Politechnika Łódzka
Instytut Elektroniki

Elementy programowania współbieżnego w MS Windows

Piotr M. Szczypiński
Instytut Elektroniki Politechniki Łódzkiej
http://www.eletel.p.lodz.pl/pms/
piotr.szczypinski@p.lodz.pl
Budynek B9, II piętro, pokój 217A

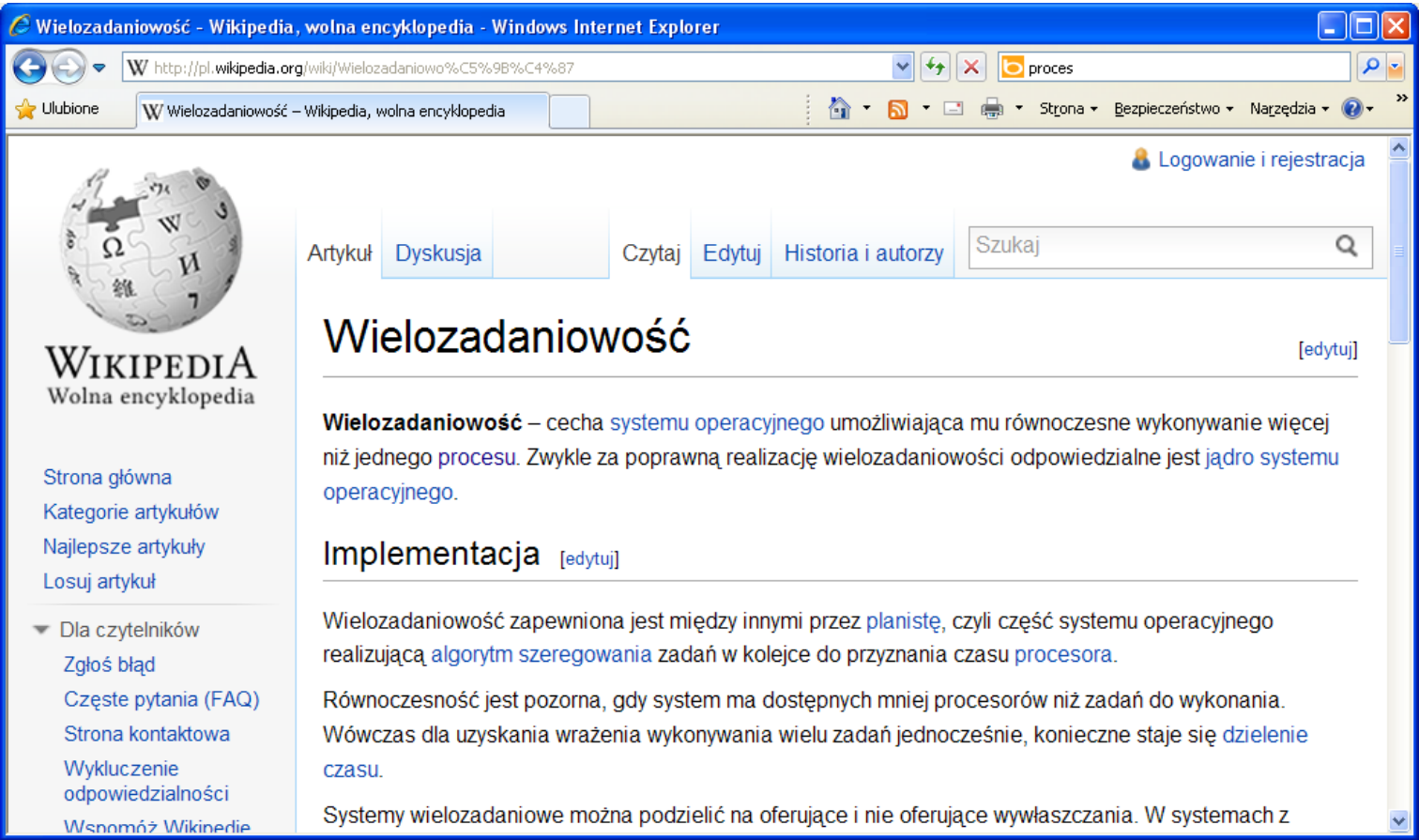
Materiał na dziś

1. Procesy, wątki, synchronizacja, współdzielenie zasobów
2. Jeden wątek wiele programów (procesów)
3. Kiedy potrzebne jest programowanie wielowątkowe
4. Narzędzia programowania
5. Źródła informacji
6. Klasyczny program systemu Windows:
    - Tworzenie okna i pętla obsługi zdarzeń
    - Blokowanie interfejsu graficznego podczas obliczeń
    - Dodanie wątku obliczeniowego w programie
    - Uruchamianie programów zewnętrznych – elementy synchronizacji
7. Sekcje krytyczne i inne narzędzia synchronizacji
8. Wprowadzenie do programowania technologii multimedialnych

Procesy, wątki...

Procesy, wątki...

Procesy, wątki...



Procesy, wątki...

Procesy, wątki...

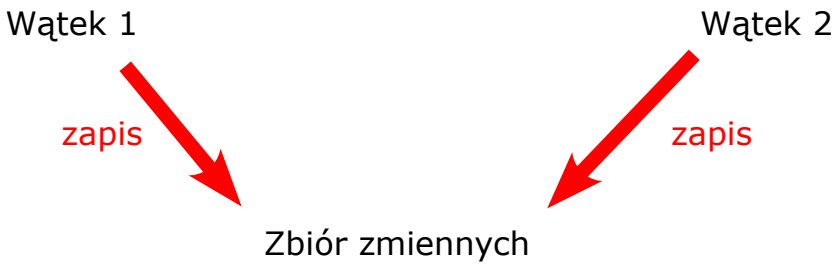
Czy możliwa jest wielozadaniowość przy wykorzystaniu pojedynczego wątku?

