unit (3) the sequence is divided into 4 equal parts, the elements in each part are summed and divided by 1000 on a separate processor unit, the resulting values are added together on one processor unit (4) the sequence is divided into 4 equal parts, the arithmetic mean of each part is found on a separate processor device, the resulting values are added together on one processor device and divided by 1000	
11.	1.4	Select conditions feasibility schedules parallel algorithm. (1) on each computing unit all operations are performed one after another	

		(2) the number of operations performed on each computing device is constant (3) computing devices performing different operations cannot exchange information with each other (4) each operation is performed on no more than one computing device	
12.	4	What is called the length of the critical path in a graph representing some parallel algorithm? (1) the length of the maximum path in the graph (2) the maximum length of a path in a graph consisting of identical operations (3) average height of the algorithm graph (4) the diameter of the algorithm graph, which determines the minimum theoretically possible execution time of the algorithm	
13.	1,3,4	How characterized by acceleration parallel algorithm? (1) minimum execution time of a sequential algorithm (2) the ratio of the number of processors to the number of execution threads (3) minimum execution time of a parallel algorithm (4) input data size	
14.	1	What is the efficiency of a parallel algorithm? (1) the ratio of the speedup of the algorithm to the number of processors (2) the product of the minimum execution time of a parallel algorithm and the number of processors (3) the ratio of the size of the input data to the size of the output data (4) the total time spent by all processors to execute the algorithm	
15.	2,3	Select the correct statements.	

		(1) the acceleration of a program using parallel computing depends only on the number of computing nodes (2) the total execution time of a parallel task is not less than the execution time of the longest sequential fragment (3) in real problems, adding new processors can increase the calculation time (4) in real problems, program acceleration using parallel computing cannot be achieved by adding computing nodes	
16.	1,3,4	Which parts of the program are sequential? (1) reading input data from hard disk (2) Write output data to multiple hard drives (3) synchronization in a parallel program (4) critical section in a parallel program	
17.	1,1	Find the acceleration of scaling according to Gustavson's law some parallel programs, if it is known that the time of the sequential part of the program is equal to 1000 ms, time of the part of the program that can be parallelized, equals 100 ms, quantity processors are equal 10. Round your answer to tenths.	
18.	1,1	Find the acceleration of scaling according to Gustavson's law some parallel programs, if it is known that the time of the sequential part of the program is equal to 800 ms, time of the part of the program that can be parallelized, equals 100 ms, quantity processors are equal 10. Round your answer to tenths.	
19.	3	What is task parallelism?	

		(1) combination of the original calculations into a larger complex problem (2) decomposition of the original problem into several smaller ones by a recursive procedure (3) decomposition of the original problem into several smaller ones by a linear procedure (4) merging similar tasks into a larger one using a recursive procedure	
20.	4	What type of computing problem is called embarrassingly parallel? (1) tasks to which the divide and conquer paradigm does not apply (2) tasks with a large number of internal connections (3) any tasks whose computation can be implemented using a parallel algorithm (4) tasks with a large number of computational subtasks that do not have dependencies between themselves	
21.	1.3	When is pipeline processing used for computing? (1) data arrives in a continuous, one-way, regular stream (2) data comes in the form of an unstable two-way stream (3) each element of the data set must be processed in several stages (4) data is received once and requires an individual approach to processing and analysis	
22.	1.4	When is event-based coordination used for computation? (1) data arrives as a regular one-way stream	

		(2) the data structure used is characterized by unpredictability of interactions between its constituent parts (3) each element of the data set must be processed in several stages (4) a two-way data flow is used	
23.	1,2,4	Select methods solutions Withlocal interactions between subtasks. (1) Gauss-Seidel method (2) Red-Black method (3) reduction using the “divide and conquer” method (4) chaotic relaxation method	
24.	2.4	When is dynamic scheduling with load balancing used for computing? (1) the number of subtasks is much greater than the number of processors (2) the number of processors in a computing system changes over time (3) subtasks vary greatly in size (4) uniform loading of processors is required, but the computing system contains heterogeneous processors	
25.	1.3	In which cases is static scheduling with load balancing used for computing? (1) the number of subtasks is much larger than the number of processors (2) the number of processors in a computing system changes over time (3) between subtasks arise unstructured interactions (4) uniform loading of processors is required, but the computing system contains heterogeneous processors	

