

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет Вычислительной техники
Кафедра "Вычислительная техника"

"УТВЕРЖДАЮ"

Зав. кафедрой ВТ

 профессор М.А. Митрохин

« 03 » марта 2021г

ЗАДАНИЕ

на курсовое проектирование по курсу

Распределенные вычисления
Студенту Кливару Владимиру Викторову Группа 17ВВ2
Тема проекта Разработка переносимой программы для мастера облачного типа

Исходные данные (технические требования) на проектирование

- 1) КС приложение для управления вычислительными процессами из двух частей: клиент и сервер, обеспечивающих распределенную вычислительную систему облачного типа
- 2) Обеспечение нормирования нескольких клиентов
- 3) Клиент с возможностью нормирования и оптимизации от сервера
- 4) Отображение информации о состоянии сервера
- 5) Оптимизация использования ресурсов клиента
- 6) Клиент должен решать задачу вычисления определенного значения $E^x dx$
- 7) Вычисление методом явных интегрирований
- 8) Графический интерфейс для клиента и сервера
- 9) Обмен данными по TCP/IP с библиотекой WinSock
- 10) Язык программирования C++.

Объем работы по курсу

1. Расчетная часть

2. Графическая часть

Диаграммы ЧМЛ

3. Экспериментальная часть

Тестирование разработанной вычислительной системы

Срок выполнения проекта по разделам

1 В соответствии с графиком

2

3

4

5

6

7

8

Дата выдачи задания "03" марта 2021

Дата защиты проекта "15" мая 2021

Руководитель Федюшин РИ

Задание получил "03" марта 2021 г.

Студент Акивов В.В.

Министерство образования Российской Федерации
Пензенский государственный университет
Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовой работе
по курсу «Распределенные вычисления»
на тему «Разработка параллельной программы
для кластера облачного типа»

Выполнил:

студент группы 17ВВ2

Живаев В.В.

Приняли:

Федюнин Р.Н.

Слепцов Н.В.

Пенза 2021

Содержание

Введение.....	4
1 Постановка задачи.....	6
2 Анализ предметной области и требований задачи	8
3 Разработка клиентской части приложения.....	10
4 Разработка серверной части приложения.....	12
5 Описание программы.....	14
6 Описание результатов работы программы и их анализ	18
7 Руководство пользователя.....	19
Заключение	21
Список используемых источников.....	22
Приложение А Листинг клиентской части приложения.....	23
Приложение А.1 Файл «MyForm.h».....	23
Приложение А.2 Файл «CClient.cpp».....	32
Приложение В Листинг серверной части приложения	37
Приложение В.1 Файл «MyForm.h»	37
Приложение В.2 Файл «SServer.cpp».....	51
Приложение В.4 Файл «FormUpdate.cpp»	53
Приложение В.5 Файл «Client.cpp».....	54
Приложение В.8 Файл «Integral.h».....	57
Приложение С Результаты работы программы	58
Приложение С.1 – Вид клиента в процессе вычисления	58
Приложение С.2 – Вид сервера в процессе вычисления.....	58
Приложение С.3 – Вид сервера с результатом вычисления	59
Приложение С.4 – Эталонный результат вычисления	59
Приложение С.5 – Завершение работы клиентом	60
Приложение С.6 – Расход ресурсов системы.....	60

Введение

«Клиент-сервер» — вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами. Фактически клиент и сервер — это программное обеспечение. Обычно эти программы расположены на разных вычислительных машинах и взаимодействуют между собой через вычислительную сеть посредством сетевых протоколов, но они могут быть расположены также и на одной машине. Программы-серверы ожидают от клиентских программ запросы и предоставляют им свои ресурсы в виде данных (например, загрузка файлов посредством HTTP, FTP, BitTorrent, потоковое мультимедиа или работа с базами данных) или в виде сервисных функций (например, работа с электронной почтой, общение посредством систем мгновенного обмена сообщениями или просмотр web-страниц во всемирной паутине). Поскольку одна программа-сервер может выполнять запросы от множества программ-клиентов, её размещают на специально выделенной вычислительной машине, настроенной особым образом, как правило, совместно с другими программами-серверами, поэтому производительность этой машины должна быть высокой. Из-за особой роли такой машины в сети, специфики её оборудования и программного обеспечения, её также называют сервером, а машины, выполняющие клиентские программы, соответственно, клиентами.

TCP — это ориентированный на соединения протокол, предназначенный для обеспечения надежной передачи данных между процессами, выполняемыми или на одном и том же компьютере, или на разных компьютерах. Термин "ориентированный на соединения" означает, что два процесса или приложения прежде, чем обмениваться какими-либо данными должны установить TCP-соединение. В этом TCP отличается от протокола UDP, являющегося протоколом "без установления соединения", позволяющим

выполнять широковещательную передачу данных неопределенному числу клиентов.

Когда приложение отправляет данные, используя ТСР, они перемещаются вниз по стеку протоколов. Данные проходят по всем уровням и в конце концов передаются через сеть как поток битов. Каждый уровень в наборе протоколов ТСР/IP добавляет к данным некоторую информацию в форме заголовков.

Когда пакет прибывает на конечный узел в сети, он снова проходит через все уровни снизу доверху. Каждый уровень проверяет данные, отделяя от пакета свою информацию в заголовке и наконец данные достигают серверного приложения в той же самой форме, в какой они покинули приложение-клиент.

В данной работе будет разработано приложение, которое выполняет вычисление интеграла, при этом параллелизм вычислений будет обеспечен как в рамках вычислительной сети, так и в рамках отдельного узла.

1 Постановка задачи

Требуется разработать клиент-серверное приложение, представляющее из себя параллельную программу для кластера облачного типа, которое позволяет вычислять определённый интеграл для заданной функции.

Требования к разработке:

1) Клиент-серверное приложение для управления вычислительным процессом должно состоять из двух частей: клиента и сервера, которые должны образовывать единую распределенную вычислительную систему облачного типа.

2) Серверное приложение должно обеспечивать одновременное подключение нескольких клиентов.

3) Клиентское приложение должно обеспечивать подключение к серверу (порт и IP сервера вводятся в окне клиента) и иметь возможность осуществить отключение от сервера в любой момент.

4) В окне серверного приложения должен отображаться прогресс решения вычислительной задачи и примерное время, оставшееся до ее решения с учетом *текущей* скорости вычислительного процесса, а также указываться список подключившихся клиентов и *шаг* интегрирования.

5) Для каждого клиента должна быть предусмотрена возможность указания предельно допустимого уровня использования вычислительных ресурсов (процессорного времени) в процентах. Соответствующее значение вводится на *клиентской* стороне.

6) Кластер должен решать задачу вычисления определенного интеграла $\int_{0.5}^1 e^{-x} dx$.

7) Для интегрирования должен использоваться метод *правых прямоугольников*.

8) Приложения (клиент и сервер) должны обладать интуитивно понятным графическим интерфейсом.

9) Для обмена данными использовать протокол TCP/IP и библиотека WinSock.

10) Язык программирования C++, среда разработки Microsoft Visual Studio.

Графическая часть: Диаграммы языка UML:

- 1) диаграмма классов;
- 2) диаграмма деятельности;
- 3) диаграмма вариантов использования;
- 4) диаграмма развёртывания;
- 5) диаграмма последовательности.

Экспериментальная часть: тестирование распределенной вычислительной системы.

2 Анализ предметной области и требований задачи

Для того чтобы лучше понять, что необходимо для реализации поставленной задачи, рассмотрим предметную область.

Исходя из требования к задаче, необходимо выбрать соответствующие инструменты для её реализации.

Для реализации был выбран язык программирования C++. Клиентская и серверная части приложения должны обладать графическим интерфейсом для комфортной работы пользователя с программой, исходя из этого был использован Windows/CLI для организации приложения типа “Windows Forms”.

Основная задача приложения заключается в вычислении определённого интеграла. Это подразумевает, что необходимо обрабатывать большой объем данных, в том числе от нескольких клиентов сразу, по заданному алгоритму, который представляет собой математическую задачу. Исходя из вышеизложенного, можно сделать вывод, что необходимо организовать параллельный алгоритм вычисления интеграла, а также обработку клиентских запросов в многопоточном режиме. Тем самым мы достигнем большей эффективности на всем процессе исполнения. В C++ имеется возможность относительно просто создавать дополнительные процессы и нити, что позволит реализовать параллельную обработку нескольких запросов, исходя из этого требования будет использован WinApi – «Windows.h» пакет для решения задач СП.

Для передачи информации между клиентом и сервером будет использована библиотека WinSock, которая позволяет создавать сокеты и на их основе организовывать передачу информации между клиентом и сервером, в данной работе на основе TCP протокола.

С описанными инструментами можно реализовать поставленную задачу в соответствии со всеми изложенными требованиями.

Метод прямоугольников — метод численного интегрирования функции одной переменной, заключающийся в замене подынтегральной функции на многочлен нулевой степени, то есть константу, на каждом элементарном отрезке. Если рассмотреть график подынтегральной функции, то метод будет заключаться в приближённом вычислении площади под графиком суммированием площадей конечного числа прямоугольников, ширина которых будет определяться расстоянием между соответствующими соседними узлами интегрирования, а высота — значением подынтегральной функции в этих узлах.

Если отрезок $[a, b]$ является элементарным и не подвергается дальнейшему разбиению, значение интеграла можно найти по

- 1) Формуле левых прямоугольников:

$$\int_a^b f(x) dx \approx f(a)(b - a).$$

- 2) Формуле правых прямоугольников:

$$\int_a^b f(x) dx \approx f(b)(b - a).$$

- 3) Формуле средних прямоугольников:

$$\int_a^b f(x) dx \approx f\left(\frac{a+b}{2}\right)(b - a).$$

3 Разработка клиентской части приложения

В соответствии с требованиями, клиентское приложение должно обеспечивать: подключение к серверу (порт и IP сервера вводятся в окне клиента) и иметь возможность осуществить отключение от сервера в любой момент; нагрузка на ЦП клиента не должна превышать некоторый процент (который далее будет указан в виде конфигурации клиентской части); выполнять параллельный расчет интеграла методом правых прямоугольников.

