МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО»

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

дисциплина «Программирование сетевых приложений»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

Организация распределённых вычислений с

использованием сокет TCP/IP средствами WinAPI

Выполнил: студент гр. ИТП-41

Курченок Т.Ю.

Принял: преподаватель

Гуменников Е.Д.

Гомель 2022

**Цель работы:** научиться организовывать распределенные вычисления с использованием сокетов *TCP*/*IP* средствами *WinAPI*.

**Задание:**

Распределенное вычисление площади поверхности фигуры одним из предложенных по варианту методов. Исходными данными для вычисления являются:

● отрезок [A;B];

● точность вычислений;

● количество клиентов, решающих задачу.

Вычислить площадь фигуры, описанную следующей функцией

методом левых прямоугольников.

**Ход выполнения**

В ходе выполнения лабораторной работы было разработано два консольных приложения Server и Client. Server принимает заданное количество клиентов и две переменные начало и конец отрезка, на которых считается заданная функция. Client каждый клиент считает свою часть отрезка, которой делиться поровну на всех клиентов. На рисунках 1 – 3 представлены результаты работы сервера и двух клиентов.

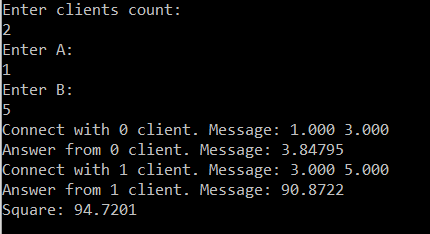


Рисунок 1 – Работа сервера

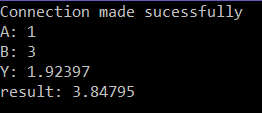


Рисунок 2 – Работа 1 клиента

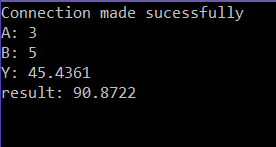


Рисунок 3 – Работа 2 клиента

Результаты были проверены с помощью онлайн калькулятора. Результаты представлены на рисунке 4.

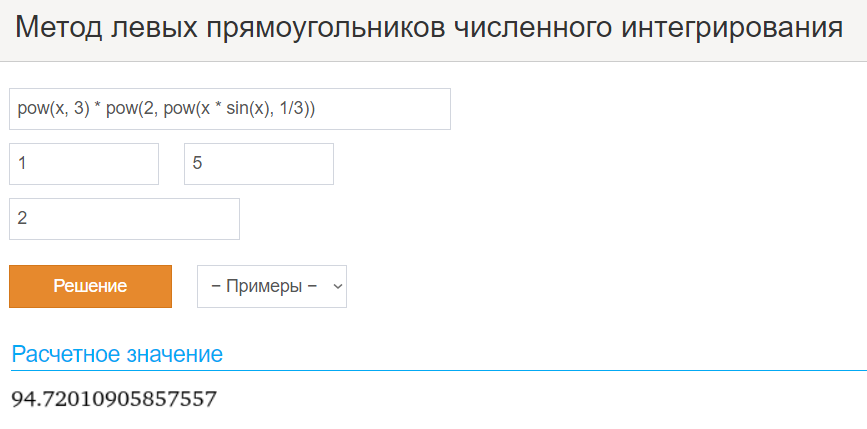


Рисунок 4 – Результаты вычислений калькулятора

Так как значения совпадают, можно сделать вывод, что программа работает правильно.

Следует отметить, что так как значения, высчитываемые функцией, не велики, разница между количеством клиентов будет мало заметной в скорости вычислений.

**Вывод**: в ходе выполнения данной лабораторной работы было разработано сетевое приложение, которое организует распределенные вычисления с использованием сокетов *TCP*/*IP* средствами *WinAPI*.

