***Министерство образования Республики Беларусь***

***Учреждение образования***

***«Брестский государственный технический университет»***

***Кафедра ИИТ***

**Лабораторная работа №1**

**По дисциплине АОИС за III семестр**

**Тема: «Разработка многопоточного приложения на основе библиотеки thread»**

**Выполнил:**

Студент группы ИИ-18

2-го курса

Омельчук В.А

**Проверила:**

**Слинко Е.В.**

Брест 2020

Цель работы: изучить основные принципы работы с библиотекой thread. Получить навыки разработки многопоточных приложений на языке C++.

Вариант 10

1. Реализовать игру «камень-ножницы-бумага». 3 различных потока генерируют случайным образом свои ходы. Программа запускает 3 потока и принимает их результат (ход) и выводит результат хода на экран и в файл. Провести не менее 1000 ходов. По завершении выполнения ходов вывести результат игры.

Код:

#include <iostream>

#include <ctime>

#include <fstream>

using namespace std;

HANDLE hSemaphore;

int a = 1, b = 1, c = 1;

void aa(void\* pParams)

{

srand(time(0));

int s = 0;

int n = \*((int\*)pParams);

while (n) {

s = (rand()/2) % 3 + 1;

WaitForSingleObject(hSemaphore, INFINITE);

a = s;

n--;

}

\_endthread();

}

void bb(void\* pParams)

{

srand(time(0));

//Sleep(1000);

int s = 0;

int n = \*((int\*)pParams);

while (n) {

s = rand() % 3 + 1;

WaitForSingleObject(hSemaphore, INFINITE);

b = s;

n--;

}

\_endthread();

}

void cc(void\* pParams)

{

srand(time(0));

//Sleep(1000);

int s = 0;

int n = \*((int\*)pParams);

while (n) {

s = (rand()\*2) % 3 + 1;

WaitForSingleObject(hSemaphore, INFINITE);

c = s;

n--;

}

\_endthread();

}

int main(void)

{

ofstream fout;

fout.open("file.txt");

srand(time(0));

setlocale(LC\_ALL, "Russian");

int aaa = 0, bbb = 0, ccc = 0;

int n = 1000;

\_beginthread(aa, 0, &n);

\_beginthread(bb, 0, &n);

\_beginthread(cc, 0, &n);

hSemaphore = CreateSemaphore(NULL, 3, 3, NULL);

while (n)

{

Sleep(20);

cout << a << " " << b << " " << c << endl;

fout << a << " " << b << " " << c << endl;

switch (a) {

case 1: switch (b) {

case 1:switch (c) {

case 2:ccc++; break;

case 3:aaa++; bbb++; break;

default:break;

}break;

case 2:switch (c) {

case 1:bbb++; break;

case 2:bbb++; ccc++; break;

default:break;

}break;

case 3:switch (c) {

case 1:aaa++; ccc++; break;

case 3:aaa++; break;

default:break;

}break;

default:break;

}break;

case 2:switch (b) {

case 1:switch (c) {

case 1:aaa++; break;

case 2:aaa++; ccc++; break;

default:break;

}break;

case 2:switch (c) {

case 1:aaa++; bbb++; break;

case 3:ccc++; break;

default:break;

}break;

case 3:switch (c) {

case 2:bbb++; break;

case 3:bbb++; ccc++; break;

default:break;

}break;

default:break;

}break;

case 3:switch (b) {

case 1:switch (c) {

case 1:bbb++;ccc++; break;

case 3:bbb++; break;

default:break;

}break;

case 2:switch (c) {

case 2:aaa++; break;

case 3:aaa++; ccc++; break;

default:break;

}break;

case 3:switch (c) {

case 1:ccc++; break;

case 2:bbb++; aaa++; break;

default:break;

}break;

default:break;

}break;

default:break;

}

ReleaseSemaphore(hSemaphore, 3, NULL);

n--;

}

cout << aaa << " " << bbb << " " << ccc << endl;

fout << "\t"<<aaa << " " << bbb << " " << ccc << endl;

cout<<"The end!" << endl;

\_endthread();

CloseHandle(hSemaphore);

fout.close();

system("Pause");

return 0;

}

Вывод: изучил основные принципы работы с библиотекой thread. Получил навыки разработки многопоточных приложений на языке C++.