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 WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)

{

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

DWORD bytes;//типо счетчик реально прочитанных символов

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

{

char\* str = "Сюда записана строка";

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

CloseHandle(file);

}

HANDLE fileRead = 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

bytes1;//типо счетчик реально прочитанных символов

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

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

{

ReadFile(fileRead, text, size, &bytes1, NULL);//читаем

CloseHandle(fileRead);

}

MessageBoxA(NULL, text, "Прочитанное сообщение", MB\_OK);

return 0;

}

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);

}

}

}

4. Получение кода клавиши в формате ASCII и использованием 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 = MapVirtualKeyA(pHook->vkCode, NULL) << 16;

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

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

{

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 <stdio.h>

#include <malloc.h>

char\* ConvertToString(int a);

char\* unification(char\* a, char\* b);

int main()

{

system("chcp 1251>null");

int num = 1243;

char\* str = ConvertToString(num);

char\* s = "Строка: ";

char\* result = unification(s, str);

printf("%s", result);

return 0;

}

// Функция преобразования числа в строку

char\* ConvertToString(int a)

{

int i = 0, k, l1 = 0;

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

if (a < 0)

{

sign = 1;

a = -a;

l1++;

}

int n1 = a;

while (a > 0) // Определение количества элементов для строки

{

a = a / 10;

l1++;

}

char\* str1 = calloc(l1, sizeof(char));

a = n1;

while (a > 0)

{

k = a % 10;

a = a / 10;

str1[i] = k + '0';

i++;

}

if (sign == 1)

{

str1[i] = '-';

i++;

}

i--;

char\* str2 = calloc(l1, sizeof(char));

while (i >= 0)

{

str2[a] = str1[i];

i--;

a++;

}

str2[a] = '\0';

return(str2);

}

// Соединение двух строк в одну

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

{

int l1 = 0, l2 = 0;

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

{

l1++;

}

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

{

l2++;

}

char\* c = calloc(l1 + l2, sizeof(char));

for (int i = 0; i < l1; i++) {

c[i] = a[i];

}

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

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

}

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

return c;

}

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);

int a = 4 + num;

}

// 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.

#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. Программа для считывания нескольких числовых значений из строки и из записи в числовой массив (разделитель - пробел). Количество значений в строке заранее неизвестно.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <Windows.h>

#include <stdio.h>

#include <math.h>

int main()

{

char str[] = "15 20 41 770 1 4";

char\* istr;

int i = 0;

int k = 0;

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

{

if (str[i] == ' ')

{

k++;

}

i++;

}

float \*massive = calloc(k + 1, sizeof(float));

istr = strtok(str, " ");

int j = 0;

while (istr != NULL)

{

massive[j] = atof(istr);

j++;

istr = strtok(NULL, " ");

}

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

{

printf("%g\n", massive[i]);

}

return 0;

}

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));

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)

{

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 <stdio.h>

#include <Windows.h>

int hour = 0;

int minute = 0;

int second = 0;

void WorkHours() // Поток имитирующий работу часов

{

while (1)

{

second++;

if (second == 60)

{

minute += 1;

second = 0;

}

if (minute == 60)

{

minute = 0;

hour += 1;

}

if (hour == 24)

{

hour = 0;

}

Sleep(1000);

}

}

void ShowHours() // Вывод часов

{

while (1)

{

unsigned int start\_time = clock();

system("cls");

printf("%d:%d:%d\n", hour, minute, second);

unsigned int end\_time = clock();

unsigned int search\_time = end\_time - start\_time;

Sleep(1000 - search\_time);

}

}

int main()

{

system("chcp 1251 >null");

HANDLE hThread[2];

system("chcp 1251>nul");

hThread[0] = CreateThread(NULL, 0, WorkHours, NULL, 0, 0);

hThread[1] = CreateThread(NULL, 0, ShowHours, NULL, 0, 0);

int command;

while (1)

{

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

switch (command)

{

case 1:

printf("Пауза выключена\n");

ResumeThread(hThread[0]);

