1. Создание и отладка файла DLL. Для демонстрации работы написать две элементарные функции в библиотеке и вызвать их из основной программы

Создаем два проекта, один настраиваем как длл вин апи(подсистему меняем на винду, а в общих настройках exe на dll), второй как обычно. В коде программы, где path прописываем путь к длл.

Код DLL:

\_\_declspec(dllimport) int Summa(float a, float b);

\_\_declspec(dllimport) int Razn(float a, float b);

int Summa(float a, float b)

{

return a + b;

}

int Razn(float a, float b)

{

return a - b;

}

Код программы:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <Windows.h>

#include <malloc.h>

#include <stdio.h>

#define PATH L"..\\Debug\\DLL.dll"

typedef(\*MyFun)(float, float);

typedef(\*MyFunRazn)(float, float);

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPWORD pCmdLine, int nCmdShow)

{

HINSTANCE myDll = LoadLibrary(PATH);

if (myDll == NULL)

{

return 1;

}

MyFun Sum = (MyFun)GetProcAddress(myDll, "Summa");

int summa = 0;

summa = Sum(2, 2);

char\* a = calloc(100, 1);

sprintf(a, "%d", summa);

MessageBoxA(NULL, a, "Сумма", MB\_OK);

MyFunRazn razn = (MyFunRazn)GetProcAddress(myDll, "Razn");

int r = razn(5, 6);

char \*str = calloc(100, 1);

sprintf(str, "%d", r);

MessageBoxA(NULL, str, "Разность", MB\_OK);

return 0;

}

2. Программа для считывания и записи строкового значения с использованием текстового файла. Для работы с файлами используются функции WinAPI (WinAPI)

#include <Windows.h>

#include <stdio.h>

int WINAPI wWinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)

{

HANDLE file = CreateFile(L"1.txt", GENERIC\_READ, 0, NULL, OPEN\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);//открываем для чтения

DWORD size = 100,//кол-во символов которые надо прочитать

bytes;//типо счетчик реально прочитанных символов

char\* text = calloc(size + 1, 1);//буфер куда записывается прочитанное

if (file != INVALID\_HANDLE\_VALUE) //если файл есть

{

ReadFile(file, text, size, &bytes, NULL);//читаем ебать

CloseHandle(file);//закрываем нахуй ибо нахуй он нам нужен теперь

}

file = CreateFile(L"answer.txt", GENERIC\_WRITE, 0, NULL,

CREATE\_ALWAYS, //если нет файла создаст, если есть перезапишет(удалит и создаст заново)

FILE\_ATTRIBUTE\_NORMAL, NULL);//открываем для записи

WriteFile(file, text, size, &bytes, NULL);//записываем (size и bytes как и при чтении)

}

3. Получение названия клавиши с использованием функции GetKeyNameText (Windows HOOK). Название клавиши выводить в MessageBox

#include <Windows.h>

#include <malloc.h>

LRESULT CALLBACK LogKey(int iCode, WPARAM wParam, LPARAM lParam);

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, PWSTR pCmdLine, int nCmdShow)

{

HHOOK hHook = SetWindowsHookExW(WH\_KEYBOARD\_LL, LogKey, NULL, 0);

MSG msg = { 0 };

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

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

UnhookWindowsHookEx(hHook);

}

LRESULT CALLBACK LogKey(int iCode, WPARAM wParam, LPARAM lParam)

{

if (wParam != WM\_KEYDOWN)

{

LPKBDLLHOOKSTRUCT pHook = (LPKBDLLHOOKSTRUCT)lParam;

DWORD iKey = MapVirtualKey(pHook->vkCode, NULL) << 16;

LPWSTR str = calloc(1000, 1);

if (GetKeyNameTextW(iKey, str, 1000) == 0)

{

MessageBox(NULL, L"При опредление название клавиши возникла ошибка", L"Информация", MB\_OK);

}

else

{

MessageBox(NULL, str, L"Название клавиши", MB\_OK);

}

}

}

Для мышки:

#include <Windows.h>

#include <stdio.h>

LRESULT CALLBACK LogMouse(int iCode, WPARAM wParam, LPARAM lParam);

DWORD LKey = 0, RKey = 0, CKey = 0, TKey = 0;

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE HpREViNSTANCE, PSTR pCmdLine, int nCmdShow)

{

HHOOK hHook = SetWindowsHookExW(WH\_MOUSE\_LL, LogMouse, NULL, NULL);

MSG msg = { 0 };

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

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

UnhookWindowsHookEx(hHook);

return 0;

}

LRESULT CALLBACK LogMouse(int iCode, WPARAM wParam, LPARAM lParam)

{

if (wParam == 513)

{

LKey++;

LPWSTR String = (LPWSTR)calloc(150, sizeof(WCHAR));

swprintf(String, 150, L"Внимание! Лимит нажатий на левую кнопку мыши исчерпан. Вы нажали %d раз", LKey);

MessageBox(NULL, String, L"Зафиксировано несанкционированное нажатие кнопки мыши", MB\_OK);

free(String);

}

if (wParam == 516)

{

RKey++;

LPWSTR String = (LPWSTR)calloc(150, sizeof(WCHAR));

swprintf(String, 150, L"Внимание! Лимит нажатий н а правую кнопку фмыши исчерпан. Вы нажали %d раз", RKey);

MessageBox(NULL, String, L"Зафиксировано несанкционированное нажатие кнопки мыши", MB\_OK);

free(String);

}

if (wParam == 520)

{

CKey++;

LPWSTR String = (LPWSTR)calloc(150, sizeof(WCHAR));

swprintf(String, 150, L"Внимание! Лимит нажатия на колёсико мыши исчерпан. Вы нажали %d раз", CKey);

MessageBox(NULL, String, L"Зафиксировано несанкционированное нажатие кнопки мыши", MB\_OK);

free(String);

}

if (wParam == 522)

{

TKey++;

LPWSTR String = (LPWSTR)calloc(150, sizeof(WCHAR));

swprintf(String, 150, L"Внимание! Лимит прокрутки колёсика мыши исчерпан. Вы прокрутил %d раз", TKey);

MessageBox(NULL, String, L"Зафиксировано несанкционированное нажатие кнопки мыши", MB\_OK);

free(String);

}

return CallNextHookEx(NULL, iCode, wParam, lParam);

}

4. Получение кода клавиши в формате ASCII и использованием Windows HOOK. Название клавиши выводить в MessageBox (ascii таблица символа)

#include <Windows.h>

#include <malloc.h>

LRESULT CALLBACK LogKey(int iCode, WPARAM wParam, LPARAM lParam);

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, PWSTR pCmdLine, int nCmdShow)

{

HHOOK hHook = SetWindowsHookExW(WH\_KEYBOARD\_LL, LogKey, NULL, 0);

MSG msg = { 0 };

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

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

UnhookWindowsHookEx(hHook);

}

LRESULT CALLBACK LogKey(int iCode, WPARAM wParam, LPARAM lParam)

{

if (wParam != WM\_KEYDOWN)

{

LPKBDLLHOOKSTRUCT pHook = (LPKBDLLHOOKSTRUCT)lParam;

DWORD iKey = MapVirtualKeyA(pHook->vkCode, NULL) << 16;

LPSTR\* str = calloc(100, 1);

if (GetKeyNameTextA(iKey, str, 100) == 0)

//  извлекает строку, которая представляет название клавиши.

