準備２．対象受信プログラム：操作手順ｖｓソースコード

操作手順（実行手順）

1. サーバーを起動
・サーバーソケットを初期化

103行 int server_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

111行 int bind_status = bind(server_socket, (struct sockaddr *)&server_address, server_address_len);

114行 int listen_status = listen(server_socket, 10);

・クライアントソケットを初期化して受信データを待つ

121行 int client_socket = accept(server_socket, &client_address, &client_address_len);

サーバが受信
・サーバーは受信データの長さを受信

26行 int ret = recv(client_socket, (char*)&datalength, 4, 0);

・続けてデータ（ファイル名＋半角スペース＋内容）を受信

42行 int retnsize = recv(client_socket, (char*)buf, recvsize, 0);

・ファイル名と内容を分離

55行 strncpy_s(recv_filename, (char *)recv_data, BUF_SIZE);

69行 strncpy_s(saveFile_text, (char *)&recv_data[null_index + 1], BUF_SIZE);

・C:¥ZZZフォルダに受信ファイル名で内容を保存

74行 strncpy_s(saveFile_filename, "C:¥¥zzz¥¥", BUF_SIZE);

75行 strncat_s(saveFile_filename, (char *)recv_filename, BUF_SIZE - strlen(saveFile_filename));

83行 if (fopen_s(&fp, (char*)saveFile_filename, "wb") == 0)
{
fputs((const char*)saveFile_text, fp);
fclose(fp);
returnCode = 0;
88行 }

・保存に成功したら０、失敗したら１をクライアントに送信

94行 send(client_socket, (const char*)&returnCode, 1, 0);

96行 send(client_socket, (const char*)&rlen, 4, 0);

・ソケットをクローズ

128行 closesocket(client_socket);

行No	ソースコード
1	#define _WINSOCK_DEPRECATED_NO_WARNINGS
2	#include <conio.h>
25	memset(datalength, 0, 4);
26	int ret = recv(client_socket, (char*)&datalength, 4, 0);
27	if (ret == 4)
41	int recvsize = rest > 4 ? 4 : rest;
42	int retnsize = recv(client_socket, (char*)buf, recvsize, 0);
54	memset(recv_filename, 0, BUF_SIZE);
55	strncpy_s(recv_filename, (char *)recv_data, BUF_SIZE);
56	{
68	memset(saveFile_text, 0, BUF_SIZE);
69	strncpy_s(saveFile_text, (char *)&recv_data[null_index + 1], BUF_SIZE);
70	{
73	memset(saveFile_filename, 0, BUF_SIZE);
74	strncpy_s(saveFile_filename, "C:¥¥zzz¥¥", BUF_SIZE);
75	strncat_s(saveFile_filename, (char *)recv_filename, BUF_SIZE - strlen(saveFile_filename));
76	//デモ用として画面に表示する
82	FILE* fp = NULL;
83	if (fopen_s(&fp, (char*)saveFile_filename, "wb") == 0)
84	{
85	fputs((const char*)saveFile_text, fp);
86	fclose(fp);
87	returnCode = 0;
88	}
89	}
93	//W4_send_saveFile_return
94	send(client_socket, (const char*)&returnCode, 1, 0);
95	int rlen = 0;
96	send(client_socket, (const char*)&rlen, 4, 0);
97	}
111	int bind_status = bind(server_socket, (struct sockaddr *)&server_address, server_address_len); //L4_bind_status
112	if (bind_status == 0) //L3_listen_status
113	{
114	int listen_status = listen(server_socket, 10); //L4_listen_status
115	if (listen_status == 0) //L3_client_socket
120	socklen_t client_address_len = sizeof(client_address);
121	int client_socket = accept(server_socket, &client_address, &client_address_len);
122	sTimeStart = GetTickCount(); //処理開始時間
127	//L4_client_socket_close
128	closesocket(client_socket);

1．従来プログラムの構造解析(シナリオ関数定義体準備)：構文種別化（１／３）

オリジナルソース　(1 / 3)	構文種別	ベクトル識別	ベクトル名
#include <conio.h>			
#include <stdio.h>			
#include <stdlib.h>			
#include <winsock2.h>			
#include <errno.h>			
#include <string.h>			
#include <mbctype.h>			
#include <string>			
using namespace std;			
typedef int socklen_t;			
#define BUF_SIZE 4096			
struct sockaddr_in server_address;			
void readData_(int client_socket)			
{			
int returnCode = -1;			

