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 длл:

#include <Windows.h>

BOOL WINAPI DLLMain()

{

return 0;

}

\_\_declspec(dllexport) int sum(int a, int b);//разрешение на экспорт функций из длл

\_\_declspec(dllexport) int razn(int a, int b);

int sum(int a, int b)

{

return a + b;

}

int razn(int a, int b)

{

return a - b;

}

Код программы:

#include <stdio.h>

#include <Windows.h>

#define PATH L"C://Users//Dima//source//repos//Project1//x64//Debug//Project2.dll"//путь меняется

typedef int(cdecl\* sum)(int a, int b);

typedef int(cdecl\* razn)(int a, int b);// экспорт функций

int main()

{

HINSTANCE hMyDLL;

if ((hMyDLL = LoadLibrary(PATH)) == NULL) return 1;// проверка подключения к длл

sum s = (sum)GetProcAddress(hMyDLL, "sum");

razn r = (razn)GetProcAddress(hMyDLL, "razn");//вызов фунций

printf("%d\n", s(5, 2));

printf("%d", r(5, 2));

FreeLibrary(hMyDLL);// очистка

}

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 <stdio.h>

LRESULT CALLBACK KeyboardProc(int nCode, WPARAM wParam, LPARAM lParam)// метод получения клавиши

{

PKBDLLHOOKSTRUCT hHook = (PKBDLLHOOKSTRUCT)lParam; // создание хука

// Если событие - нажатие клавиши

if (nCode == HC\_ACTION && (wParam == WM\_KEYDOWN || wParam == WM\_SYSKEYDOWN))

{

KBDLLHOOKSTRUCT\* p = (KBDLLHOOKSTRUCT\*)lParam;

// Получаем название нажатой клавиши

TCHAR szKeyName[256];

DWORD iKey = MapVirtualKey(p->vkCode, NULL) << 16;// получение кода клавиши

if (!((p->vkCode <= 32))) {

iKey |= 0x1 << 24;

}

GetKeyNameText(iKey, szKeyName, 256);

// Выводим название клавиши в MessageBox

MessageBox(NULL, szKeyName, L"Key Pressed", MB\_OK);

}

return CallNextHookEx(hHook, nCode, wParam, lParam);

}

int main()

{

HHOOK hHook;

// Устанавливаем хук на обработку событий клавиатуры

hHook = SetWindowsHookEx(WH\_KEYBOARD\_LL, KeyboardProc, NULL, 0);

// Обрабатываем сообщения окна

MSG msg;

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

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

// Удаляем хук

UnhookWindowsHookEx(hHook);

return 0;

}

4. Получение кода клавиши в формате ASCII и использованием Windows HOOK. Название клавиши выводить в MessageBox

#include <Windows.h>

#include <stdio.h>

LRESULT CALLBACK KeyboardProc(int nCode, WPARAM wParam, LPARAM lParam)

{

PKBDLLHOOKSTRUCT hHook = (PKBDLLHOOKSTRUCT)lParam;

// Если событие - нажатие клавиши

if (nCode == HC\_ACTION && (wParam == WM\_KEYDOWN || wParam == WM\_SYSKEYDOWN))

{

KBDLLHOOKSTRUCT\* p = (KBDLLHOOKSTRUCT\*)lParam;

// Получаем название нажатой клавиши

TCHAR szKeyName[256];

int iKey = MapVirtualKeyA(p->vkCode, NULL);//получение кода клавиши

WCHAR procID[256];

swprintf\_s(procID, 256, L"%d", iKey);

MessageBox(NULL, procID, L"Key Pressed", MB\_OK);

}

return CallNextHookEx(hHook, nCode, wParam, lParam);

}

int main()

{

HHOOK hHook;

// Устанавливаем хук на обработку событий клавиатуры

hHook = SetWindowsHookEx(WH\_KEYBOARD\_LL, KeyboardProc, NULL, 0);

// Обрабатываем сообщения окна

MSG msg;

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

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

// Удаляем хук

UnhookWindowsHookEx(hHook);

return 0;

}

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

#include <Windows.h>

#include <stdio.h>

char\* int\_to\_string(int n)

{

int nDigits = 1; // счетчик разрядов

// Определяем, сколько разрядов имеет число

int new\_n = n;

while (new\_n / 10 != 0)// считаем разряды

{

nDigits++;

new\_n /= 10;

