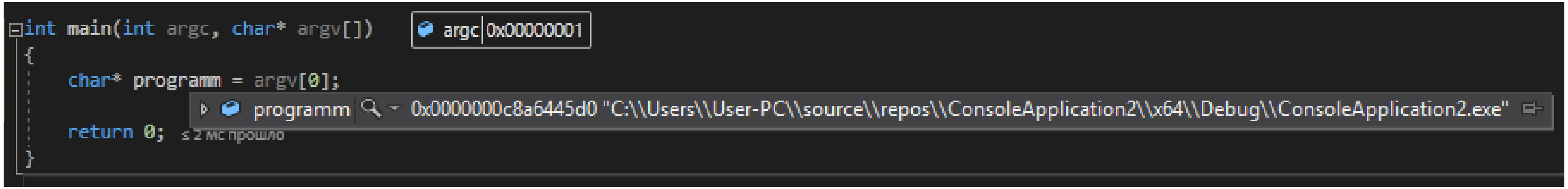
СИСТЕМНОЕ ПРОГРАММИРОВАНИЕ

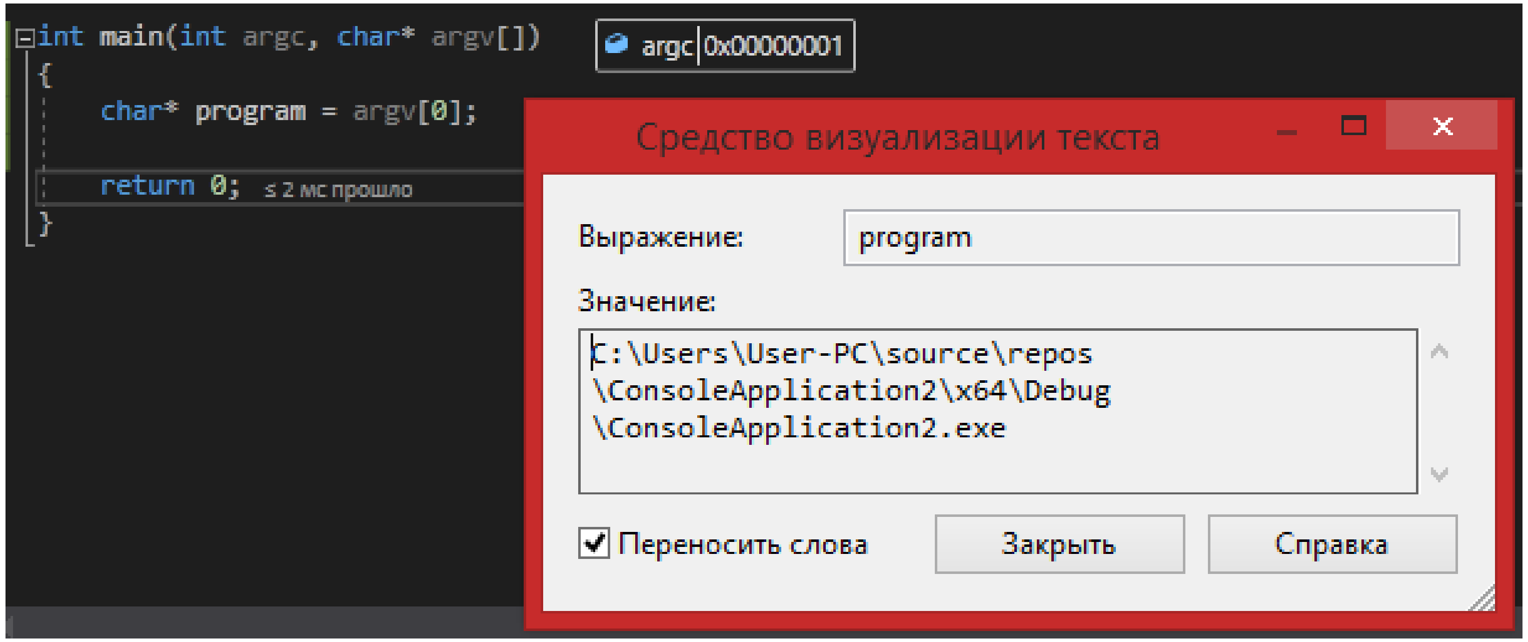
ЛЕКЦИЯ № 5

ПРЕПОДАВАТЕЛЬ: ХУСТОЧКА А.В.

