Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики

и радиоэлектроники»

Специальность «Программная инженерия»

Кафедра инженерной психологии и эргономики

Учебная дисциплина «Основы алгоритмизации и программирования»

ОТЧЕТ

по лабораторной работе №12

«Очереди»

Вариант 14

Подготовил: Зайцев Е. А.

Проверил: Усенко Ф.В.

Минск 2025

**Цель работы:** сформировать знания и умения по работе с подпрограммами, приобрести навыки написания программ с использованием очередей.

**Задание:** Система состоит из трех процессоров P1, P2, P3, очереди F, стека S и распределителя задач R. В систему поступают запросы на выполнение задач трёх типов – T1, T2 и T3, каждая для своего процессора. Поступающие запросы ставятся в очередь. Если в начале очереди находится задача Ti и процессор Pi свободен, то распределитель R ставит задачу на выполнение в процессор Pi, а если процессор Pi занят, то распределитель R отправляет задачу в стек и из очереди извлекается следующая задача. Если в вершине стека находится задача, процессор которой в данный момент свободен, то эта задача извлекается и отправляется на выполнение.

Код программы приведен ниже:

#include <iostream>

#include <queue>

#include <stack>

using namespace std;

enum TaskType {

T1 = 1,

T2 = 2,

T3 = 3

};

struct Task {

TaskType type;

int id;

};

int main() {

setlocale(LC\_ALL, "Russian");

queue<Task> taskQueue;

stack<Task> taskStack;

bool processorBusy[3] = { false, false, false }; // Состояние занятости процессоров

int processorFinishTime[3] = { 0, 0, 0 }; // Время, когда процессор освободится

int currentTime = 0; // Текущее время

// Добавляем задачи в очередь

taskQueue.push({ T1, 1 });

taskQueue.push({ T2, 2 });

taskQueue.push({ T3, 3 });

taskQueue.push({ T1, 4 });

taskQueue.push({ T2, 5 });

taskQueue.push({ T3, 6 });

taskQueue.push({ T3, 7 });

while (!taskQueue.empty() || !taskStack.empty()) {

// Проверяем завершение задач на процессорах

for (int i = 0; i < 3; ++i) {

if (processorBusy[i] && currentTime >= processorFinishTime[i]) {

cout << "Процессор " << i + 1 << " завершил работу и сейчас свободен." << endl;

processorBusy[i] = false;

}

}

// Проверяем стек на задачи для свободных процессоров

if (!taskStack.empty() && !processorBusy[taskStack.top().type - 1]) {

Task t = taskStack.top();

taskStack.pop();

processorBusy[t.type - 1] = true;

processorFinishTime[t.type - 1] = currentTime + 5;

cout << "Процессор " << t.type << " выполняет задачу " << t.id << endl;

continue;

}

// Обрабатываем очередь

if (!taskQueue.empty()) {

Task t = taskQueue.front();

if (!processorBusy[t.type - 1]) {

taskQueue.pop();

processorBusy[t.type - 1] = true;

processorFinishTime[t.type - 1] = currentTime + 5;

cout << "Процессор " << t.type << " выполняет задачу " << t.id << endl;

}

else {

taskStack.push(t);

taskQueue.pop();

cout << "Задача " << t.id << " типа " << t.type

<< " перемещена в стек потому что процессор " << t.type

<< " занят" << endl;

}

}

currentTime++; // Увеличиваем текущее время

}

return 0;

}

Результат работающей программы представлен на рисунке 1:

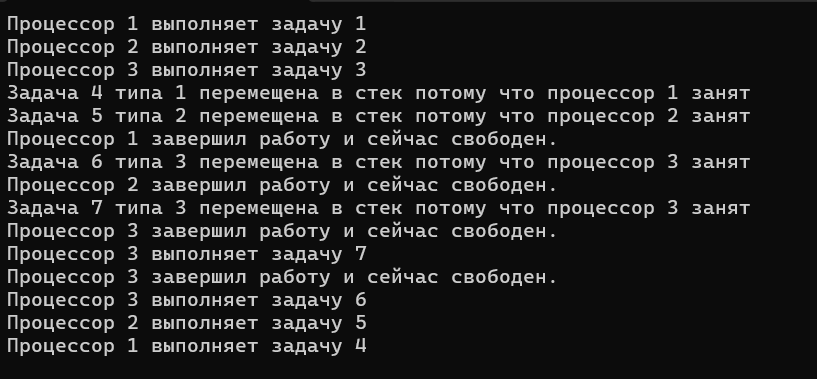


Рисунок 1 – Результат работы программы

Блок-схема работы программы представлена на рисунке 2.

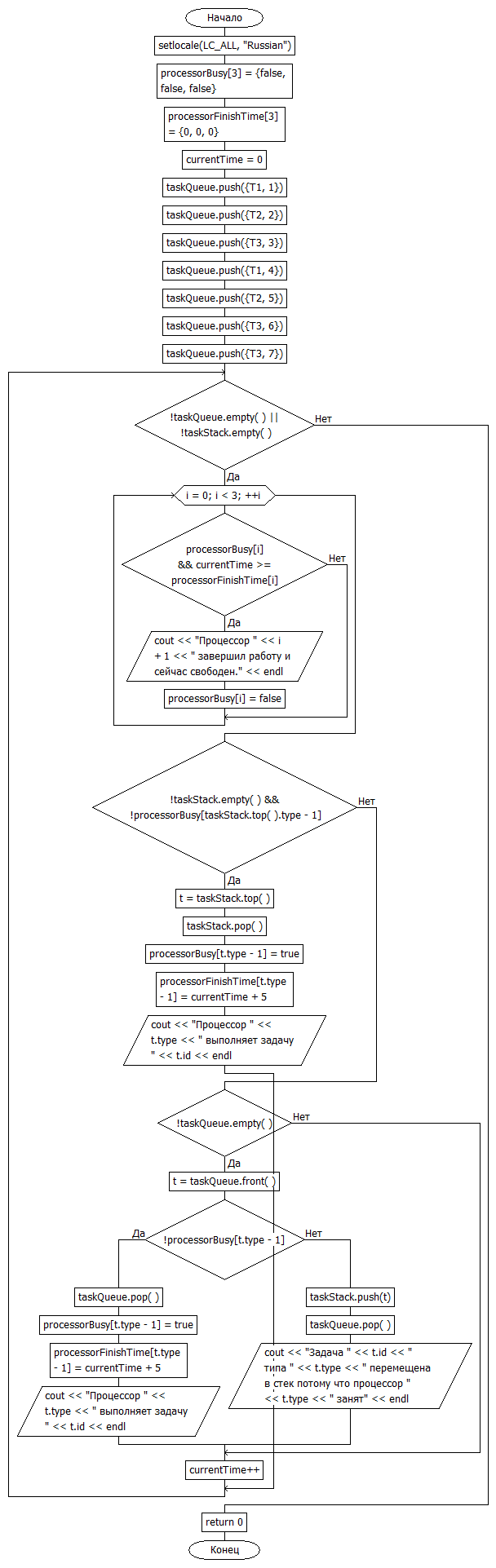


Рисунок 2

**Вывод:** в ходе выполнения работы была достигнута цель данной лабораторной работы: сформировать знания и умения по работе с подпрограммами, приобрести навыки написания программ с использованием очередей.