}

char\* s = "";// задаем переменную в которую записывать будем

int i = 0;

if (n < 0) {

n = -n;

nDigits++;

}

s = malloc(nDigits);// выделение памяти для переменной

for (i; i < nDigits; i++)

{

s[i] = n % 10 + '0';

if (n == 0) {

s[i] = '-';

}

else {

n /= 10;

}

}

s[nDigits] = '\0';

int j;

char c;

for (i = 0, j = nDigits - 1; i < j; i++, j--) {

c = s[i];

s[i] = s[j];

s[j] = c;

}

return s;// программа не выдает ничего в консоль, смотреть по точке остановы на этой строчке

}

char\* Concat(char\* str1, char\* str2) {

int count = 0;// счетчик символов первой переменной

while (str1[count] != '\0')// считает количество символов первой переменно

count++;

int count2 = 0;// счетчик символов второй переменной

while (str2[count2] != '\0')// считает количество символов второй переменной

count2++;

count += count2;

char\* result = malloc(count + 1);// выделение памяти под склеенную строчку

result[count] = '\0';// завершение строчки

count2 = 0;

while (str1[count2] != '\0')// цикл записи первой строчки в новую переменную

{

result[count2] = str1[count2];

count2++;

}

count = 0;

while (str2[count] != '\0')// цикл записи второй строчки в новую переменную

{

result[count2] = str2[count];

count++;

count2++;

}

return result;// смотреть результат по точке остановы на этой строке

}

int main()

{

int\_to\_string(9);

Concat("dfb", "sbvdb");

}

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()

{

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

us->a = 10;

us->b = 2;

printf("%d\n", sum(us));

}

Заголовочный:

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <malloc.h>

#include <Windows.h>

#include "Struct.c"

#include "Func.c"

Функция:

sum(struct s\* us)

{

return us.a + us.b;// передаем параметры структуры в переменные

}// надо точки ставить, они потом превратятся в стрелки

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>

int GetLevels(int a, int n)// функция для добавления нулей в соответствии с их количеством

{

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

{

a \*= 10;

}

return a;

}

int string\_to\_int(LPWSTR str) // основная функция{

int result = 0; //ответ

int is\_negative = 0;// проверка на минус

int i = 0;

int count = 0;

if (str[i] == L'-') // проверка на минус{

is\_negative = 1;

i++;

}

int index = i;

while (str[index] != '\0') // счет количества символов в строке

{

count++;

index++;

}

while (str[i] != L'\0') {

if (str[i] >= L'0' || str[i] <= L'9') // проверка, число ли это{

result += (int)(str[i] - '0') \* (GetLevels(10, count) / 10);// посимвольно переводит строку в число

count--;

}

i++;

}

if (is\_negative) // если отрицательное, меняет знак на -{

result = -result;

}

return result; // смотреть результат по этой строке по точке остановы

}

int main()

{

string\_to\_int(L"-325");

}

8. Программа для записи строки в системный буфер обмена

#include <Windows.h>

#include <stdio.h>

void CBInput(LPWSTR text)

{

HANDLE hMem = GlobalAlloc(GMEM\_MOVEABLE, (wcslen(text) + 1) \* sizeof(LPWSTR)); // выделение памяти под результат

memcpy(GlobalLock(hMem), text, (wcslen(text) + 1) \* sizeof(LPWSTR));

GlobalUnlock(hMem);

OpenClipboard(NULL);

EmptyClipboard();

SetClipboardData(CF\_UNICODETEXT, hMem);// занесение строки в буфер обмена

CloseClipboard();

}

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

{

CBInput(L"yu,yu,");// передаем строку, которая попадет в буфер

}

9. Программа для считывания данных из системного буфера обмена

#include <Windows.h>

#include <stdio.h>

LPWSTR CBOuput()

{

LPWSTR text = NULL; // куда запишется результат

OpenClipboard(NULL);

HANDLE CBtext = GetClipboardData(CF\_UNICODETEXT); // метод получения содержимого из буфера обмена

text = (LPWSTR)GlobalLock(CBtext); // приравнивание содержимого к переменной

GlobalUnlock(CBtext);

CloseClipboard();

return text; // здесь по точке остановы можно посмотреть результат

}

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

{

CBOuput();