26.	processes processing data, V which several machine operations can be performed simultaneously	Parallel computing refers to…	
27.	- independence of the functioning of individual computer devices - this requirement applies equally to all the main components of the computing system - to input-output devices, to processing processors and to memory devices; redundancy of elements of the computing system - the organization of redundancy can be carried out in the following main forms: usage specialized devices such as separate processors for integer and real arithmetic, devices multi-level memory (registers, cache); duplication of computer devices by using, for example, several identical processing processors or several random access memory devices.	Achieving parallelism is possible only if the following architectural requirements are met: principles constructions computing system	
28.	These are computers that have multiple processors that share memory. This class includes most of the multi-core computers sold on the market today.	Multiprocessor computing systems are	
29.	represent set computers, connected by high-speed communication lines. Each computer has its own memory and exchanges messages with other computers in the system to transfer data. This class includes clusters. A cluster is understood as computing complex, considered as a single entity, with some dedicated computer playing the role	Multicomputer computing complexes - …	

	servers. Because the computers that make up a cluster can be regular computers, clusters are relatively inexpensive. Most of the supercomputers in the Top 500 are clusters		
30.	consist of many nodes, each of which can be multicomputer, multiprocessor, graphics or vector processor. Such complexes are usually supercomputers	Hybrid computing systems - …	
31.	creating a parallelism resource (obtaining a parallel algorithm) in problem solving processes and control the implementation of this parallelism Withpurpose to achieve the greatest efficiency of using multiprocessor computing technology	Parallel computing problem– …	
32.	a group of computers connected to a local area network and capable of operating as a single computing resource	Cluster – …	
33.	a data processing technology in which a large task is distributed among many people for execution computers, united computer network or the Internet	Distributed computing –	
34.	If, when solving a problem, processors perform the same sequence of calculations but use different data, then we speak of data parallelism. For example, when searching a database, each processor can work with its own part of the database. If processors perform different tasks of one task, perform different functions, then we speak of functional parallelism.	Data parallelism. Functional parallelism.	

35.	Let the operation be divided into micro-operations. Let us arrange the micro-operations in the order of execution and allocate a separate part of the device for each execution. At the first moment of time, the input data is received for processing in the first part. After the first micro-operation is executed, the first part passes the results of its work to the second part, and itself takes new data. When the input arguments have passed all stages of processing, the result of the operation execution will appear at the output of the device. In this way, functional parallelism is implemented. Each part of the device is called a pipeline stage, and the total number of stages is the pipeline length.	Pipelining of operation data (pipelining parallelism at the operation level)	
36.	The classification is based on the concept of a flow, which is a sequence of commands or data processed by the processor. Based on the number of command flows and data flows, four architecture classes are distinguished: - SISD (Single Instruction stream/Single Data stream) – one instruction stream and one data stream; - SIMD (Single Instruction stream/Multiple Data stream) - one command stream and multiple data streams; - MISD (Multiple Instruction stream/Single Data stream) - multiple command streams and one data stream; - MIMD (Multiple Instruction stream/Multiple Data stream) - multiple instruction streams and multiple data streams.	M. Flynn's classification	
37.	consist of a set of processors that have shared common memory with a single address space and operate under the control of a single operating system. The disadvantage is that the number of processors that have access to common memory cannot be made larger. There is a limit to the increase in the number of processors, exceeding	Symmetrical multiprocessors (SMP)	

	which leads to a rapid increase in losses due to interprocessor data exchange.		
38.	The basic idea of the classification is that multiple command streams can be processed either by a pipeline scheme in time-sharing mode, or each stream is processed by its own device. According to this, the following MIMD computers are distinguished: - conveyor; - switchable (with shared memory and distributed memory); - networks implemented in the form of: a regular lattice, a hypercube, a hierarchical structure and a variable configuration.	R. Hockney's Systematization of MIMD Computers	
39.	T. Feng's classification also allows one to construct estimates of the achievable degree of parallelism. It is based on two characteristics: - the number n of bits in a machine word processed in parallel; - the number of words m processed simultaneously by the computing system.	T. Feng's classification	
40.	This classification is based on an explicit description of parallel and pipeline processing. Three levels of data processing are distinguished: - level of program implementation; - level of command execution; - bit processing level.	Classification by V. Handler	
41.	1.	Assuming that the only shared resource is RAM, what are the problems that are typical for parallel computing on shared memory systems: 1. Data Race	