Пользователю необходимо указать численное значение нагрузки (являющееся процентом общей нагрузки на ЦП), заполнить поля «IP» и «Port» для описания конечной точки на сервере, куда будет осуществлена попытка подключения и нажать кнопку «Подключиться», чтобы установить связь с сервером. Статус клиента отображается в строке статуса приложения, как и данные очередного расчета.

Согласно требованиям, интерфейс клиента имеет вид как на рисунке 1.

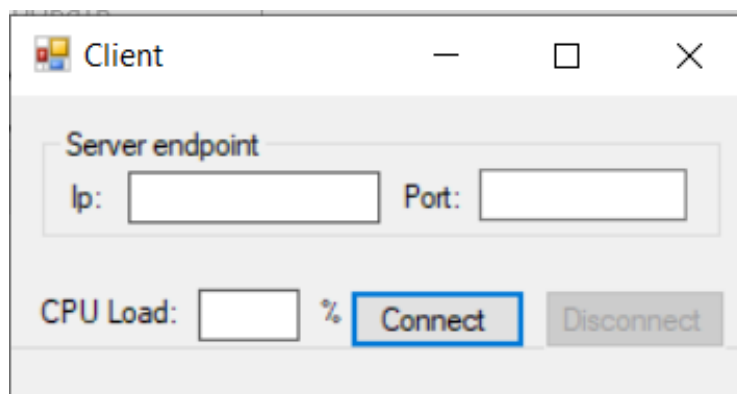


Рисунок 1 - Интерфейс клиентского приложения.

После подключения клиента к серверу, последний начинает его «обслуживание» т.е. выдачу задач. После начала выдачи задач или частей задачи, данные очередного вычисления будут указаны в строке статуса клиента. При этом вычисление организовано так, чтобы ресурсы ЦП клиента не расходовались больше, чем указано при начальной настройке.

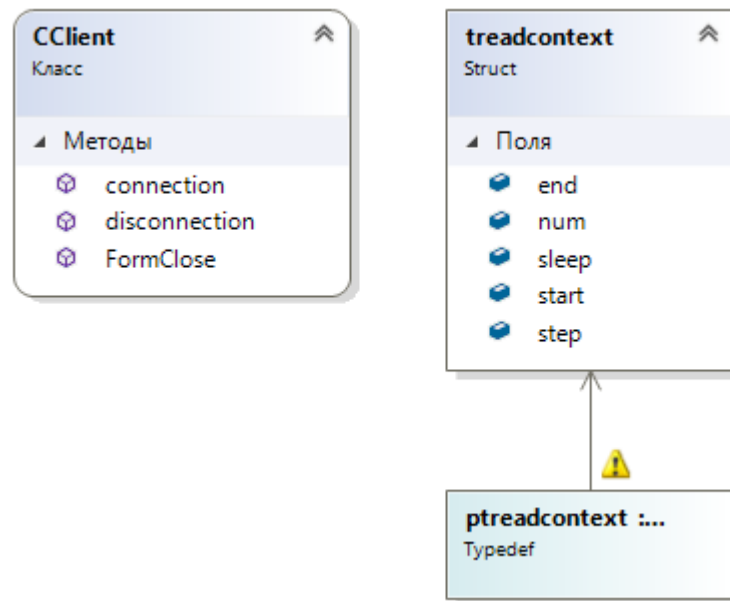


Рисунок 2 - UML-диаграмма классов клиента.

4 Разработка серверной части приложения

По указанным в постановке задачи требованиям серверное приложение должно обеспечивать отображение таких секций как:

- Настройка начальных условий для выполнения вычислений
- Настройка конечной точки сервера
- Список подключенных агентов-вычислителей
- Статус запущенного вычисления (прогресс, результат)

Согласно требованиям, интерфейс сервера имеет вид как на рисунке 4.

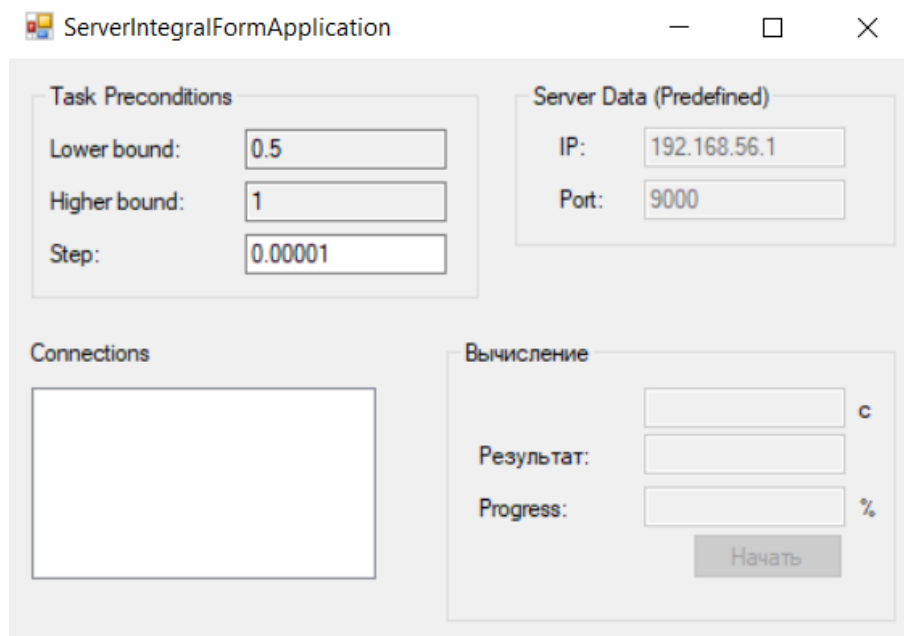


Рисунок 3 - Интерфейс серверного приложения.

После нажатия кнопки «Начать» в секции вычисления будет запущен метод класса сервера, который организует потоки прослушивания подключения новых агентов-вычислителей и менеджмента (назначения) очереди задач свободным ресурсам (агентам). По ходу процесса вычисления и получения сервером промежуточных результатов, а также метаданных выполнения задачи агентами будет производиться обновление основного view серверной части для отображения прогресса решения.

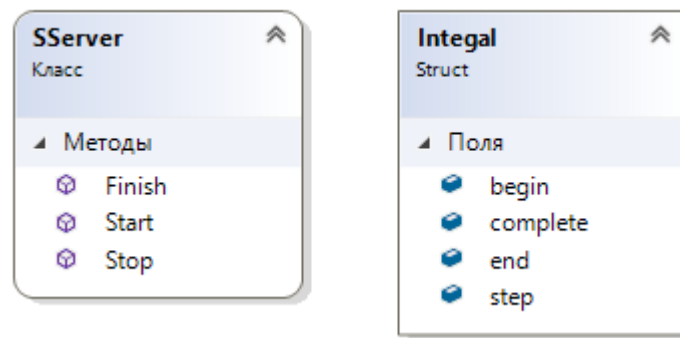


Рисунок 4 - UML-диаграмма классов сервера.

5 Описание программы

Приложение состоит из 2 частей:

- Клиент
- Сервер

При запуске сервера пользователь должен ввести следующие параметры для настройки:

- Данные задачи (пределы интегрирования)
- Данные серверной конечной точки (для подключения клиентов)

После заполнения указанной выше информации пользователь может нажать кнопку «Начать». После нажатия последней программа-сервер переходит в режим ожидания подключений новых агентов и ожидания назначения задач новым агентом, такое ожидание организовано путем создания системы из нескольких потоков.

После запуска сервера и перевода его в режим ожидания пользователь-клиент в клиентской части приложения может заполнить данные конечной точки сервера и осуществить подключения, тем самым предоставив свой вычислительный ресурс.

После подключения нового клиента поток-менеджер задач на сервере начинает назначение задач. Менеджер организован на основе критических секций и выдает задачу на вычисление интеграла порционно любому свободному клиенту. Клиент в свою очередь при получении части задачи своими средствами организует её многопоточное вычисление с разделением исходной задачи на более мелкие части.

На основе изложенного выше можно составить UML-диаграмму развертывания.

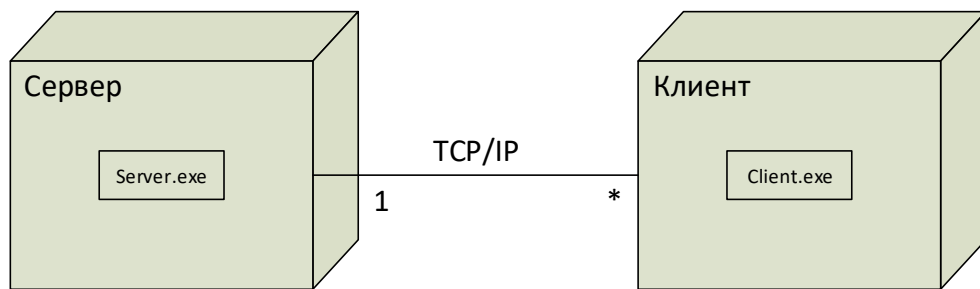


Рисунок 5 - UML-диаграмма развертывания.

На основе изложенного выше можно составить UML-диаграммы вариантов использования для клиента и сервера.

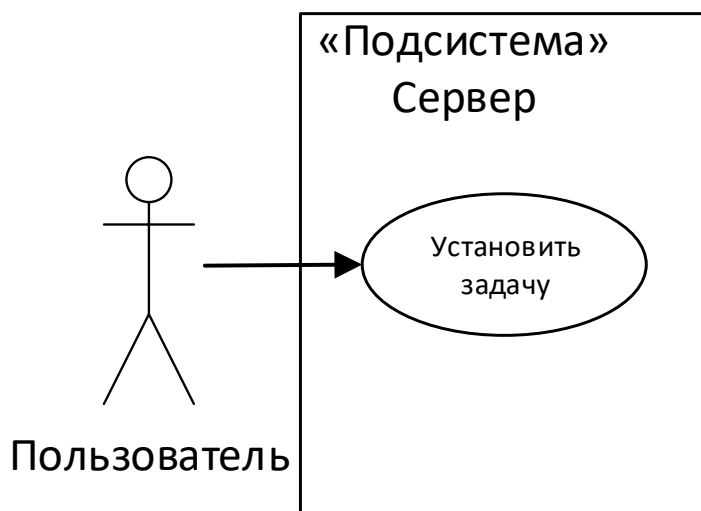


Рисунок 6 - UML-диаграмма вариантов использования сервера.

