**Правительство Российской Федерации**

Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет «Высшая школа экономики»

Факультет компьютерных наук

**Отчет к домашней работе №4.1**

**По дисциплине**

**“Архитектура вычислительных систем”**

Работу выполнил

Студент группы БПИ-197 \_\_\_\_\_\_\_\_\_\_\_\_\_ А. А. Синенко

подпись, дата

Работу проверил \_\_\_\_\_\_\_\_\_\_\_\_\_ А. И. Легалов

подпись, дата

Москва 2020

Содержание

[Постановка задачи 3](#_Toc56510232)

[Уточнение задачи 3](#_Toc56510233)

[Тестирование программы 3](#_Toc56510234)

[Использованная литература 4](#_Toc56510235)

[Приложение 5](#_Toc56510236)

# Постановка задачи

В цехе по заточке булавок все необходимые операции осуществляются тремя рабочими. Первый из них берет булавку и проверяет ее на предмет кривизны. Если булавка не кривая, то рабочий передает ее своему напарнику. Напарник осуществляет собственно заточку и передает заточенную булавку третьему рабочему, который осуществляет контроль качества операции. Требуется создать многопоточное приложение, моделирующее работу цеха. При решении использовать парадигму «производитель-потребитель».

# Уточнение задачи

1. Необходимо при помощи библиотеки OpenMP по работе с потоками в C++ написать программу, моделирующую работу цеха по производству булавок (pin).
2. Выводить действия рабочих в консоль.
3. Использовать парадигму “Производитель-потребитель” (1, стр. 1).

Реализация программы

* Для хранения информации о производимой булавке создана структура pin.
* Для моделирования конвейеров созданы три очереди (queue) с булавками conveer1, conveer2, conveer3.
* Поле producted для хранения произведенных деталей.
* Поле defective для хранения произведенных деталей.
* Методы-потоки firstWorker, secondWorker, thirdWorker для моделирования работы трех рабочих.
* В методеint main() реализовано создание трех потоков, ожидание завершения этих потоков и корректное завершение программы.

# Тестирование программы

При корректном запуске программа выводит информацию о любом действии рабочих индивидуально.

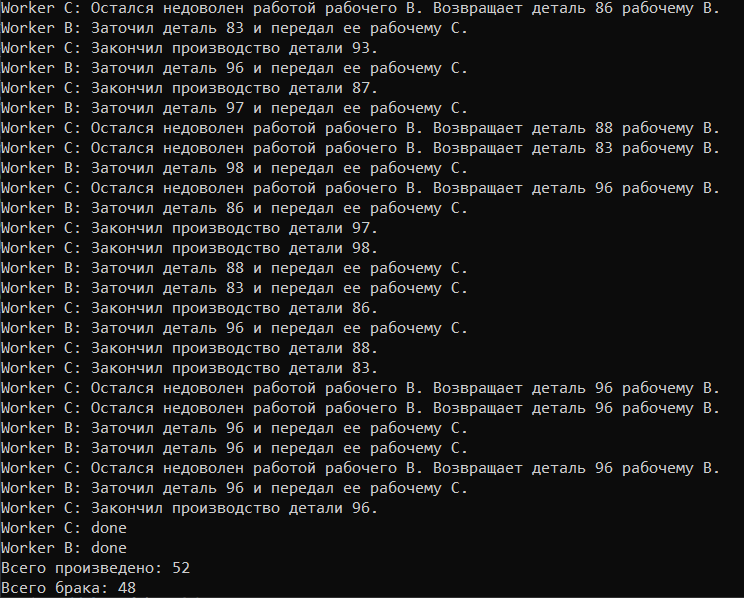


Рисунок – демонстрация работы программы (последние строки).

