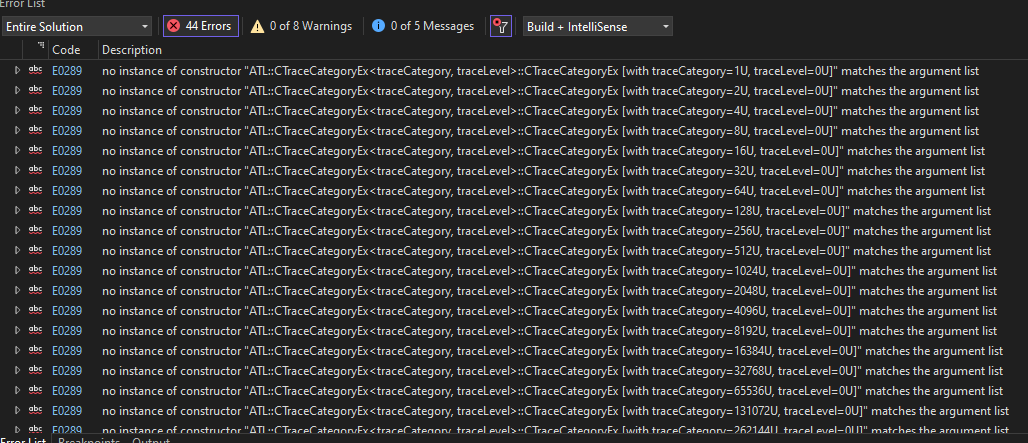
Sisteme de Operare

**Nicoleta Radu – Tema 2**

Comentarii Tema –

Conversia numarului obtinut din calculul factorialului nu a fost posibila intrucat libraria “atlstr” (#include <atlstr.h>) produce o multitidine de erori in program. Nu am gasit o alternativa pentru tranformarea din integer In char[] decat cea care foloseste “sprintf\_s “ / “sprintf”. Comunicarea intre client si server a fost totusi efectuata si functioneaza din testele facute de mine. Mai jos am atasat o poza cu erorile care apar numai din includerea librariei in program.



# Client

#undef UNICODE

#include <windows.h>

#include "resource.h"

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

BOOL CALLBACK DlgProc(HWND hDlg, UINT iMsg,

WPARAM wParam, LPARAM lParam);

BOOL dlgActive = FALSE; //variabila pentru caseta de dialog

HWND hwndMain; //variabila handle a ferestrei principale

// Variabile pentru tema

LPCTSTR GetFromClip = "GetFromClipboard";

HWND hDialog;

HGLOBAL hGlobalMemory;

char\* pGlobalMemory;

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,

PSTR szCmdLine, int iCmdShow)

{

static char szAppName[] = "Client";

HWND hwnd;

MSG msg;

WNDCLASSEX wndclass;

wndclass.cbSize = sizeof(wndclass);

wndclass.style = CS\_HREDRAW | CS\_VREDRAW; //optiuni pentru stilul clasei

wndclass.lpfnWndProc = WndProc;

wndclass.cbClsExtra = 0;

wndclass.cbWndExtra = 0;

wndclass.hInstance = hInstance;

wndclass.hIcon = LoadIcon(NULL, IDI\_APPLICATION);

// incarca pictograma fisierelor de tip aplicatie

wndclass.hCursor = LoadCursor(NULL, IDC\_ARROW); // incarca cursorul sageata

wndclass.hbrBackground = (HBRUSH)GetStockObject(WHITE\_BRUSH);

// fondul ferstrei de culoare alba

wndclass.lpszMenuName = NULL;

wndclass.lpszClassName = szAppName;

wndclass.hIconSm = LoadIcon(NULL, IDI\_APPLICATION);

RegisterClassEx(&wndclass);

hwnd = CreateWindow(szAppName, //numele clasei inregistrat cu RegisterClass

"Client", // text pentru bara de titlu a ferestrei

WS\_OVERLAPPEDWINDOW, // stilul ferestrei

CW\_USEDEFAULT, // pozitia orizontala implicit�

CW\_USEDEFAULT, // pozitia verticala implicita

CW\_USEDEFAULT, // latimea implicita

CW\_USEDEFAULT, // inaltimea implicita

NULL, // handle-ul ferestrei parinte

NULL, // handle-ul meniului ferestrei

hInstance, // proprietara ferestrei

NULL);