{

MessageBoxA(NULL, "При опредление название клавиши возникла ошибка", "Информация", MB\_OK);

}

else

{

MessageBoxA(NULL, str, "Название клавиши", MB\_OK);

}

}

}

**5. Программа для преобразования числа в строку и склейки двух строк. Разработать без использования системных функций**

#include "Header.h"

void main()

{

system("chcp 1251>nul");

char c1[] = "asdsdad";

char c2[] = "павпавпвап";

char\* c = conc(c1, c2);

printf("%s", c);

int a = 12223;

printf("\n");

char\* c3 = cvs(a);

printf("%s", c3);

}

#pragma once

#pragma once

#include <stdio.h>

#include <math.h>

char\* conc(char\*, char\*);

char\* conc(char\* a, char\* b)

{

int l1 = 0, l2 = 0; // длина строки

while (a[l1] != '\0')

{

l1++;

}

while (b[l2] != '\0')

{

l2++;

}

char\* c = malloc(l1 + l2);

for (int i = 0; i < l1; i++) // записываем по длине строки

{

c[i] = a[i];

}

for (int i = 0; i < l2; i++)

{

c[i + l1] = b[i];

}

c[l1 + l2] = '\0';

return c;

}

char\* cvs(int a)

{

int b = a, l1 =0,z=0;

while (b) {

b /= 10; // считаем сколько цифр в этом цисле

l1++;

}

b = a;

char\* x = malloc(l1-1); // выделяем память по чаровскую строку

for (int i = l1-1; i >= 0 ; i--)

{

x[i] = b % 10 + '0'; // запись в строку, а 0 для форматирования

b /= 10; // делим б для след итерации

}

x[l1] = '\0';

return x;

}

6. Программа с использованием нескольких программных модулей (каждый модуль должен располагаться в отдельном файле .c). В рамках модулей должна быть как минимум одна пользовательская функция и одна структура. Также для этих модулей должен быть написан заголовочный файл, содержащий всю необходимую информацию для корректного функционирования модулей

Создаем в проекте три файла с и один заголовочный h. Вставляем туда код снизу и исключаем из проекта файл структуры и функции (пкм по файлу – исключить из проекта)

Основной код:

#include "Header.h"

int main()

{

system("chcp 1251>null");

struct s\* us = calloc(1, sizeof(struct s));

us->a = 5;

us->b = 15;

printf("Сумма чисел: %d\n", sum(us));

}

Заголовочный:

#pragma once

#include <stdio.h>

#include <malloc.h>

#include "Struct.c"

#include "Func.c"

Функция:

int sum(struct s\* us)

{

return us->a + us->b;

}

Структура:

struct s

{

int a;

int b;

};

7. Программа для преобразования строки в число. Строка хранится в виде указателя LPWSTR. Реализовать без использования системных функций

#include <Windows.h>

#include <stdio.h>

int ConvertToInt(LPWSTR\* s);

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, PWSTR pCmdLine, int nCmdShow)

{

LPWSTR str = L"345";

int num = ConvertToInt(str);

}

// Преобразования строки в число

int ConvertToInt(LPWSTR s)

{

int number = 0;

int sign = 0; // знак числа числа 0 - положительное, 1 — отрицательное

int i = 0;

if (s[0] == '-')

{

sign = 1;

i++;

}

while (s[i] != '\0')

{

number = number + (s[i] - '0');

number = number \* 10;

i++;

}

number = number / 10;

if (sign == 1)

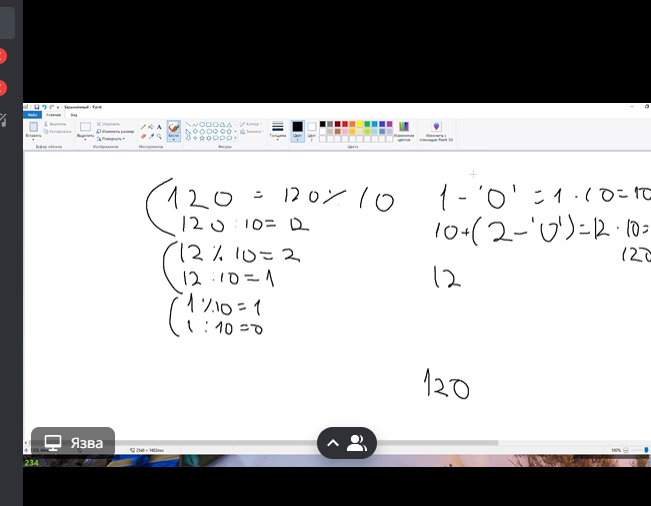
{

number = -number;

}

return(number);

}



8. Программа для записи строки в системный буфер обмена

#include <Windows.h>

int WINAPI wWinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPWSTR pCmdLine, int nCmdShow)

{

LPWSTR str = L"Строка помещена в буфер обмена";

ClicboardInputText(str);

return 0;

}

int ClicboardInputText(LPWSTR text)

{

DWORD len;

len = wcslen(text) + 1;

HANDLE hMem = GlobalAlloc(GMEM\_MOVEABLE, len \* sizeof(LPWSTR));

memcpy(GlobalLock(hMem), text, len \* sizeof(LPWSTR));

GlobalUnlock(hMem);

OpenClipboard(0);

EmptyClipboard();

SetClipboardData(CF\_UNICODETEXT, hMem);

CloseClipboard();

return 0;

}

**9. Программа для считывания данных из системного буфера обмена**

#include <Windows.h>

#include <malloc.h>

TCHAR\* ClipboardOutputText();

int WINAPI wWinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPWSTR pCmdLine, int nCmdShow)

{

LPWSTR str = calloc(100, 1);

str = ClipboardOutputText();

MessageBox(NULL, str, L"Значение системного буфера", MB\_OK);

return 0;

}

TCHAR\* ClipboardOutputText()