}

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

#include <Windows.h>

#include <stdio.h>

CRITICAL\_SECTION cs; // объявление критической секции

int count = 0;

DWORD WINAPI thread\_func(int thread\_num) {

for (int i = 0; i < 10; i++) {

EnterCriticalSection(&cs); // запуск критической секции

printf("Thread %d: %d\n", thread\_num, count);

count++;

LeaveCriticalSection(&cs); // закрытие критической секции

Sleep(100);

}

return 0;

}

int main() {

InitializeCriticalSection(&cs); // создание критической секции

HANDLE thread1 = CreateThread(NULL, 0, thread\_func, 1, 0, NULL);// создание потоков и передача в них функции

HANDLE thread2 = CreateThread(NULL, 0, thread\_func, 2, 0, NULL);

WaitForSingleObject(thread1, INFINITE); // не заканчивать работу потоков, пока не закончится работа программы

WaitForSingleObject(thread2, INFINITE);

DeleteCriticalSection(&cs); // удаление

return 0;

}

11. Программа 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 <Windows.h>

int pow(int a, int b) //

{

for (size\_t i = 0; i < b; i++)

{

a \*= 10;

}

return a;

}

int main()

{

LPSTR str = "10 52 654";

int index = 0;

int count = 0;

int countall = 0;

while (str[index] != L'\0')

{

if (str[index] == L' ')

{

count++;

}

countall++;

index++;

}

index = 0;

int\* mass = calloc(count + 1, sizeof(int));

int\* countt = calloc(countall, sizeof(int));

int\* min = calloc(count + 1, sizeof(int));

int ind = 0;

for (int i = 0; i < count + 1; i++)

{

countt[i] = 0;

}

while (str[index] != L'\0')

{

while (str[index] != L' ')

{

if (str[index] == L'\0')

{

break;

}

countt[ind] += 1;

if (str[index] == L'-')

{

min[ind] = 1;

}

index++;

}

ind++;

index++;

}

index = 0;

int ii = 0;

int iii = 0;

int pr = 0;

for (int i = 0; i < count + 1; i++)

{

if (str[index] != L' ')

{

iii = countt[i];

while (countt[i] > ii)

{

if (str[index] != L'-')

{

pr = (int)(str[index] - '0');

mass[i] += pow(pr, iii) / 10;

ii++;

index++;

iii--;

}

else

{

ii++;

index++;

iii--;

}

}

ii = 0;

}

else

{

i--;

}

index++;

}

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

{

if (min[i] == 1)

{

mass[i] = -mass[i];

}

}

for (int i = 0; i < count + 1; i++)

{

printf("%d\n", mass[i]);

}

}

12. Запись и считывание строковых параметров в системный реестр.

#include <Windows.h>

#include <stdio.h>

int main()

{

HKEY hkey;

HKEY hMyKey;

RegOpenKeyW(HKEY\_CURRENT\_USER, NULL, &hkey); // выбор главной ветви реестра

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

{

if (RegSetValueEx(hMyKey, L"test", 0, REG\_SZ, L"Message", 8 \* sizeof(WCHAR)) == ERROR\_SUCCESS) // условие создания строкового параметра

{

WCHAR text[256];

DWORD size = sizeof(WCHAR) \* 256; // выделение памяти под считывание

if (RegGetValueW(hMyKey, NULL, L"test", RRF\_RT\_REG\_SZ, NULL, text, &size) == ERROR\_SUCCESS) // условие для считывания с проверкой на исключение

{

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

}

}

}

}

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

#include <Windows.h>

#include <stdio.h>

HHOOK hHook = NULL;

int count;

int count1;

WPARAM ButtonCode;

LPWSTR str[256];

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

{

if (nCode == HC\_ACTION) // проверка на нажатие мыши

{

switch (wParam)

{

case WM\_LBUTTONDOWN:// обработка левой кнопки мыши

if (ButtonCode != wParam)

ButtonCode = wParam;

count++;

swprintf\_s(str, 256, L"Левая клавиша мыши нажата %d раз", count);

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

break;

case WM\_RBUTTONDOWN: // обработка правой кнопки мыши

if (ButtonCode != wParam)

ButtonCode = wParam;

count1++;

swprintf\_s(str, 256, L"Правая клавиша мыши нажата %d раз", count1);

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

break;