Czy możliwe jest wykonywanie wielu wątków w systemie z pojedynczym procesorem (pojedynczym rdzeniem)?

Czy ma sens tworzenie wielu wątków w systemie z jednym rdzeniem procesora?

Co się może stać, jeśli dwa wątki wykonywane równocześnie będą korzystać z tych samych zmiennych?

...synchronizacja, współdzielenie...

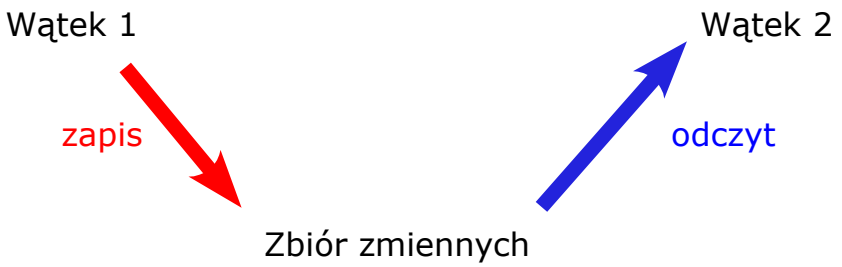


Problem – pojedynczy rdzeń, przełączanie wątków w czasie:

„Wątek 1” zapisał część danych, system go wstrzymał i uruchomił „Wątek 2”, „Wątek 2” zapisał wszystkie dane, system go wstrzymał i uruchomił „Wątek 1”, który dokończył zapis danych. Dane są niespójne.

Konieczna jest synchronizacja wątków!

...synchronizacja, współdzielenie...

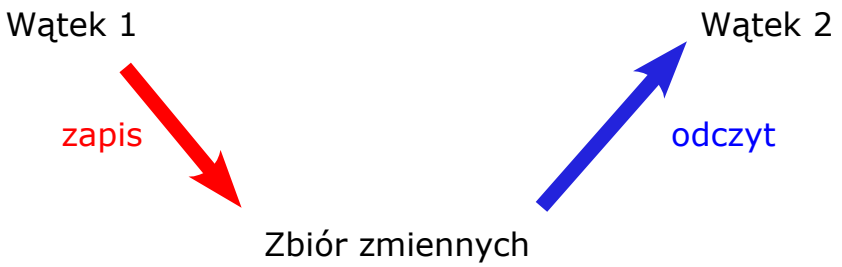


Problem – pojedynczy rdzeń, przełączanie wątków w czasie:

„Wątek 1” zapisał część danych, zaczął robić coś innego lub system go wstrzymał. Dane są niespójne. „Wątek 2” odczytuje dane (np. parametry metody obliczeniowej) i wykonuje obliczenia z wykorzystaniem nonsensownych parametrów.