//R2_recv_datalength
unsigned char datalength[4] = { 0 };
memset(datalength, 0, 4);
int ret = recv(client_socket, (char*)&datalength, 4, 0);
	入力文	R2	R2_recv_datalength

if (ret == 4)	(制御の条件文)		
{			
//許容される数値であることを確認 Substitution statement Substitution statement Substitution statement	代入文	L4	L4_recv_datalength

//R2_recv_data
unsigned char data[BUF_SIZE] = { 0 };
memset(data, 0, BUF_SIZE);
int rest = recv_datalength;
int pos = 0;
while (rest > 0)
{
unsigned char buf[4] = { 0 };
int recvsize = rest > 4 ? 4 : rest;
int retnsize = recv(client_socket, (char*)buf, recvsize, 0);
if (retnsize <= 4)
{
memcpy_s(&data[pos], BUF_SIZE, buf, retnsize);
pos += retnsize;
}
rest -= recvsize;
	入力文	R2	R2_recv_data

}			
unsigned char recv_data[BUF_SIZE] = { 0 };			
memcpy(recv_data, data, BUF_SIZE);			

//L4_recv_data_filename
char recv_filename[BUF_SIZE];
memset(recv_filename, 0, BUF_SIZE);
strncpy_s(recv_filename, (char *)recv_data, BUF_SIZE);
	代入文	L4	L4_recv_data_filename

{			

1．従来プログラムの構造解析(シナリオ関数定義体準備)：構文種別化（２／３）

オリジナルソース　（2 / 3）	構文種別	ベクトル識別	ベクトル名

//L4_recv_data_text
int null_index = 0;
for (int i = 0; i < recv_datalength; i++)
{
if (recv_data[i] == 0x00)
{
null_index = i;
break;
}
}
char saveFile_text[BUF_SIZE];
memset(saveFile_text, 0, BUF_SIZE);
strncpy_s(saveFile_text, (char *)&recv_data[null_index + 1], BUF_SIZE);
	代入文	L4	L4_recv_data_text

{			

//L4_saveFile_filename
char saveFile_filename[BUF_SIZE];
memset(saveFile_filename, 0, BUF_SIZE);
strncpy_s(saveFile_filename, "C:¥¥zzz¥¥", BUF_SIZE);
strncat_s(saveFile_filename, (char *)recv_filename, BUF_SIZE - strlen(saveFile_filename));
	代入文	L4	L4_saveFile_filename

//デモ用として画面に表示する
printf("%s¥n", saveFile_filename);
printf("%s", saveFile_text);			

//W4_saveFile
FILE* fp = NULL;
if (fopen_s(&fp, (char*)saveFile_filename, "wb") == 0)
{
fputs((const char*)saveFile_text, fp);
fclose(fp);
	出力文	W4	W4_saveFile

returnCode = 0;			
}			
}			
}			
}			
}			

//W4_send_saveFile_return
unsigned char ret = (unsigned char)W4_saveFile.f4;
send(client_socket, (const char*)&returnCode, 1, 0);
int rlen = 0;
send(client_socket, (const char*)&rlen, 4, 0);
	出力文	W4	W4_send_saveFile_return

}			

void run()
{
//WSA初期化
WSADATA data;
WSAStartup(MAKEWORD(2, 0), &data);			

int server_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP); //L2_server_socket	定値文	L2	L2_server_socket
if (server_socket > 0)	（制御の条件文）		
{			

memset(&server_address, 0, sizeof(sockaddr_in));//L2_server_address
server_address.sin_family = AF_INET;
server_address.sin_addr.s_addr = inet_addr("127.0.0.1");
server_address.sin_port = htons(10000);
	定置文	L2	L2_server_address

socklen_t server_address_len = sizeof(sockaddr_in); //L2_server_address_len	定置文	L2	L2_server_address_len
int bind_status = bind(server_socket, (struct sockaddr *)&server_address, server_address_len); //L4_bind_status	代入文	L4	L4_bind_status

1．従来プログラムの構造解析(シナリオ関数定義体準備)：構文種別化（３／３）

オリジナルソース　（3 / 3）	構文種別	ベクトル識別	ベクトル名
if (bind_status == 0) //L3_listen_status	条件文	L3	L3_listen_status
{			
int listen_status = listen(server_socket, 10); //L4_listen_status	代入文	L4	L4_listen_status
if (listen_status == 0) //L3_client_socket	条件文	L3	L3_client_socket
{			

//L4_client_socket
sockaddr client_address;
memset(&client_address, 0, sizeof(client_address));
socklen_t client_address_len = sizeof(client_address);
int client_socket = accept(server_socket, &client_address, &client_address_len);
	代入文	L4	L4_client_socket

if (client_socket > 0) //L3_client_socket_close	条件文	L3	L3_client_socket_close
{			
readData_(client_socket);			