{

LPWSTR text = NULL;

OpenClipboard(NULL);

HANDLE hClipboardData = GetClipboardData(CF\_UNICODETEXT);

text = (LPWSTR)GlobalLock(hClipboardData);

GlobalUnlock(hClipboardData);

CloseClipboard();

EmptyClipboard();

return text;

}

**10. Создание программы с критической секцией. Программа должна содержать минимум два потока. Использование критической секции в функции потока должно быть обосновано.**

Если убрать критическую секцию, то там данные некорректно подсчитываются, это связано с тем что в один и тот же момент записывали свое значение разные потоки

#include <Windows.h>

#include <stdio.h>

int count;

HANDLE hF[2];

CRITICAL\_SECTION cs; // объявление критической секции

void thread\_func()

{

for (int i = 0; i < 100; i++)

{

EnterCriticalSection(&cs); // запуск критической секции

count++;

LeaveCriticalSection(&cs); // закрытие критической секции

Sleep(50);

}

}

int main()

{

system("chcp 1251>nul");

count = 0;

InitializeCriticalSection(&cs); // создание критической секции

hF[0] = CreateThread(NULL, 0, thread\_func, NULL, 0, 0);// создание потоков и передача в них функции

hF[1] = CreateThread(NULL, 0, thread\_func, NULL, 0, 0);

WaitForMultipleObjects(2, hF, FALSE, INFINITE);

DeleteCriticalSection(&cs); // удаление

printf("%d", count);

return 0;

}

**11. П рограмма для считывания нескольких числовых значений из строки и из записи в числовой массив (разделитель - пробел). Количество значений в строке заранее неизвестно.**

#include <stdio.h>

#include <stdlib.h>//стандартная библиотека функций, она нужна для atoi

int main()

{

system("chcp 1251>nul");//локализация на русский

printf("Введите строку\n");

char str[10000];

gets(str);

//строка, содержащая в себе числа, но в формате символов

int k = 0;//переменная, которая будет хранить количество элементов числового массива

char\* Poisk = strtok(str, " ");//strtok позволяет разделить строку на отдельные "слова" по разделителю (у нас пробел). Poisk будет хранить указатель на первое "слово"

float Massive[100];//объявляем числовой массив, в который переведем числа из символов в реальные числа

while (Poisk != NULL)//пока не закончились отдельные "слова" в строке будем делать перево

{

Massive[k] = atof(Poisk);//atoi переводит символьную строку в число, если это получится. Поскольку у нас в строке написаны цифры, перевод получится без проблем. Записываем их в массив

k++;//прибавляем счетчик элементов массива

Poisk = strtok(NULL, " ");//переходим к следующему "слову" в строке

}

for (int i = 0; i < k; i++)//цикл будет выполнятся, пока мы не выведем все элементы массива (k, посчитали выше)

{

printf(" число %f\n", Massive[i]);//вывод на экран элемента массива

}

}

12. Запись и считывание строковых параметров в системный реестр.

#include <Windows.h>

#include <stdio.h>

#include <malloc.h>

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, PSTR pCmdLine, int nCmdShow)

{

HKEY hKey;

HKEY hMyKey;

if (RegOpenKeyW(HKEY\_CURRENT\_USER, NULL, &hKey) != ERROR\_SUCCESS)

{

MessageBox(NULL, L"В работе программы возникла проблема", L"Информация", MB\_OK);

return 0;

} // выбор главной ветви реестра

if (RegCreateKeyW(hKey, L"MyKey", &hMyKey) != ERROR\_SUCCESS) // условие создания ключа

{

MessageBox(NULL, L"При создание ключа возникла ошибка", L"Информация", MB\_OK);

return 0;

}

LPWSTR str = L"Строковый параметр";

DWORD StrParamLen = wcslen(str);

if (RegSetValueExW(hMyKey, L"MyStrParam", 0, REG\_SZ, str, StrParamLen \* sizeof(WCHAR)) != ERROR\_SUCCESS) // условие создания строкового параметра

{

MessageBoxA(NULL, "При создание строкового параметра возникла ошибка", "Информация", MB\_OK);

}

LPDWORD len = 512;

LPDWORD type = NULL;

LPWSTR text = malloc(512);

if (RegGetValueW(hKey, L"MyKey", L"MyStrParam", RRF\_RT\_REG\_SZ, &type, text, &len) == ERROR\_SUCCESS) // условие для считывания с проверкой на исключение

{

MessageBoxW(NULL, text, L"Информация", MB\_OK); // передача значения из реестра в бокс

}

else

{

MessageBoxA(NULL, "При считывание данных из реестра возникла ошибка", "Информация", MB\_OK);

}

if (RegDeleteKey(hKey, L"MyKey") != ERROR\_SUCCESS)

{

MessageBoxA(NULL, "При удаление ключа возникла ошибка", "Информация", MB\_OK);

}

RegCloseKey(hMyKey);

RegCloseKey(hKey);

}

13. Обработка нажатия клавиши мыши в системе (выписать в messagebox какая клавиша нажата и сколько раз)

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <Windows.h>

#include <stdio.h>

int count;

int count1;

LRESULT CALLBACK MouseProc(int iCode, WPARAM wParam, LPARAM lParam) // метод считывания нажатия

{

if (wParam == 513)

{

count++;

LPWSTR string = (LPWSTR)calloc(100, sizeof(WCHAR));

swprintf(string, 100, L"Левая клавиша мыши нажата %d раз", count);

MessageBox(NULL, string, L"Левая кнопка мыши", MB\_OK);

free(string);

}

if (wParam == 516)

{

count1++;

LPWSTR string = (LPWSTR)calloc(100, sizeof(WCHAR));

swprintf(string, 100, L"Правая клавиша мыши нажата %d раз", count);

MessageBox(NULL, string, L"Правая кнопка мыши", MB\_OK);

free(string);

}

return CallNextHookEx(NULL, iCode, wParam, lParam);

}

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, PWSTR pCmdLine, int nCmdShow)

{

HHOOK hHook = SetWindowsHookEx(WH\_MOUSE\_LL, MouseProc, NULL, 0); // устанавливаем хук

MSG msg;

while (GetMessage(&msg, NULL, 0, 0)) // цикл для непрерывного перехвата

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

UnhookWindowsHookEx(hHook); // закрытие хука

return 0;

}

14. Использование системного буфера обмена для передачи строковых значений между процессами

Основной код:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <strsafe.h>

#include <Windows.h>

#include < stdlib.h >

LPWSTR ClipboardOutputText();

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR pCmdLine, int nCmdShow)

{

int p = 1;

while (p)