Konieczna jest synchronizacja wątków!

...synchronizacja, współdzielenie...



Problem – dwa rdzenie, dwa wątki wykonywane w tym samym czasie:

„Wątek 1” przetwarza blok danych wstępnie, „Wątek 2” odczytuje dane przetworzone przez pierwszy i przetwarza je dalej. Algorytm jest kontynuowany ponieważ wciąż pojawiają się nowe bloki danych.

Problem: „Wątek 2” może gubić, nie nadążać przetwarzać kolejnych bloków danych lub „Wątek 2” jest zbyt szybki i może przetwarzać przetwarzać niektóre bloki danych powtórnie

Konieczna jest synchronizacja wątków!

...synchronizacja, współdzielenie...

Mechanizmy synchronizacji wątków:

semafor – licznik określający liczbę dostępnych zasobów, w miarę zajmowania zasobów jego wartość jest zmniejszana, wartość zero oznacza brak zasobów – wątek, który chce uzyskać dostęp do zasobów musi w takim przypadku zostać wstrzymany.

mutex – semafor binarny, przyjmuje wartość 1 lub 0, stosowany do blokowania dostępu do zmiennych w pamięci lub kodu programu.

sekcja krytyczna – fragment kodu programu, który w danej chwili może być wykorzystywany przez co najwyżej jeden wątek, np. dlatego, że korzysta ze współdzielonych zasobów.

Kiedy potrzebne jest...

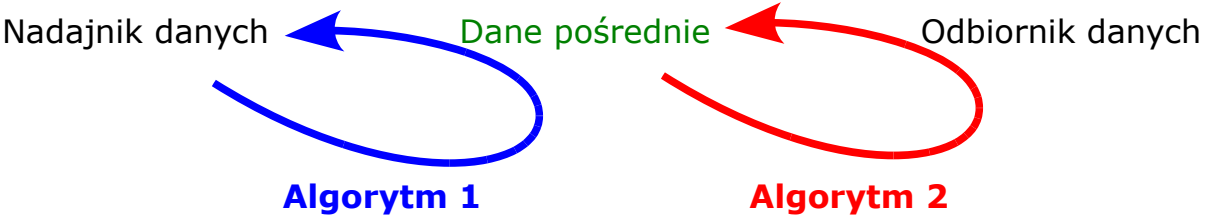
1. Rozdzielenie wątków interfejsu graficznego i obliczeniowego (sensowne nawet przy pojedynczym rdzeniu).
   Zapobiega blokowaniu interfejsu graficznego użytkownika podczas wykonywania obliczeń.

Kiedy potrzebne jest...

2. Zrównoleglenie obliczeń, ten sam algorytm wykonywany współbieżnie na różnych danych (np. podział obrazu na części i filtracja poszczególnych części)

Kiedy potrzebne jest...

3. Przetwarzanie potokowe (ang. *pipeline*) – alogorytm jest podzielony na mniejsze algorytmy, które wykonywane są kolejno na nadchodzących w czasie danych (np. danych multimedialnych wideo lub dźwięku).