ПЕРЕДАЧА АРГУМЕНТОВ В ПРИЛОЖЕНИЕ ЧЕРЕЗ КОНСОЛЬ



```
int main(int argc, char* argv[])    argc 0x00000001
{
    char* programm = argv[0];
      programm  0x0000000c8a6445d0 "C:\\Users\\User-PC\\source\\repos\\ConsoleApplication2\\x64\\Debug\\ConsoleApplication2.exe"
    return 0;  ≤ 2 мс прошло
}
```



```
int main(int argc, char* argv[])    argc 0x00000001
{
    char* program = argv[0];

    return 0;  ≤ 2 мс прошло
}
```

Средство визуализации текста

Выражение: program

Значение:

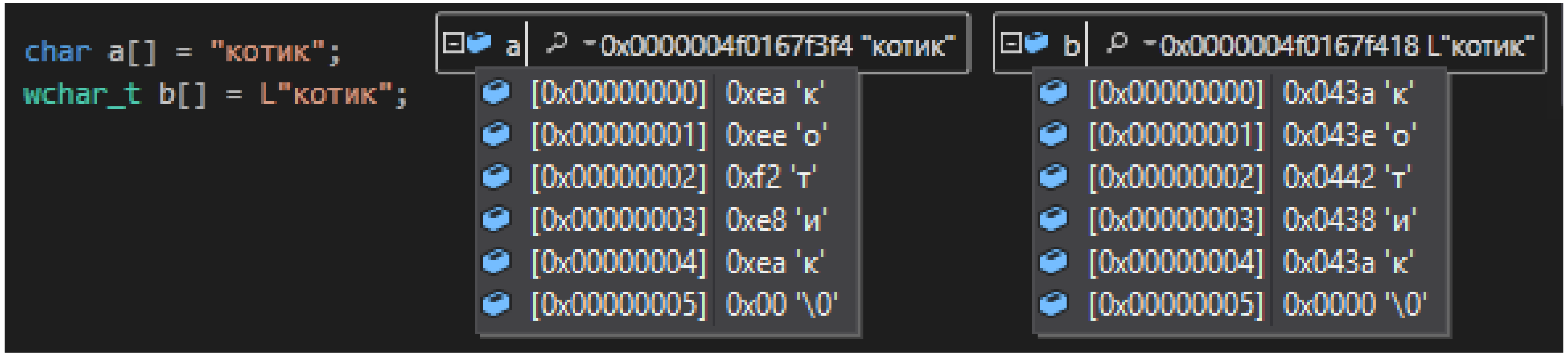
```
C:\Users\User-PC\source\repos
\ConsoleApplication2\x64\Debug
\ConsoleApplication2.exe
```

ФУНКЦИИ С ПЕРЕМЕННЫМ КОЛИЧЕСТВО АРГУМЕНТОВ

```c
#include <stdarg.h>

int sum(int n, ...)
{
    int result = 0;
    va_list arguments;
    va_start(arguments, n);
    for (int i = 0; i < n; i++) {
        result += va_arg(arguments, int);
    }
    va_end(arguments);
    return result;
}

int main(int argc, char* argv[])
{
    int sum1 = sum(3, 5, 10, 15);
    return 0;
}
```

```c
void print_say(int n, ...)
{
    va_list arguments;
    va_start(arguments, n);
    for (int i = 0; i < n; i++) {
      if (i % 2 != 0) printf("says ");
      printf("%s ", va_arg(arguments, char*));
    }
    va_end(arguments);
}

int main(int argc, char* argv[])
{
    print_say(2, "kotik", "meow");
    return 0;
}
```

