

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук
Департамент программной инженерии
Дисциплина: «Архитектура вычислительных систем»

**Домашнее задание №4 по дисциплине
«Архитектура вычислительных систем»**

На тему:

«Задача про экзамен»

Пояснительная записка

Выполнил:
Моторкин Владимир,
студент гр. БПИ198.

Москва
2020

Содержание

1. Текст задания.....	2
2. Применяемые расчетные методы.....	2
2.1. Теория решения задания.....	2
2.2. Описание переменных и функций программы.....	3
Class Student.....	3
Class Teacher.....	3
Другие переменные и функции программы	3
3. Тестирование программы.....	4
ПРИЛОЖЕНИЕ 1.....	6
Список литературы	6
ПРИЛОЖЕНИЕ 2.....	7
Код программы.....	7

1. Текст задания

Вариант 21.

Задача про экзамен. Преподаватель проводит экзамен у группы студентов. Каждый студент заранее знает свой билет и готовит по нему ответ. Подготовив ответ, он передает его преподавателю. Преподаватель просматривает ответ и сообщает студенту оценку. Требуется создать многопоточное приложение, моделирующее действия преподавателя и студентов. При решении использовать парадигму «клиент-сервер».

2. Применяемые расчетные методы

2.1. Теория решения задания

В данном задании необходимо использовать парадигму «клиент-сервер». В качестве сервера выступает учитель, который принимает запросы от студентов на проверку их ответа.

При сдаче работы студентом, его номер добавляется в очередь, учитель замечает, что очередь работ не пустая. Учитель берёт верхний id из очереди и выставляет ему случайную оценку от 1 до 10 включительно. Учитель сообщает оценку студента. Студент тоже объявляет свою оценку. Учитель проверяет работы 1-3 секунды, студенты решают задание 5-15 секунд.

В критических сессиях производится работа с очередью, массивом результатов и вывод в консоль сообщений.

2.2. Описание переменных и функций программы

Class Student

Тип	Название	Описание
Private bool	hasScore	Флаг наличия оценки
Private int	score	Оценка
Private int	number	Номер студента
Private void	printMessage	Флаг для отметки завершения проверки билетов
Public void	startExam	Функция начала выполнения задания
конструктор	Student	Задаёт отсутствие оценки и номер студента.

Class Teacher

Тип	Название	Описание
Public static void	checkWork	Функция проверки работы
Public static void	startExam	Функция начала экзамена
Тип	Название	Описание
Public static void	checkWork	Функция проверки работы
Public static void	startExam	Функция начала экзамена

Другие переменные и функции программы

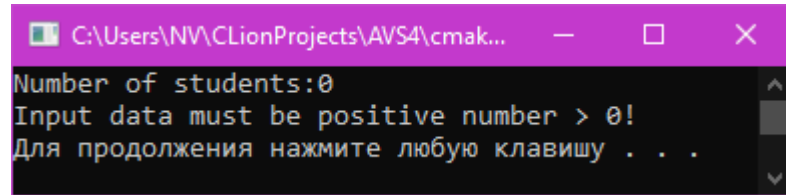
Тип	Название	Описание
Queue<int>	works	Функция проверки работы
Vector<int>	results	Функция начала экзамена
void	threadStudentFunction	Функция потока студента, которая создаёт экземпляр студента и вызывает startExam()
void	fillArray	Заполняет массив results нулями.

3. Тестирование программы

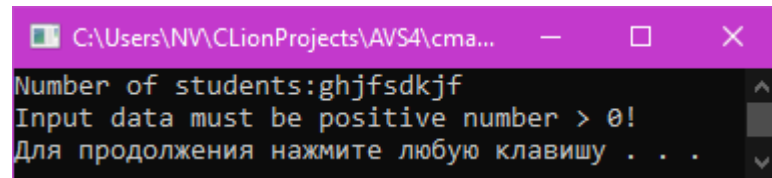
```
C:\Users\NV\CLionProjects\AVS4\cmake-build-debu...
Number of students: 4
Student #4: I am starting an exam!
Student #2: I am starting an exam!
Student #3: I am starting an exam!
Student #1: I am starting an exam!
Student #2: I have finished!
Teacher: I'm checking work of student #2!
Student #1: I have finished!
Teacher: I have checked work of student #2! Score is 8
Teacher: I'm checking work of student #1!
Student #2: My score is 8.
Teacher: I have checked work of student #1! Score is 1
Student #1: My score is 1.
Student #3: I have finished!
Teacher: I'm checking work of student #3!
Student #4: I have finished!
Teacher: I have checked work of student #3! Score is 5
Teacher: I'm checking work of student #4!
Student #3: My score is 5.
Teacher: I have checked work of student #4! Score is 9
Student #4: My score is 9.
Для продолжения нажмите любую клавишу . . .
```

```
C:\Users\NV\CLionProjects\AVS4\cmake-build-debu...
Number of students:1
Student #1: I am starting an exam!
Student #1: I have finished!
Teacher: I'm checking work of student #1!
Teacher: I have checked work of student #1! Score is 8
Student #1: My score is 8.
Для продолжения нажмите любую клавишу . . .
```