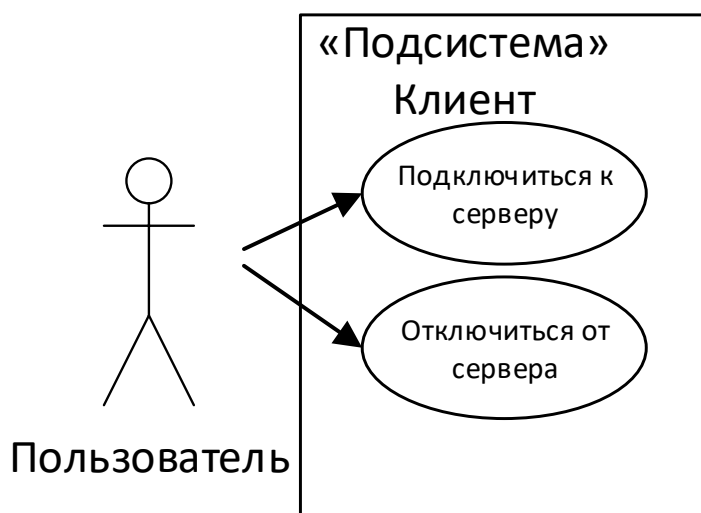


Рисунок 7 - UML-диаграмма вариантов использования клиента.

На основе изложенного выше можно составить UML-диаграмму деятельности.



Рисунок 8 - UML-диаграмма деятельности.

На основе изложенного выше можно составить UML-диаграмму последовательности.

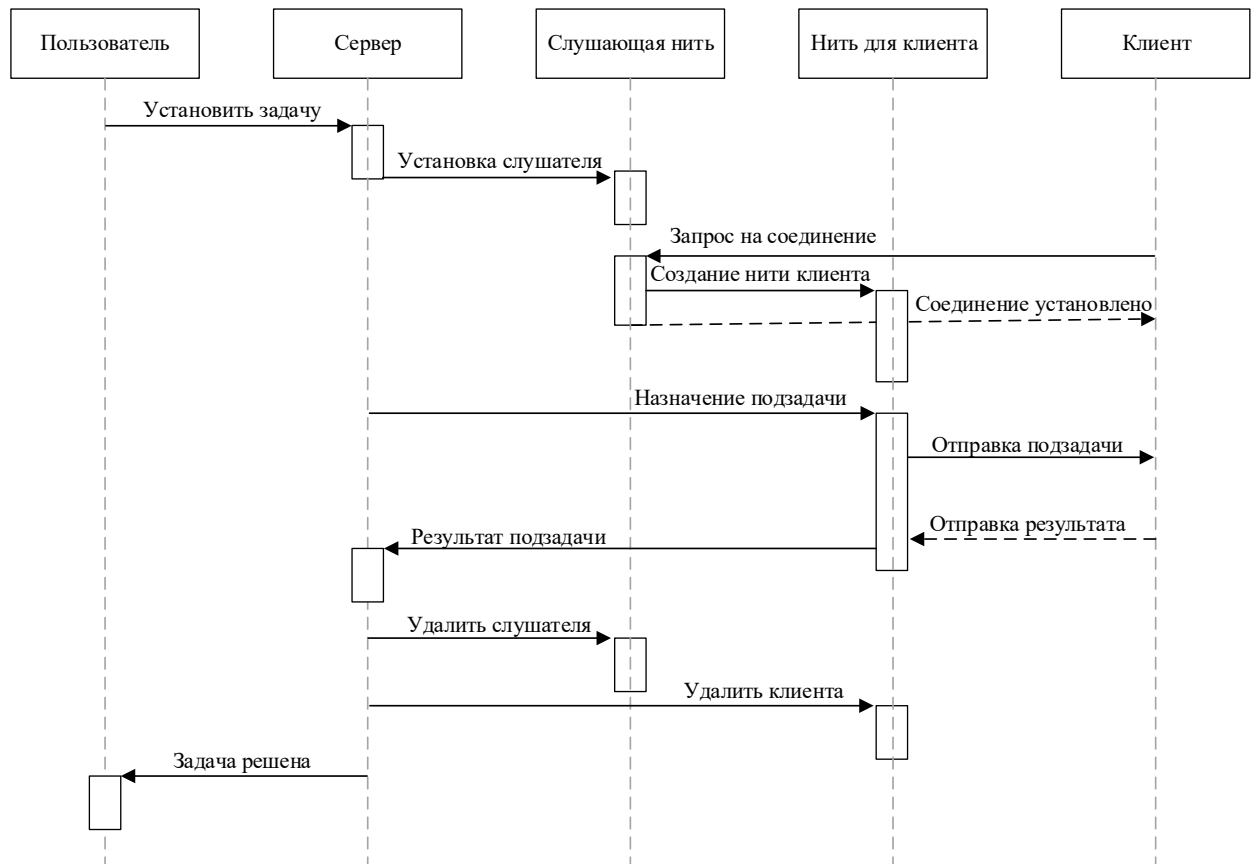


Рисунок 9 - UML-диаграмма последовательности.

6 Описание результатов работы программы и их анализ

В качестве среды разработки была выбрана программа MS Visual Studio 2017.

Тестирование проводилось в рабочем порядке, в процессе разработки и после завершения написания программы. В ходе тестирования было выявлено и исправлено все проблемы, связанные с взаимодействием нитей между собой.

В результате разработки, отладки и тестирования было разработано приложение параллельной обработки данных для кластера облачного типа для вычисления интеграла, которое отвечает заявленным требованиям.

В рамках предполагаемого улучшения данного комплекса приложений для серверной части можно предусмотреть функцию healthCheck (проверка жизнеспособности определенной ноды-агента) для своевременного получения информации о работающих в данный момент агентах.

Также одним из предполагаемых улучшений могло бы стать вычисление динамически задаваемого интеграла. Для обеих последних задач будут соответственно новые структуры данных для передачи, а также для второй задачи необходим строковый парсер, который смог бы собирать необходимый интеграл для вычисления.

7 Руководство пользователя

Приложение состоит из клиентской и серверной части.

В клиентской части пользователь должен ввести в поле «IP:» IP-адрес сервера, в поле «Port:» ввести порт сервера и в поле «Нагрузка:» задать предельно допустимый уровень использования вычислительных ресурсов (Загрузки ЦП) в процентах. После необходимо нажать кнопку «Подключиться» и клиентское приложение попытается подключиться к серверу. Если соединение установлено, то сервер, а точнее поток сервера, отвечающий за менеджмент задач и обслуживание клиентов начинает выдавать предварительно определенные части исходного задания для вычисления агентом, в ходе которого пользователь может нажать кнопку «Отключиться» и тогда клиентское приложение разорвёт соединение сервером. Также в секции статуса клиентской части (нижний бар окна) пользователь может увидеть интервал интеграла, полученный на вычисление в текущем моменте. Если соединение с сервером не удалось установить, то необходимо проверить поля «IP:» и «Порт:», а также соединение с сетью. Если подключиться все равно не удастся, то возможно сервер не работает.

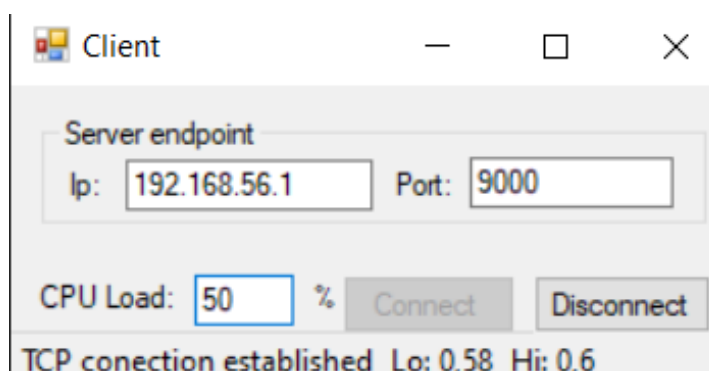


Рисунок 10 – Клиент в процессе вычисления.

В серверной части пользователь определяет задачу для вычисления. Чтобы установить задачу пользователю необходимо шаг интегрирования в поле «Шаг:». После нажатия кнопки «Начать» сервер начинает ожидать клиентов и, если таковые имеются, начинается раздача частей исходного

задания. Текущие клиенты отображаются в поле «Список подключенных клиентов:». Процент готовности задачи отображает элемент метка «Progress:», а примерное время (в секундах), оставшееся до завершения – поле «Remaining:». После готовности результата, его можно посмотреть в поле «Result:». И после пользователь может назначить другую задачу.

The screenshot shows a window titled "ServerIntegralFormApplication". It contains several sections:

- Task Preconditions:** Three input fields with values "0.5", "1", and "0.000001" labeled "Lower bound:", "Higher bound:", and "Step:" respectively.
- Server Data (Predefined):** Two input fields with values "192.168.56.1" and "9000" labeled "IP:" and "Port:" respectively.
- Connections:** A large text box containing the number "1248".
- Вычисление (Calculation):** A section with three input fields: "Remaining:" with value "4" and unit "с", "Результат:" (empty), and "Progress:" with value "16" and unit "%". Below these is a button labeled "Начать" (Start).

Рисунок 11 – Сервер во время вычисления.

Заключение

При выполнении данной курсовой работы были получены навыки разработки параллельной программы для кластера облачного типа, которое выполняет вычисление определённого интеграла. Так же были получены навыки программирования на языке C++. Освоены навыки работы с библиотекой WinSock. Получен опыт разработки клиент-серверного приложения с соответствующей архитектурой, а также решались задачи построения распределенного приложения на основе стека протокола с установлением соединения. Были решены проблемы синхронизации и проблемы обмена сообщений между клиентом и сервером, задачи структурирования отправляемых/получаемых наборов данных. Также получены навыки разработки параллельных программ на основе WinAPI. Изучены принципы построения и организации графического интерфейса на основе пакетов Windows/CLI и приложения типа “Windows Forms”.

Список используемых источников

1. MSDN.
2. Страуструп Б. «Язык программирования C++».
3. Лафоре Р. «Объектно-ориентированное программирование в C++»
4. Дейтел Х. «Как программировать на C++».