{

ClipboardInputText(L"HeLLO"); //

Sleep(1000);

STARTUPINFO si;

PROCESS\_INFORMATION pi = { 0 };

ZeroMemory(&si, sizeof(si));

si.cb = sizeof(si);

ZeroMemory(&pi, sizeof(pi)); // все что сверху это для создания процесса

if (!CreateProcessA(NULL, "C:\\Users\\Dima\\source\\repos\\Project8\\x64\\Debug\\Project1.exe", NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))// проверка на запуск процесса

{

}

WaitForSingleObject(pi.hProcess, INFINITE);

CloseHandle(pi.hProcess);

CloseHandle(pi.hThread);

p = 0;

}

return 0;

}

int ClipboardInputText(LPWSTR buffer)// метод для записи строки в буфер обмена

{

DWORD len;

HANDLE hMem;

len = wcslen(buffer) + 1;

hMem = GlobalAlloc(GMEM\_MOVEABLE, len \* sizeof(LPWSTR));

// **GlobalAlloc** выделяет глобальный блок памяти.

memcpy(GlobalLock(hMem), buffer, len \* sizeof(LPWSTR));

// Функция memcpy копирует num байтов первого блока памяти, на который ссылается указатель srcptr, во второй блок памяти, на который ссылается указатель destptr.

GlobalUnlock(hMem);

OpenClipboard(0);

EmptyClipboard(); // Функция **EmptyClipboard** очищает буфер обмена и освобождает дескрипторы

SetClipboardData(CF\_UNICODETEXT, hMem);

// Функция **SetClipboardData** помещает данные в буфер обмена в заданном формате

CloseClipboard();

return 0;

}

Получение и вывод в процессе

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <strsafe.h>

#include <Windows.h>

#include < stdlib.h >

LPWSTR ClipboardOutputText();

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR pCmdLine, int nCmdShow)

{

LPSTR\* Data = ClipboardOutputText();

MessageBox(NULL, Data, L"Информация", MB\_OK);

return 0;

}

TCHAR\* ClipboardOutputText()// метод для получения данных из буфера обмена

{

TCHAR\* Mess = NULL;

OpenClipboard(NULL); //открыть буфер обмена

HANDLE hClipboardData = GetClipboardData(CF\_UNICODETEXT); //записать в буфер обмена данные соответствующего типа

Mess = (TCHAR\*)GlobalLock(hClipboardData); //считать из глобального участка памяти, привести это все к строке

GlobalUnlock(hClipboardData); //освободить глобальные участки памяти

CloseClipboard(); //закрыть буфер обмена, сделать его доступным для других приложений

EmptyClipboard(); //очистить буфер обмена

return Mess;

}

15. Запись и считывание числовых параметров в системный реестр

#include <Windows.h>

#include <stdio.h>

#include <malloc.h>

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, PSTR pCmdLine, int nCmdShow)

{

HKEY hKey;

HKEY hMyKey;

if (RegOpenKeyW(HKEY\_CURRENT\_USER, NULL, &hKey) != ERROR\_SUCCESS)

{

MessageBox(NULL, L"В работе программы возникла проблема", L"Информация", MB\_OK);

return 0;

} // выбор главной ветви реестра

if (RegCreateKeyW(hKey, L"MyKey", &hMyKey) != ERROR\_SUCCESS) // условие создания ключа

{

MessageBox(NULL, L"При создание ключа возникла ошибка", L"Информация", MB\_OK);

return 0;

}

DWORD parametr = 220;

if (RegSetValueExW(hMyKey, L"MyParam", 0, REG\_DWORD, &parametr, 4) != ERROR\_SUCCESS) // условие создания строкового параметра

{

MessageBoxA(NULL, "При создание числового параметра возникла ошибка", "Информация", MB\_OK);

}

DWORD len = 512;

DWORD type = NULL;

DWORD param = 0;

if (RegGetValueW(hKey, L"MyKey", L"MyParam", RRF\_RT\_DWORD, &type, &param, &len) == ERROR\_SUCCESS) // условие для считывания с проверкой на исключение

{

LPWSTR OutputString = malloc(512);

swprintf(OutputString, 512, L"%d", param);

MessageBoxW(NULL, OutputString, L"Информация", MB\_OK);

}

else

{

MessageBoxA(NULL, "При считывание данных из реестра возникла ошибка", "Информация", MB\_OK);

}

if (RegDeleteKey(hKey, L"MyKey") != ERROR\_SUCCESS)

{

MessageBoxA(NULL, "При удаление ключа возникла ошибка", "Информация", MB\_OK);

}

RegCloseKey(hMyKey);

RegCloseKey(hKey);

}

16. Разработка программы-секундомера. в одном потоке идет отсчет времени, а в другом осуществляется управление секундомером

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <windows.h>

#include <stdio.h>

VOID FF(VOID);

DWORD WINAPI Timer(BOOL[2]);

VOID ChangeTime(VOID);

DWORD WINAPI Sec();

SYSTEMTIME t; // struct

int main()

{

system("chcp 1251");

system("cls");

FF();

return 0;

}

VOID FF(VOID)

{

BOOL F[2];

F[0] = TRUE;

F[1] = FALSE;

DWORD cmd = 0;

GetLocalTime(&t);

HANDLE Potok = CreateThread(NULL, 0, Timer, F, 0, 0);

while (1)

{

scanf\_s("%d", &cmd);

if (cmd == 6) {

SuspendThread(Potok);

printf("Пауза включена!\n");

}

else if (cmd == 7) {

printf("Пауза выключена!\n");

ResumeThread(Potok);

}

if (cmd == 0)

{

SuspendThread(Potok);

system("cls");

printf\_s("0-Остановка\n1-Запуск\n2-Вкл/выкл секундомер\n6-Пауза вкл\n7-Пауза выкл\n");

}

if (cmd == 1)

{

ResumeThread(Potok);

system("cls");

}

if (cmd == 2)

{

F[1] = !F[1];

if (F[1])

{

t.wHour = 0;

t.wMinute = 0;

t.wSecond = 0;

t.wMilliseconds = 0;

}

else

{

GetLocalTime(&t);

}

system("cls");

}

}

}

DWORD WINAPI Timer(BOOL F[2])

{

SYSTEMTIME staroe, novoe; // два времени для сравнения чтобы понять прошла секунда или нет

GetLocalTime(&staroe);

GetLocalTime(&novoe);

while (1)

{

GetLocalTime(&staroe);

if (F[1]) // флаг меняющий режим работы секундомера

{

if ((staroe.wMilliseconds / 100) != (novoe.wMilliseconds / 100) || novoe.wHour != staroe.wHour || novoe.wMinute != staroe.wMinute || novoe.wSecond != staroe.wSecond)