		2. distribute – collect (Map – Reduce)	
42.	1.	Assuming that the only shared resource is RAM, what are the problems typical for parallel computing performed on distributed memory systems (clusters): 1. synchronization 2. Data Race 3. clinch 4. blocking	
43.	1.	If we assume that the only shared resource is RAM, then what is meant by data race? 1. competing threads when writing data to the same word of computer memory 2. clearing memory from data 3. computers compete for the fastest writing of data into the computer's memory 4. competition of threads when writing data to different words of computer memory	
44.	1.	What is locking in parallel computing? 1. blocking a critical section of a program in order to exclude the possibility of its simultaneous execution in parallel processes	

		2. blocking a memory area to prevent its use in parallel processes 3. blocking a program to prevent its use in parallel processes 4. blocking the program except for critical sections to enable simultaneous execution of critical sections in several parallel processes	
45.	1.	Vector processors according to Flynn's classifications belong to the class: 1. SIMD 2. SISD 3. MISD 4. MIMD	
46.	1.	From Amdahl's law it follows: 1. for some task, it is impossible to achieve 10-fold acceleration by increasing the number of processors used 2. for any task, you can achieve arbitrarily high acceleration by increasing the number of processors used 3. for any task, you can achieve more than 10-fold acceleration by increasing the number of processors used	

47.	1.	Breaking down the original problem into subproblems of smaller dimension: 1. often leads to a reduction in overall running time in both sequential and parallel algorithms 2. possible only in sequential algorithms 3. possible only in parallel algorithms 4. always leads to an increase in the total running time 5. always results in a reduction in overall running time	
48.	1.	Indicate the true statements: 1. for multi-computer complexes, where each computer in the complex has its own memory, the problems of data races and deadlocks are not so relevant 2. data race always results in exceptions 3. clinch always results in exceptions	
49.	1.	In this course, parallel computing means parallel execution of: 1. the same program 2. different programs on one computer 3. different programs on different computers	

		4. one program in different execution sessions	
50.	1.	What such multiprogram mode work? 1. parallel execution of different programs 2. parallel execution of one program 3. execution of multimedia applications 4. parallel development of several programs	
51.	1.	Priority, program counter, stack can belong to: 1. Flow 2. The process 3. Set of instructions 4. To the programmer	
52.	1. Decomposition of the task into subtasks that are implemented independently. 2. Definition for the formed set of subtasks of information interactions. 3. Scaling subtasks, determining the number of processors. 4. Determining the system architecture, assigning subtasks to processors, and creating a schedule.	General scheme of stages of development of parallel algorithms	
53.	1. Finding computations in a program that can be implemented in parallel. 2. Subsequent distribution of data across local memory modules of processors. 3. Subsequent coordination of parallelism of computations and data distribution.	Multithreading implies:	

54.	This view parallelism is being implemented compiler and consists of replacing multiple executions of similar operations with one operation over a set of data. Processing is performed on vector processors. In programs processed by the compiler, compilation directives are specified, which assumes the use of parallel computing languages.	Data-level parallelism	
55.	This type of parallelism involves replacing sequential algorithms of some calculations with parallel ones. This concerns search, sorting algorithms, etc. The organization of the parallelization process is carried out by using various parallel programming tools, such as special libraries, environment variables, compiler directives, etc. An example is OpenMP technology	Parallelism at the algorithm level	
56.	- minimize memory conflicts; - optimize the distribution of the computing load between processors; - optimize data distribution; - optimize the program structure; - replace sequential parts of the algorithm with parallel ones; - solve problems typical of sequential computers and traditional sequential programming.	To improve the program's performance, you can:	
57.	flops is defined as the number of arithmetic operations per second performed on floating- point real numbers	Unit productivity parallel computer -	
58.	creation of a parallelism resource (obtaining a parallel algorithm) in the decision processes	Parallel computing problem	

