**NAME: DHANRAJ SUBHASH KORE**

**BATCH: B-3 ROLL NO: 60**

**CNT-LAB ASSIGNMENT**

**TCP CLIENT SERVER CALCULATOR**

**Server.cpp**

#undef UNICODE

#define WIN32\_LEAN\_AND\_MEAN

#include <windows.h>

#include <winsock2.h>

#include <ws2tcpip.h>

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

#include <iostream>

#include <string>

// Need to link with Ws2\_32.lib

#pragma comment (lib, "Ws2\_32.lib")

#define MAX 80

#define PORT 8080

#define SA struct sockaddr

// Function designed for chat between client and server.

void func(int sockfd)

{

char buff[MAX];

int result;

int n,i;

int a, b;

char op;

// infinite loop for chat

for (;;) {

memset(&buff, 0, MAX);

// read the message from client and copy it in buffer

recv(sockfd, buff, sizeof(buff), 0);

int a = std :: stoi(buff);

printf("From client: %s\t To client : ", buff);

recv(sockfd, buff, sizeof(buff), 0);

op = \*buff;

printf("From client: %s\t To cleint : ", buff);

recv(sockfd, buff, sizeof(buff), 0);

int b = std::stoi(buff);

printf("From client: %s\t To client: ", buff);

if (op== '+')

{

result = a + b ;

}

else if (op == '-')

{

result = a - b;

}

else if (op == '\*')

{

result = a \* b;

}

else

{

result = a / b;

}

// print buffer which contains the client contents

printf("From client: %d\t To result : ", result);

// copy server message in the buffer

std ::string str = std ::to\_string(result);

for (n = 0; str[n] != '\0'; n++)

{

buff[n] = str[n];

}

buff[n] = '\0';

// and send that buffer to client

send(sockfd, buff, sizeof(buff), 0);

// if msg contains "Exit" then server exit and chat ended.

if (strncmp("exit", buff, 4) == 0) {

printf("Server Exit...\n");

break;

}

}

}

void acceptFile(int sockfd)

{

const int BUFFER\_SIZE = 256;

char buff[BUFFER\_SIZE];

size\_t datasize;

FILE\* fd;

fopen\_s(&fd, "tmp.txt", "wb");

while ((datasize = recv(sockfd, buff, sizeof(buff), 0)) > 0)

{

fwrite(&buff, 1, datasize, fd);

memset(&buff, 0, sizeof(buff));

}

fclose(fd);

}

// Driver function

int main()

{

int sockfd, connfd, len, iResult;

struct sockaddr\_in servaddr, cli;

WSADATA wsaData;

iResult = WSAStartup(MAKEWORD(2, 2), &wsaData);

// socket create and verification

sockfd = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);

if (sockfd == -1) {

printf("socket creation failed...\n");

exit(0);

}

else

printf("Socket successfully created..\n");

memset(&servaddr, 0, sizeof(servaddr));

// assign IP, PORT

servaddr.sin\_family = AF\_INET;

servaddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

servaddr.sin\_port = htons(PORT);

// Binding newly created socket to given IP and verification

if ((bind(sockfd, (SA\*)&servaddr, sizeof(servaddr))) != 0) {

printf("socket bind failed...\n");

exit(0);

}

else

printf("Socket successfully binded..\n");

// Now server is ready to listen and verification

if ((listen(sockfd, 5)) != 0) {

printf("Listen failed...\n");

exit(0);

}

else

printf("Server listening..\n");

len = sizeof(cli);

// Accept the data packet from client and verification

connfd = accept(sockfd, (SA\*)&cli, &len);

if (connfd < 0) {

printf("server acccept failed...\n");

exit(0);

}

else

printf("server acccept the client...\n");