{

// проверяем отличается ли системное время со старым

system("cls");

novoe = staroe;

printf\_s("%d:%d:%d:%d\n", t.wHour, t.wMinute, t.wSecond, t.wMilliseconds); // секундомер

t.wMilliseconds++;

if (t.wMilliseconds == 10)

{

t.wMilliseconds = 0;

t.wSecond++;

}

if (t.wSecond == 60)

{

t.wSecond = 0;

t.wMinute++;

}

if (t.wMinute == 60)

{

t.wMinute = 0;

t.wHour++;

}

if (t.wHour == 24)

{

t.wHour = 0;

}

}

}

}

ExitThread(0);

}

17. Разработка программы для вычисления факториала в отдельном потоке. В программе должна быть предусмотрена валидация данных.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <Windows.h>

void FacKU(int n)

{

if (n == 0) // если факториал = 0 , возвращаем 1

return 1;

if (n < 1)

return 0;

int a = 1;

for (int i = 1; i <= n; i++)

{

a \*= i;

}

printf("%d\n", a);

}

int main()

{

HANDLE hThread = NULL;

int n;

scanf("%d", &n);

if (n != NULL)

{

hThread = CreateThread(NULL, 0, FacKU, n, NULL, NULL);// создание потока

}

WaitForSingleObject(hThread, INFINITE);

}

18. Создание и настройка проекта WinAPI. Вывод сообщения в messagebox. Текст сообщения считывается из текстового файла в кодировке Юникод

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <Windows.h>

#include <stdio.h>

int WINAPI WinMain()

{

DWORD d;

HANDLE hFile = CreateFile(L"1.txt",//путь к файлу

GENERIC\_READ | GENERIC\_WRITE,//флаги на открытие как на чтение, так и на запись

FILE\_SHARE\_READ | FILE\_SHARE\_WRITE,//совместный доступ только на чтение

NULL,//структура безопасности по умолчанию

OPEN\_ALWAYS,//режим создания файла (открыть, перезаписать и т.п.)

FILE\_ATTRIBUTE\_NORMAL,//атрибуты файла по умолчанию

NULL);//шаблон файла отсутствует

LPCSTR str = calloc(100, 1);

ReadFile(hFile, str, 100, &d, NULL);

MessageBoxA(NULL, str, NULL, MB\_OK);

}

Решение егора

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <Windows.h>

#include <stdio.h>

int WINAPI WinMain()

{

DWORD d;

HANDLE hFile = CreateFile(L"C:\\Users\\1piga\\OneDrive\\Рабочий стол\\4 курс\\1 Семестр\\Экзамен системное программирование\\Zadanie18\\x64\\Debug\\1.txt",//путь к файлу

GENERIC\_READ | GENERIC\_WRITE,//флаги на открытие как на чтение, так и на запись

FILE\_SHARE\_READ | FILE\_SHARE\_WRITE,//совместный доступ только на чтение

NULL,//структура безопасности по умолчанию

OPEN\_ALWAYS,//режим создания файла (открыть, перезаписать и т.п.)

FILE\_ATTRIBUTE\_NORMAL,//атрибуты файла по умолчанию

NULL);//шаблон файла отсутствует

LPWSTR str = calloc(100, 1);

ReadFile(hFile, str, 100, &d, NULL);

MessageBox(NULL, str, L"Содержимое файла", MB\_OK);

}

19. Программа с запуском стороннего процесса. Имя запускаемого процесса выбирается с помощью системного диалогового окна.

#include <Windows.h>

int main()

{

OPENFILENAME ofn;

WCHAR szFile[250] = { 0 };

ZeroMemory(&ofn, sizeof(ofn)); // Функция ZeroMemory предназначена для обнуления памяти

ofn.lStructSize = sizeof(ofn); // Определяет длину структуры, в байтах.

ofn.lpstrFile = szFile; // Указатель на буфер, который содержит имя файла, используемое, чтобы инициализировать поле редактирования

ofn.nMaxFile = sizeof(szFile); // определяет размер буфера, в TCHARs, указанного членом lpstrFile.

ofn.lpstrFilter = L"All\0\*.\*\0exe\0\*.EXE\0"; // Указатель на буфер, содержащий пары строк фильтров с нулевым символом в конце. Последняя строка в буфере должна быть завершена двумя символами ПУСТО (NULL).

ofn.Flags = OFN\_PATHMUSTEXIST | OFN\_FILEMUSTEXIST;

// OFN\_PATHMUSTEXIST Определяет, что пользователь может вводить с клавиатуры только допустимые пути и имена файла.

// OFN\_FILEMUSTEXIST Определяет, что пользователь может вводить с клавиатуры имена только существующих файлов в поле ввода Имя файла(File Name).

if (GetOpenFileNameW(&ofn) == TRUE)

{

LPSTARTUPINFOA sti = calloc(1, sizeof(STARTUPINFO));

LPPROCESS\_INFORMATION li = calloc(1, sizeof(PROCESS\_INFORMATION));

CreateProcessW(

ofn.lpstrFile,

NULL,

NULL,

NULL,

FALSE,

0,

NULL,

NULL,

sti,

li

);

WaitForSingleObject(li->hProcess, 1);

}

}

20. Использование условных блоков для анализа значений, возвращаемых системной функцией.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <Windows.h>

#include <stdio.h>

int main()

{

int a = 0;

if (!scanf("%d", &a)) //можно любую системную функцию

{

printf("%s", "no number");

}

else

{

printf("%s", "number");

}

}

21. Программа с импортом функции из файла DLL. Функция принимает структуру, содержащую координаты двух точек, а возвращает длину отрезка, который образуется этими точками

Основной код программы:

#include "Header.h"

#define PATH L"C:\\Users\\stron\\source\\repos\\ConsoleApplication4\\x64\\Debug\\ConsoleApplication4.dll"

typedef double(\_cdecl\* getSize)(struct cord\*);

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, PSTR pCmdLine, int nCmdShow)

{

HINSTANCE hMyDLL;

if ((hMyDLL = LoadLibrary(PATH)) == NULL) return 1;

getSize getSqize = (getSize)GetProcAddress(hMyDLL, "getSize");

cord\* cord = malloc(sizeof(cord));

cord->x1 = 2; // значение для координат

cord->y1 = 1;

cord->x2 = 6;

cord->y2 = 4;

double result = getSqize(cord);

FreeLibrary(hMyDLL);

return 0;

}

Код заголовка

#pragma once

#include <Windows.h>

#include <Math.h>

typedef struct coordinats {

int x1;

int x2;

int y1;

int y2;

}cord;

Код длл

#include "C:\Users\stron\source\repos\ConsoleApplication4\ConsoleApplication1\Header.h"

\_\_declspec(dllimport) double getSize(cord\* cord);

double getSize(cord\* cord)

{

// получаем длину отрезка

// координаты конца - координаты начала все в квадрате + тоже самое и квадрате

double d = sqrt((pow(cord->x2 - cord->x1, 2) + pow(cord->y2 - cord->y1, 2)), 2);

return d;

}

22. Отправка и получение данных из именованного канала. Данные содержатся в строковой форме. Перед отправкой следующей строки клиенту необходимо дождаться ответа от сервера.