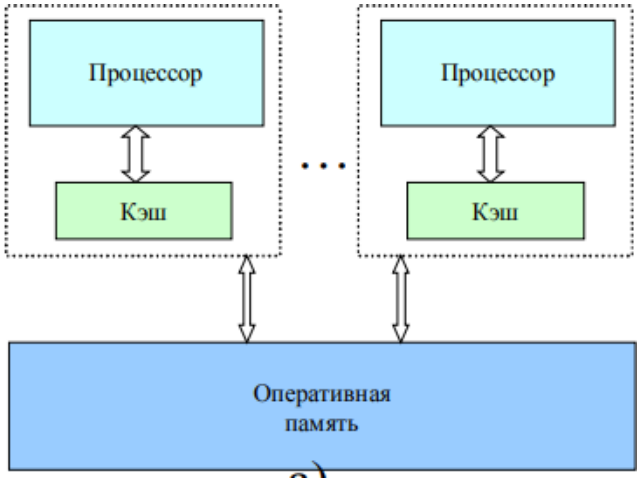
	tasks and managing the implementation of this parallelism in order to achieve the greatest efficiency of use multiprocessor computer technology		
59.	a method for solving complex computational problems using multiple computers connected into a parallel computing system. Loosely coupled, heterogeneous computing systems are allocated to a separate class of distributed systems – Grid	Distributed computing	
60.	an algorithm whose operations can be executed simultaneously (not necessarily independently); it is assumed that the operations or sets of operations to be executed simultaneously are specified explicitly or implicitly. A strict concept of a parallel algorithm has not been introduced	Parallel algorithm	
61.	a parallel algorithm written in some programming system oriented towards computing systems parallel architecture	Parallel program	
62.	If, when solving a problem, processors perform the same sequence of calculations but use different data, then we speak of data parallelism. For example, when searching a database, each processor can work with its own part of the database. If processors perform different tasks of one task, perform different functions, then we speak of functional parallelism.	Data Parallelism. Functional Parallelism	
63.	Let the operation be divided into micro-operations. Let us arrange the micro-operations in the order of execution and allocate a separate part of the device for each execution. At the first moment of time	Conveyor belt processing data operations (pipeline parallelism at the operation level)	

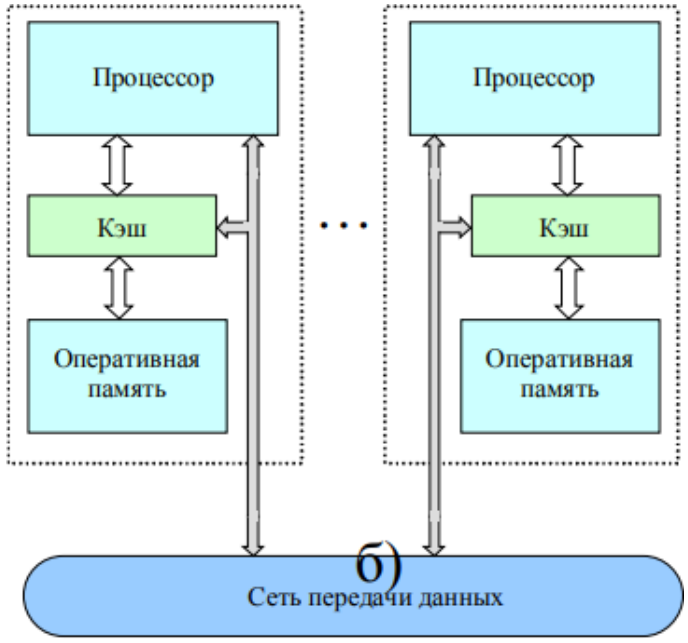
	input data is sent to the first part for processing. After the first micro-operation is performed, the first part passes the results of its work to the second part, and takes new data itself. When the input arguments have passed all stages of processing, the result of the operation will appear at the output of the device. In this way, functional parallelism is implemented. Each part of the device is called a pipeline stage, and the total number of stages is the pipeline length		
64.	the number of operations actually performed on average per unit of time.	Real performance computing systems	
65.	the maximum number of operations that can be performed by the system per unit of time.	Peak performance computing systems	
66.	the ratio of the actual processor operation time on a given segment to the length of the entire segment. The workload of a computing system consisting of identical processors is the arithmetic mean of the workloads of all processors. It follows from the definition that the workload p satisfies the condition $0 \leq p \leq 1$	CPU load at a given time interval	
67.	the ratio of the execution time of an algorithm on one processor (on one processor core) to the parallel execution time. Speedup depends on the choice of a parallel algorithm and on how adequate this algorithm is to the architecture of the computing system.	Acceleration of the implementation of an algorithm on a computing system consisting of identical processors –	
68.	systems with common shared memory (but each processor also has its own cache memory).	Multiprocessors	
69.	distributed memory systems. When memory is physically distributed among processors.	Multicomputers	