Приложение А

Листинг клиентской части приложения

Приложение А.1

Файл «MyForm.h»

```
#pragma once
#include <msclr\marshal_cppstd.h>
#include "CClient.h"

namespace Client {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;

    /// <summary>
    /// Сводка для MyForm
    /// </summary>
    public ref class MyForm : public
System::Windows::Forms::Form
    {
    public:
        CClient* Client;
        void ConnectButton()
        {
            button1->Enabled = true;
            button2->Enabled = false;
        }

        void ConnectButtonEnable()
        {
            if (button1->InvokeRequired)
                Invoke(gcnew Action(this,
&MyForm::ConnectButton));
            else
                button1->Enabled = true;
                button2->Enabled = false;
        }
    }
}
```

```

        void UpdateStatus(double state)
        {
            //this->button1->Text = "Подключиться";
            if (state == 1)
                this->toolStripStatusLabel1->Text = "Работа
завершена";
            else if (state == 2)
                this->toolStripStatusLabel1->Text = "Сервер
разорвал соединение";
            toolStripStatusLabel2->Text = "0";
            toolStripStatusLabel3->Text = "0";
        }

        MyForm(void)
        {
            InitializeComponent();
        }

    protected:
        ~MyForm()
        {
            if (components)
            {
                delete components;
            }
        }

    public: System::Windows::Forms::TextBox^   textBox1;
    protected:
    private: System::Windows::Forms::Label^   label1;
    public: System::Windows::Forms::Button^   button1;
    private: System::Windows::Forms::Label^   label2;
    private: System::Windows::Forms::TextBox^   textBox2;
    private: System::Windows::Forms::Label^   label3;
    private: System::Windows::Forms::TextBox^   textBox3;

    private: System::Windows::Forms::GroupBox^   groupBox1;
    private: System::Windows::Forms::StatusStrip^   statusStrip1;
    public: System::Windows::Forms::ToolStripStatusLabel^
toolStripStatusLabel1;
    public: System::Windows::Forms::Button^   button2;
    public: System::Windows::Forms::ToolStripStatusLabel^
toolStripStatusLabel2;
    public:
    public: System::Windows::Forms::ToolStripStatusLabel^
toolStripStatusLabel3;
    private: System::Windows::Forms::Label^   label4;
    public:

    private:
        /// <summary>

```

```

        /// Обязательная переменная конструктора.
        /// </summary>
        System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
        /// <summary>
        /// Требуемый метод для поддержки конструктора — не
изменяйте
        /// содержимое этого метода с помощью редактора кода.
        /// </summary>
        void InitializeComponent(void)
        {
            this->textBox1 = (gcnew
System::Windows::Forms::TextBox());
            this->label1 = (gcnew
System::Windows::Forms::Label());
            this->button1 = (gcnew
System::Windows::Forms::Button());
            this->label2 = (gcnew
System::Windows::Forms::Label());
            this->textBox2 = (gcnew
System::Windows::Forms::TextBox());
            this->label3 = (gcnew
System::Windows::Forms::Label());
            this->textBox3 = (gcnew
System::Windows::Forms::TextBox());
            this->groupBox1 = (gcnew
System::Windows::Forms::GroupBox());
            this->statusStrip1 = (gcnew
System::Windows::Forms::StatusStrip());
            this->toolStripStatusLabel1 = (gcnew
System::Windows::Forms::ToolStripStatusLabel());
            this->toolStripStatusLabel2 = (gcnew
System::Windows::Forms::ToolStripStatusLabel());
            this->toolStripStatusLabel3 = (gcnew
System::Windows::Forms::ToolStripStatusLabel());
            this->button2 = (gcnew
System::Windows::Forms::Button());
            this->label4 = (gcnew
System::Windows::Forms::Label());
            this->groupBox1->SuspendLayout();
            this->statusStrip1->SuspendLayout();
            this->SuspendLayout();
            //
            // textBox1
            //
            this->textBox1->Location =
System::Drawing::Point(44, 21);
            this->textBox1->Margin =
System::Windows::Forms::Padding(4);
            this->textBox1->MaxLength = 15;

```

```

        this->textBox1->Name = L"textBox1";
        this->textBox1->Size = System::Drawing::Size(128,
22);

        this->textBox1->TabIndex = 0;
        this->textBox1->Text = L"192.168.56.1";
        this->textBox1->KeyPress += gcnew
System::Windows::Forms::KeyPressEventHandler(this,
&MyForm::textBox1_KeyPress);
        //
        // label1
        //
        this->label1->AutoSize = true;
        this->label1->Location =
System::Drawing::Point(11, 25);
        this->label1->Margin =
System::Windows::Forms::Padding(4, 0, 4, 0);
        this->label1->Name = L"label1";
        this->label1->Size = System::Drawing::Size(23,
17);

        this->label1->TabIndex = 1;
        this->label1->Text = L"Ip:";
        //
        // button1
        //
        this->button1->Location =
System::Drawing::Point(174, 91);
        this->button1->Margin =
System::Windows::Forms::Padding(4);
        this->button1->Name = L"button1";
        this->button1->Size = System::Drawing::Size(91,
28);

        this->button1->TabIndex = 2;
        this->button1->Text = L"Connect";
        this->button1->UseVisualStyleBackColor = true;
        this->button1->Click += gcnew
System::EventHandler(this, &MyForm::button1_Click);
        //
        // label2
        //
        this->label2->AutoSize = true;
        this->label2->Location =
System::Drawing::Point(182, 24);
        this->label2->Margin =
System::Windows::Forms::Padding(4, 0, 4, 0);
        this->label2->Name = L"label2";
        this->label2->Size = System::Drawing::Size(38,
17);

        this->label2->TabIndex = 4;
        this->label2->Text = L"Port:";
        //
        // textBox2

```

```

        //
        this->textBox2->Location =
System::Drawing::Point(224, 20);
        this->textBox2->Margin =
System::Windows::Forms::Padding(4);
        this->textBox2->MaxLength = 5;
        this->textBox2->Name = L"textBox2";
        this->textBox2->Size = System::Drawing::Size(105,
22);

        this->textBox2->TabIndex = 3;
        this->textBox2->Text = L"9000";
        this->textBox2->KeyPress += gcnew
System::Windows::Forms::KeyPressEventHandler(this,
&MyForm::textBox2_KeyPress);
        //
        // label3
        //
        this->label3->AutoSize = true;
        this->label3->Location =
System::Drawing::Point(12, 94);
        this->label3->Margin =
System::Windows::Forms::Padding(4, 0, 4, 0);
        this->label3->Name = L"label3";
        this->label3->Size = System::Drawing::Size(76,
17);

        this->label3->TabIndex = 6;
        this->label3->Text = L"CPU Load:";
        //
        // textBox3
        //
        this->textBox3->Location =
System::Drawing::Point(96, 91);
        this->textBox3->Margin =
System::Windows::Forms::Padding(4);
        this->textBox3->MaxLength = 2;
        this->textBox3->Name = L"textBox3";
        this->textBox3->Size = System::Drawing::Size(51,
22);

        this->textBox3->TabIndex = 5;
        this->textBox3->KeyPress += gcnew
System::Windows::Forms::KeyPressEventHandler(this,
&MyForm::textBox3_KeyPress);
        //
        // groupBox1
        //
        this->groupBox1->Controls->Add(this->textBox1);
        this->groupBox1->Controls->Add(this->label1);
        this->groupBox1->Controls->Add(this->textBox2);
        this->groupBox1->Controls->Add(this->label2);
        this->groupBox1->Location =
System::Drawing::Point(16, 15);

```

```

        this->groupBox1->Margin =
System::Windows::Forms::Padding(4);
        this->groupBox1->Name = L"groupBox1";
        this->groupBox1->Padding =
System::Windows::Forms::Padding(4);
        this->groupBox1->Size = System::Drawing::Size(348,
53);

        this->groupBox1->TabIndex = 9;
        this->groupBox1->TabStop = false;
        this->groupBox1->Text = L"Server endpoint";
        //
        // statusStrip1
        //
        this->statusStrip1->ImageScalingSize =
System::Drawing::Size(20, 20);
        this->statusStrip1->Items->AddRange(gcnew
cli::array< System::Windows::Forms::ToolStripItem^ >(3) {
            this->toolStripStatusLabel1,
            this->toolStripStatusLabel2, this-
>toolStripStatusLabel3
        });
        this->statusStrip1->Location =
System::Drawing::Point(0, 123);
        this->statusStrip1->Name = L"statusStrip1";
        this->statusStrip1->Padding =
System::Windows::Forms::Padding(1, 0, 19, 0);
        this->statusStrip1->Size =
System::Drawing::Size(380, 22);
        this->statusStrip1->SizingGrip = false;
        this->statusStrip1->TabIndex = 10;
        this->statusStrip1->Text = L"statusStrip1";
        //
        // toolStripStatusLabel1
        //
        this->toolStripStatusLabel1->Name =
L"toolStripStatusLabel1";
        this->toolStripStatusLabel1->Size =
System::Drawing::Size(0, 17);
        //
        // toolStripStatusLabel2
        //
        this->toolStripStatusLabel2->Name =
L"toolStripStatusLabel2";
        this->toolStripStatusLabel2->Size =
System::Drawing::Size(0, 17);
        //
        // toolStripStatusLabel3
        //
        this->toolStripStatusLabel3->Name =
L"toolStripStatusLabel3";

