***Сетевое программирование сокетов: понятие, серверные функции, клиентские функции сокета.***

***Дана матрица целых чисел размером АхВ. Вывести на экран все строки с четными номерами, у которых первый элемент больше последнего.***

**Сетевое программирование сокетов** на C++ — это способ объединить или связать два узла друг с другом через сеть, чтобы они могли легко обмениваться данными без потери информации.

**Серверные функции сокетов включают:**

listen(). Используется для прослушивания входящих соединений на сокете. Обычно применяется к серверному сокету, привязанному к определённому адресу и порту с помощью функции bind().

accept(). Принимает входящие соединения на сокете. Обычно используется на серверном сокете, который настроен прослушивать входящие соединения с помощью функции listen().

**Клиентские функции сокетов включают:**

connect(). Инициирует установление связи на сокете, используя дескриптор сокета и информацию из структуры, которая содержит адрес сервера и номер порта, на который нужно установить связь.

send() и recv(). Используются после установления связи для отправки и получения данных через ссылку.

**Сетевое программирование с использованием сокетов** — это фундаментальный способ создания приложений, способных обмениваться данными по сети, будь то локальная сеть или интернет. Сокеты предоставляют абстракцию, позволяющую приложениям взаимодействовать друг с другом через сеть, независимо от физической среды передачи данных.

Понятие сокета

**Сокет (socket)** — это конечная точка (endpoint) в двунаправленном канале связи между процессами. Это комбинация IP-адреса и номера порта, которая идентифицирует конкретный процесс, получающий или отправляющий данные по сети. Простыми словами, это как “дверь” в сетевое соединение. Сокеты могут быть представлены как абстракция сетевого соединения, позволяющая отправлять и получать данные, при этом детали сетевой инфраструктуры скрыты от программиста.

**Сокеты классифицируют по двум основным параметрам:**

Тип протокола:

TCP (Transmission Control Protocol): Обеспечивает надежное, потоковое, ориентированное на соединение взаимодействие. Данные передаются в порядке их отправки, с контролем ошибок и повторной передачей.

UDP (User Datagram Protocol): Обеспечивает ненадежное, дейтаграммное, без установления соединения взаимодействие. Данные передаются в виде отдельных пакетов, без гарантии порядка и доставки.

Тип адреса:

IPv4: Использует 32-битные адреса.

IPv6: Использует 128-битные адреса.

**Серверный сокет выполняет следующие основные операции:**

socket():

Создание сокета: Функция socket() создает новый сокет и возвращает его дескриптор (целочисленный идентификатор).

Аргументы:

domain (например, AF\_INET для IPv4, AF\_INET6 для IPv6): Семейство протоколов, которое будет использоваться.

type (например, SOCK\_STREAM для TCP, SOCK\_DGRAM для UDP): Тип сокета (например, потоковый или дейтаграммный).

protocol (обычно 0, автоматически выбирается протокол): Протокол, который будет использоваться.

bind():

Привязка сокета к адресу: Функция bind() связывает сокет с конкретным IP-адресом и номером порта. Это необходимо для того, чтобы клиенты могли подключаться к серверу.

Аргументы:

sockfd: Дескриптор сокета, полученный при вызове socket().

addr: Указатель на структуру с IP-адресом и номером порта (например, sockaddr\_in для IPv4).

addrlen: Размер структуры адреса.

listen():

Перевод сокета в режим прослушивания: Функция listen() переводит сокет в режим ожидания входящих подключений. Она определяет, сколько входящих подключений может быть поставлено в очередь до их принятия.

Аргументы:

sockfd: Дескриптор сокета.

backlog: Максимальное количество ожидающих подключений в очереди.

accept():

Принятие подключения: Функция accept() принимает входящее подключение от клиента и создает новый сокет для этого соединения. Возвращает дескриптор нового сокета, который будет использоваться для связи с конкретным клиентом.

Аргументы:

sockfd: Дескриптор сокета, который слушает входящие соединения.

addr: Указатель на структуру, в которую будет записан адрес клиента.

addrlen: Указатель на переменную, в которой будет записан размер структуры адреса клиента.

Блокирующая операция: accept() блокирует выполнение программы до тех пор, пока не будет принято входящее подключение.

recv()/send() (или read()/write()):

Обмен данными: После установки соединения, сервер использует recv() для получения данных от клиента и send() для отправки данных клиенту (или read()/write(), которые также могут использоваться для операций ввода/вывода с сокетами).

Аргументы:

sockfd: Дескриптор сокета клиента, полученный при вызове accept().

buf: Буфер для данных.

len: Размер буфера.

flags (обычно 0): Флаги для управления операцией.

close():

Закрытие сокета: Функция close() закрывает сокет, освобождая связанные с ним ресурсы.

**Клиентский сокет выполняет следующие основные операции:**

socket():

Создание сокета: Функция socket() создает новый сокет для клиента. Как и на сервере, указывается домен, тип и протокол.

connect():

Установление соединения с сервером: Функция connect() пытается установить соединение с сервером, указанным в аргументах (IP-адрес и порт сервера).

Аргументы:

sockfd: Дескриптор сокета, созданный socket().

addr: Указатель на структуру с IP-адресом и портом сервера.

addrlen: Размер структуры адреса.

Блокирующая операция: connect() блокирует выполнение программы до установления соединения.

recv()/send() (или read()/write()):

Обмен данными: После установки соединения, клиент использует send() для отправки данных серверу и recv() для получения данных от сервера (или read()/write() как альтернатива).

close():

Закрытие сокета: Функция close() закрывает сокет клиента после завершения сеанса связи.