// Function for chatting between client and server

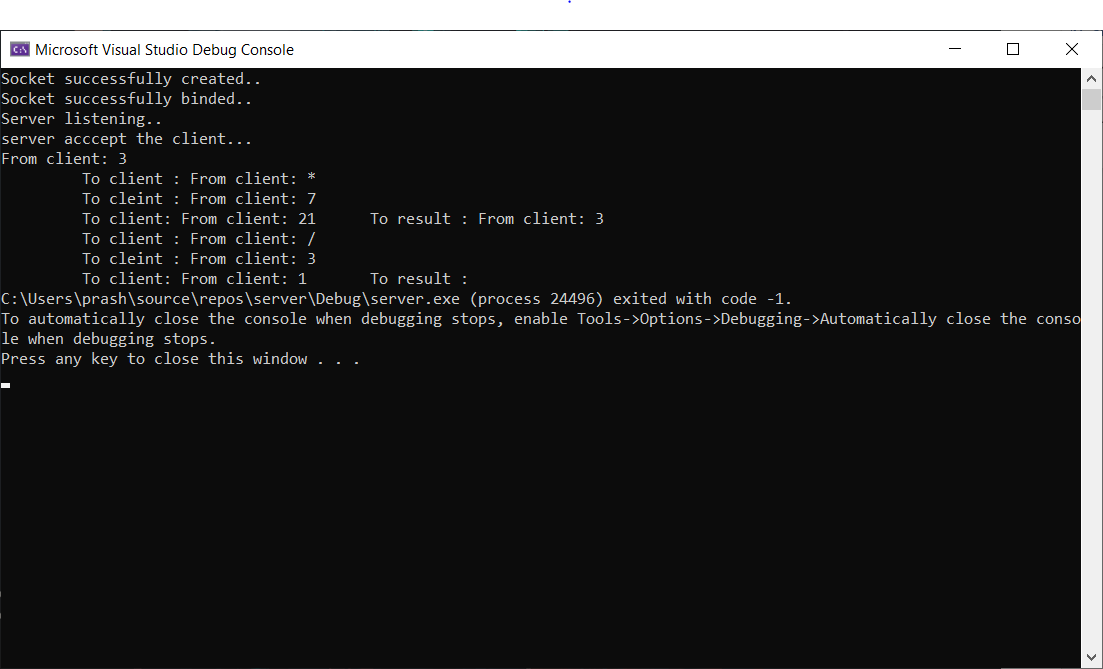
func(connfd);

//acceptFile(connfd);

// After chatting close the socket

closesocket(sockfd);

}



**Client.cpp**

#undef UNICODE

#define WIN32\_LEAN\_AND\_MEAN

#include <windows.h>

#include <winsock2.h>

#include <ws2tcpip.h>

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

#include <iostream>

#include <string>

// Need to link with Ws2\_32.lib

#pragma comment (lib, "Ws2\_32.lib")

#define MAX 80

#define PORT 8080

#define SA struct sockaddr

void func(int sockfd)

{

char buff[MAX];

int n;

for (;;) {

memset(buff, 0, sizeof(buff));

printf("enter first no. ");

n = 0;

while ((buff[n++] = getchar()) != '\n')

;

send(sockfd, buff, sizeof(buff), 0);

memset(buff, 0, sizeof(buff));

printf("enter operator ");

n = 0;

while ((buff[n++] = getchar()) != '\n')

;

send(sockfd, buff, sizeof(buff), 0);

memset(buff, 0, sizeof(buff));

printf("enter second no. ");

n = 0;

while ((buff[n++] = getchar()) != '\n')

;

send(sockfd, buff, sizeof(buff), 0);

memset(buff, 0, sizeof(buff));

recv(sockfd, buff, sizeof(buff), 0);

printf("From Server : %s", buff);

if ((strncmp(buff, "exit", 4)) == 0) {

printf("Client Exit...\n");

break;

}

}

}

void sendFile(int sockfd)

{

const int BUFFER\_SIZE = 256;

char buff[BUFFER\_SIZE];

int n;

FILE\* fd;

fopen\_s(&fd, "tmp.txt", "rb");

//size\_t rret, wret;

int bytes\_read;

while (!feof(fd)) {

if ((bytes\_read = fread(&buff, 1, BUFFER\_SIZE, fd)) >0)

send(sockfd, buff, bytes\_read, 0);

else

break;

}

fclose(fd);

}

int main()

{

int sockfd, connfd, iResult;

struct sockaddr\_in servaddr, cli;

WSADATA wsaData;

iResult = WSAStartup(MAKEWORD(2, 2), &wsaData);

// socket create and varification

sockfd = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);

if (sockfd == -1) {

printf("socket creation failed...\n");

exit(0);

}

else

printf("Socket successfully created..\n");

memset(&servaddr, 0, sizeof(servaddr));

// assign IP, PORT

servaddr.sin\_family = AF\_INET;

InetPton(AF\_INET, (PCSTR)("127.0.0.1"), &servaddr.sin\_addr.s\_addr);

servaddr.sin\_port = htons(PORT);

// connect the client socket to server socket

if (connect(sockfd, (SA\*)&servaddr, sizeof(servaddr)) != 0) {

printf("connection with the server failed...\n");

exit(0);

}

else

printf("connected to the server..\n");

// function for chat

func(sockfd);

//sendFile(sockfd);

// close the socket

closesocket(sockfd);

}