```

```

        this->toolStripStatusLabel3->Size =
System::Drawing::Size(0, 17);
        //
        // button2
        //
        this->button2->Enabled = false;
        this->button2->Location =
System::Drawing::Point(273, 91);
        this->button2->Margin =
System::Windows::Forms::Padding(4);
        this->button2->Name = L"button2";
        this->button2->Size = System::Drawing::Size(94,
28);

        this->button2->TabIndex = 11;
        this->button2->Text = L"Disconnect";
        this->button2->UseVisualStyleBackColor = true;
        this->button2->Click += gcnew
System::EventHandler(this, &MyForm::button2_Click);
        //
        // label4
        //
        this->label4->AutoSize = true;
        this->label4->Location =
System::Drawing::Point(154, 94);
        this->label4->Name = L"label4";
        this->label4->Size = System::Drawing::Size(20,
17);

        this->label4->TabIndex = 12;
        this->label4->Text = L"%";
        //
        // MyForm
        //
        this->AutoScaleDimensions =
System::Drawing::SizeF(8, 16);
        this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
        this->ClientSize = System::Drawing::Size(380,
145);

        this->Controls->Add(this->label4);
        this->Controls->Add(this->button2);
        this->Controls->Add(this->statusStrip1);
        this->Controls->Add(this->groupBox1);
        this->Controls->Add(this->label3);
        this->Controls->Add(this->textBox3);
        this->Controls->Add(this->button1);
        this->FormBorderStyle =
System::Windows::Forms::FormBorderStyle::FixedSingle;
        this->Margin = System::Windows::Forms::Padding(4);
        this->Name = L"MyForm";
        this->Text = L"Client";

```

```

        this->FormClosing += gcnew
System::Windows::Forms::FormClosingEventHandler(this,
&MyForm::MyForm_FormClosing);
        this->groupBox1->ResumeLayout(false);
        this->groupBox1->PerformLayout();
        this->statusStrip1->ResumeLayout(false);
        this->statusStrip1->PerformLayout();
        this->ResumeLayout(false);
        this->PerformLayout();

    }

#pragma endregion
    private: System::Void button1_Click(System::Object^ sender,
System::EventArgs^ e) {
        if (textBox1->Text != "" && textBox2->Text != "" &&
textBox3->Text != "")
        {
            button1->Enabled = false;
            button2->Enabled = true;
            msclr::interop::marshal_context context;
            std::string IP =
context.marshal_as<std::string>(textBox1->Text);
            int Port = Convert::ToInt32(textBox2->Text);
            int Load = Convert::ToInt32(textBox3->Text);
            Client->connection(IP, Port, Load);
        }
        else
        {
            MessageBox::Show("Please enter data", "Warning");
        }
    }

    private: System::Void button2_Click(System::Object^ sender,
System::EventArgs^ e) {
        Client->disconnection();
        button1->Enabled = true;
        button2->Enabled = false;
    }
    private: System::Void MyForm_FormClosing(System::Object^
sender, System::Windows::Forms::FormClosingEventArgs^ e) {
        Client->FormClose();
    }
    private: System::Void textBox1_KeyPress(System::Object^ sender,
System::Windows::Forms::KeyPressEventArgs^ e) {
        char number = e->KeyChar;
        if ((e->KeyChar <= 47 || e->KeyChar >= 58) && number != 8 &&
number != 46)
        {
            e->Handled = true;
        }
    }

```

```

}
private: System::Void textBox2_KeyPress(System::Object^ sender,
System::Windows::Forms::KeyPressEventArgs^ e) {
    char number = e->KeyChar;
    if ((e->KeyChar <= 47 || e->KeyChar >= 58) && number != 8)
    {
        e->Handled = true;
    }
}
private: System::Void textBox3_KeyPress(System::Object^ sender,
System::Windows::Forms::KeyPressEventArgs^ e) {
    char number = e->KeyChar;
    if ((e->KeyChar <= 47 || e->KeyChar >= 58) && number != 8)
    {
        e->Handled = true;
    }
}
};
}

```

Приложение A.2

Файл «CClient.cpp»

```
#include "CClient.h"
#include "MyForm.h"

using namespace Client;
SOCKET Connection;
int cpus = 0;
HANDLE *workers;
HANDLE hTreadCon;
bool endpoc = false;
CRITICAL_SECTION critsect;
double *sum;

/// Структура данных нити
typedef struct treadcontext
{
    int id;          ///< номер нити
    DOUBLE low;      ///< начало отрезка
    DOUBLE high;     ///< конец отрезка
    DOUBLE step;
    double sleep;
}tcon, *ptcon;

double f(double x) {
    return exp(-x);
}

DWORD WINAPI countThreadProc(CONST LPVOID lpParam)
{
    ptcon ctx = (ptcon)lpParam;
    DOUBLE starting = ctx->low;
    DOUBLE integral = 0;
    int start = 0, stop = 0, workTime = 0, rest = 0, count = 0;

    while (starting < ctx->high)
    {
        start = clock();
        while (count < 150000 && starting < ctx->high)
        {
            DOUBLE a = f(starting + ctx->step)*ctx->step;
            integral += a;
            starting += ctx->step;
            count++;
        }
        stop = clock();
        workTime = stop - start;
        count = 0;
    }
}
```

```

        rest = workTime * (100 / ctx->sleep) - workTime;
        if (rest >= 0) {
            Sleep(rest);
        }
    }

    sum[ctx->id] = integral;
    ExitThread(0);
}

DWORD WINAPI workerMain(CONST LPVOID lpParam)
{
    MyForm^ f =
safe_cast<MyForm^>(Application::OpenForms[L"MyForm"]);
    double recvBuff[4];
    int slp = *static_cast<const int*>(lpParam);
    while (true) {
        if (endpoc == true)
        {
            ExitThread(0);
        }
        recv(Connection, (char *)recvBuff, sizeof(recvBuff),
NULL);
        double lo = recvBuff[0];
        double hi = recvBuff[1];
        double step = recvBuff[2];
        double status = recvBuff[3];
        f->toolStripStatusLabel2->Text = "Lo: " + lo.ToString();
        f->toolStripStatusLabel3->Text = "Hi: " +
hi.ToString();

        if (status != 0)
        {
            closesocket(Connection);
            f->UpdateStatus(status);
            f->ConnectButtonEnable();
            ExitThread(0);
        }
        else
        {
            //Контекст нити-задачи
            ptcon ctx = new treadcontext[cpus];
            for (int i = 0; i < cpus; i++)
            {
                ctx[i].sleep = slp;
                ctx[i].low = (hi - lo) / cpus * i + lo;
                ctx[i].high = hi - (hi - lo) / cpus * (cpus -
i - 1);

                ctx[i].step = step;
                ctx[i].id = i;

```

```

        workers[i] = CreateThread(NULL, 0,
&countThreadProc, &ctx[i], 0, NULL);
    }

    WaitForMultipleObjects(cpus, workers, TRUE,
INFINITE);

    DOUBLE result = 0;
    for (int i = 0; i < cpus; i++)
    {
        result += sum[i];
    }

    cout << result << endl;

    double sndBuff[2];
    sndBuff[0] = result;
    sndBuff[1] = 0;
    send(Connection, (char *)sndBuff, sizeof(sndBuff),
NULL);
    }
}

void CClient::connection(string IP, int Port, int Load) {
    InitializeCriticalSection(&critsect);
    endpoc = false;
    SYSTEM_INFO siSysInfo;
    GetSystemInfo(&siSysInfo);
    cpus = siSysInfo.dwNumberOfProcessors;

    //МНОГОПОТОЧНЫЙ КОНТЕКСТ
    workers = new HANDLE[cpus];
    sum = new DOUBLE[cpus];

    WSADATA wsaData;
    WORD DLLVersion = MAKEWORD(2, 1);
    if (WSAStartup(DLLVersion, &wsaData) != 0) {
        cout << "Error" << endl;
        exit(1);
    }
    const char *IPc = IP.c_str();
    SOCKADDR_IN addr;
    int sizeofaddr = sizeof(addr);
    addr.sin_addr.s_addr = inet_addr(IPc);
    addr.sin_port = htons(Port);
    addr.sin_family = AF_INET;

    // TCP conn
    Connection = socket(AF_INET, SOCK_STREAM, NULL);

```

```

        if (connect(Connection, (SOCKADDR*)&addr, sizeof(addr)) !=
0) {
            MyForm^ f =
safe_cast<MyForm^>(Application::OpenForms[L"MyForm"]);
            f->toolStripStatusLabel1->Text = L"Failed to TCP
connect";
            f->ConnectButtonEnable();
        }
        else
        {
            MyForm^ f =
safe_cast<MyForm^>(Application::OpenForms[L"MyForm"]);
            f->toolStripStatusLabel1->Text = L"TCP conection
established";
            int* id = new int(Load);
            hTreadCon = CreateThread(NULL, 0, &workerMain, id, 0,
NULL);
        }
    }

void CClient::disconnection() {

    double sendBuffer[2];
    sendBuffer[0] = 0;
    sendBuffer[1] = 1;
    // send res
    send(Connection, (char *)sendBuffer, sizeof(sendBuffer),
NULL);
    closesocket(Connection);
    endpoc = true;

    for (int i = 0; i < cpus; i++)
    {
        TerminateThread(workers[i], 0);
    }

    MyForm^ f =
safe_cast<MyForm^>(Application::OpenForms[L"MyForm"]);
    f->toolStripStatusLabel1->Text = L"TCP connection close";

}

void CClient::FormClose() {
    double sendBuffer[2];
    sendBuffer[0] = 0;
    sendBuffer[1] = 1;
    // TCP close send
    send(Connection, (char *)sendBuffer, sizeof(sendBuffer),
NULL);
    closesocket(Connection);
    ExitProcess(0);
}

```

}

Приложение В

Листинг серверной части приложения

Приложение В.1

Файл «MyForm.h»

```
#pragma once
#include "SServer.h"

namespace Server {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;

    /// <summary>
    /// Сводка для MyForm
    /// </summary>
    public ref class MyForm : public
System::Windows::Forms::Form
    {
    private:
        SServer* SServer;

    public:

        void add(SOCKET str)
        {
            listBox1->Items->Add(str);
        }

        void remove(SOCKET str)
        {
            listBox1->Items->Remove(str);
        }

        void result(double res)
        {
            textBox3->Text = res.ToString();
        }

        void removeItems()
        {

