**ФЕДЕРАЛЬНОЕ АГЕНТСТВО ЖЕЛЕЗНОДОРОЖНОГО ТРАНСПОРТА**

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Петербургский государственный университет путей сообщения**

**Императора Александра I»**

**(ФГБОУ ВО ПГУПС)**

Факультет «Автоматизация и интеллектуальные технологии»

Кафедра «Информатика и информационная безопасность»

Лабораторная работа № 1

по дисциплине

«Защита информации в РИС и ЦОД»

«Построение приложения, реализующего концепцию многозадачности и распределенных вычислений»

|  |  |  |
| --- | --- | --- |
| **Выполнил обучающийся**  Курс 4  Группа КИБ-012 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | Г. Е. Груздев |
|  |  |  |
| **Проверил** | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | А.А. Корниенко |

Санкт-Петербург

2024

**Текстовое описание задачи**

Задача о нелюдимых садовниках. Имеется пустой участок земли (двумерный массив) и план сада, который необходимо реализовать. Эту задачу выполняют два садовника, которые не хотят встречаться друг с другом. Первый садовник начинает работу с верхнего левого угла сада и перемешается слева направо, сделав ряд, он спускается вниз. Второй садовник начинает работу с нижнего правого угла сада и перемещается снизу вверх, сделав ряд, он перемещается влево. Если садовник видит, что участок сада уже выполнен другим садовником, он идет дальше. Садовники должны работать параллельно. Создать многопоточное приложение, моделирую шее работу садовников. При решении задачи использовать критические секции.