Код сервера 43п:

#include <Stdio.h>

#include <Windows.h>

int main()

{

system("chcp 1251");

HANDLE hNamePipe;

LPSTR pipeName = L"\\\\.\\pipe\\MyPipe";

DWORD read\_buffer = 100;

LPWSTR buffer = calloc(read\_buffer, sizeof(char));

char message[100];

DWORD actual\_read;

BOOL Connected;

BOOL SuccessRead;

while (1)

{

hNamePipe = CreateNamedPipe(

pipeName,

PIPE\_ACCESS\_DUPLEX,

PIPE\_TYPE\_MESSAGE | PIPE\_READMODE\_MESSAGE | PIPE\_WAIT,

PIPE\_UNLIMITED\_INSTANCES,

512,

512,

INFINITE,

NULL);

Connected = ConnectNamedPipe(hNamePipe, NULL);

if (Connected) {

printf("\nКлиент подключился\n");

SuccessRead = ReadFile(hNamePipe, buffer, read\_buffer, &actual\_read, NULL);

if (SuccessRead) {

printf("\nКлиент пишет: ");

printf(buffer);

printf("\n");

printf("\nВведите сообщение для клиента: ");

gets(message);

buffer = &message;

WriteFile(hNamePipe, buffer, read\_buffer, &actual\_read, NULL);

}

}

else

{

printf("\nКлиент отключился от сервера");

}

CloseHandle(hNamePipe);

}

}

**Код Клиента никиты:**

#include "stdio.h"

#include "Windows.h"

#define SIZE\_BUFFER 512

int main()

{

system("chcp 1251 > nul");

LPSTR pipeName = "\\\\.\\pipe\Pipe";

BOOL flag = TRUE;

char message\_user[SIZE\_BUFFER];

DWORD size\_buf = SIZE\_BUFFER; // размер буффера для записи

DWORD actual\_read\_bite; // размер реально записано байт;

DWORD actual\_write\_bite; // размер реально записано байт;

LPWSTR buffer;

while (TRUE) {

char message\_user[SIZE\_BUFFER];

HANDLE myPIPE = CreateFile(pipeName, GENERIC\_READ | GENERIC\_WRITE, 0, NULL, OPEN\_EXISTING, 0, NULL);

DWORD dMode = PIPE\_READMODE\_MESSAGE;

BOOL isConnected = SetNamedPipeHandleState(pipeName, &dMode, NULL, NULL);

if (isConnected == FALSE) {

printf("Сервер временно недоступен");

flag = TRUE;

}

else {

printf("Соединение установлено успешно!");

Sleep(1000);

system("clear");

if (flag == TRUE) {

printf("Введите сообщение ");

gets(message\_user);

buffer = &message\_user;

WriteFile(pipeName, buffer, SIZE\_BUFFER, &actual\_write\_bite, NULL);

flag = FALSE;

}

buffer = (CHAR\*)calloc(size\_buf, sizeof(CHAR));

BOOL SuccesRead = ReadFile(pipeName, buffer, SIZE\_BUFFER, &actual\_read\_bite, NULL);

if (SuccesRead) {

printf("Ответ: \n");

printf(buffer);

printf("\n");

flag = TRUE;

}

}

Sleep(1000);

CloseHandle(pipeName);

}

}

Код сервера Никиты:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "stdio.h"

#include "Windows.h"

#include "math.h"

#define SIZE\_BUFFER 512

int main() {

float otvet;

system("chcp 1251 > nul");

LPSTR pipeName = "\\\\.\\pipe\\Pipe";

HANDLE hNamedPipe; // дескриптор канала

DWORD size\_buffer = SIZE\_BUFFER;

DWORD size\_buf = SIZE\_BUFFER; // размер буффера для записи

DWORD actual\_read\_bite; // размер реально записано байт;

DWORD actual\_write\_bite; // размер реально записано байт;

LPSTR buffer = (CHAR\*)calloc(size\_buffer, sizeof(CHAR));

BOOL Connection;

BOOL SuccesRead;

DWORD bukva=0;

DWORD zap = 0, ch = 0; // ЧИСЛА ОТ КЛИЕНТА

while (TRUE) {

hNamedPipe = CreateNamedPipeA(

pipeName, //имя канала

PIPE\_ACCESS\_DUPLEX, //режим доступа к каналу (односторонний/двусторонний)

PIPE\_TYPE\_MESSAGE | PIPE\_READMODE\_MESSAGE | PIPE\_WAIT, //режимы работы канала: передавать сообщения|читать сообщения|ждать

PIPE\_UNLIMITED\_INSTANCES, //количество водящих соединений к каналу. в данном случае неограничено

SIZE\_BUFFER, // объем буфера на чтение (байт)

SIZE\_BUFFER, // объем буфера на запись (байт)

INFINITE, // максимальное время ожидания сообщения

NULL); //указатель на структуру безопасности

Connection = ConnectNamedPipe(hNamedPipe, NULL);

if (Connection = TRUE) {

printf("Соединение с клиентом установлено успешно!");

Sleep(1000);

system("cls");

SuccesRead = ReadFile(hNamedPipe, buffer, size\_buffer, &actual\_read\_bite, NULL);

zap = 0; ch = 0, bukva = 0;

if (SuccesRead == TRUE) {

printf("\n Клиент ввел - " );

printf(buffer);

for (int i = 0; i < strlen(buffer); i++)

{

if (buffer[i] == '.') zap++; // количество точек

if (buffer[i] == '-' || buffer[i] == '1' || buffer[i] == '2' || buffer[i] == '3' || buffer[i] == '4' || buffer[i] == '5' || buffer[i] == '6' || buffer[i] == '7' || buffer[i] == '8' || buffer[i] == '9' || buffer[i] == '0')

{

ch++;