**Листинг приложения Server:**

**Листинг SeverTCP.cpp:**

#include "ServerTCP.h"

Server::Server(int port, int clientCount)

{

square = 0.0;

clientFinished = 0;

this->clientCount = clientCount;

clients = new SOCKET[clientCount];

clientThreads = new std::thread[clientCount];

sockVer = MAKEWORD(2, 2);

retVal = WSAStartup(sockVer, &wsaData);

//Создаем сокет

sock = socket(PF\_INET, SOCK\_STREAM, IPPROTO\_TCP);

if (sock == INVALID\_SOCKET)

{

std::cout << "Unable to create socket" << std::endl;

WSACleanup();

}

SOCKADDR\_IN sin;

sin.sin\_family = PF\_INET;

sin.sin\_port = htons(port);

sin.sin\_addr.s\_addr = INADDR\_ANY;

retVal = bind(sock, (LPSOCKADDR)&sin, sizeof(sin));

if (retVal == SOCKET\_ERROR)

{

std::cout << "Unable to bind" << std::endl;

WSACleanup();

}

}

std::thread\* Server::listenClients(double start, double end)

{

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

{

int retVal = listen(sock, 10);

if (retVal == SOCKET\_ERROR)

{

std::cout << "Unable to listen" << std::endl;

}

clients[i] = accept(sock, NULL, NULL);

if (clients[i] == INVALID\_SOCKET)

{

std::cout << "Unable to accept" << std::endl;

}

clientThreads[i] = std::thread(ProcessingClients, clients[i], clientCount, i, &clientFinished, start, end, &square);

clientThreads[i].detach();

}

return clientThreads;

}

double Server::get\_square()

{

return square;

}

void ProcessingClients(SOCKET client, int count, int index, int\* clientFinished, double start, double end, double\* square) {

const int RESVBUF = 32;

double step = (end - start) / count;

double x1 = step \* (double)index + start;

double x2 = step \* (1 + (double)index) + start;

std::stringstream stream\_x1;

stream\_x1 << std::fixed << std::setprecision(3) << x1;

std::stringstream stream\_x2;

stream\_x2 << std::fixed << std::setprecision(3) << x2;

std::string sendMessage = stream\_x1.str() + " " + stream\_x2.str();

std::cout << "Connect with " << index << " client. Message: " << sendMessage << std::endl;

int err = 0;

err = send(client, sendMessage.c\_str(), sendMessage.length(), 0);

char receiveMessageBuf[RESVBUF];

err = recv(client, receiveMessageBuf, RESVBUF, 0);

std::string receiveMessage = "";

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

{

receiveMessage += receiveMessageBuf[i];

}

double receive = atof(receiveMessage.c\_str());

std::cout << "Answer from " << index << " client. Message: " << receive << std::endl;

\*square += receive;

\*clientFinished += 1;

}

Server::~Server()

{

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

{

closesocket(clients[i]);

}

delete[] clientThreads;

delete[] clients;

closesocket(sock);

WSACleanup();

}

**Листинг Sever.cpp:**

#include "ServerTCP.h"

int main()

{

int clientCount = 0;

std::string input;

std::string a;

std::string b;

do

{

std::cout << "Enter clients count:" << std::endl;

std::cin >> input;

} while (input[0] < '0' || input[0] > '9');

clientCount = stoi(input);

Server server(1000, clientCount);

do

{

std::cout << "Enter A:" << std::endl;

std::cin >> a;

} while (a[0] == '-');

do

{

std::cout << "Enter B:" << std::endl;

std::cin >> b;

} while (b[0] == '-');

std::thread\* threads = server.listenClients(stoi(a), stoi(b));

while (server.clientFinished != clientCount);

std::cout << "Square: " << server.get\_square() << std::endl;

int tmp = getchar();

return 0;

}

**Листинг SeverTCP.h:**

#pragma once

#include <winsock2.h>

#include <thread>

#include <iostream>

#include <string>

#include <iomanip>

#include <sstream>

class Server

{

public:

Server(int port, int clientCount);

std::thread \* listenClients(double start, double end);

double get\_square();