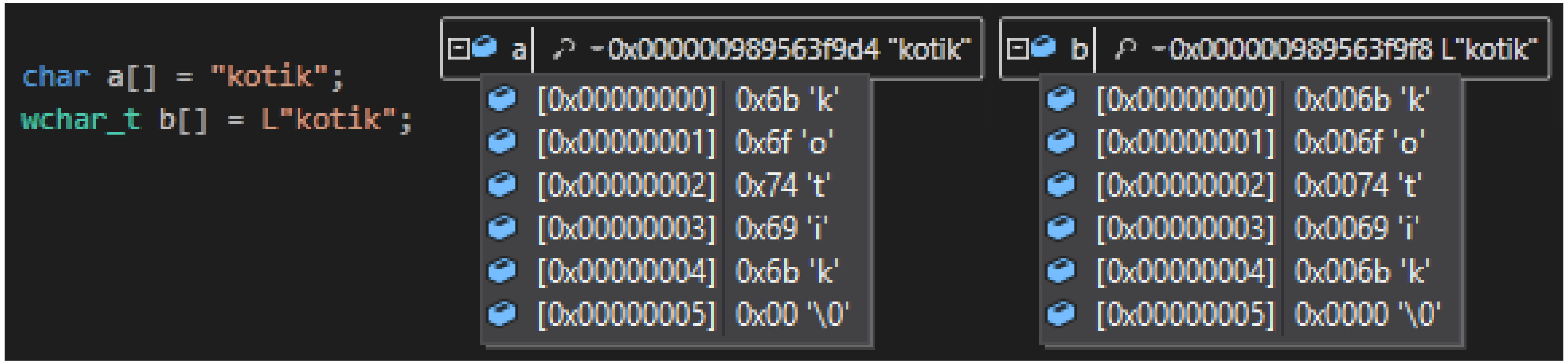
WCHAR И CHAR



```
char a[] = "котик";
wchar_t b[] = L"котик";
```

```
a  0x0000004f0167f3f4 "котик"
[0x00000000] 0xea 'к'
[0x00000001] 0xee 'о'
[0x00000002] 0xf2 'т'
[0x00000003] 0xe8 'и'
[0x00000004] 0xea 'к'
[0x00000005] 0x00 '\0'
```

```
b  0x0000004f0167f418 L"котик"
[0x00000000] 0x043a 'к'
[0x00000001] 0x043e 'о'
[0x00000002] 0x0442 'т'
[0x00000003] 0x0438 'и'
[0x00000004] 0x043a 'к'
[0x00000005] 0x0000 '\0'
```



```
char a[] = "kotik";
wchar_t b[] = L"kotik";
```

```
a  0x000000989563f9d4 "kotik"
[0x00000000] 0x6b 'k'
[0x00000001] 0x6f 'o'
[0x00000002] 0x74 't'
[0x00000003] 0x69 'i'
[0x00000004] 0x6b 'k'
[0x00000005] 0x00 '\0'
```

```
b  0x000000989563f9f8 L"kotik"
[0x00000000] 0x006b 'k'
[0x00000001] 0x006f 'o'
[0x00000002] 0x0074 't'
[0x00000003] 0x0069 'i'
[0x00000004] 0x006b 'k'
[0x00000005] 0x0000 '\0'
```

CHAR TO WCHAR | WCHAR TO CHAR

```
int
WINAPI
MultiByteToWideChar(
    _In_ UINT CodePage,
    _In_ DWORD dwFlags,
    _In_NLS_string_(cbMultiByte) LPCCH lpMultiByteStr,
    _In_ int cbMultiByte,
    _Out_writes_to_opt_(cchWideChar,return) LPWSTR lpWideCharStr,
    _In_ int cchWideChar
    );

int
WINAPI
WideCharToMultiByte(
    _In_ UINT CodePage,
    _In_ DWORD dwFlags,
    _In_NLS_string_(cchWideChar) LPCWCH lpWideCharStr,
    _In_ int cchWideChar,
    _Out_writes_bytes_to_opt_(cbMultiByte,return) LPSTR lpMultiByteStr,
    _In_ int cbMultiByte,
    _In_opt_ LPCCH lpDefaultChar,
    _Out_opt_ LPBOOL lpUsedDefaultChar
    );
```

Документация:
MultiByteToWideChar
WideCharToMultiByte

ПРИМЕР

```
wchar_t* CharToWchar(char* str)
{
    wchar_t* wstr = NULL;
    int size = MultiByteToWideChar(CP_ACP, 0, str, -1, NULL, 0);
    wstr = (wchar_t*)malloc(size * sizeof(wchar_t));
    MultiByteToWideChar(CP_ACP, 0, str, -1, wstr, size * sizeof(wchar_t));
    return wstr;
}

char* WcharToChar(wchar_t* wstr)
{
    char* str = NULL;
    int size = WideCharToMultiByte(CP_ACP, 0, wstr, -1, NULL, 0, NULL, NULL);
    str = (char*)malloc(size * sizeof(char));
    WideCharToMultiByte(CP_ACP, 0, wstr, -1, str, size * sizeof(char), NULL, NULL);
    return str;
}
```