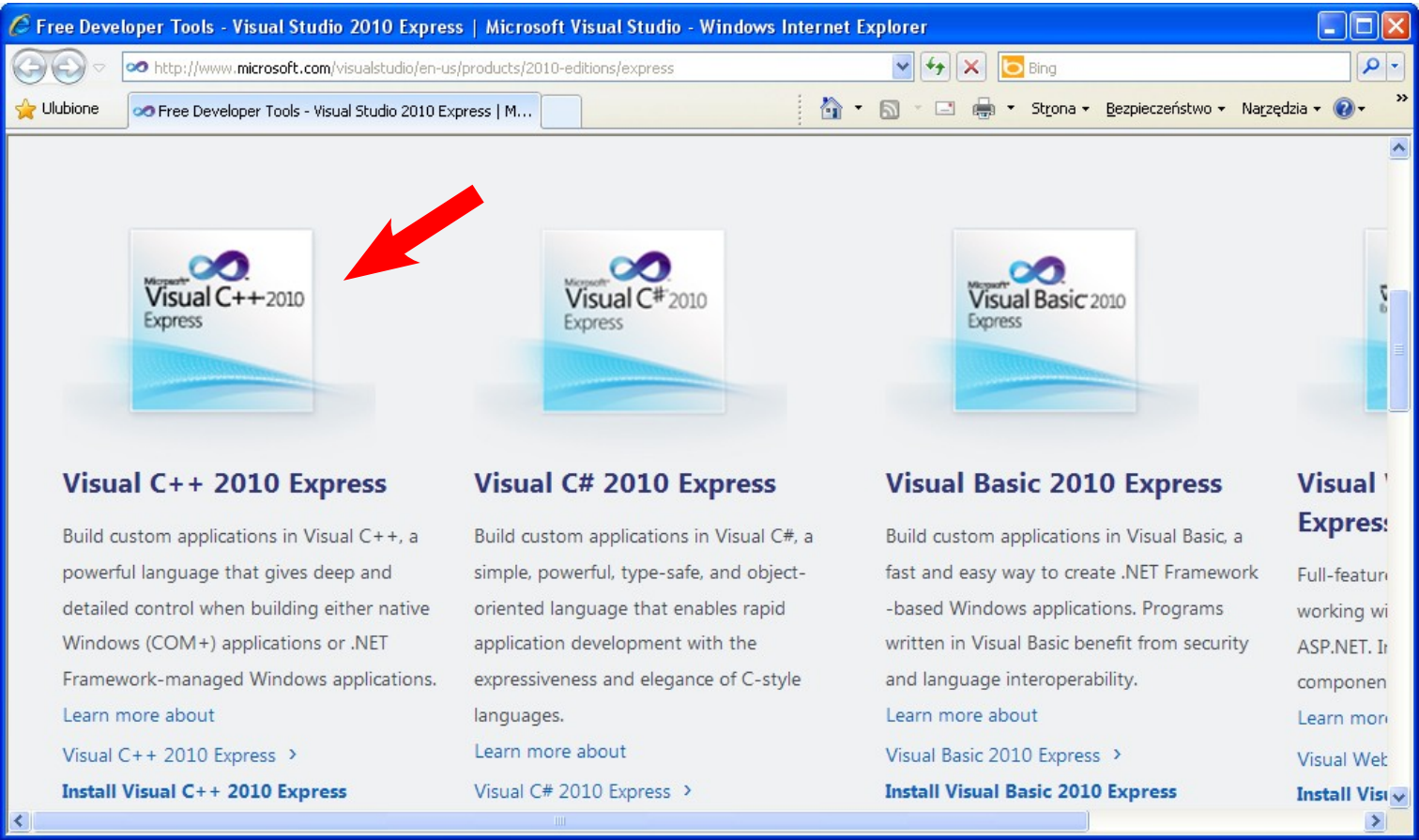


Narzędzia programowania

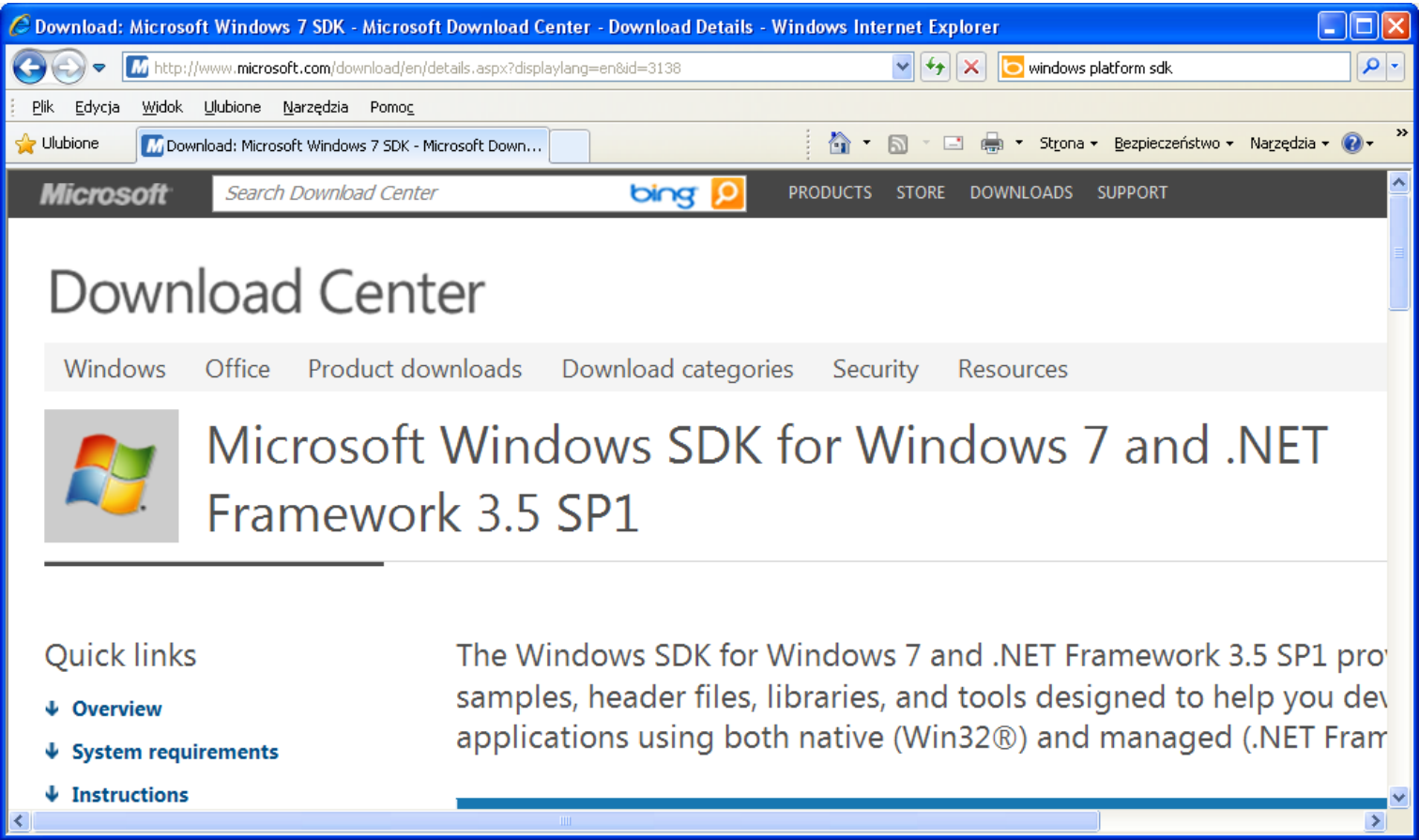
Visual Studio C++ – środowisko programistyczne

Microsoft Windows SDK – dodatkowe narzędzia i biblioteki

Narzędzia programowania



Narzędzia programowania



Źródła informacji

Charles Petzold, Programming Windows

MSDN Library

Źródła informacji

Programming Windows, 5th Edition, by Charles Petzold - Windows Internet Explorer

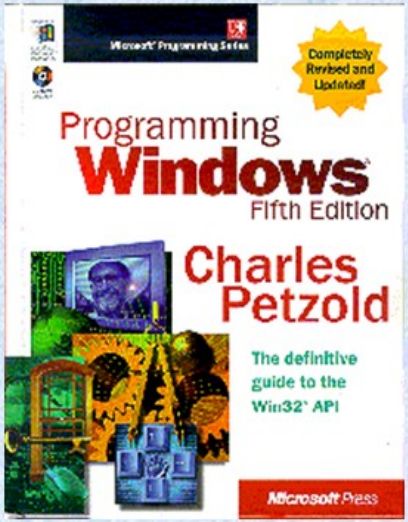
http://charlespetzold.com/pw5/index.html