```

```

        listBox1->Items->Clear();
    }

    void buttonenable()
    {
        button1->Enabled = true;
    }

    void ClientConnection(SOCKET str)
    {
        if (listBox1->InvokeRequired)
            listBox1->Invoke(gcnew Action<SOCKET>(this,
&MyForm::add), str);
        else
            listBox1->Items->Add(str);
    }

    void ClientDisconnection(SOCKET str)
    {
        if (listBox1->InvokeRequired)
            listBox1->Invoke(gcnew Action<SOCKET>(this,
&MyForm::remove), str);
        else
            listBox1->Items->Remove(str);
    }

    void Result(double res)
    {
        if (textBox3->InvokeRequired)
            textBox3->Invoke(gcnew Action<double>(this,
&MyForm::result), res);
        else
            textBox3->Text = res.ToString();

        if (listBox1->InvokeRequired)
            listBox1->Invoke(gcnew Action(this,
&MyForm::removeItems));
        else
            listBox1->Items->Clear();

        if (button1->InvokeRequired)
            button1->Invoke(gcnew Action(this,
&MyForm::buttonenable));
        else
            button1->Enabled = true;
        SServer->Finish();
    }

    void Bar(int state)
    {

```

```

        progressBar1->Value = state;
    }

    void Time(double time)
    {
        textBox4->Text = time.ToString();
    }

    void progress(double proc) {
        textBox8->Text = proc.ToString();
    }

    void UpdateBar(int state, int time)
    {
        double dState = state * 1.0;
        double progTime = ((25 - state) * (time)) / 1000;
        if (textBox4->InvokeRequired)
            Invoke(gcnew Action<double>(this,
&MyForm::Time), progTime);

        if (textBox8->InvokeRequired)
            Invoke(gcnew Action<double>(this,
&MyForm::progress), (dState / 25) * 100);
        else
            textBox8->Text = ((dState / 25) *
100).ToString();
        progressBar1->Value = state;
    }

    MyForm(void)
    {
        InitializeComponent();

        //
        //TODO: добавьте код конструктора
        //
    }

protected:
    /// <summary>
    /// Освободить все используемые ресурсы.
    /// </summary>
    ~MyForm()
    {
        if (components)
        {

```

```

        delete components;
    }
}

private: System::Windows::Forms::TextBox^  textBox1;
protected:
private: System::Windows::Forms::Label^  label1;
private: System::Windows::Forms::Label^  label2;
private: System::Windows::Forms::TextBox^  textBox2;
private: System::Windows::Forms::Label^  label3;
public: System::Windows::Forms::ListBox^  listBox1;
private: System::Windows::Forms::Button^  button1;
private: System::Windows::Forms::Label^  label4;
private: System::Windows::Forms::TextBox^  textBox3;
private: System::Windows::Forms::Label^  label5;
public: System::Windows::Forms::TextBox^  textBox4;
private: System::Windows::Forms::ProgressBar^  progressBar1;
private: System::Windows::Forms::Label^  label6;
private: System::Windows::Forms::TextBox^  textBox5;
private: System::Windows::Forms::TextBox^  textBox6;
private: System::Windows::Forms::Label^  label7;
private: System::Windows::Forms::TextBox^  textBox7;
private: System::Windows::Forms::Label^  label8;
private: System::Windows::Forms::GroupBox^  groupBox1;
private: System::Windows::Forms::GroupBox^  groupBox2;
private: System::Windows::Forms::GroupBox^  groupBox3;
private: System::Windows::Forms::Label^  label11;
private: System::Windows::Forms::Label^  label10;
public: System::Windows::Forms::TextBox^  textBox8;
private:
private: System::Windows::Forms::Label^  label9;
public:

private:
    /// <summary>
    /// Обязательная переменная конструктора.
    /// </summary>
    System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
    /// <summary>
    /// Требуемый метод для поддержки конструктора — не
изменяйте
    /// содержимое этого метода с помощью редактора кода.
    /// </summary>
    void InitializeComponent(void)
    {
        this->textBox1 = (gcnew
System::Windows::Forms::TextBox());
        this->label1 = (gcnew
System::Windows::Forms::Label());

```

```

        this->label2 = (gcnew
System::Windows::Forms::Label());
        this->textBox2 = (gcnew
System::Windows::Forms::TextBox());
        this->label3 = (gcnew
System::Windows::Forms::Label());
        this->listBox1 = (gcnew
System::Windows::Forms::ListBox());
        this->button1 = (gcnew
System::Windows::Forms::Button());
        this->label4 = (gcnew
System::Windows::Forms::Label());
        this->textBox3 = (gcnew
System::Windows::Forms::TextBox());
        this->label5 = (gcnew
System::Windows::Forms::Label());
        this->textBox4 = (gcnew
System::Windows::Forms::TextBox());
        this->progressBar1 = (gcnew
System::Windows::Forms::ProgressBar());
        this->label6 = (gcnew
System::Windows::Forms::Label());
        this->textBox5 = (gcnew
System::Windows::Forms::TextBox());
        this->textBox6 = (gcnew
System::Windows::Forms::TextBox());
        this->label7 = (gcnew
System::Windows::Forms::Label());
        this->textBox7 = (gcnew
System::Windows::Forms::TextBox());
        this->label8 = (gcnew
System::Windows::Forms::Label());
        this->groupBox1 = (gcnew
System::Windows::Forms::GroupBox());
        this->groupBox2 = (gcnew
System::Windows::Forms::GroupBox());
        this->groupBox3 = (gcnew
System::Windows::Forms::GroupBox());
        this->label11 = (gcnew
System::Windows::Forms::Label());
        this->label10 = (gcnew
System::Windows::Forms::Label());
        this->textBox8 = (gcnew
System::Windows::Forms::TextBox());
        this->label9 = (gcnew
System::Windows::Forms::Label());
        this->groupBox1->SuspendLayout();
        this->groupBox2->SuspendLayout();
        this->groupBox3->SuspendLayout();
        this->SuspendLayout();
//

```

```

        // textBox1
        //
        this->textBox1->Location =
System::Drawing::Point(141, 28);
        this->textBox1->Margin =
System::Windows::Forms::Padding(4);
        this->textBox1->Name = L"textBox1";
        this->textBox1->ReadOnly = true;
        this->textBox1->Size = System::Drawing::Size(132,
22);

        this->textBox1->TabIndex = 0;
        this->textBox1->Text = L"0.5";
        //
        // label1
        //
        this->label1->AutoSize = true;
        this->label1->Location = System::Drawing::Point(8,
32);

        this->label1->Margin =
System::Windows::Forms::Padding(4, 0, 4, 0);
        this->label1->Name = L"label1";
        this->label1->Size = System::Drawing::Size(94,
17);

        this->label1->TabIndex = 1;
        this->label1->Text = L"Lower bound:";
        //
        // label2
        //
        this->label2->AutoSize = true;
        this->label2->Location = System::Drawing::Point(8,
64);

        this->label2->Margin =
System::Windows::Forms::Padding(4, 0, 4, 0);
        this->label2->Name = L"label2";
        this->label2->Size = System::Drawing::Size(98,
17);

        this->label2->TabIndex = 3;
        this->label2->Text = L"Higher bound:";
        //
        // textBox2
        //
        this->textBox2->Location =
System::Drawing::Point(141, 60);
        this->textBox2->Margin =
System::Windows::Forms::Padding(4);
        this->textBox2->Name = L"textBox2";
        this->textBox2->ReadOnly = true;
        this->textBox2->Size = System::Drawing::Size(132,
22);

        this->textBox2->TabIndex = 2;
        this->textBox2->Text = L"1";

```

```

        //
        // label3
        //
        this->label3->AutoSize = true;
        this->label3->Location =
System::Drawing::Point(12, 171);
        this->label3->Margin =
System::Windows::Forms::Padding(4, 0, 4, 0);
        this->label3->Name = L"label3";
        this->label3->Size = System::Drawing::Size(86,
17);

        this->label3->TabIndex = 4;
        this->label3->Text = L"Connections";
        //
        // listBox1
        //
        this->listBox1->FormattingEnabled = true;
        this->listBox1->ItemHeight = 16;
        this->listBox1->Location =
System::Drawing::Point(16, 201);
        this->listBox1->Margin =
System::Windows::Forms::Padding(4);
        this->listBox1->Name = L"listBox1";
        this->listBox1->Size = System::Drawing::Size(227,
116);

        this->listBox1->TabIndex = 5;
        //
        // button1
        //
        this->button1->Location =
System::Drawing::Point(132, 122);
        this->button1->Margin =
System::Windows::Forms::Padding(4);
        this->button1->Name = L"button1";
        this->button1->Size = System::Drawing::Size(100,
28);

        this->button1->TabIndex = 6;
        this->button1->Text = L"Начать";
        this->button1->UseVisualStyleBackColor = true;
        this->button1->Click += gcnew
System::EventHandler(this, &MyForm::button1_Click);
        //
        // label4
        //
        this->label4->AutoSize = true;
        this->label4->Location =
System::Drawing::Point(17, 63);
        this->label4->Margin =
System::Windows::Forms::Padding(4, 0, 4, 0);
        this->label4->Name = L"label4";

```

```

        this->label4->Size = System::Drawing::Size(80,
17);
        this->label4->TabIndex = 10;
        this->label4->Text = L"Результат:";
        //
        // textBox3
        //
        this->textBox3->Enabled = false;
        this->textBox3->Location =
System::Drawing::Point(100, 63);
        this->textBox3->Margin =
System::Windows::Forms::Padding(4);
        this->textBox3->MaxLength = 7;
        this->textBox3->Name = L"textBox3";
        this->textBox3->ReadOnly = true;
        this->textBox3->Size = System::Drawing::Size(132,
22);
        this->textBox3->TabIndex = 9;
        //
        // label5
        //
        this->label5->AutoSize = true;
        this->label5->Location =
System::Drawing::Point(17, 30);
        this->label5->Margin =
System::Windows::Forms::Padding(4, 0, 4, 0);
        this->label5->Name = L"label5";
        this->label5->Size = System::Drawing::Size(79,
17);
        this->label5->TabIndex = 8;
        this->label5->Text = L"Remaining:";
        this->label5->Click += gcnew
System::EventHandler(this, &MyForm::label5_Click);
        //
        // textBox4
        //
        this->textBox4->Enabled = false;
        this->textBox4->Location =
System::Drawing::Point(100, 32);
        this->textBox4->Margin =
System::Windows::Forms::Padding(4);
        this->textBox4->Name = L"textBox4";
        this->textBox4->ReadOnly = true;
        this->textBox4->Size = System::Drawing::Size(132,
22);
        this->textBox4->TabIndex = 7;
        //
        // progressBar1
        //
        this->progressBar1->Location =
System::Drawing::Point(23, 155);