ФУНКЦИИ ДЛЯ БАЗОВОЙ РАБОТОЙ С ФАЙЛАМИ В ОС WINDOWS

```
WINBASEAPI
HANDLE
WINAPI
CreateFileW(
    _In_ LPCWSTR lpFileName,
    _In_ DWORD dwDesiredAccess,
    _In_ DWORD dwShareMode,
    _In_opt_ LPSECURITY_ATTRIBUTES lpSecurityAttributes,
    _In_ DWORD dwCreationDisposition,
    _In_ DWORD dwFlagsAndAttributes,
    _In_opt_ HANDLE hTemplateFile
    );
```

```
WINBASEAPI
BOOL
WINAPI
CloseHandle(
    _In_ _Post_ptr_invalid_ HANDLE hObject
    );
```

Документация:
CreateFileW
CloseHandle

ПРИМЕР

```
WCHAR* name = L"C:\\file.txt";
HANDLE file = INVALID_HANDLE_VALUE;

file = CreateFileW(name,
    FILE_GENERIC_READ | FILE_GENERIC_WRITE,
    FILE_SHARE_READ | FILE_SHARE_WRITE,
    0, OPEN_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);

CloseHandle(file);
```

ФУНКЦИИ ДЛЯ БАЗОВОЙ РАБОТОЙ С ФАЙЛАМИ В ОС WINDOWS

```
WINBASEAPI
_Must_inspect_result_
BOOL
WINAPI
ReadFile(
    _In_ HANDLE hFile,
    _Out_writes_bytes_to_opt_(nNumberOfBytesToRead, *lpNumberOfBytesRead) __out_data_source(FILE) LPVOID lpBuffer,
    _In_ DWORD nNumberOfBytesToRead,
    _Out_opt_ LPDWORD lpNumberOfBytesRead,
    _Inout_opt_ LPOVERLAPPED lpOverlapped
    );
```

Документация:

ReadFile

WriteFile

GetFileSize

```
WINBASEAPI
BOOL
WINAPI
WriteFile(
    _In_ HANDLE hFile,
    _In_reads_bytes_opt_(nNumberOfBytesToWrite) LPCVOID lpBuffer,
    _In_ DWORD nNumberOfBytesToWrite,
    _Out_opt_ LPDWORD lpNumberOfBytesWritten,
    _Inout_opt_ LPOVERLAPPED lpOverlapped
    );
```

```
WINBASEAPI
DWORD
WINAPI
GetFileSize(
    _In_ HANDLE hFile,
    _Out_opt_ LPDWORD lpFileSizeHigh
    );
```

ПРИМЕР

```
int read_bytes = 0;
int buffer_size = 1024;
char* buffer = NULL;
buffer = (char*)malloc(buffer_size * sizeof(char));
if (!ReadFile(file, buffer, buffer_size, &read_bytes, NULL)) {
// ошибка
}
if (buffer) free(buffer);
```

```
int written_bytes = 0;
char* buffer = "kotik";
if (!WriteFile(file, buffer, strlen(buffer) + 1, &written_bytes, NULL)) {
// ошибка
}
```

ФУНКЦИИ ДЛЯ БАЗОВОЙ РАБОТОЙ С ФАЙЛАМИ В ОС LINUX

```c
int open(const char* pathname, int flags);
int creat(const char* pathname, mode_t mode);
ssize_t read(int fd, void* buf, size_t count);
ssize_t write(int fd, void* buf, size_t count);
int close(int fd);
```

Документация:
open
read
write
close

ОТОБРАЖЕНИЕ ФАЙЛОВ В ПАМЯТЬ

```
WINBASEAPI
_Ret_maybenull_
HANDLE
WINAPI
CreateFileMappingW(
    _In_ HANDLE hFile,
    _In_opt_ LPSECURITY_ATTRIBUTES lpFileMappingAttributes,
    _In_ DWORD flProtect,
    _In_ DWORD dwMaximumSizeHigh,
    _In_ DWORD dwMaximumSizeLow,
    _In_opt_ LPCWSTR lpName
    );
```