charles petzold

Plik Edycja Widok Ulubione Narzędzia Pomoc

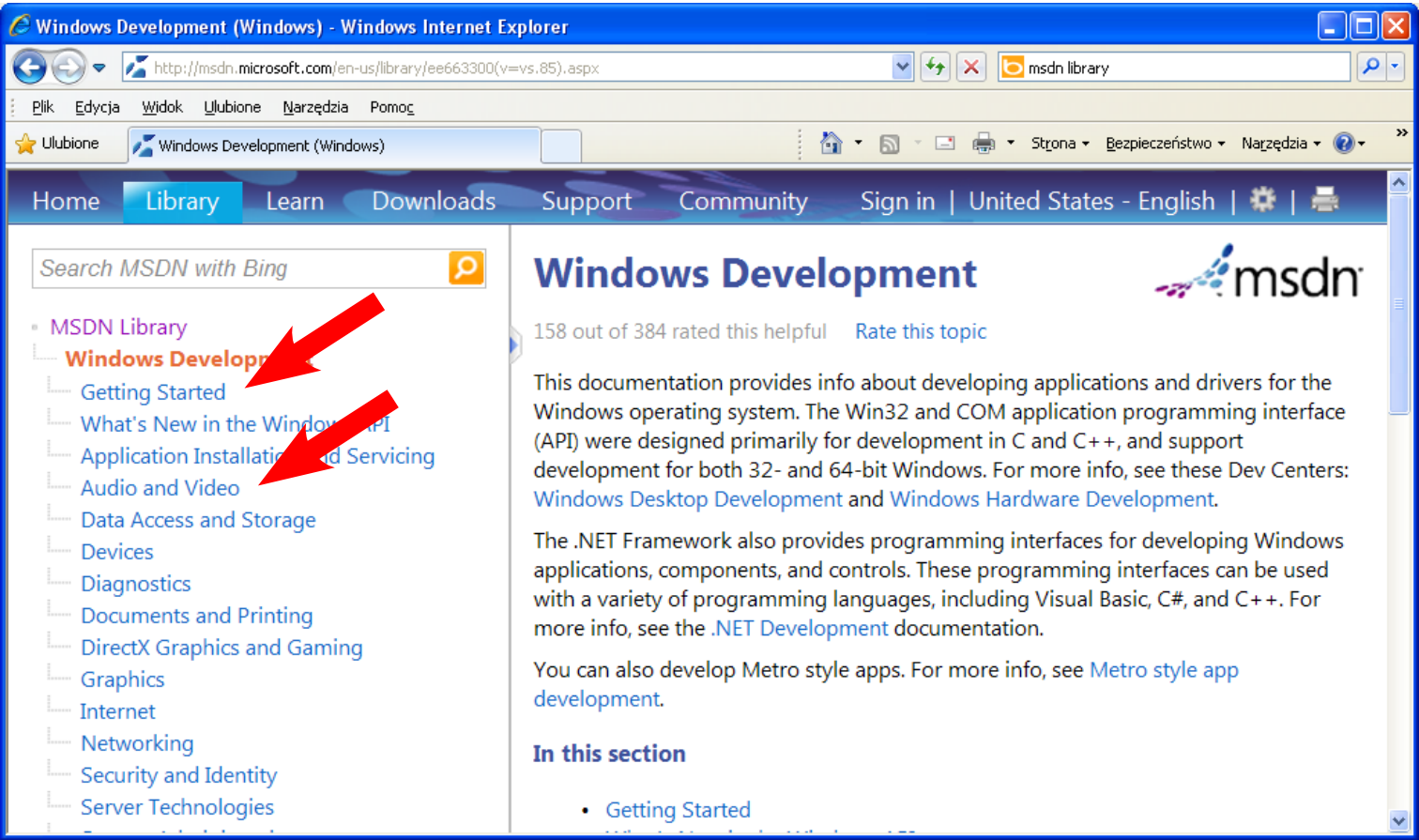
Ulubione | Programming Windows, 5th Edition, by Charles Petzold | Strona | Bezpieczeństwo | Narzędzia