//L4_client_socket_close
closesocket(client_socket);
	代入文	L4	L4_client_socket_close

}			
}			
}			
}			
//WSA終了			
WSACleanup();			
}			
int main(void)			
{			
START:			
//デモ用として画面に表示する			
printf("¥nクライアントプログラムから操作してください。¥n¥n");			
run();			
printf("¥n¥n繰り返しますか？(y/n)");			
int c = _getch();			
if (c == 0x59 || c == 0x79)	（制御の条件文）		
{			
printf("¥n");			
goto START;			
}			
return 0;			
}			

1－2．シナリオ関数定義体準備：構文種別化

	①ベクトル名	②実行命令
主語ベクトル	L4_bind_status	bind_status = bind(L2_server_socket.f4, (struct sockaddr *)&L2_server_address.f4, L2_server_address_len.f4);
	L4_listen_status	listen_status = listen(L2_server_socket.f4, 10);

	L4_client_socket	sockaddr client_address;
		memset(&client_address, 0, sizeof(client_address));
		socklen_t client_address_len = sizeof(client_address);
		
		client_socket = accept(L2_server_socket.f4, &client_address, &client_address_len);

	L4_client_socket_close	closesocket(L4_client_socket.f4);
	L4_recv_datalength	int datalength = R2_recv_datalength.f4;

	L4_recv_data_filename	char saveFile_filename[BUF_SIZE];
		memset(saveFile_filename, 0, BUF_SIZE);
		strncpy_s(saveFile_filename, (char *)R2_recv_data.f4, BUF_SIZE);

	L4_recv_data_text	//R2ベクトルの第4領域から該当項目を取得する処理
		int null_index = 0;
		for (int i = 0; i < L4_recv_datalength.f4; i++)
		{
		if (R2_recv_data.f4[i] == 0x00)
		{
		null_index = i;
		break;
		}
		}
		char saveFile_text[BUF_SIZE];
		memset(saveFile_text, 0, BUF_SIZE);
		strncpy_s(saveFile_text, (char *)&R2_recv_data.f4[null_index + 1], BUF_SIZE);

	L4_saveFile_filename	memset(saveFile_filename, 0, BUF_SIZE);
		strncpy_s(saveFile_filename, "C:¥¥zzz¥¥", BUF_SIZE);
		strncat_s(saveFile_filename, (char *)L4_recv_data_filename.f4, BUF_SIZE - strlen(saveFile_filename));

	W4_saveFile	int ret = 1;
		FILE* fp = NULL;
		if (fopen_s(&fp, (char*)L4_saveFile_filename.f4, "wb") == 0)
		{
		fputs((const char*)L4_recv_data_text.f4, fp);
		fclose(fp);
		ret = 0;
		}

	W4_send_saveFile_return	unsigned char ret = (unsigned char)W4_saveFile.f4;
		send(L4_client_socket.f4, (const char*)&ret, 1, 0);
		int rlen = 0;
		send(L4_client_socket.f4, (const char*)&rlen, 4, 0);

	L2_server_socket	server_socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

	L2_server_address	memset(&server_address, 0, sizeof(sockaddr_in));
		server_address.sin_family = AF_INET;
		server_address.sin_addr.s_addr = inet_addr("127.0.0.1");
		server_address.sin_port = htons(10000);

	L2_server_address_len	server_address_len = sizeof(sockaddr_in);
	R2_recv_datalength	int ret = recv(L4_client_socket.f4, (char*)&datalength, 4, 0);

	R2_recv_data	//一度に長いバッファに読み込むのではなく4バイトずつ読み込む
		memset(recv_data, 0, BUF_SIZE);
		int rest = L2_recv_datalength.f4;
		int ret = 0;
		while(rest > 0)
		{
		unsigned char buf[4] = { 0 };
		int recvsize = rest > 4 ? 4 : rest;
		int retnsize = recv(L4_client_socket.f4, (char*)buf, recvsize, 0);
		if (retnsize <= 4)
		{
		memcpy_s(&recv_data[ret], BUF_SIZE, buf, retnsize);
		ret += retnsize;
		}
		rest -= recvsize;
		}

	L3_listen_status	if (L4_bind_status.f4 == 0)
	L3_client_socket	if (L4_listen_status.f4 == 0)
	L3_client_socket_close	if (W4_send_saveFile_return.f4 == 1)
	L3_saveFile	