```

```

        this->progressBar1->Margin =
System::Windows::Forms::Padding(4);
        this->progressBar1->Maximum = 25;
        this->progressBar1->Name = L"progressBar1";
        this->progressBar1->Size =
System::Drawing::Size(234, 10);
        this->progressBar1->TabIndex = 11;
        this->progressBar1->Visible = false;
        //
        // label6
        //
        this->label6->AutoSize = true;
        this->label6->Location = System::Drawing::Point(8,
96);
        this->label6->Margin =
System::Windows::Forms::Padding(4, 0, 4, 0);
        this->label6->Name = L"label6";
        this->label6->Size = System::Drawing::Size(41,
17);
        this->label6->TabIndex = 13;
        this->label6->Text = L"Step:";
        //
        // textBox5
        //
        this->textBox5->Location =
System::Drawing::Point(141, 92);
        this->textBox5->Margin =
System::Windows::Forms::Padding(4);
        this->textBox5->Name = L"textBox5";
        this->textBox5->Size = System::Drawing::Size(132,
22);
        this->textBox5->TabIndex = 12;
        this->textBox5->Text = L"0.000001";
        this->textBox5->KeyPress += gcnew
System::Windows::Forms::KeyPressEventHandler(this,
&MyForm::textBox5_KeyPress);
        //
        // textBox6
        //
        this->textBox6->Enabled = false;
        this->textBox6->Location =
System::Drawing::Point(85, 27);
        this->textBox6->Margin =
System::Windows::Forms::Padding(4);
        this->textBox6->Name = L"textBox6";
        this->textBox6->ReadOnly = true;
        this->textBox6->Size = System::Drawing::Size(132,
22);
        this->textBox6->TabIndex = 14;
        //
        // label7

```

```

        //
        this->label7->AutoSize = true;
        this->label7->Location =
System::Drawing::Point(25, 63);
        this->label7->Margin =
System::Windows::Forms::Padding(4, 0, 4, 0);
        this->label7->Name = L"label7";
        this->label7->Size = System::Drawing::Size(38,
17);

        this->label7->TabIndex = 16;
        this->label7->Text = L"Port:";
        //
        // textBox7
        //
        this->textBox7->Enabled = false;
        this->textBox7->Location =
System::Drawing::Point(85, 59);
        this->textBox7->Margin =
System::Windows::Forms::Padding(4);
        this->textBox7->Name = L"textBox7";
        this->textBox7->ReadOnly = true;
        this->textBox7->Size = System::Drawing::Size(132,
22);

        this->textBox7->TabIndex = 15;
        //
        // label8
        //
        this->label8->AutoSize = true;
        this->label8->Location =
System::Drawing::Point(25, 31);
        this->label8->Margin =
System::Windows::Forms::Padding(4, 0, 4, 0);
        this->label8->Name = L"label8";
        this->label8->Size = System::Drawing::Size(24,
17);

        this->label8->TabIndex = 17;
        this->label8->Text = L"IP:";
        //
        // groupBox1
        //
        this->groupBox1->Controls->Add(this->label1);
        this->groupBox1->Controls->Add(this->textBox1);
        this->groupBox1->Controls->Add(this->textBox2);
        this->groupBox1->Controls->Add(this->label2);
        this->groupBox1->Controls->Add(this->textBox5);
        this->groupBox1->Controls->Add(this->label6);
        this->groupBox1->Location =
System::Drawing::Point(16, 15);
        this->groupBox1->Margin =
System::Windows::Forms::Padding(4);
        this->groupBox1->Name = L"groupBox1";

```

```

        this->groupBox1->Padding =
System::Windows::Forms::Padding(4);
        this->groupBox1->Size = System::Drawing::Size(296,
133);

        this->groupBox1->TabIndex = 18;
        this->groupBox1->TabStop = false;
        this->groupBox1->Text = L"Task Preconditions";
        //
        // groupBox2
        //
        this->groupBox2->Controls->Add(this->textBox6);
        this->groupBox2->Controls->Add(this->textBox7);
        this->groupBox2->Controls->Add(this->label8);
        this->groupBox2->Controls->Add(this->label7);
        this->groupBox2->Location =
System::Drawing::Point(336, 15);
        this->groupBox2->Margin =
System::Windows::Forms::Padding(4);
        this->groupBox2->Name = L"groupBox2";
        this->groupBox2->Padding =
System::Windows::Forms::Padding(4);
        this->groupBox2->Size = System::Drawing::Size(252,
101);

        this->groupBox2->TabIndex = 19;
        this->groupBox2->TabStop = false;
        this->groupBox2->Text = L"Server Data
(Predefined)";
        //
        // groupBox3
        //
        this->groupBox3->Controls->Add(this->label11);
        this->groupBox3->Controls->Add(this->label10);
        this->groupBox3->Controls->Add(this->textBox8);
        this->groupBox3->Controls->Add(this->label9);
        this->groupBox3->Controls->Add(this->textBox3);
        this->groupBox3->Controls->Add(this->button1);
        this->groupBox3->Controls->Add(this->textBox4);
        this->groupBox3->Controls->Add(this->label5);
        this->groupBox3->Controls->Add(this->label4);
        this->groupBox3->Controls->Add(this->
>progressBar1);
        this->groupBox3->Location =
System::Drawing::Point(291, 171);
        this->groupBox3->Margin =
System::Windows::Forms::Padding(4);
        this->groupBox3->Name = L"groupBox3";
        this->groupBox3->Padding =
System::Windows::Forms::Padding(4);
        this->groupBox3->Size = System::Drawing::Size(262,
156);

        this->groupBox3->TabIndex = 20;

```

```

        this->groupBox3->TabStop = false;
        this->groupBox3->Text = L"Вычисление";
        //
        // label11
        //
        this->label11->AutoSize = true;
        this->label11->Location =
System::Drawing::Point(239, 35);
        this->label11->Name = L"label11";
        this->label11->Size = System::Drawing::Size(15,
17);

        this->label11->TabIndex = 15;
        this->label11->Text = L"c";
        //
        // label10
        //
        this->label10->AutoSize = true;
        this->label10->Location =
System::Drawing::Point(237, 95);
        this->label10->Name = L"label10";
        this->label10->Size = System::Drawing::Size(20,
17);

        this->label10->TabIndex = 14;
        this->label10->Text = L"%";
        //
        // textBox8
        //
        this->textBox8->Enabled = false;
        this->textBox8->Location =
System::Drawing::Point(100, 92);
        this->textBox8->Margin =
System::Windows::Forms::Padding(4);
        this->textBox8->Name = L"textBox8";
        this->textBox8->ReadOnly = true;
        this->textBox8->Size = System::Drawing::Size(132,
22);

        this->textBox8->TabIndex = 13;
        //
        // label9
        //
        this->label9->AutoSize = true;
        this->label9->Location =
System::Drawing::Point(17, 95);
        this->label9->Name = L"label9";
        this->label9->Size = System::Drawing::Size(69,
17);

        this->label9->TabIndex = 12;
        this->label9->Text = L"Progress:";
        //
        // MyForm
        //

```

```

        this->AutoScaleDimensions =
System::Drawing::SizeF(8, 16);
        this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
        this->ClientSize = System::Drawing::Size(601,
357);

        this->Controls->Add(this->groupBox3);
        this->Controls->Add(this->groupBox2);
        this->Controls->Add(this->groupBox1);
        this->Controls->Add(this->listBox1);
        this->Controls->Add(this->label3);
        this->FormBorderStyle =
System::Windows::Forms::FormBorderStyle::FixedSingle;
        this->Margin = System::Windows::Forms::Padding(4);
        this->Name = L"MyForm";
        this->Text = L"ServerIntegralFormApplication";
        this->FormClosing += gcnew
System::Windows::Forms::FormClosingEventHandler(this,
&MyForm::MyForm_FormClosing);
        this->groupBox1->ResumeLayout(false);
        this->groupBox1->PerformLayout();
        this->groupBox2->ResumeLayout(false);
        this->groupBox2->PerformLayout();
        this->groupBox3->ResumeLayout(false);
        this->groupBox3->PerformLayout();
        this->ResumeLayout(false);
        this->PerformLayout();

    }

#pragma endregion
    private: System::Void button1_Click(System::Object^ sender,
System::EventArgs^ e) {

        if (textBox5->Text != "")
        {
            button1->Enabled = false;
            double stepst = double::Parse(textBox5->Text-
>Replace(".", ","));

            char* a = SServer->Start(stepst);

            System::String^ str;
            for (int i = 0; a[i] != '\0'; i++)
            {
                str += wchar_t(a[i]);
            }

            textBox6->Text = str;
            textBox7->Text = "9000";
            progressBar1->Value = 0;
            textBox3->Text = "";

```

```

        }
        else
        {
            MessageBox::Show("Please enter data", "Warning");
        }
    }

    private: System::Void MyForm_FormClosing(System::Object^
sender, System::Windows::Forms::FormClosingEventArgs^ e) {
        SServer->Stop();
    }
    private: System::Void textBox5_KeyPress(System::Object^
sender, System::Windows::Forms::KeyPressEventArgs^ e) {
        char number = e->KeyChar;
        if ((e->KeyChar <= 47 || e->KeyChar >= 58) && number !=
8 && number != 44 && number != 46)
        {
            e->Handled = true;
        }
    }
    private: System::Void label5_Click(System::Object^ sender,
System::EventArgs^ e) {
    }
};
}

```

Приложение В.2

Файл «SServer.cpp»

```
#include "SServer.h"

SOCKET sListen;
SOCKADDR_IN addr;
bool work = false;
SOCKET newConnection;

void ClientConnect() {
    int sizeofaddr = sizeof(addr);

    while (work == false) {
        newConnection = accept(sListen, (SOCKADDR*)&addr,
&sizeofaddr);
        if (newConnection == SOCKET_ERROR) {
            ExitThread(0);
        }
        if (newConnection != 0 ) {
            Connect(newConnection);
        }
    }
}

char* SServer::Start(double step) {
    work = false;
    WSADATA wsaData;
    WORD DLLVersion = MAKEWORD(2, 1);
    if (WSAStartup(DLLVersion, &wsaData) != 0) {
        exit(1);
    }
    hostent* localHost;
    char* localIP;

    localHost = gethostbyname("");
    localIP = inet_ntoa(*(struct in_addr *)*localHost->h_addr_list);

    addr.sin_addr.s_addr = inet_addr(localIP);
    addr.sin_port = htons(9000);
    addr.sin_family = AF_INET;

    sListen = socket(AF_INET, SOCK_STREAM, NULL);
    bind(sListen, (SOCKADDR*)&addr, sizeof(addr));
    listen(sListen, SOMAXCONN);
}
```

```

        CreateThread(NULL, NULL,
(LPTHREAD_START_ROUTINE)ClientConnect, NULL, NULL, NULL);
        counting(step);
        return localIP;
    }

void SServer::Stop()
{
    StopServer();
}

void SServer::Finish()
{
    closesocket(sListen);
    work = true;
}

