**Лабораторная работа №3**

1. Теоретические сведения.

При выполнении параллельных потоков возможна проблема синхронизации, когда два или более потоков обрабатывают разделяемые данные и конечный результат зависит от соотношения скоростей потоков.

Для решения проблемы синхронизации существует несколько различных методов.

Разделяемые переменные: вводится дополнительная переменная, которая отвечает за доступ к критической секции. Например – алгоритм Деккера:

/\* Номер потока: 1 или 2. Приведен код потока 1. \*/

flag1 = false; flag2 = false; turn = 1;

/\* flagi – намерение i-го потока войти в КС \*/

/\* turn – номер потока, имеющего право войти первым \*/

while( true ){

flag1 = true;

l1:if( flag2 ){

if( turn == 1 )

goto l1;

else{

flag1 = false;

while( turn == 2 ){}

}

}

else{

CSi(); turn = 2; flag1 = false;

}

}

Или алгоритм Петерсона:

/\* i – номер потока (0 или 1) \*/

int ready[2] = {0, 0}; /\* Намерение войти в КС \*/

int turn = 0; /\* Приоритетный поток \*/

while( true ) {

ready[i] = 1;

turn = 1 - i;

while( ready[1-i] && turn == 1-i )

;

CSi();

ready[i] = 0;

NCSi();

}

2. Задание:

Есть глобальная строковая переменная. Есть несколько потоков (количество потоков задается пользователем с клавиатуры), которые работают с глобальной переменной, каждый поток 10 раз записывает в строку свой уникальный символ. Например так

|  |
| --- |
| // Функция потока  void f1(char key)  {     for( i=0; i < 10; i ++)    {         Strcat(str, key);         Sleep(1000)    }  } |

1.     Написать программу, которая будет запускать эти потоки,  посмотреть результаты.

2.     Используя переменные блокировки правильно реализовать работу программы, чтобы не происходило нарушения работы.

3.     Реализовать вариант программы с использованием семафоров или мьютексов.

3. Рекомендации:

* + При написании программ желательно вставить паузу, чтобы код потоков выполнялся в течении нескольких квантов времени.