# Использованная литература

1. https://studydocs.ru/studfiles/112/916/656121/9.doc.html (электронный ресурс)
2. <https://docs.microsoft.com/ru-ru/cpp/parallel/openmp/reference/openmp-library-reference?view=msvc-160> (электронный ресурс)

# Приложение

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171 | #include <iostream>  #include <queue>  #include <omp.h>  **using** **namespace** std;  **const** **size\_t** pinsCount = **100**;  //const size\_t pinsCount = 1000;  //const size\_t pinsCount = 10000;  // Прогресс рабочих (0 - все работают, 1 - A закончил работу, 2 - C и B закончили работу)  **int** status = **0**;  // Класс детали  **struct** pin  {  // Идентификатор детали  **int** id = **0**;  // Является ли деталь кривой  **bool** isCurved = false;  // Качественная ли заточка детали  **bool** isGoodQuality = false;  pin(**int** id)  {  isCurved = rand() % **2**;  **this**->id = id;  }  };  // Рабочий A (производитель)  **void** **first\_worker**(queue<pin>& input, queue<pin>& output, queue<pin>& defective)  {  **while** (!input.empty())  {  // Берется деталь  pin takenPin = input.front();  input.pop();  **if** (!takenPin.isCurved)  {  // Передается рабочему B  output.push(takenPin);  #pragma omp critical  {  cout << "Worker A: Проверил деталь " << takenPin.id << " и передал рабочему B" << endl;  }  }  **else**  {  // Бракуется  defective.push(takenPin);  #pragma omp critical  {  cout << "Worker A: Деталь " << takenPin.id << " оказалась кривой." << endl;  }  }    status = **0**;  }  cout << "Worker A: done" << endl;  status = **1**;  }  // Рабочий B (потребитель для A и производитель для C)  **void** **second\_worker**(queue<pin>& input, queue<pin>& output)  {  **while** (true)  {  **while** (!input.empty())  {  // Берется деталь  pin takenPin = input.front();  input.pop();  // Заточка и передача рабочему C  takenPin.isGoodQuality = rand() % **2**;  output.push(takenPin);  #pragma omp critical  {  cout << "Worker B: Заточил деталь " << takenPin.id << " и передал ее рабочему C." << endl;  }  }  **if** (status == **2**)  **break**;  }  cout << "Worker B: done" << endl;  }  // Рабочий C (потребитель)  **void** **third\_worker**(queue<pin>& input, queue<pin>& secondWorkerInput, queue<pin>& output)  {  **while** (true)  {  **while** (!input.empty())  {  // Берется деталь  pin takenPin = input.front();  input.pop();  **if** (takenPin.isGoodQuality)  {  // Заканчивает производство детали  output.push(takenPin);  #pragma omp critical  {  cout << "Worker C: Закончил производство детали " << takenPin.id << "." << endl;  }  // Условие окончания работы цеха  **if** (input.empty() && secondWorkerInput.empty() && status == **1**)  status = **2**;  }  **else**  {  // Возвращается рабочему B на доработку  secondWorkerInput.push(takenPin);  #pragma omp critical  {  cout << "Worker C: Остался недоволен работой рабочего B. "  << "Возвращает деталь " << takenPin.id << " рабочему B." << endl;  }  }  }  **if** (status == **2**)  **break**;  }  cout << "Worker C: done" << endl;  }  // Входная точка программы  **int** **main**()  {  setlocale(LC\_ALL, "Russian");  **if** (pinsCount == **0**)  {  cout << "Всего произведено: 0" << endl;  cout << "Всего брака: 0" << endl;  **return** **0**;  }  // Обнуление состояния  status = **0**;  // Вход для рабочего A  queue<pin> conveer1;  **for** (**int** i = **0**; i < pinsCount; i++)  conveer1.emplace(pin(i + **1**));  // Вход для рабочего B  queue<pin> conveer2;  // Вход для рабочего C  queue<pin> conveer3;  // Произведенные детали  queue<pin> producted;  // Забракованные детали  queue<pin> defective;  // Параллелизм по секциям в три потока  #pragma omp parallel sections num\_threads(**3**)  {  #pragma omp section  {  first\_worker(conveer1, conveer2, defective);  }  #pragma omp section  {  second\_worker(conveer2, conveer3);  }  #pragma omp section  {  third\_worker(conveer3, conveer2, producted);  }  }  // Итог  cout << "Всего произведено: " << producted.size() << endl;  cout << "Всего брака: " << defective.size() << endl;  // Выход из программы.  **return** **0**;  } |