}

if (buffer[i] >= 'A' && buffer[i] <= 'Z' || buffer[i] >= 'a' && buffer[i] <= 'z' ||

buffer[i] >= 'А' && buffer[i] <= 'Я' || buffer[i] >= 'а' && buffer[i] <= 'я' ||

buffer[i] == 'Ё' || buffer[i] == 'ё') {

bukva++;

}

}

if (bukva > 0) {

sprintf(buffer, "Вы ввели букву!");

}

if (ch + zap == strlen(buffer) && zap < 2) {

// если 1.4 = 2 + 1(точка) = 3

otvet = atof(buffer);

otvet = pow(otvet, 2);

printf("\n Квадрат числа - %3.5f\n", otvet);

sprintf(buffer, "%3.5f", otvet);

}

else {

sprintf(buffer, "%s", "Невозможно возвести в степень\n");

}

WriteFile(hNamedPipe, buffer, size\_buffer, &actual\_read\_bite, NULL);

}

}

else {

printf("Клиент был отключен от сервер\n");

}

CloseHandle(hNamedPipe);

}

}

Код клиента:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <Stdio.h>

#include <Windows.h>

int main()

{

system("chcp 1251");

BOOL flag\_awser = TRUE;

char message[100];

DWORD read\_buffer = 100;

DWORD actual\_read;

DWORD actual\_write;

LPWSTR buffer = (CHAR\*)calloc(read\_buffer, sizeof(char));

HANDLE hNamePipe;

LPSTR pipeName = L"\\\\.\\pipe\\MyPipe";

BOOL SuccessRead;

BOOL isSuccess;

DWORD dwMode = PIPE\_READMODE\_MESSAGE;

while (1)

{

hNamePipe = CreateFile(pipeName, GENERIC\_READ | GENERIC\_WRITE, 0, NULL, OPEN\_EXISTING, 0, NULL);

isSuccess = SetNamedPipeHandleState(hNamePipe, &dwMode, NULL, NULL);

if (!isSuccess) {

printf("\nСервер не отвечает\n");

flag\_awser = TRUE;

}

else

{

if (flag\_awser) {

printf("Введите сообщение для сервера: \n");

gets(message);

buffer = &message;

WriteFile(hNamePipe, buffer, read\_buffer, &actual\_write, NULL);

flag\_awser = FALSE;

}

SuccessRead = ReadFile(hNamePipe, buffer, 100, &actual\_read, NULL);

if (SuccessRead) {

printf("\nСервер пишет: ");

printf(buffer);

printf("\n");

flag\_awser = TRUE;

if (buffer == NULL)

printf("Пусто");

}

}

Sleep(100);

CloseHandle(hNamePipe);

}

}

23. Создание линейного односвязного списка из n элементов. Значение элемента генерируется случайным образом. Поместить в отдельные указатели адрес минимального и максимального элемента списка. Вывести список и значения минимального и максимального элемента через указатель

#include <Windows.h>

#include <stdio.h>

#include <locale.h>

typedef struct LOS { // создаем структуру

int arg;

int min;

int max;

struct LOS\* next;

}LOS;

void GetMinMax(LOS\* los) {

LOS\* head = los; // запоминаем бошку (самый первый элемент ЛОСа)

int max;

int min = los->arg; // берем первое значение

while (los) {

if (los->next != NULL) { // если не последний элемент

int n = los->next->arg; // ищем следующий элемент

if (los->arg > n) // если текущий элемент больше чем след элемент, то текущий становится максимальным

max = los->arg;

}

if (min > los->arg) // а если текущий меньше минимального то он становится минимальным

min = los->arg;

los = los->next; // переходим к следующей структуре(элементу)

}

los = head; // заново формируем бошку

// указали для структуры минимальный и максимальный элемент

los->min = min;

los->max = max;

}

void WriteLOS(LOS\* los) {

LOS\* head = los; // передаем бошку

while (los)

{

printf("значение - %d\n", los->arg); // выводим поочередно

los = los->next;

}

los = head;

printf("min - %d; max - %d", los->min, los->max);

}

LOS\* createLOS(int count) { // генерируем рандом

LOS\* los = calloc(1, sizeof(LOS)); // выделяем память под структуру

los->arg = rand(); // заполняем начальное значением случайным

los->next = NULL; // следующее равно нулю. т.к мы не знаем что хранится у следующего элемента(вдруг вообще элемент 1)

LOS\* headCopy, \* next;

headCopy = los; // создаем

for (size\_t i = 0; i < count - 1; i++)

{

next = calloc(1, sizeof(LOS)); // выделяем память для следующей структуры для поулчения следующего элмеента

next->arg = rand(); // следующий заполянем рандомом

headCopy->next = next; // значением предыдущего ссылаемся на новую структуру

headCopy = next; // предыдущее становится следующее

}

headCopy->next = NULL; //

return los;

}

main() {

setlocale(LC\_ALL, "Rus");

LOS\* los = createLOS(10);

GetMinMax(los);

WriteLOS(los);

}

24. Программа для считывания числового значения из файла (с сохранением его в числовом формате). Исходный файл является бинарным

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <Windows.h>

#include <stdio.h>

int main()

{

FILE\* file = NULL;

int number;

file = fopen("bin.bin", "wb");

if (file == NULL) {

printf("Error opening file");

}

scanf("%d", &number);

fwrite(&number, sizeof(int), 1, file);

fclose(file);

file = fopen("bin.bin", "rb");

if (file == NULL) {

printf("Error opening file");

}

fread(&number, sizeof(int), 1, file);

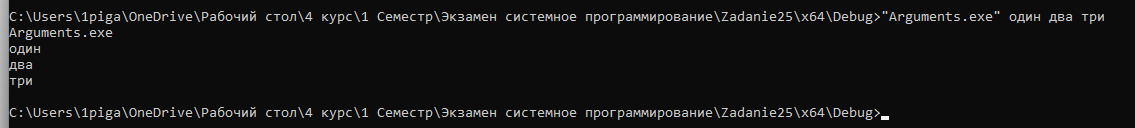
printf("%d", number);

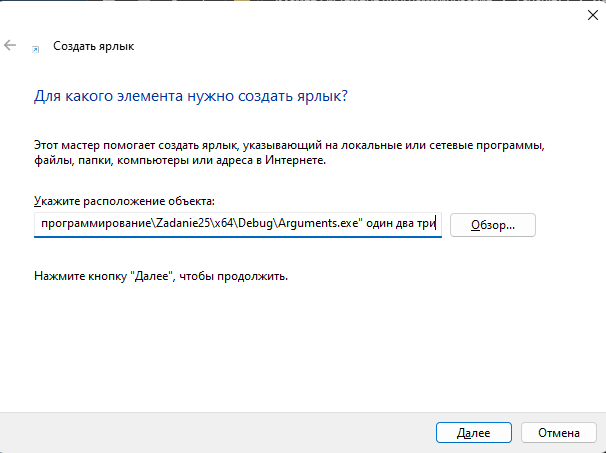
fclose(file);

