**Лабораторная работа: Инкремент и декремент переменной, изменение значений переменной**

**Теоретический материал**

В многопоточных приложениях важно корректно управлять общими ресурсами, такими как переменные. Одним из способов обеспечения потокобезопасности является использование атомарных операций, которые гарантируют, что операция будет выполнена целиком без вмешательства других потоков.

**Инкремент и декремент переменной** — это операции, которые изменяют значение переменной на единицу. Они могут быть реализованы с помощью атомарных функций, таких как InterlockedIncrement и InterlockedDecrement, что позволяет избежать состояния гонки при работе с общими переменными.

**Изменение значений переменной** также может быть выполнено с использованием атомарных операций, таких как InterlockedExchange, что позволяет безопасно менять значение переменной, предотвращая конфликты между потоками.

**Цели и задания**

1. Научиться использовать атомарные операции для инкремента и декремента переменной.
2. Изучить методы изменения значений переменной в многопоточном окружении.
3. Реализовать многопоточную программу, которая демонстрирует инкремент и декремент переменной.

**Демонстрационный код**

#include <iostream>

#include <windows.h>

#include <io.h>

#include <fcntl.h>

LONG sharedVariable = 0; // Общая переменная для потоков

DWORD WINAPI IncrementThreadFunction(LPVOID lpParam) {

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

LONG incrementedValue = InterlockedIncrement(&sharedVariable);

std::wcout << L"Поток " << GetCurrentThreadId() << L" инкрементировал значение: " << incrementedValue << std::endl;

Sleep(100); // Задержка для демонстрации

}

return 0;

}

DWORD WINAPI DecrementThreadFunction(LPVOID lpParam) {

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

LONG decrementedValue = InterlockedDecrement(&sharedVariable);

std::wcout << L" Поток " << GetCurrentThreadId() << L" декрементировал значение: " << decrementedValue << std::endl;

Sleep(100); // Задержка для демонстрации

}

return 0;

}

int main() {

// Настройка консоли для вывода Unicode

\_setmode(\_fileno(stdout), \_O\_U16TEXT);

HANDLE incrementThreads[2];

HANDLE decrementThreads[2];

// Создание потоков для инкремента

for (int i = 0; i < 2; ++i) {

incrementThreads[i] = CreateThread(NULL, 0, IncrementThreadFunction, NULL, 0, NULL);

}

// Создание потоков для декремента

for (int i = 0; i < 2; ++i) {

decrementThreads[i] = CreateThread(NULL, 0, DecrementThreadFunction, NULL, 0, NULL);

}

// Ожидание завершения всех потоков

WaitForMultipleObjects(2, incrementThreads, TRUE, INFINITE);

WaitForMultipleObjects(2, decrementThreads, TRUE, INFINITE);

std::wcout << L"Конечное значение переменной: " << sharedVariable << std::endl;

// Закрываем дескрипторы потоков

for (int i = 0; i < 2; ++i) {

CloseHandle(incrementThreads[i]);

CloseHandle(decrementThreads[i]);

}

return 0;

}

**Ожидаемый результат**

При запуске программы вы должны увидеть вывод, показывающий, как потоки инкрементируют и декрементируют общую переменную. Конечное значение переменной будет равно разности между инкрементами и декрементами.

**Самостоятельные задания**

1. **Задание 1:** Измените программу так, чтобы она инкрементировала переменную на 2 вместо 1. Используйте InterlockedAdd для этой операции.
   * **Подсказка:** Замените InterlockedIncrement на InterlockedAdd с добавлением 2.
2. **Задание 2:** Добавьте еще один поток, который будет выполнять инкрементирование переменной и выводить результат.
   * **Подсказка:** Создайте новый поток с той же функцией инкрементации.

**Контрольные вопросы**

1. Что такое атомарные операции?
2. Какую роль играют InterlockedIncrement и InterlockedDecrement в многопоточном программировании?
3. Почему важно использовать атомарные операции при работе с общими переменными?
4. Какое значение будет у sharedVariable после выполнения двух потоков инкрементации и двух потоков декрементации?
5. Как можно предотвратить состояние гонки в многопоточной программе?
6. Какие еще атомарные операции вы знаете?
7. Что произойдет, если два потока попытаются одновременно изменить одну и ту же переменную без атомарных операций?
8. Как вы можете вывести текущее значение переменной в конце работы программы?
9. В чем разница между инкрементом и декрементом переменной?
10. Как вы можете протестировать свою программу на наличие ошибок?