```

Приложение В.4

Файл «FormUpdate.cpp»

```
#include "FormUpdate.h"
#include "MyForm.h"

using namespace Server;
void ClientConnection(SOCKET soc) {
    MyForm^ f =
safe_cast<MyForm^>(Application::OpenForms[L"MyForm"]);
    f->ClientConnection(soc);
}

void ClientDisconnection(SOCKET soc) {
    MyForm^ f =
safe_cast<MyForm^>(Application::OpenForms[L"MyForm"]);
    f->ClientDisconnection(soc);
}

void Result(double res) {
    MyForm^ f =
safe_cast<MyForm^>(Application::OpenForms[L"MyForm"]);
    f->Result(res);
}

void ProgressBar(int state,int time)
{
    MyForm^ f =
safe_cast<MyForm^>(Application::OpenForms[L"MyForm"]);
    f->UpdateBar(state,time);
}
```

Приложение В.5

Файл «Client.cpp»

```
#pragma once

#include "Client.h"
#include "Integral.h"
#include "FormUpdate.h"

public delegate void InvokeDelegate();

CRITICAL_SECTION critsect;
int counter = 0;
double sum = 0;
bool flag = false;
bool stopwork = false;
int check = 0;
int worktime = 0;

void ClientHandler(SOCKET soc) {
    ClientConnection(soc);

    double recBuffer[2];
    SOCKET socket = soc;
    double sendBuffer[4];
    int start = 0, stop = 0;

    while (true) {
        for (int i = 0; i < count; i++)
        {
            start = clock();
            EnterCriticalSection(&critsect);
            bool status = integral[i].complete;
            bool workstatus = stopwork;
            LeaveCriticalSection(&critsect);
            if (workstatus == false)
            {

                if (flag == false)
                {
                    if (status == false)
                    {
                        EnterCriticalSection(&critsect);
                        integral[i].complete = true;
                        sendBuffer[0] = integral[i].begin;
                        sendBuffer[1] = integral[i].end;
                        sendBuffer[2] = integral[i].step;
                        sendBuffer[3] = 0;
                    }
                }
            }
        }
    }
}
```

```

        LeaveCriticalSection(&critsect);
        send(socket, (char *)sendBuffer,
sizeof(sendBuffer), NULL);
        if (recv(socket, (char *)recBuffer,
sizeof(recBuffer), NULL) != SOCKET_ERROR) {
            stop = clock();
            double state = recBuffer[1];
            if (state == 0)
            {
                double res =
recBuffer[0];

                EnterCriticalSection(&critsect);

                sum += res;
                integral[i].complete =
true;

                check++;
                worktime = stop - start;
                ProgressBar(check,
worktime);

                if (check == count) {
                    flag = true;
                    Result(sum);

                LeaveCriticalSection(&critsect);
                }

                LeaveCriticalSection(&critsect);
            }
            else
            {
                ClientDisconnection(soc);

                EnterCriticalSection(&critsect);
                integral[i].complete =
false;

                LeaveCriticalSection(&critsect);
                ExitThread(0);
            }
        }
        else
        {
            ClientDisconnection(soc);

            EnterCriticalSection(&critsect);
            integral[i].complete = false;

            LeaveCriticalSection(&critsect);
            ExitThread(0);
        }
    }
}

```

```

        }
    }
    else
    {
        sendBuffer[0] = 0;
        sendBuffer[1] = 0;
        sendBuffer[2] = 0;
        sendBuffer[3] = 1;
        send(socket, (char *)sendBuffer,
sizeof(sendBuffer), NULL);
        ExitThread(0);
    }
}
else
{
    sendBuffer[0] = 0;
    sendBuffer[1] = 0;
    sendBuffer[2] = 0;
    sendBuffer[3] = 2;
    send(socket, (char *)sendBuffer,
sizeof(sendBuffer), NULL);
    ExitProcess(0);
}
}
}

void Connect(SOCKET newConnection) {
    CreateThread(NULL, NULL,
(LPTHREAD_START_ROUTINE)ClientHandler, (LPVOID) (newConnection),
NULL, NULL);
}

void counting(double step) {
    counter = 0;
    sum = 0;
    flag = false;
    stopwork = false;
    check = 0;
    InitializeCriticalSection(&critsect);
    countinteg(step);
}

void StopServer() {
    stopwork = true;
    Sleep(1000);
}

```

Приложение В.8

Файл «Integral.h»

```
#pragma once
#define count 25

struct Integral {
    double begin;
    double end;
    double step;
    bool complete;
};

Integral integral[count] = {};

void countinteg(double st) {

    double begin = 0.5;
    double end = 1.0;
    double step = st;
    double exambegin;
    double examend;
    // разбиение интеграла
    for (int i = 0; i < count; i++)
    {
        exambegin = (end - begin) / count * i + begin;
        examend = end - (end - begin) / count * (count - i -
1);
        integral[i].begin = exambegin;
        integral[i].end = examend;
        integral[i].step = step;
        integral[i].complete = false;
    }
}
```

Приложение С

Результаты работы программы

Приложение С.1 – Вид клиента в процессе вычисления

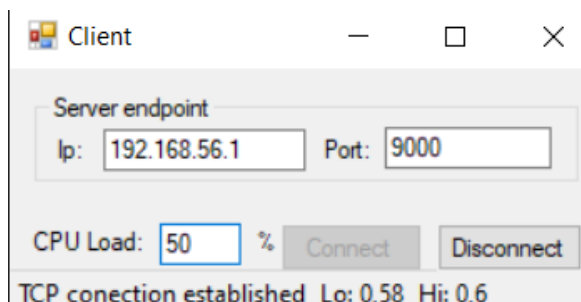


Рисунок 12 – Окно клиента.

Приложение С.2 – Вид сервера в процессе вычисления

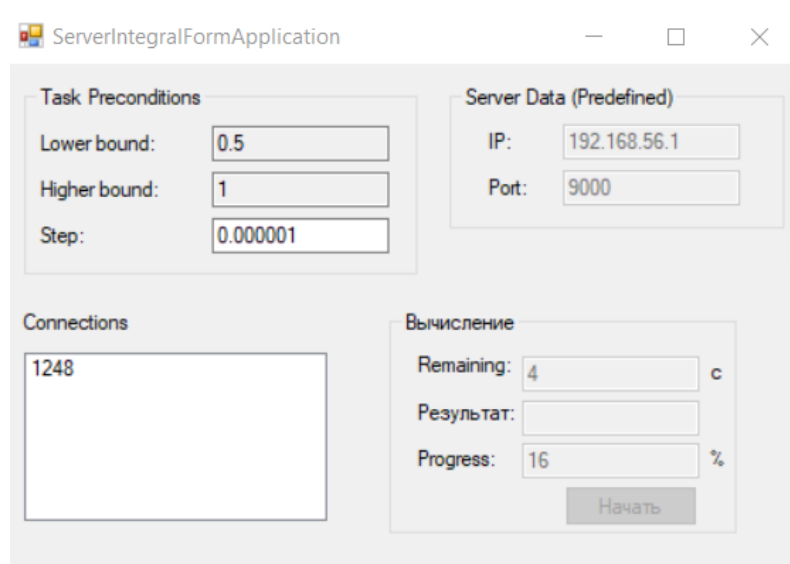


Рисунок 13 – Окно сервера.

Приложение С.3 – Вид сервера с результатом вычисления

The screenshot shows a window titled "ServerIntegralFormApplication". It contains four main sections:

- Task Preconditions:** Lower bound: 0.5, Higher bound: 1, Step: 0.000001.
- Server Data (Predefined):** IP: 192.168.56.1, Port: 9000.
- Connections:** An empty rectangular box.
- Вычисление (Calculation):** Remaining: 0 с, Результат: 0.23865109921558, Progress: 100 %, and a button labeled "Начать" (Start).

Рисунок 14 - Окно сервера с результатом.

Приложение С.4 – Эталонный результат вычисления

The screenshot displays the reference result for the calculation of the definite integral. It includes the following elements:

- ОПРЕДЕЛЁННЫЙ ИНТЕГРАЛ:** $\int_{0.5}^1 f(x) dx =$
- Exact result:** $\frac{1}{\sqrt{e}} - e^{-1}$
- В приближении:** 0.2386512185411911
- Buttons:** "Упростить" (Simplify) and a green checkmark icon.

Рисунок 15 – Результат вычисления.

Приложение С.5 – Завершение работы клиентом

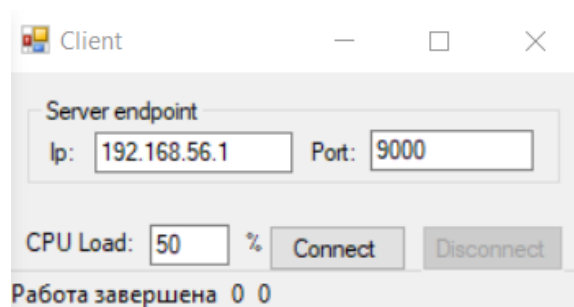


Рисунок 17 - Результат нагрузки на ЦП

Приложение С.6 – Расход ресурсов системы

Диспетчер задач

Файл Параметры Вид

Процессы	Производительность	Журнал приложений	Автозагрузка	Пользователи
Имя	Состояние	50% ЦП	29 Память	
Приложения (8)				
> Client.exe (32 бита)		24,8%	5,6 МБ	
> Client.exe (32 бита)		24,4%	8,1 МБ	

Рисунок 16 - Результат нагрузки на ЦП