**Структурная схема взаимодействия синхронизируемых потоков**

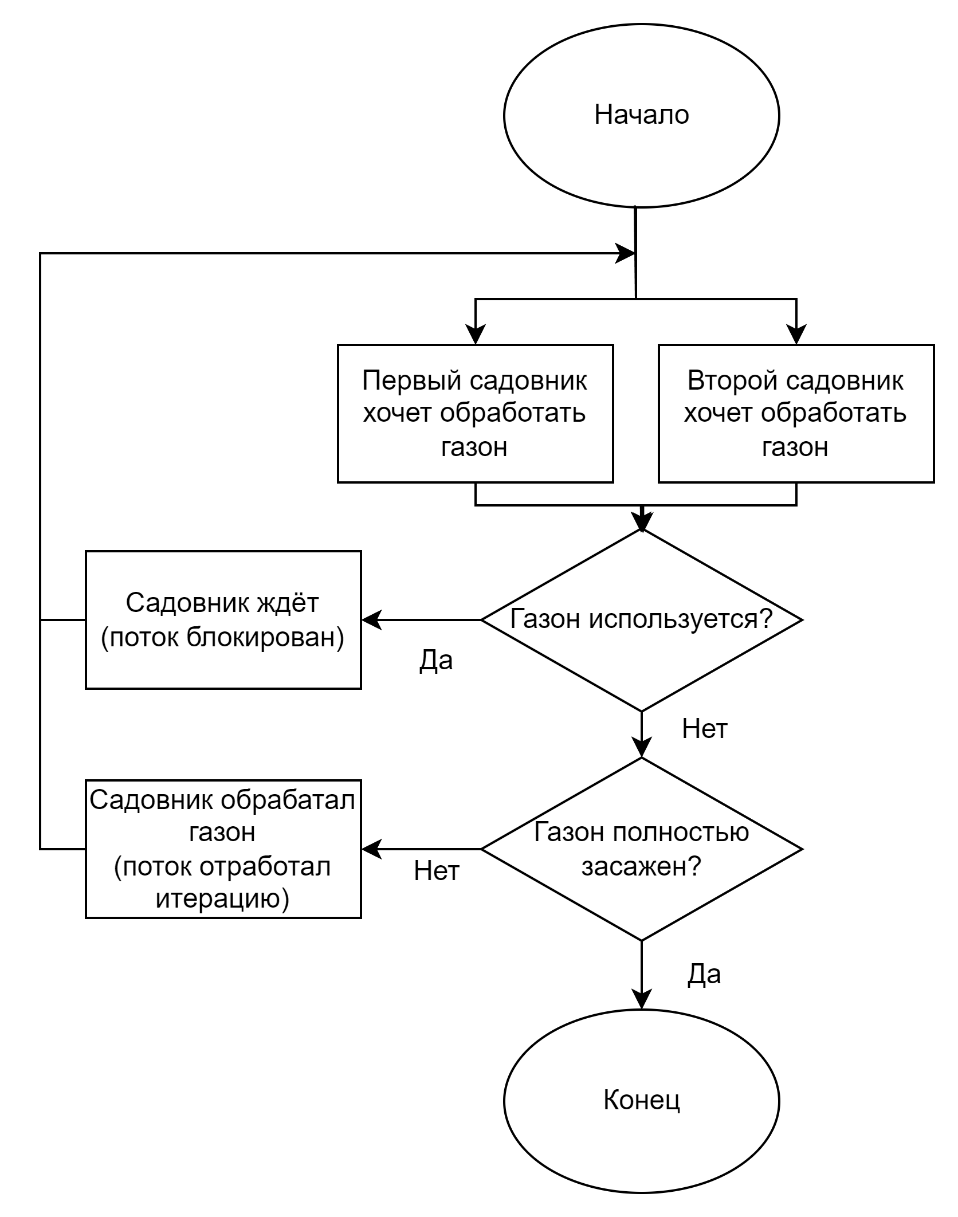


Рисунок 1.1 – Структурная схема жизненного цикла потока (концептуально)

**Краткое описание использованных объектов синхронизации**

**Критический участок** является наиболее простым средством синхронизации задач для обеспечения последовательного доступа к ресурсам. При вхождении задачи в критический участок никакая другая задача не может начать ее выполнение до того, как работающая с ним задача не выйдет из нее. Попытка такой задачи войти в критическую секцию приведет к переводу ее в состояние "блокирован".

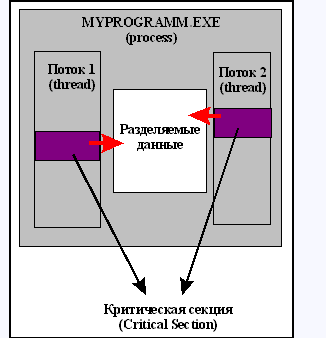


Рисунок 1.2 — Схема работы критической секции

Критическая секция (Critical Section) — это участок кода, в котором поток (thread) получает доступ к ресурсу (например переменная), который доступен из других потоков. Объект критическая секция обеспечивает синхронизацию. Этим объектом может владеть только один поток, что и обеспечивает синхронизацию. Для работы с критическими секциями есть ряд функций API и тип данных CRITICAL\_SECTION. Для использования критической секции нужно создать переменную данного типа, и проинициализировать ее перед использованием с помощью функции InitializeCriticalSection(). Для того, чтобы войти в секцию нужно вызвать функцию EnterCriticalSection(), а после завершения работы LeaveCriticalSection(). Что будет, если поток обратится к секции, в которой сейчас другой поток? Тот поток, который обратится будет блокирован пока критическая секция не будет освобождена. Саму критическую секцию можно удалить функцией DeleteCriticalSection(). Для того, чтобы обойти блокировку потока при обращении к занятой секции есть функция TryEnterCriticalSection(), которая позволяет проверить критическую секцию на занятость.

**Листинг разработанного приложения**

#include <iostream>

#include <vector>

#include <chrono>

#include <thread>

#include <windows.h>

#pragma warning(suppress : 6387)

CRITICAL\_SECTION cs;

std::vector<std::vector<int>> array(5, std::vector<int>(5, 0));

void print\_array(int point, int i, int j) {

EnterCriticalSection(&cs);

if (point == 1) {

if (array[i][j] == 2) {

LeaveCriticalSection(&cs);

return;

}

array[i][j] = 1;

}

if (point == 2) {

if (array[j][i] == 1) {

LeaveCriticalSection(&cs);

return;

}

array[j][i] = 2;

}

for (const auto& row : array) {

for (int value : row) {

std::cout << value << ' ';

}

std::cout << '\n';

}

std::cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

LeaveCriticalSection(&cs);

return;

}

DWORD WINAPI FirstGardener(LPVOID lpParam) {

for (int i = 0; i < 5; ++i) {

for (int j = 0; j < 5; ++j) {

print\_array(1, i, j);

std::this\_thread::sleep\_for(std::chrono::milliseconds(500));

}

}

ExitThread(0);

}

DWORD WINAPI SecondGardener(LPVOID lpParam) {

for (int i = 4; i > -1; --i) {

for (int j = 4; j > -1; --j) {

print\_array(2, i, j);

std::this\_thread::sleep\_for(std::chrono::milliseconds(500));

}

}

ExitThread(0);

}

int main() {

InitializeCriticalSection(&cs);

HANDLE gardener1, gardener2;

gardener1 = CreateThread(NULL, 0, FirstGardener, NULL, 0, NULL);

gardener2 = CreateThread(NULL, 0, SecondGardener, NULL, 0, NULL);

WaitForSingleObject(gardener1, INFINITE);

WaitForSingleObject(gardener2, INFINITE);

DeleteCriticalSection(&cs);

CloseHandle(gardener1);

CloseHandle(gardener2);

return 0;

}

**Выводы**

В ходе выполнения лабораторной работы была решена задача нелюдимых садовниках с помощью реализации многопоточного приложения на языке C++.