SetWindowPos(hwnd, HWND\_BOTTOM, 0, 0, 0, 0, SWP\_NOACTIVATE);

// schimba dimensiunea, pozitia si ordinea z a ferestrei copil, a ferestrei pop-up

ShowWindow(hwnd, SW\_HIDE);

UpdateWindow(hwnd);

hwndMain = hwnd;

while (GetMessage(&msg, NULL, 0, 0))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return msg.wParam;

}

LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)

{

// Variabile tema

static HINSTANCE hInstance;

static UINT WM\_GETFROMCLIP;

switch (iMsg)

{

case WM\_CREATE: // operatiile ce se executa la crearea ferestrei

// se creaza caseta de dialog

WM\_GETFROMCLIP = RegisterWindowMessage(GetFromClip);

hInstance = ((LPCREATESTRUCT)lParam)->hInstance;

if (!dlgActive) {

DialogBox(hInstance, MAKEINTRESOURCE(IDD\_DIALOG),

hwnd, DlgProc);

PostMessage(hwnd, WM\_CLOSE, 0, 0);

// insereaza un nou mesaj nou in coada de asteptare

}

return 0;

case WM\_DESTROY:

PostQuitMessage(0); // insereaza un mesaj de incheiere

return 0;

}

if (iMsg == WM\_GETFROMCLIP)

{

SendMessage(hDialog, WM\_GETFROMCLIP, 0, 0);

return 0;

}

return DefWindowProc(hwnd, iMsg, wParam, lParam);

}

/////////////////////////////////////////////////////////////

BOOL CALLBACK DlgProc(HWND hDlg, UINT iMsg, WPARAM wParam, LPARAM lParam) {

// Variabile tema - to send

char text[100];

static char bufferInput[100];

static UINT WM\_GETFROMCLIP;

HWND hwndReceive;

// Variabile tema - to receive

BOOL bAvailable;

HANDLE hClipMemory;

RECT rect; //obiect dreptunghi

char\* pMyCopy, \* pClipMemory;

static char bufferResponse[100] = "Astept msg...";

switch (iMsg) {

case WM\_INITDIALOG:

// setari initiale

hDialog = hDlg;

WM\_GETFROMCLIP = RegisterWindowMessage(GetFromClip);

return TRUE;

case WM\_CLOSE:

//dlgActive = FALSE;

EndDialog(hDlg, 0);

return TRUE;

case WM\_COMMAND:

switch (LOWORD(wParam)) {

case IDC\_BUTTON\_FACTORIAL:

// citeste un text dintr-o caseta de editare

GetDlgItemText(hDialog, IDC\_EDIT, bufferInput, 100);

// cauta fereastra cu numele server

hwndReceive = FindWindow("Server", "Server");

;

// (SET)Daca o gaseste, scrie in zona de memorie

if (hwndReceive != NULL)

{

SendMessage(hwndReceive, WM\_GETFROMCLIP, 0, 0); //notificare receive pt. preluare date

hGlobalMemory = GlobalAlloc(GHND, strlen(bufferInput) + 1); //alocare memorie in heap

pGlobalMemory = (char\*)GlobalLock(hGlobalMemory); //obtinerea unui pointer la memoria

for (UINT i = 0; i < strlen(bufferInput); i++) //copiere date in heap

\*(pGlobalMemory++) = bufferInput[i];

GlobalUnlock(hGlobalMemory);

OpenClipboard(hDialog); //deschidere clipboard

SetClipboardData(CF\_TEXT, hGlobalMemory);//transmitere date

CloseClipboard();//inchidere clipboard

}

else

{

MessageBox(hDialog, "Nu am gasit Serverul!", "Eroare", MB\_OK);

}

// Receptionez rezultatul si il salvez in clipboard

// (GET) data

bAvailable = IsClipboardFormatAvailable(CF\_TEXT);

if (bAvailable == FALSE) {

//SetDlgItemText(hDialog, IDC\_EDIT\_REZULTAT, "Not available");

return 0;

}

if (iMsg == WM\_GETFROMCLIP) {

OpenClipboard(hDialog);

//obtinere handle la memoria clipboard

hClipMemory = GetClipboardData(CF\_TEXT);

//alocare dinamica de memorie (buffer)

pMyCopy = (char\*)malloc(GlobalSize(hClipMemory));