}

}

return CallNextHookEx(hHook, nCode, wParam, lParam);

}

int main()

{

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));

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

GlobalUnlock(hMem);

OpenClipboard(0);

EmptyClipboard();

SetClipboardData(CF\_UNICODETEXT, hMem);

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)

{

int p = 1;

while (p)

{

LPSTR\* Data = ClipboardOutputText();

MessageBoxW(NULL, Data, NULL, MB\_OK | MB\_ICONWARNING);

Sleep(1000);

p = 0;

}

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

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <Windows.h>

#include <stdio.h>

#include <time.h>

HANDLE hConsole;

main()

{

DWORD st = 100;

DWORD size = sizeof(st);

HKEY hMyKey;

if (RegCreateKeyW(HKEY\_CURRENT\_USER, L"NewMyKey", &hMyKey) == ERROR\_SUCCESS)

{

if (RegSetValueEx(hMyKey, L"nameparamdword", 0, REG\_DWORD, (const BYTE\*)&st, sizeof(st)) == ERROR\_SUCCESS)// то же что и в 12, только с числом (reg dword)

{

if (RegGetValueW(hMyKey, NULL, L"nameparamdword", RRF\_RT\_DWORD, NULL, (LPBYTE)&st, &size) == ERROR\_SUCCESS)

{

WCHAR buf[256];

swprintf\_s(buf, 256, L"%d", st);

MessageBox(NULL, buf, L"Оповещение", MB\_OK);

}

}

}

}

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

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <Windows.h>

int times(long int t)

{

int sec = t;

int min = 0;

int hours = 0;

while (1)

{

Sleep(1000);

sec++;

if (sec == 60)

{

min++;

sec = 0;

if (min == 60)

{

hours++;

min = 0;

if (hours == 24)

{

hours = 0;

}

}

}

system("cls");

printf\_s("%d:%d:%d\n", hours, min, sec);

}

}

int main()

{

system("chcp 1251 >null");

DWORD choose = -1;

HANDLE hTread[1];

printf("\n1 - секундомер\n0 - Остановить\n");

while (1)

{

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

switch (choose)

{

case 1:

hTread[0] = CreateThread(NULL, 0, times, 0, NULL, NULL);// запускаем поток с секуномером

break;

case 0:

SuspendThread(hTread[0]);// завершаем

break;

default:

break;

}

}

WaitForSingleObject(hTread, INFINITE);

}

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>

int FacKU(int n) {

if (n == 0)// проверка

return 1;

if (n < 1)

return 0;

int a = 1;

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

{

a \*= i;

}

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

return a;

}

main() {

HANDLE hThread;

while (1) {

char str[256];

scanf("%s", str);

int n = atoi(str);

if (n != NULL) {

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

}

}

}

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);

}

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

[#define](https://vk.com/im?sel=273775725&st=%23define) \_CRT\_SECURE\_NO\_WARNINGS   
  
[#include](https://vk.com/im?sel=273775725&st=%23include) <windows.h>   
[#include](https://vk.com/im?sel=273775725&st=%23include) <Commdlg.h>   
  
OPENFILENAME ofn;   
  
  
// a another memory buffer to contain the file name   
char szFile[100];   
  
  
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)   
{   
  
// open a file name   
ZeroMemory(&ofn, sizeof(ofn));   
ofn.lStructSize = sizeof(ofn);   
ofn.hwndOwner = NULL;   
ofn.lpstrFile = szFile;   
ofn.lpstrFile[0] = '\0';   
ofn.nMaxFile = sizeof(szFile);   
//ofn.lpstrFilter = "All\0\*.\*\0Text\0\*.TXT\0";   
ofn.nFilterIndex = 1;   
ofn.lpstrFileTitle = NULL;   
ofn.nMaxFileTitle = 0;   
ofn.lpstrInitialDir = NULL;   
ofn.Flags = OFN\_PATHMUSTEXIST | OFN\_FILEMUSTEXIST;   
  
GetOpenFileName(&ofn);   
  
  
char buffer[500];   
wcstombs(buffer, ofn.lpstrFile, 500);   
  
  
LPSTARTUPINFO lp = calloc(1, sizeof(STARTUPINFO));   
LPPROCESS\_INFORMATION lpi = calloc(1, sizeof(PROCESS\_INFORMATION));   
  