70.	In a multiprocessor system, it is necessary to coordinate the exchange of data between processors and, when using shared memory, between processors and shared memory. Synchronization is the timing of the execution of parallel tasks. It involves waiting for the execution of a task to reach a special point, called a synchronization point. After all tasks have reached a synchronization point, the execution of tasks can continue until the next synchronization point. When using a message-passing model (usually MPI – Message Passing Interface), the work of processors is synchronized by data exchange functions.	Synchronization (processor operation).	
71.	A computational grain (or tile) is a set of algorithm operations that are executed atomically: computations belonging to a single grain cannot be interrupted by synchronization or data exchange required to perform these operations. Granularity is a measure of the ratio of the number of computations performed in a parallel task to the number of data transfers. Fine-grained parallelism means very few computations per data transfer. Coarse-grained parallelism means intensive computations per data transfer (data is transferred in large portions). The finer the granularity, the more synchronization points.	Grain	
72.	an object created by the operating system within a process (a process is an executable application with its own virtual address space) that executes instructions	Thread, lightweight process	

	programs. A process can have one or more threads running in the context of that process. Threads allow parallel execution of processes and simultaneous execution of different parts of a program by one process on different processors.		
73.	An algorithm has natural parallelism if it can be broken down into independently executable parts.	Natural parallelism	
74.	An algorithm has internal parallelism if it can be broken down into parts that can be executed in parallel (but not necessarily independently).	Internal parallelism	
75.	Identification (indication) of operations or sets of operations of a sequential algorithm that can be executed simultaneously	Parallelization	
76.	Parallelization performed before the start of the algorithm (program) execution	Static parallelization	
	Parallelization performed during the execution of an algorithm (program)	Dynamic parallelization	
77.	Scalability of this parallel algorithm when implemented on a given parallel system means that the system performance is proportional to the number of processors (cores) it contains. A distinction is made between the concepts of weak and strong scalability. Weak (not in the sense of "bad") scalability (weak scaling) means a linear increase in the size of the problem with an increase in the number of processors with a fixed execution time of the algorithm. Strong scaling means a linear increase in acceleration with an increase in the number of processors with a fixed size of the problem.	Scalability	

78.	is a programming strategy that postpones the evaluation of an expression or variable until the value is needed. It is the opposite of direct, or immediate, evaluation, in which values are calculated immediately when the calculation is called. Deferred evaluation allows for greater optimization of calculations and reduces memory load by avoiding unnecessary repetitions of calculations.	Lazy evaluation	
79.	OrepMR - this is a parallel standard shared-memory programming environment. OpenMP provides programmers with a set of pragmas, procedures, and environment variables that make it easy to parallelize code written in Fortran, C, or C++. A pragma is a compiler directive that specifies how to process the code that follows the pragma. When OpenMP pragmas are used in a program, they instruct an OpenMP-aware compiler to create an executable that will be executed in parallel using multiple threads. OpenMP pragmas provide a uniform and portable interface for parallelizing programs across different architectures and systems.	Parallel programming standard OpepMR	
80.	Message Passing Interface (MPI) is a software interface for information transfer that allows processes performing the same task to exchange messages. The MPI standard is aimed at distributed memory systems where data transfer costs are high, while OpenMP is aimed at shared memory systems (for	MPI Message Passing Interface	

	multi-core processors with shared cache memory). Both technologies can be used together to optimize the use of multi-core systems in a single cluster.		
81.	Architecture of multiprocessor systems with shared memory - systems with uniform access to memory		