Ввод некорректных данных



```
C:\Users\NV\CLionProjects\AVS4\cma...  —  □  ×  
Number of students:0  
Input data must be positive number > 0!  
Для продолжения нажмите любую клавишу . . .
```



```
C:\Users\NV\CLionProjects\AVS4\cma...  —  □  ×  
Number of students:ghjfsdkjf  
Input data must be positive number > 0!  
Для продолжения нажмите любую клавишу . . .
```

Список литературы

1. Потоки, блокировки и условные переменные в C++11 [Часть 1].
[Электронный ресурс] // URL: <https://habr.com/ru/post/182610/> (дата обращения: 17.11.2020)
2. Потоки, блокировки и условные переменные в C++11 [Часть 2].
[Электронный ресурс] // URL: <https://habr.com/ru/post/182626/> (дата обращения: 17.11.2020)
3. [C++] часть 3: синхронизация потоков в ресторане. [Электронный ресурс]
// URL: <https://yandex.ru/turbo/nuancesprog.ru/s/p/6546/> (дата обращения: 17.11.2020)

Код программы

```
1. #include <iostream>
2. #include <thread>
3. #include <queue>
4. #include <string>
5. #include <omp.h>
6.
7. using namespace std;
8.
9. queue<int> works;
10. vector<int> results;
11. class Student
12. {
13.     bool hasScore;
14.     int score;
15.     int number;
16.     void printMessage(string message)
17.     {
18. #pragma omp critical(print)
19.     {
20.         cout << "Student #" << number << ": " << message << endl;
21.     }
22. }
23.
24. public:
25.     Student(int n)
26.     {
```



```
27.    number = n;
28.    hasScore = false;
29.    // Условие для генерации псевдослучайных чисел.
30.    srand(static_cast<unsigned>(n * n + static_cast<unsigned>(time(0))));
31. }
32. void startExam()
33. {
34.     printMessage("I am starting an exam!");
35.     int time = rand() % 11 + 5;
36.     this_thread::sleep_for(chrono::seconds(time));
37.     {
38.#pragma omp critical(queue)
39.     {
40.         printMessage("I have finished!");
41.         works.push(number);
42.     }
43. }
44. while (!hasScore)
45. {
46.     int a;
47.#pragma omp critical(scores)
48.     {
49.         a = results[number - 1];
50.     }
51.     if (a != 0)
52.     {
53.#pragma omp critical(scores)
54.     {
```

```

55.         score = results[number - 1];
56.     }
57.     printMessage("My score is " + to_string(score) + ".");
58.     hasScore = true;
59. }
60. }
61. }
62.};
63.
64.class Teacher
65.{
66.public:
67.    static void checkWork(int n)
68.    {
69.#pragma omp critical(print)
70.    {
71.        cout << "Teacher: I'm checking work of student #" << n << "!" << endl;
72.    }
73.    int time = rand() % 3 + 1;
74.    this_thread::sleep_for(chrono::seconds(time));
75.    int score = rand() % 10 + 1;
76.
77.#pragma omp critical(print)
78.    {
79.        cout << "Teacher: I have checked work of student #" << n << "!" << "Score
        is " << score << endl;
80.    }
81.#pragma omp critical(scores)

```

```
82.    {
83.        results[n - 1] = score;
84.    }
85. }
86. static void startExam(int numberOfStudents)
87. {
88.     int numberOfCheckedWorks = 0;
89.     while (numberOfCheckedWorks != numberOfStudents)
90.     {
91.         while (!works.empty())
92.         {
93.             int work;
94. #pragma omp critical(queue)
95.             {
96.                 work = works.front();
97.                 works.pop();
98.             }
99.             checkWork(work);
100.             numberOfCheckedWorks++;
101.         }
102.     }
103. }
104. };
105.
106. void fillArray(int n)
107. {
108.     for (int i = 0; i < n; i++)
109.     {
```

```
110.         results.push_back(0);
111.     }
112. }
113.
114. void threadStudentFunction(int a)
115. {
116.     Student stud = Student(a);
117.     stud.startExam();
118. }
119.
120. int main() {
121.     int numberOfStudents;
122.     cout << "Number of students:";
123.     cin >> numberOfStudents;
124.     if (numberOfStudents <= 0) {
125.         cout << "Input data must be positive number > 0!" << endl;
126.         system ("pause");
127.         return 0;
128.     }
129.     results = vector<int>(numberOfStudents);
130.     fillArray(numberOfStudents);
131.
132.     #pragma omp parallel num_threads(numberOfStudents + 1)
133.     {
134.         auto numberOfThreads = omp_get_thread_num();
135.         if (numberOfThreads == 0)
136.         {
137.             Teacher::startExam(numberOfStudents);
```

```
138.     }
139.     else
140.     {
141.         threadStudentFunction(omp_get_thread_num());
142.     }
143. }
144. //Чтобы сразу не пропадало окно.
145. system ("pause");
146. return 0;
147. }
```