***Министерство образования Республики Беларусь***

***Учреждение образования***

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

***Кафедра ИИТ***

**Лабораторная работа №5**

**По дисциплине АОИС за III семестр**

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

**Выполнил:**

Студент группы ИИ-15 (1)

2-го курса

Волк И. А.

**Проверил:**

Михно Е. В.

Брест 2018

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

Задание.

1. Один поток открывает и считывает файл, второй поток вычисляет среднее значение всей последовательности чисел, и среднее, максимальное и минимальное значение каждых 50 значений, третий поток выводит данные в новый файл в сгруппированном виде, блоками по 50 значений, с указанием среднего, максимального и минимального значения по каждому блоку.

Код программы:

#include <iostream>

#include <fstream>

#include <conio.h>

#include <iomanip>

#include <thread>

#include <mutex>

#include <array>

#include <chrono>

const int N = 50;

void **fillRand**(const std::string& filename, int numOfBlocks)

{

std::ofstream stream(filename, std::ios::out | std::ios::ate);

if (!stream)

{

std::cerr << "Cannot open the file! (" << filename.c\_str() << ")" << std::endl;

return;

}

stream.clear();

srand(time(0));

while (numOfBlocks--)

{

for (int j = 0; j < N; j++)

stream << rand() % 50 << ((j != N - 1) ? " " : "");

if (numOfBlocks != 0)stream << "\n";

}

stream.close();

}

void **clear**(std::string filename)

{

std::ofstream stream(filename);

if (!stream)

{

std::cerr << "Cannot open the file! (" << filename.c\_str() << ")" << std::endl;

return;

}

stream.clear();

stream.close();

}

struct **PrResult**

{

int min;

int max;

double average;

};

enum **ThreadQueue**

{

READ = 0,

PROCESS,

WRITE

};

int **strToInt**(std::string str)

{

int res = 0;

for (int i = 0; i < str.length(); i++)

{

res += (str[i] - 48) \* pow(10, str.length() - i - 1);

}

return res;

}

bool proc = true;

void **read**(std::ifstream& stream, int arr[N], ThreadQueue& q)

{

bool toBreak = false;

while (!toBreak)

{

while (q != READ) std::this\_thread::sleep\_for(std::chrono::microseconds(10));

toBreak = false;

for (int i = 0; i < N; i++)

{

std::string str;

while (true)

{

char ch;

stream.read(&ch, sizeof(char));

if (stream.eof())

{

toBreak = true;

proc = false;

break;

}

if (ch == '\n')

{

break;

}

else if (ch == ' ')

{

break;

}

else if (ch < 48 || ch > 57 && ch != ' ')

{

std::cerr << "Error. Unexpected character \"" << ch << "\"!" << std::endl;

toBreak = true;

break;

}

str += ch;

}

arr[i] = strToInt(str);

q = PROCESS;

}

}

}

void **write**(std::ofstream& stream, int arr[N], PrResult &result, ThreadQueue& q)

{

while (proc)

{

while (q != WRITE) std::this\_thread::sleep\_for(std::chrono::microseconds(10));

for (int i = 0; i < N; i++)

{

if (i % 10 == 0)

stream << '\n';

else

stream << ' ';

stream << arr[i];

}

stream << std::endl << "Min - " << result.min << std::endl;

stream << "Max - " << result.max << std::endl;

stream << "Average - " << result.average << std::endl;

q = READ;

}

}

void **process**(int arr[N], PrResult &result, ThreadQueue& q)

{

while (proc)

{

while (q != PROCESS) std::this\_thread::sleep\_for(std::chrono::microseconds(10));

result.min = arr[0];

result.max = arr[0];

result.average = 0;

for (int i = 0; i < N; i++)

{

if (arr[i] < result.min)

result.min = arr[i];

if (arr[i] > result.max)

result.max = arr[i];

result.average += arr[i];

}

result.average /= N;

q = WRITE;

}

}

int **main**()

{

clear("res/2.txt");

int arr[N];

PrResult res;

ThreadQueue q = READ;

fillRand("res/1.txt", 4);

std::ofstream out("res/2.txt");

std::ifstream in("res/1.txt");

std::thread readThread(read, std::ref(in), arr, std::ref(q));

std::thread processThread(process, arr, std::ref(res), std::ref(q));

std::thread writeThread(write, std::ref(out), arr, std::ref(res), std::ref(q));

readThread.join();

processThread.join();

writeThread.join();

out.close();

in.close();

std::cin.get();

return 0;

}

Исходные данные (1.txt):

26 19 44 17 28 9 24 48 34 25 44 33 22 17 49 22 28 18 4 10 11 19 20 33 7 21 35 14 3 20 33 49 2 36 28 43 48 46 32 33 34 9 6 25 25 29 7 23 31 45

26 9 12 25 27 9 39 32 34 38 22 19 31 39 20 20 19 42 6 1 6 30 19 47 33 46 39 3 47 30 33 44 46 37 18 26 0 10 40 5 5 20 14 32 39 31 13 14 21 2

36 19 30 6 19 45 17 9 22 41 20 19 3 23 35 36 40 19 43 40 27 23 19 28 12 32 48 49 17 37 38 41 38 37 0 32 45 46 47 38 2 40 37 13 27 45 25 18 21 46

45 17 28 27 12 1 1 49 23 34 18 16 35 29 49 17 25 46 5 4 6 46 27 14 4 17 27 48 27 25 9 22 33 10 17 48 17 19 24 2 11 26 28 0 18 31 29 45 7 29

Результат выполнения (2.txt):

26 19 44 17 28 9 24 48 34 25

44 33 22 17 49 22 28 18 4 10

11 19 20 33 7 21 35 14 3 20

33 49 2 36 28 43 48 46 32 33

34 9 6 25 25 29 7 23 31 45

Min - 2

Max - 49

Average - 25.76

26 9 12 25 27 9 39 32 34 38

22 19 31 39 20 20 19 42 6 1

6 30 19 47 33 46 39 3 47 30

33 44 46 37 18 26 0 10 40 5

5 20 14 32 39 31 13 14 21 2

Min - 0

Max - 47

Average - 24.4

36 19 30 6 19 45 17 9 22 41

20 19 3 23 35 36 40 19 43 40

27 23 19 28 12 32 48 49 17 37

38 41 38 37 0 32 45 46 47 38

2 40 37 13 27 45 25 18 21 46

Min - 0

Max - 49

Average - 29

45 17 28 27 12 1 1 49 23 34

18 16 35 29 49 17 25 46 5 4

6 46 27 14 4 17 27 48 27 25

9 22 33 10 17 48 17 19 24 2

11 26 28 0 18 31 29 45 7 29

Min - 0

Max - 49

Average - 22.94

Вывод: изучил обучение и функционирование релаксационных ИНС в качестве ассоциативной памяти при решении задач распознавания образов.