Лабораторная работа №2 – часть 4. Написание собственного сетевого приложения

В результате этой лабораторной должны быть представлены два приложения: клиентское и серверное. Клиент должен будет обращаться к серверу, и передавать ему некоторые данные. При получении данных сервер должен высылать подтверждение клиенту подтверждения. После передачи данных сервер по команде клиента должен произвести некоторые вычисления и вернуть клиенту ответ.

Для передачи данных клиент должен посылать серверу текстовое сообщение в формате data <значение>. В ответ сервер должен посылать сообщение ok, если данные в порядке, и error, если в данных ошибка. Для получения результата клиент должен послать серверу команду get. Ответ сервера должен иметь формат result <значение>. После возвращения результата сервер должен обнулять полученные ранее данные и быть готов к новой порции значений для обработки.

Клиентское приложение должен иметь поля и кнопки для отправки значений и получения результатов.

Серверное приложение должно отображать лог взаимодействия с клиентом.

Варианты заданий

Общая постановка задания дана выше. Каждый студент должен выполнить задание лабораторной работы в соответствии со своим номером варианта:

№ варианта	ФИО	Формулировка задания лабораторной работы
1.	Азаров Алексей Александрович	Реализовать клиент-серверное приложение обмена сообщениями между сервером и несколькими клиентами.
2.	Акаев Магомед Захарович	Требуется разработать клиент-серверное приложение (то есть и клиент, и сервер), позволяющие работать по сети. Работу с сетью необходимо реализовать на нижнем уровне (создание сокета, прослушивание порта и т.п.) Сервер должен позволять задать номер порта, по которому он будет расположен; клиенты должны позволять задать номер порта и IP-адрес сервера (в случае запуска на одной машине - localhost или 127.0.0.1). Сервер может быть консольным приложением (без графического интерфейса).
3.	Гамаюнов Даниил Сергеевич	Реализовать клиент-серверное приложение «Текстовый калькулятор». Клиент содержит поле для ввода примера в текстовом виде (например, 4+(3*0.74-sin(0.1))), по кнопке эта строка отправляется на сервер. Сервер вычисляет результат и возвращает его клиенту, либо возвращает сообщение об ошибке (непарные скобки и т.п.).
4.	Ганзин Юрий	Реализовать клиент-серверное приложение обмена шифрованными сообщениями между несколькими клиентами (через сервер)
5.	Гринченко Станислав Александрович	Реализовать авторизацию клиента на сервере с подтверждением (авторизован/не авторизован). Максимальное количество попыток - 3
6.	Ковалев Михаил Александрович	Реализовать клиент-серверное приложение обмена растровыми изображениями
7.	Маруняк Евгений Владиславович	Реализовать клиент-серверное приложение обмена сообщениями между личным настольный компьютером (сервер) и личным ноутбуком (клиентом) (или любым другим компьютером/ноутбуком подключенным к сети Интернет).
8.	Мередов	Требуется разработать клиент-серверное приложение (то есть

	Мерген Байрамалыевич	и клиент, и сервер), позволяющие работать по сети. Работу с сетью необходимо реализовать на нижнем уровне (создание сокета, прослушивание порта и т.п.) Сервер должен позволять задать номер порта, по которому он будет расположен; клиенты должны позволять задать номер порта и IP-адрес сервера (в случае запуска на одной машине - localhost или 127.0.0.1). Сервер может быть консольным приложением (без графического интерфейса).
9.	Мукабенова Элеонора Вячеславовна	Реализовать авторизацию клиента на сервере с подтверждением (авторизован/не авторизован). Максимальное количество попыток - 3
10.	Пелихов Вячеслав Игоревич	Реализовать клиент-серверное приложение обмена растровыми изображениями
11.	Пронин Тимур Русланович	Реализовать клиент-серверное приложение обмена сообщениями между личным настольный компьютером (сервер) и личным ноутбуком (клиентом) (или любым другим компьютером/ноутбуком подключенным к сети Интернет).
12.	Самойлов Никита Сергеевич	Реализовать клиент-серверное приложение обмена сообщениями между сервером и несколькими клиентами.
13.	Супрунова Екатерина Петровна	Клиент записывает текст в файл и отправляет его серверу. сервер записывает данный текст в обратном порядке записывает в файл и отправляет клиенту. клиент получает файл считывает результат и выводит на экран.
14.	Трошечкин Даниил Сергеевич	Требуется разработать клиент-серверное приложение (то есть и клиент, и сервер), позволяющие работать по сети. Работу с сетью необходимо реализовать на нижнем уровне (создание сокета, прослушивание порта и т.п.) Сервер должен позволять задать номер порта, по которому он будет расположен; клиенты должны позволять задать номер порта и IP-адрес сервера (в случае запуска на одной машине - localhost или 127.0.0.1). Сервер может быть консольным приложением (без графического интерфейса).
15.	Ханнанов Родион Русланович	Реализовать клиент-серверное приложение обмена растровыми изображениями
16.	Щемелев Максим Александрович	Реализовать клиент-серверное приложение обмена сообщениями между личным настольный компьютером (сервер) и личным ноутбуком (клиентом) (или любым другим компьютером/ноутбуком подключенным к сети Интернет).
17.	Ядрин Вячеслав Владимирович	Реализовать авторизацию клиента на сервере с подтверждением (авторизован/не авторизован). Максимальное количество попыток - 3