Документация:
CreateFileMappingW
MapViewOfFile
UnmapViewOfFile

```
WINBASEAPI
_Ret_maybenull_  __out_data_source(FILE)
LPVOID
WINAPI
MapViewOfFile(
    _In_ HANDLE hFileMappingObject,
    _In_ DWORD dwDesiredAccess,
    _In_ DWORD dwFileOffsetHigh,
    _In_ DWORD dwFileOffsetLow,
    _In_ SIZE_T dwNumberOfBytesToMap
    );

WINBASEAPI
BOOL
WINAPI
UnmapViewOfFile(
    _In_ LPCVOID lpBaseAddress
    );
```

ПРИМЕР

```
HANDLE mapFile = CreateFileMappingW(file, NULL, PAGE_READWRITE, 0, 0, 0);
LPVOID address = MapViewOfFile(mapFile, FILE_MAP_ALL_ACCESS, 0, 0, GetFileSize(file,0));
UnmapViewOfFile(address);
CloseHandle(mapFile);
```

ФУНКЦИИ ДЛЯ ПОЛУЧЕНИЯ ИНФОРМАЦИИ ОБ ОШИБКАХ

```
WINBASEAPI
_Check_return_
_Post_equals_last_error_
DWORD
WINAPI
GetLastError(
    VOID
);
WINBASEAPI
_Success_(return != 0)
DWORD
WINAPI
FormatMessageW(
    _In_     DWORD dwFlags,
    _In_opt_ LPCVOID lpSource,
    _In_     DWORD dwMessageId,
    _In_     DWORD dwLanguageId,
    _When_((dwFlags & FORMAT_MESSAGE_ALLOCATE_BUFFER) != 0, _At_((LPWSTR*)lpBuffer, _Outptr_result_z_))
    _When_((dwFlags & FORMAT_MESSAGE_ALLOCATE_BUFFER) == 0, _Out_writes_z_(nSize))
             LPWSTR lpBuffer,
    _In_     DWORD nSize,
    _In_opt_ va_list *Arguments
);
```

Документация:
GetLastError
FormatMessage

ПРИМЕР

```
int error = GetLastError();
char* desc = NULL;
FormatMessageA(FORMAT_MESSAGE_FROM_SYSTEM | FORMAT_MESSAGE_ALLOCATE_BUFFER,
               NULL, error, 0, &desc, 0, 0);
LocalFree(desc);



char str[] = "You name is %1 and you are %2!d! years old";
char* name = "kotiche";
int age = 25;
char* desc = NULL;
DWORD_PTR argumets[] = {(DWORD_PTR)name, (DWORD_PTR)age};
FormatMessageA(FORMAT_MESSAGE_ARGUMENT_ARRAY | FORMAT_MESSAGE_FROM_STRING |
               FORMAT_MESSAGE_ALLOCATE_BUFFER, str, 0, 0,
               &desc, 0, (va_list*)argumets);
LocalFree(desc);
```

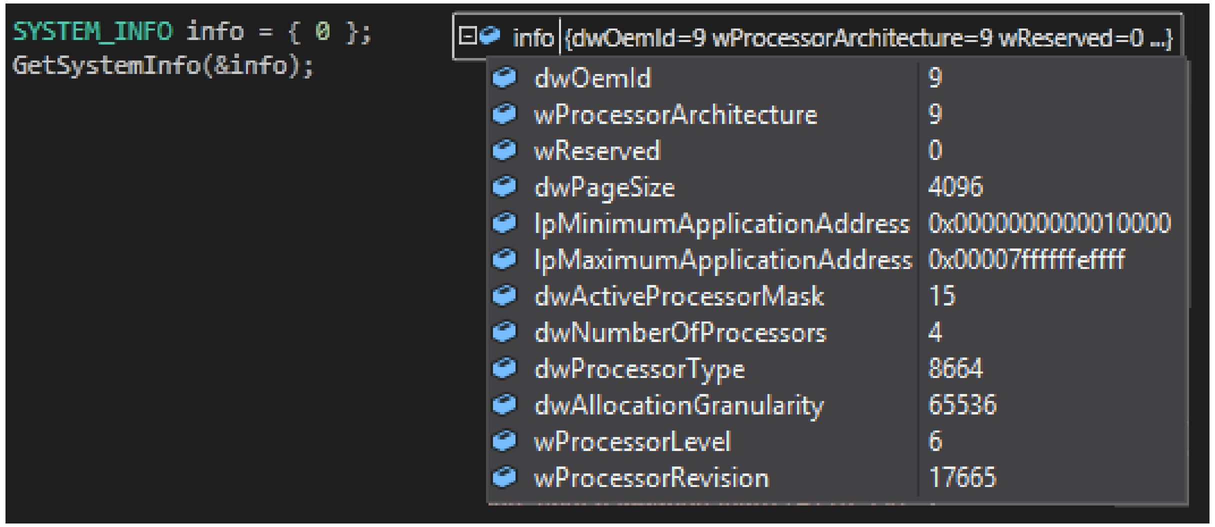
ФУНКЦИИ ДЛЯ ПОЛУЧЕНИЯ ИНФОРМАЦИИ О СИСТЕМЕ

```
WINBASEAPI
VOID
WINAPI
GetLocalTime(
    _Out_ LPSYSTEMTIME lpSystemTime
    );

WINBASEAPI
VOID
WINAPI
GetSystemTime(
    _Out_ LPSYSTEMTIME lpSystemTime
    );

WINBASEAPI
VOID
WINAPI
GetSystemInfo(
    _Out_ LPSYSTEM_INFO lpSystemInfo
    );
```