CreateProcessA(   
buffer,   
" ",   
NULL,   
NULL,   
TRUE,   
NULL,   
NULL,   
NULL,   
lp,   
lpi   
);   
  
WaitForSingleObject(lpi->hProcess, INFINITE);   
  
return 0;   
}

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:

#define PATH L"DLLCode.dll"

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

typedef struct coordinats {

COORD point1;

COORD point2;

}cord;

int main()

{

HINSTANCE hMyDLL;

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

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

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

cord->point1.X = 2;

cord->point1.Y = 1;

cord->point2.X = 6;

cord->point2.Y = 4;

double result = getSqize(cord);

FreeLibrary(hMyDLL);

return 0;

}

Код длл

#include <Windows.h>

typedef struct coordinats {

COORD point1;

COORD point2;

}cord;

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

BOOL WINAPI DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)

{

switch (fdwReason)

{

case DLL\_PROCESS\_ATTACH:

break;

case DLL\_THREAD\_ATTACH:

break;

case DLL\_THREAD\_DETACH:

break;

case DLL\_PROCESS\_DETACH:

break;

}

return TRUE;

}

double getSize(cord\* cord)

{

double d = sqrt((pow(cord->point2.X - cord->point1.X, 2) + pow(cord->point2.Y - cord->point1.Y, 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:

#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:

#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 {

int arg;

int min;

int max;

struct LOS\* next;

}LOS;

void GetMinMax(LOS\* los) {

LOS\* head = los;

int max;

int min = los->arg;

while (los) {

if (los->next != NULL) {

int n = los->next->arg;

if (los->arg > n)

max = los->arg;

}

if (min > los->arg)

min = los->arg;

los = los->next;

}

los = head;

los->min = min;

los->max = max;

}

void WriteLOS(LOS\* los) {

LOS\* head = los;

while (los)

{

printf("значение - %d\n", los->arg);

los = los->next;

}

los = head;

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

}

LOS\* createLOS(int count) {

LOS\* los = calloc(1, sizeof(LOS));

los->arg = rand();

los->next = NULL;

LOS\* headCopy, \* next;

headCopy = los;

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

{

next = calloc(1, sizeof(LOS));

next->arg = rand();

headCopy->next = next;

headCopy = next;

}

headCopy->next = NULL;

return los;

}

main() {

setlocale(LC\_ALL, "Rus");

LOS\* los = createLOS(5);

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

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 \_CRT\_SECURE\_NO\_WARNINGS

#include <Windows.h>

#include <stdio.h>

int main()

{

HANDLE hReadPipe, hWritePipe;

char message[] = "Hello, child process!";

char buffer[256];

SECURITY\_ATTRIBUTES sa = { sizeof(sa),NULL,TRUE };

DWORD bytesWritten, bytesRead;

BOOL success;

// Создаем анонимный канал

success = CreatePipe(&hReadPipe, &hWritePipe, &sa, 256);

if (!success)

{

printf("Error creating pipe\n");

return 1;

}

// Создаем дочерний процесс

STARTUPINFO si;

ZeroMemory(&si, sizeof(si));

si.cb = sizeof(si);

si.hStdError = hWritePipe;

si.hStdOutput = hWritePipe;

si.dwFlags |= STARTF\_USESTDHANDLES;

PROCESS\_INFORMATION pi;

ZeroMemory(&pi, sizeof(pi));

success = CreateProcessA(NULL, "..\\Debug\\Child.exe", NULL, NULL, TRUE, CREATE\_NEW\_CONSOLE, NULL, NULL, &si, &pi);

if (!success)

{

printf("Error creating child process\n");

return 1;

}

while (TRUE) {

// Читаем из канала

success = ReadFile(hReadPipe, buffer, sizeof(buffer), &bytesRead, NULL);

if (!success)

{

printf("Error reading from pipe\n");

return 1;

}

buffer[bytesRead] = '\0';

printf("Received message: %s\n", buffer);

}

}

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

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <Windows.h>

#include <stdio.h>

int main()

{

HANDLE hReadPipe = GetStdHandle(STD\_OUTPUT\_HANDLE);

while (TRUE)

{

char message[256];

DWORD bytesWritten;

scanf("%s", message);

WriteFile(hReadPipe, message, strlen(message), &bytesWritten, NULL);

}

return 0;

}