~Server();

int clientFinished;

private:

SOCKET sock;

SOCKET\* clients;

std::thread\* clientThreads;

WSADATA wsaData;

WORD sockVer;

int retVal;

int clientCount;

double square;

};

void ProcessingClients(SOCKET client, int count, int index, int\* clientFinished, double start, double end, double\* square);

**Листинг приложения Client:**

**Листинг Client TCP.cpp:**

#include "ClientTCP.h"

ClientTCP::ClientTCP(int port)

{

ver = MAKEWORD(2, 2);

retVal = WSAStartup(ver, (LPWSADATA)&wsaData);

hostEnt = gethostbyname("localhost");

if (!hostEnt)

{

std::cout << "Unable to collect gethostbyname" << std::endl;

WSACleanup();

}

sock = socket(PF\_INET, SOCK\_STREAM, IPPROTO\_TCP);

if (sock == SOCKET\_ERROR)

{

std::cout << "Unable to create socket" << std::endl;

WSACleanup();

}

SOCKADDR\_IN serverInfo;

serverInfo.sin\_family = PF\_INET;

if (hostEnt != nullptr)

{

serverInfo.sin\_addr = \*((LPIN\_ADDR)\*hostEnt->h\_addr\_list);

}

serverInfo.sin\_port = htons(port);

retVal = connect(sock, (LPSOCKADDR)&serverInfo, sizeof(serverInfo));

if (retVal == SOCKET\_ERROR)

{

std::cout << "Unable to connect" << std::endl;

WSACleanup();

} else {

std::cout << "Connection made sucessfully" << std::endl;

}

}

ClientTCP::~ClientTCP()

{

closesocket(sock);

WSACleanup();

}

void ClientTCP::sendMessage(std::string message)

{

retVal = send(sock, message.c\_str(), message.length(), 0);

if (retVal == SOCKET\_ERROR)

{

std::cout << "Unable to send" << std::endl;

WSACleanup();

}

}

std::string ClientTCP::receiveMessage(int size)

{

char \*szResponse = new char[size];

retVal = recv(sock, szResponse, size, 0);

if (retVal == SOCKET\_ERROR)

{

std::cout << "Unable to receive" << std::endl;

WSACleanup();

}

return to\_string(szResponse, size);

}

std::string to\_string(char\* arr, int size) {

std::string s = "";

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

s = s + arr[i];

}

return s;

}

**Листинг Client.cpp:**

#include <iostream>

#include <cmath>

#include <cstdlib>

#include "ClientTCP.h"

double function(double x) {

return pow(x, 3) \* pow(2, pow(x \* sin(x), 1./3));

}

double method\_left\_rectangles(double x1, double x2, double y1) {

return y1 \* (x2 - x1);

}

int main()

{

std::string message;

double x1, x2, y1;

double result = 0;

ClientTCP client(1000);

message = client.receiveMessage(15);

x1 = atof(message.substr(0, message.find(' ')).c\_str());

x2 = atof(message.substr(message.find(' ') + 1, message.length() - message.find(' ') - 1).c\_str());

y1 = function(x1);

std::cout << "A: " << x1 << std::endl;

std::cout << "B: " << x2 << std::endl;

std::cout << "Y: " << y1 << std::endl;

result = method\_left\_rectangles(x1, x2, y1);

std::cout << "result: " << result << std::endl;

client.sendMessage(std::to\_string(result));

int tmp = getchar();

return 0;

}

**Листинг Client TCP.h:**

#pragma once

#define \_WINSOCK\_DEPRECATED\_NO\_WARNINGS

#include <string>

#include <winsock2.h>

#include <iostream>

class ClientTCP

{

public:

ClientTCP(int port);

~ClientTCP();

void sendMessage(std::string message);

std::string receiveMessage(int size);

private:

SOCKET sock;

WSADATA wsaData;

LPHOSTENT hostEnt;

WORD ver;

int retVal;

};

std::string to\_string(char\* arr, int size);