ResumeThread(hThread[1]);

break;

case 2:

printf("Включена пауза\n");

SuspendThread(hThread[0]);

SuspendThread(hThread[1]);

break;

default:

CloseHandle(hThread[0]);

CloseHandle(hThread[1]);

break;

}

}

}

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)

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"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));

ofn.lStructSize = sizeof(ofn);

ofn.lpstrFile = szFile;

ofn.nMaxFile = sizeof(szFile);

ofn.lpstrFilter = L"All\0\*.\*\0exe\0\*.EXE\0";

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

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\\1piga\\OneDrive\\Рабочий стол\\4 курс\\1 Семестр\\Экзамен системное программирование\\Zadanie21\\Debug\\DLLCode.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;

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 "..//Zadanie21/Header.h"

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

double getSize(cord\* cord)

{

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:

#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;

struct LOS\* next;

}LOS;

void GetMinMax(LOS\* los, int \*min1, int \*max1) {

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;

\*min1 = min;

\*max1 = max;

}

void WriteLOS(LOS\* los, int \*min1, int \*max1) {

LOS\* head = los;

while (los)

{

printf("%d\n", los->arg);

los = los->next;

}

los = head;

printf("min - %d; max - %d", \*min1, \*max1);

}

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() {

system("chcp 1251>nul");

LOS\* los = createLOS(5);

int\* min = 0;

int\* max = 0;

GetMinMax(los, &min, &max);

WriteLOS(los, &min, &max);

}

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

Дочерний процесс

#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;

}

Первый способ:

#include <Windows.h>

#define PATH "..\\x64\\Debug\\Arguments.exe"

int main()

{

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

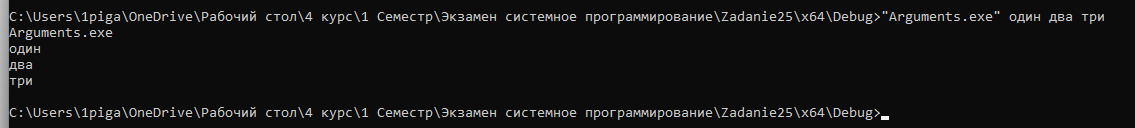
);

WaitForSingleObject(lppi->hProcess, INFINITE);

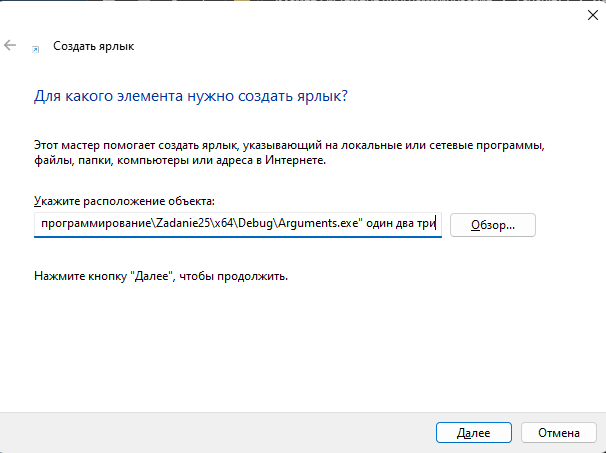
return 0;

}

Второй способ



Тритий способ



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";

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

HANDLE hRead, hWrite;

BOOL b = CreatePipe(&hRead, &hWrite, &sa, 256);

DWORD f;

b = WriteFile(hWrite, command\_line, 256, &f, NULL);

LPCSTR cmd = calloc(4, 1);

sprintf(cmd, "%d %d", hWrite, hRead);

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

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;

WaitForSingleObject(li->hProcess, INFINITE);

GetExitCodeProcess(li->hProcess, &ecode);

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

DWORD d1;

BOOL l = ReadFile(hRead, str, 100, &d1, NULL);

printf("%s", str);

CloseHandle(li->hProcess);

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);

printf("%s\n", buffer);

free(buffer);

DWORD f;

char\* str = "Это дочерний процесс";

BOOL k = WriteFile(hWrite, str, strlen(str), &f, NULL);

return 0;

}