//obtinere pointer la memoria clipboard

pClipMemory = (char\*)GlobalLock(hClipMemory);

strcpy(pMyCopy, pClipMemory); //copiere date

strcpy\_s(bufferResponse, pMyCopy);

SetDlgItemText(hDialog, IDC\_EDIT\_REZULTAT, bufferResponse);

GlobalUnlock(hClipMemory);

CloseClipboard();

return 0;

}

return TRUE;

case IDC\_CANCEL: // operatii ce se executa la actiunea apasarii but. CANCEL

MessageBox(hDialog, "Abandon!", "Ati selectat CANCEL", MB\_OK);

//dlgActive = FALSE;

EndDialog(hDialog, 0);

return TRUE;

}

//break;

}

return FALSE;

}

# Server

#undef UNICODE

#include <windows.h>

#include "resource.h"

//#include <atlstr.h>

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

BOOL CALLBACK DlgProc(HWND hDlg, UINT iMsg,

WPARAM wParam, LPARAM lParam);

BOOL dlgActive = FALSE; //variabila pentru caseta de dialog

HWND hwndMain; //variabila handle a ferestrei principale

// Variabile pentru tema

LPCTSTR GetFromClip = "GetFromClipboard";

HWND hDialog;

HGLOBAL hGlobalMemory;

char\* pGlobalMemory;

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,

PSTR szCmdLine, int iCmdShow)

{

static char szAppName[] = "Server";

HWND hwnd;

MSG msg;

WNDCLASSEX wndclass;

wndclass.cbSize = sizeof(wndclass);

wndclass.style = CS\_HREDRAW | CS\_VREDRAW; //optiuni pentru stilul clasei

wndclass.lpfnWndProc = WndProc;

wndclass.cbClsExtra = 0;

wndclass.cbWndExtra = 0;

wndclass.hInstance = hInstance;

wndclass.hIcon = LoadIcon(NULL, IDI\_APPLICATION);

// incarca pictograma fisierelor de tip aplicatie

wndclass.hCursor = LoadCursor(NULL, IDC\_ARROW); // incarca cursorul sageata

wndclass.hbrBackground = (HBRUSH)GetStockObject(WHITE\_BRUSH);

// fondul ferstrei de culoare alba

wndclass.lpszMenuName = NULL;

wndclass.lpszClassName = szAppName;

wndclass.hIconSm = LoadIcon(NULL, IDI\_APPLICATION);

RegisterClassEx(&wndclass);

hwnd = CreateWindow(szAppName, //numele clasei inregistrat cu RegisterClass

"Server", // text pentru bara de titlu a ferestrei

WS\_OVERLAPPEDWINDOW, // stilul ferestrei

CW\_USEDEFAULT, // pozitia orizontala implicit�

CW\_USEDEFAULT, // pozitia verticala implicita

CW\_USEDEFAULT, // latimea implicita

CW\_USEDEFAULT, // inaltimea implicita

NULL, // handle-ul ferestrei parinte

NULL, // handle-ul meniului ferestrei

hInstance, // proprietara ferestrei

NULL);

SetWindowPos(hwnd, HWND\_BOTTOM, 0, 0, 0, 0, SWP\_NOACTIVATE);

// schimba dimensiunea, pozitia si ordinea z a ferestrei copil, a ferestrei pop-up

ShowWindow(hwnd, SW\_HIDE);

UpdateWindow(hwnd);

hwndMain = hwnd;

while (GetMessage(&msg, NULL, 0, 0))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return msg.wParam;

}

LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)

{

// Variabile tema

static HINSTANCE hInstance;

static UINT WM\_GETFROMCLIP;

switch (iMsg)

{

case WM\_CREATE: // operatiile ce se executa la crearea ferestrei

// se creaza caseta de dialog

WM\_GETFROMCLIP = RegisterWindowMessage(GetFromClip);

hInstance = ((LPCREATESTRUCT)lParam)->hInstance;

if (!dlgActive) {

DialogBox(hInstance, MAKEINTRESOURCE(IDD\_DIALOG),

hwnd, DlgProc);

PostMessage(hwnd, WM\_CLOSE, 0, 0);

// insereaza un nou mesaj nou in coada de asteptare

}

return 0;

case WM\_DESTROY:

PostQuitMessage(0); // insereaza un mesaj de incheiere

return 0;

}

if (iMsg == WM\_GETFROMCLIP)

{

SendMessage(hDialog, WM\_GETFROMCLIP, 0, 0);

return 0;

}

return DefWindowProc(hwnd, iMsg, wParam, lParam);

}

/////////////////////////////////////////////////////////////

int Factorial(int yourVariable)

{

int result{ 1 };

for (size\_t i = 1; i <= yourVariable; i++)

{

result = result \* i;

}

return result;

}

BOOL CALLBACK DlgProc(HWND hDlg, UINT iMsg, WPARAM wParam, LPARAM lParam) {

// Variabile tema - to send

char text[100] = "abcdefghijklmnopqrstuvwyz";

static char bufferInput[100];

int num, factorial = 1, verif = 0;

static UINT WM\_GETFROMCLIP;

HWND hwndReceive;

// Variabile tema - to receive

BOOL bAvailable;

HANDLE hClipMemory;

RECT rect; //obiect dreptunghi

char\* pMyCopy, \* pClipMemory;

char bufferResponse[100];

static char BufferSend[100];

switch (iMsg) {

case WM\_INITDIALOG:

// setari initiale

hDialog = hDlg;

WM\_GETFROMCLIP = RegisterWindowMessage(GetFromClip);

return TRUE;

case WM\_CLOSE:

//dlgActive = FALSE;

EndDialog(hDialog, 0);

return TRUE;

}

if (iMsg == WM\_GETFROMCLIP)

{

// Verificam daca clipboardul este disponibil

bAvailable = IsClipboardFormatAvailable(CF\_TEXT);

if (!bAvailable)

{

return 0;

}

// (GET) Copiere date din zona de memorie clipboard in "bufferInput[]"

OpenClipboard(hDialog);

hClipMemory = GetClipboardData(CF\_TEXT);

pMyCopy = (char\*)(GlobalLock(hClipMemory));

pClipMemory = (char\*)GlobalLock(hClipMemory);

strcpy(pMyCopy, pClipMemory);

strcpy\_s(bufferInput, pMyCopy);

GlobalUnlock(hClipMemory);

CloseClipboard();

GetClientRect(hDialog, &rect);

InvalidateRect(hDialog, &rect, TRUE);

// Afisare numar in edit control

SetDlgItemText(hDialog, IDC\_EDIT\_REZULTAT, bufferInput);

// Transformare numarul primit in integer

num = atoi(bufferInput);

if (strcmp(bufferInput, text))

{

verif = 1; // daca nu s-a transmis un numar

}

else

{

if (num > 15 || num < 0)

{

verif = 2;

}

else

{

if (!strcmp(bufferInput, ""))

{

verif = 3;

}

else

{

factorial = Factorial(num);

sprintf\_s(bufferResponse, "%ld", factorial);

}

}

}

switch (verif)

{

case 1:

strcpy(BufferSend, "Nu am primit un numar valid"); break;

case 2:

strcpy(BufferSend, "Am primit numar > 15. Domeniul este [0,16)"); break;

case 3:

strcpy(BufferSend, "Sir vid!"); break;

case 4:

strcpy(BufferSend, text); break;

}

// (SET) pune valoarea calculata in clipboard

hwndReceive = FindWindow("Client", "Client");

if (hwndReceive != NULL)

{

SendMessage(hwndReceive, WM\_GETFROMCLIP, 0, 0); //notificare receive pt. preluare date

hGlobalMemory = GlobalAlloc(GHND, strlen(bufferResponse) + 1); //alocare memorie in heap

pGlobalMemory = (char\*)GlobalLock(hGlobalMemory); //obtinerea unui pointer la memoria

for (UINT i = 0; i < strlen(bufferResponse); i++) //copiere date in heap

\*(pGlobalMemory++) = bufferResponse[i];

GlobalUnlock(hGlobalMemory);

OpenClipboard(hDialog); //deschidere clipboard

SetClipboardData(CF\_TEXT, hGlobalMemory);//transmitere date

CloseClipboard();//inchidere clipboard

}

else

{

MessageBox(hDlg, "Nu am gasit Clientul!", "Eroare", MB\_OK);

SetDlgItemText(hDialog, IDC\_EDIT\_REZULTAT, "Nu am gasit client");

}

}

return FALSE;

}