Полезные конструкции языка C# для выполнения лабораторной работы. Синтаксис C#.

Ниже приведены конструкции и операторы, позволяющие удобно работать с разными типами.
Простые типы данных
int – целое число;
double – вещественное число;
string – строка символов;
char – один символ.

Списки
Для хранения списка однотипных данных удобно использовать списки:
List<type> myList; - объявить переменную-список, содержащих значения типа type;
myList = new List<type>(); - создать новый список и сохранить его в переменную myList;
myList.Add(a); - добавить в список myList значение a;
myList[i] - обратиться к значению списка myList с номером I (нумерация в списках начинается с нуля)
myList.Count – получить количество значений в списке;
myList.Clear(); - очистить список.

Для выполнения действий в цикле над всеми значения списка удобно применить следующую конструкцию:

```
foreach (type value in myList)
{
//некоторый полезные код
}
```

В этом случае код в фигурных скобках будет последовательно применен ко всем значениям в списке
myList. Текущее значение для данной итерации будет храниться в переменной value.

Пример:

```
List<int> values;
values = new List<int>(); values.Add(4); values.Add(5); values.Add(-1);
int sum = 0; //здесь будем хранить сумму
foreach (int value in values)
{
sum = sum + value;
}
//теперь sum = 8;
```

Преобразование числа в строку
Для этого можно применить метод ToString, которым обладают все переменные в C#.

```
string s1, s2; int a = 4; double b = 5.5;
s1 = a.ToString(); //s1=”4” s2 = b.ToString(); //s2=”5,5”
```

Преобразование строки в число
Для преобразования строки в целое число используется метод Int32.Parse(), а для вещественного числа
– Double.Parse():

```
string s1 = “4”;
string s2 = “5,5”;
int a = Int32.Parse(s1); //a=4
double b = Double.Parse(s2); //b=4.5
```

Кроме того, бывает нужно сделать проверку, действительно ли преобразуемая в число строка является числом. Для этого удобно использовать метод TryParse:

```
string s = “123”; int number; bool is_number;
is_number = Int32.TryParse(s, out number);
```

После выполнения этого кода, если строка s – это число, то is_number будет иметь значение true, а переменная number содержать распознанное число. Если переданная строка s – не число, то is_number будет равно false.

Аналогичный код может преобразовать строку в вещественное число.

```
string s = “123,5”; double number; bool is_number;
is_number = Double.TryParse(s, out number);
```

Действия над строками string s1 = “abc”;
string s2 = “xyz”;
string s3 = s1 + “ “ + s2; //слияние строк: s3=”abc xyz”;
string s4 = s3.SubString(2); //s4 = “c xyz” – выбрать из строки s3 все символы начиная с третьего; string s5 = s3.SubString(2,3); //s5 = “c x” – выбрать из строки s3 все 3 символа, начиная с третьего; string s6 = s1.ToUpper(); // заменить строчные буквы на прописные, s6=”ABC”;
string s7 = s6.ToLower(); // заменить прописные буквы на строчные, s7=”abc”;
bool b1 = s2.Contains(“yz”); //входит ли последовательность символов “yz” в строку s1; b1 = true

Математические операции
Math.Pow(x,y) – возвести x в степень y
Math.Sin, Math.Cos, Math.Tan – тригонометрические функции
Math.PI – число пи.
int a = (int)Math.Round(b); - a – результат округления b