82.	Architecture of multiprocessor systems with shared memory - systems with non-uniform access to memory	 б)	
83.	1) the operation of multiple processes is controlled; 2) data exchange between processes is organized; 3) determinism of behavior is lost due to asynchrony of access to data; 4) non-local and dynamic errors predominate; 5) the possibility of dead-end situations appears; 6) problems arise with program scalability and balancing the load of computing nodes.	What are the features of parallel programs?	
84.	- Portability – MPI allows to significantly reduce the severity of the problem of portability of parallel programs between different	Advantages of MPI	

	computer systems - parallel A program developed in the C or Fortran algorithmic language using the MPI library will generally run on any computing platform for which an implementation of the standard is available. - Increased efficiency – MPI helps to increase the efficiency of parallel computing, since currently there are implementations of the standard for almost every type of computing system that take into account the capabilities of the computer hardware used to the maximum extent.		
85.	- Ease of use – the developer does not create a new parallel program, but adds the necessary directives and, possibly, library calls to the text of the sequential program functions, indicating compiler methods for distributing computations and data between threads, as well as access to them. The main idea of OpenMP is to parallelize the cycles that usually carry the main computational load. - Flexibility – OpenMP provides the developer with fairly broad control over the behavior of a parallel program. - Reuse – an OpenMP program can in many cases be used as a regular sequential one if it is necessary to ensure its execution on a single- processor platform. In this case, to get rid of the OpenMP implementation in the executable module, it is enough to rebuild it as a sequential one	Advantages of OpenMP	

	compiler. Directives OpenMP will be ignored, and library function calls can be replaced with stubs, the text of which is given in the standard specifications		
86.	4	Using C# you can create: 1. Processes 2. Streams 3. Neither one nor the other 4. Processes and threads	
87.	1.	The CLR will not stop an application (meaning unloading the current application domain) until: 1. all foreground threads will not be terminated 2. not all threads will be completed at all 3. the main thread of the application will not be terminated 4. the GUI thread of the application will not be terminated	
88.	1.	1. Operating system is responsible for scheduling threads for launch 2. Programmer 3. Common Language Runtime (CLR) 4. Application flow management system	
89.	1.	Operating room system plans execution streams based 1. Priorities 2. Tasks 3. Instructions 4. CPU Loads	
90.	1.	Select the correct description of the Task class (TPL). 1. This class describes a single task that runs asynchronously in one of the threads from the thread pool.	

		2. Represents by yourself independent a sequence of instructions that exists within a process 3. It is a sequence of instructions that allocates a specific address space in memory and is isolated from other processes.	
91.	1.	Parallel.For method 1. Allows loop iterations to be performed in parallel 2. Allows you to iterate through a loop sequentially 3. Used to comply with the MPI standard 4. Does not exist in C#	
92.	1.	Parallelism is 1. A method for handling multiple requests simultaneously 2. A type of multithreading 3. A type of multitasking	
93.	1.	The CLR will not stop an application (meaning unloading the current application domain) until: 1. all foreground threads will not be terminated 2. not all threads will be completed at all 3. the main thread of the application will not be terminated 4. the GUI thread of the application will not be terminated	
94.	1.	The only difference between foreground threads and background threads is that: 1. A background thread is automatically terminated if all foreground threads in its process are stopped 2. There is no difference 3. Background threads have lower priority 4. Priority threads have higher priority	

95.	1.	By default, the created thread (.NET Framework) automatically becomes: 1. Priority 2. Background 3. Workers 4. The main	
96.	1,2	What types of threads are defined in the .NET Framework? 1. Priority 2. Background 3. Worker 3. Main 4. Main	
97.	1,2,3,4,5	What states can a thread be in? 1. Running 2. Ready to perform 3. Suspended 4. Blocked 5. Completed 6. Waiting for an event 7. Waiting for input	
98.	1.	Why is thread-based multitasking organized? 1. Parallel execution of individual parts of one program 2. Parallel execution of programs 3. Parallel execution of instructions 4. There is no correct answer	
99.	1.	Why is process-based multitasking organized? 1. Parallel execution of programs 2. Parallel execution of threads 3. Parallel execution of instructions	

		4. All answers are correct	
100.	1,2	What are the different types of multitasking? 1. Process-based multitasking 2. Thread-based multitasking 3. Instruction-based multitasking 4. Command-based multitasking	