Разработать программу, обеспечивающую получение следующей системной информации:
COLOR_3DFACE - Цвет фона трехмерных элементов интерфейса
COLOR_CAPTIONTEXT - Цвет текста заголовков окон
COLOR_BACKGROUND - Цвет фона окон

```cpp
#include <Windows.h>
#include <iostream>

int main() {
    // Получаем дескриптор текущего модуля
    HINSTANCE hInstance = GetModuleHandle(NULL);

    // Получаем значения трех системных цветов
    COLORREF color3DFace = GetSysColor(COLOR_3DFACE); // Цвет фона трехмерных элементов интерфейса
    COLORREF colorCaptionText = GetSysColor(COLOR_CAPTIONTEXT); // Цвет текста заголовков окон
    COLORREF colorBackground = GetSysColor(COLOR_BACKGROUND); // Цвет фона окон

    // Выводим полученные значения на экран
    std::cout << "COLOR_3DFACE: " << std::hex << color3DFace << std::endl;
    std::cout << "COLOR_CAPTIONTEXT: " << std::hex << colorCaptionText << std::endl;
    std::cout << "COLOR_BACKGROUND: " << std::hex << colorBackground << std::endl;

    return 0;
}
```

```
[Running] cd "d:\docker-test\тк
COLOR_3DFACE: f0f0f0
COLOR_CAPTIONTEXT: 0
COLOR_BACKGROUND: 0
```

COLOR_3DFACE: f0f0f0 - это светло-серый цвет, который часто используется для отображения фоновых элементов интерфейса.

COLOR_CAPTIONTEXT: 0 - это черный цвет текста заголовка окна.
COLOR_BACKGROUND: 0 - это черный цвет фона.
Эти значения выглядят стандартными

Разработать программу, обеспечивающую получение следующей системной информации: GetLocalTime, GetTimeZoneInformation, EnuinDate Formats

```
#include <iostream>
#include <windows.h>
#include <locale.h>

BOOL CALLBACK EnumDateFormatsProc(LPTSTR lpDateFormatString) {
    std::wcout << lpDateFormatString << std::endl;
    return TRUE;
}

int main() {
    // Получение локального времени
    SYSTEMTIME sysTime;
    GetLocalTime(&sysTime);
    std::cout << "Локальное время: " << sysTime.wHour << ":" <<
sysTime.wMinute << ":" << sysTime.wSecond << std::endl;

    // Перечисление форматов даты для английской локали (en-US)
    std::cout << "Форматы даты:" << std::endl;
    EnumDateFormats(EnumDateFormatsProc,
MAKELCID(MAKELANGID(LANG_ENGLISH, SUBLANG_ENGLISH_US), SORT_DEFAULT),
DATE_SHORTDATE);

    return 0;
}
```

```
[Running] cd "d:\docker-test\
Локальное время: 14:32:48
Форматы даты:
M/d/yyyy
M/d/yy
MM/dd/yy
MM/dd/yyyy
yy/MM/dd
yyyy-MM-dd
dd-MMM-yy
M/d/yyyy
M/d/yy
MM/dd/yy
MM/dd/yyyy
```

Разработать программу, обеспечивающую получение следующей системной информации: CharToOem, GetCursor, GetLocaleInfo, OemToCharBuff

```cpp
#include <iostream>
#include <windows.h>

int main() {
    // Получение информации о кодовой странице
    UINT codePage = GetConsoleCP();
    std::cout << "консольная страница: " << codePage << std::endl;

    // Преобразование символов из ANSI в OEM
    const char* ansiString = "Hello, world!";
    char oemString[256];
    CharToOemA(ansiString, oemString);
    std::cout << "ANSI to OEM: " << oemString << std::endl;

    // Получение позиции курсора
    CONSOLE_SCREEN_BUFFER_INFO csbi;
    GetConsoleScreenBufferInfo(GetStdHandle(STD_OUTPUT_HANDLE), &csbi);
    COORD cursorPos = csbi.dwCursorPosition;
```

```
    std::cout << "Позиция курсора: X = " << cursorPos.X << ", Y = " <<
cursorPos.Y << std::endl;

    // Получение информации о локали
    LCID locale = LOCALE_USER_DEFAULT;
    char localeInfo[256];
    GetLocaleInfoA(locale, LOCALE_SENGLANGUAGE, localeInfo,
sizeof(localeInfo));
    std::cout << "Локальная информация: " << localeInfo << std::endl;

    // Преобразование символов из OEM в ANSI
    char ansiBuffer[256];
    OemToCharA(oemString, ansiBuffer);
    std::cout << "OEM to ANSI: " << ansiBuffer << std::endl;

    return 0;
}
```

```
[Running] cd "d:\docker-test\тк
консольная страница: 866
ANSI to OEM: Hello, world!
Позиция курсора: X = 0, Y = 0
Локальная информация: Russian
OEM to ANSI: Hello, world!
```