}

25. Создание консольной утилиты, которая печатает список переданных ей аргументов из командной строки. При демонстрации работы программы рассмотреть не менее трех способов передачи аргументов в запускаемый процесс

Три способа создания: Ярлык(создаем ярлык, туда прописываем путь на основной проект и аршументы), дочерний процесс, cmd





Мы создаем два проекта. Один из них дочерний, в дочерний кладем:  
#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <Windows.h>

int main(int argc, char\* argv[])

{

system("chcp 1251>nul");

for (int i = 0; i < argc; i++)

{

printf("%s\n", argv[i]);

}

return 0;

}

Главный код программы второго проекта

//Подсистема - консоль

//Пример D:\Проекты\ConsoleC\Debug\ConsoleC.exe 1 2 3

#include <stdio.h>

#include <Windows.h>

#define PATH "C:\\Users\\stron\\source\\repos\\ConsoleApplication2\\Debug\\ConsoleApplication1.exe"

// ссылка на дочерний процесс

int main(int argc, char\* argv[])

{

//for (int i = 0; i < 100111; i++) это для ярлыка

//{

// printf("%d \n", i);

//}

LPSTARTUPINFO lpsi = calloc(1, sizeof(STARTUPINFO));

LPPROCESS\_INFORMATION lppi = calloc(1, sizeof(PROCESS\_INFORMATION));

char\* cmd = calloc(100, 1);

cmd = "Один два три";

CreateProcessA(

PATH,

cmd,

NULL,

NULL,

TRUE,

NULL,

NULL,

NULL,

lpsi,

lppi

);

WaitForSingleObject(lppi->hProcess, INFINITE);

return 0;

}

26. Создание функции, аргументом которой является указатель на функцию. Создать не менее двух вспомогательных функций и передать их в качестве аргумента исходной функции. Продемонстрировать специфику работы исходной функции в зависимости от переданных параметров

#include <stdio.h>

int main()

{

int (\*f)(int, int); // создаем указатель на функцию

int (\*f1)(int, int);

f = sum(5, 3);

f1 = razn(5, 3);

printf("%d", ymn(f, f1));

}

int sum(int a, int b)

{

return a + b;

}

int razn(int a, int b)

{

return a - b;

}

int ymn(int (\*f)(int, int), int (\*f1)(int, int))// передаем функции сложения и вычитания входными параметрами в функцию умножения

{

int a = f;

int b = f1;

return a \* b;

}

27. Использование анонимного канала для передачи строковых значений между процессами

Основной код программы:

#define PATH "..\\Debug\\Child.exe"

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <Windows.h>

#include <stdio.h>

#include <math.h>

int main()

{

system("chcp 1251>nul");

char\* command\_line = "Главный процесс"; // строка для передачи в дочерний процесс

SECURITY\_ATTRIBUTES sa = { sizeof(sa), NULL, TRUE }; // для вызова createprocess

HANDLE hRead, hWrite; // дескрипторы для чтения и записи

BOOL b = CreatePipe(&hRead, &hWrite, &sa, 256); // создаем анонимный канал

// переменная для дескриптора чтения (входной канал)

// переменная для дескриптора записи (выходной канал)

/ привилегии доступа

// размер буфера канала (

DWORD f;

b = WriteFile(hWrite, command\_line, 256, &f, NULL); // для помещения command\_line в канал

LPCSTR cmd = calloc(4, 1); // в одну строку помещаем дескриптор для чтения и записи

sprintf(cmd, "%d %d", hWrite, hRead); // объединяем строки hWrite, hRead для передачи в командую строку (скидываем в дочерний процесс)

LPSTARTUPINFOA sti = calloc(1, sizeof(STARTUPINFO)); // обязательные по синтаксису

**BOOL CreateProcess(**

**LPCTSTR** *lpApplicationName* **,** // имя исполняемого модуля

**LPTSTR** *lpCommandLine***,** // командная строка

**LPSECURITY\_ATTRIBUTES** *lpProcessAttributes* **,** // SD (дескриптор безопасности)

**LPSECURITY\_ATTRIBUTES** *lpThreadAttributes***,** // SD

**BOOL** *bInheritHandles***,** // дескриптор параметра наследования

**DWORD** *dwCreationFlags***,** // флажки создания

**LPVOID** *lpEnvironment***,** // новый блок конфигурации

**LPCTSTR** *lpCurrentDirectory***,** // имя текущего каталога

**LPSTARTUPINFO** *lpStartupInfo***,** // информация предустановки

**LPPROCESS\_INFORMATION** *lpProcessInformation* // информация о процессе

**);**

LPPROCESS\_INFORMATION li = calloc(1, sizeof(PROCESS\_INFORMATION));

if (!CreateProcessA(

PATH,

cmd,

NULL,

NULL,

TRUE,

0,

NULL,

NULL,

sti,

li

))

{

printf("CreateProcess failed (%d).\n", GetLastError());

return;

}

DWORD ecode; // для заполненности обязательности парамтера GetExitCodeProcess

WaitForSingleObject(li->hProcess, INFINITE);// дожидаемся дочернего процесса

GetExitCodeProcess(li->hProcess, &ecode); //  извлекает данные о состоянии завершения работы заданного процесса.

char\* str = calloc(100, 1);

DWORD d1;

BOOL l = ReadFile(hRead, str, 100, &d1, NULL); // считываем строку из child

printf("%s", str);// выводим

CloseHandle(li->hProcess); // закрываем все чтобы не было утечки

CloseHandle(li->hThread);

CloseHandle(hRead);

CloseHandle(hWrite);

return 0;

}

Второй проект: создаем с названием Child, если другое, меняем путь

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <Windows.h>

#include <stdio.h>

#include <math.h>

int main(int argc, char\* argv[])

{

HANDLE hWrite = (HANDLE)atoi(argv[0]);

HANDLE hRead = (HANDLE)atoi(argv[1]);

DWORD d1;

LPSTR buffer = calloc(256, 1);

BOOL b = ReadFile(hRead, buffer, 256, &d1, NULL); // прочитали строку

printf("%s\n", buffer); // вывели на консоль

free(buffer);

DWORD f;

char\* str = "Это дочерний процесс";

BOOL k = WriteFile(hWrite, str, strlen(str), &f, NULL); // засовываем ее в канал. и тма они взаимодействуют между дочерним и главным

return 0;

}