第三回(課題 2)ServerAssignment.cpp コード

```
#include "02ServerAssignment.h"
//---------------定数--------------------------
#define  NONBLOCKING_MODE 1
#define  BUFFER_SIZE       1024
#define PORT 7000
//------------------------------------------------------------------------------
// サーバ側コマンド入力スレッド
void ServerAssignment02::Exit()
{
	while (loop) {
		std::string input;
		std::cin >> input;
		if (input == "exit")
		{
			loop = false;
		}
	}
}

//-----実行処理-----------------------------------------------------------------
void ServerAssignment02::Execute()
{
	WSADATA wsaData;

	// TODO 01_01 課題作成する事
	// WinsockAPI を初期化
		//(WSAStartup 関数)
	int wsaStartUp = WSAStartup(MAKEWORD(2, 2), &wsaData);
	if (wsaStartUp != 0) {//失敗
		std::cout << "1_1 ERR" << std::endl;
		return;			//失敗なので終了
	}
```

第三回(課題 2)ServerAssignment.cpp コード

```
// TODO 01_02 課題作成する事
// サーバの受付設定
struct sockaddr_in addr;
//[addr.sin_family]の設定
//[addr.sin_port]の設定
//[addr.sin_addr.S_un.S_addr]の設定
addr.sin_family = AF_INET;  //IPV4 でネットワーク接続
addr.sin_port = htons(PORT);          //ポート 7000 番
addr.sin_addr.S_un.S_addr = INADDR_ANY; //誰でも OK

// TODO 01_03 課題作成する事
// ソケットの作成
SOCKET sock;
          //socket 関数でソケット作成：[インターネット接続]、[UDP 接続] 設定
          //生成できなければエラー表示して、WSACleanup()し終了
sock = socket(AF_INET, SOCK_DGRAM, 0); //UDP ソケット
if (sock == INVALID_SOCKET) {
          std::cout << "1_3 ERR" << std::endl;
          WSACleanup(); //ソケットの作り損ねを消す
          return;             //ソケット作り損ねたので、clean して戻る
}

// TODO 01_04 課題作成する事
// ノンブロッキング設定
u_long mode = NONBLOCKING_MODE;
          //(ioctlsocket 関数でのノンブロック指定)
int res = ioctlsocket(sock, FIONBIO, &mode);//止まらない
if (res != 0) {//ERR
          std::cout << "1_4 ERR" << std::endl;
          return;  //設定できなかったのでおしまい
}

// TODO 01_05 課題作成する事
// ソケットと受付情報を紐づける
//(bind 関数)
res = bind(sock, reinterpret_cast<sockaddr*>(&addr), sizeof(addr));
```

第三回(課題 2)ServerAssignment.cpp コード

```
	if (res != 0) {//ERR
		std::cout << "1_5 ERR" << std::endl;
		return;
	}

	std::cout << "Server Initialize OK." << std::endl;

	// クライアントからの受付処理
	int size = sizeof(struct sockaddr_in);

//↑↑ここまでは課題 1 のメソッドごとコピー&ペーストで OK です↑↑

	// サーバ側からコマンド入力で終了されるまでループする。
	// キーボードで exit を入力するとループを抜けるための別スレッドを立ち上げる
	std::thread th(&ServerAssignment02::Exit, this);

	PlayerInformation plInfoOld;//予備課題情報表示用　ストックしておいて新しいデータと比較
	ZeroMemory(&plInfoOld, sizeof(PlayerInformation));

	do {
		// プレイヤー情報構造体
		PlayerInformation plInfo{};
		int len = sizeof(sockaddr);
		int size = 0;
		//今回新規作成部分 S01
		// TODO 02_01
		// ソケット sock からプレイヤー情報データ受信(recvfrom 関数)
		// recvfrom の受信データ保存場所は plInfo
		size = recvfrom(sock, reinterpret_cast<char*>(&plInfo), sizeof(plInfo), 0, reinterpret_cast<sockaddr*>(&client), &len);
		if (size > 0)	//データ受信した！
		{
			//予備課題 02_s
			//position(x,y,z)/angle/state)を plInfoOld のデータと比較、
```

第三回(課題 2)ServerAssignment.cpp コード

```
                        //      全て同じなら下にある、プレイヤー情報表示を行わない
ようにする。
                        //比較後次回用に、plInfo を plInfoOld にストック









                        // プレイヤー情報表示
                        std::cout << "position :(" << plInfo.position.x << "," <<
plInfo.position.y << "," << plInfo.position.z << ")" << std::endl;
                        std::cout << "angle :(" << plInfo.angle.x << "," <<
plInfo.angle.y << "," << plInfo.angle.z << ")" << std::endl;
                        std::cout << "state : " << static_cast<int>(plInfo.state) <<
std::endl;
                        //今回新規作成部分 S02
                        // TODO 02_02 plInfo をクライアントに送信(送り返す)
                        //sendto の戻り値は size に代入（直下エラー判断しています。）
```

```
size=sendto(sock,reinterpret_cast<char*>(&plInfo), sizeof(plInfo),0,
reinterpret_cast<sockaddr*>(&client), static_cast<int>(sizeof(sockaddr_in)));
```

```
                        if (size < 0)
                        {
                                std::cout << "sendto failed. error code : " <<
WSAGetLastError() << std::endl;
                        }
                }
                else
                {
                        if (WSAGetLastError() != WSAEWOULDBLOCK)
                        {
                                std::cout << "recvfrom failed. error code : " <<
WSAGetLastError() << std::endl;
                        }
                }
```

第三回(課題 2)ServerAssignment.cpp コード

```
	} while (loop);
	th.join();                              //コマンド入力スレッド終了を待つ

	//ここから下も、前回のコピー＆ペーストでOK
	// TODO 01_08
	// サーバソケットの切断
	res = closesocket(sock);    //ソケット閉じる
	if (res <= 0) {//ERR
		std::cout << "1_8 ERR" << std::endl;
	}

	// TODO 01_09
	// WSA 終了
	res = WSACleanup();
	if (res != 0) {
		std::cout << "1_9ERR" << std::endl;
	}

	std::cout << "Cleanup WinsockAPI Success." << std::endl;
}
```