Разница между GetLocalTime() и GetSystemTime(), лишь в том, что GetLocalTime() возвращает время, скорректированное с часовым поясом

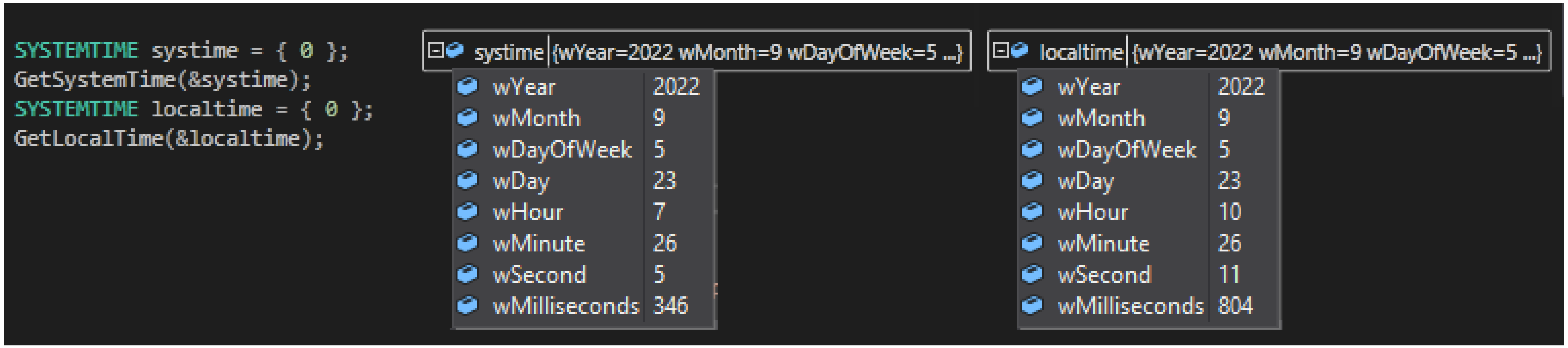
Документация:
GetLocalTime
GetSystemTime
GetSystemInfo

ПРИМЕР



```
SYSTEM_INFO info = { 0 };
GetSystemInfo(&info);
```

```
⊟ info {dwOemId=9 wProcessorArchitecture=9 wReserved=0 ...}
    dwOemId                       9
    wProcessorArchitecture        9
    wReserved                     0
    dwPageSize                    4096
    lpMinimumApplicationAddress   0x0000000000010000
    lpMaximumApplicationAddress   0x00007ffffffeffff
    dwActiveProcessorMask         15
    dwNumberOfProcessors          4
    dwProcessorType               8664
    dwAllocationGranularity       65536
    wProcessorLevel               6
    wProcessorRevision            17665
```



```
SYSTEMTIME systime = { 0 };
GetSystemTime(&systime);
SYSTEMTIME localtime = { 0 };
GetLocalTime(&localtime);
```

```
⊟ systime {wYear=2022 wMonth=9 wDayOfWeek=5 ...}
    wYear          2022
    wMonth         9
    wDayOfWeek     5
    wDay           23
    wHour          7
    wMinute        26
    wSecond        5
    wMilliseconds  346
```

```
⊟ localtime {wYear=2022 wMonth=9 wDayOfWeek=5 ...}
    wYear          2022
    wMonth         9
    wDayOfWeek     5
    wDay           23
    wHour          10
    wMinute        26
    wSecond        11
    wMilliseconds  804
```