Programming Windows, 5th Edition

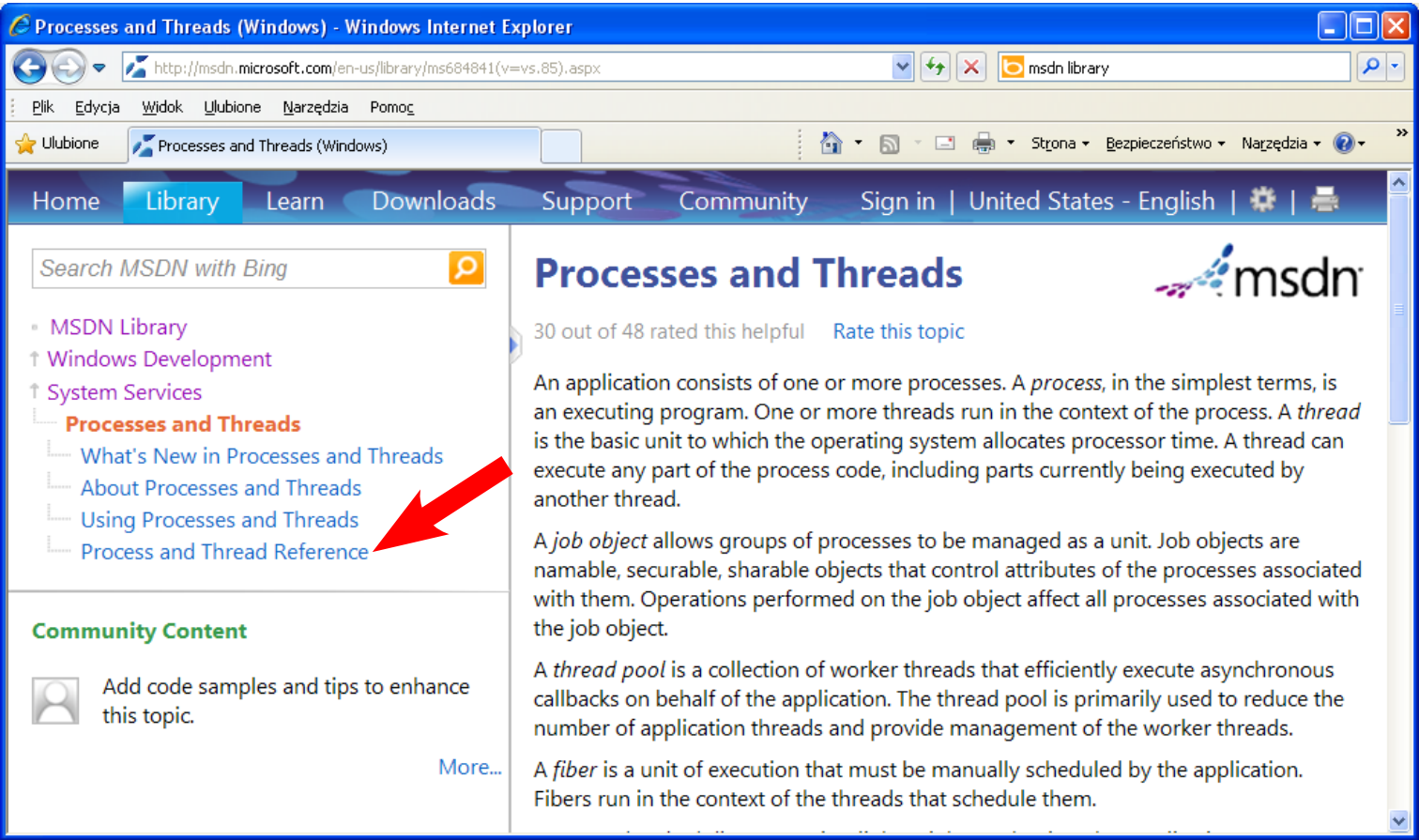


Programming Windows is a tutorial for programmers wishing to write applications for Windows using the C programming language and the native Win32 application programming interface (API). Programs written using this book run under every version of Windows from Windows 95 through Windows XP and beyond. Knowledge of C is required.

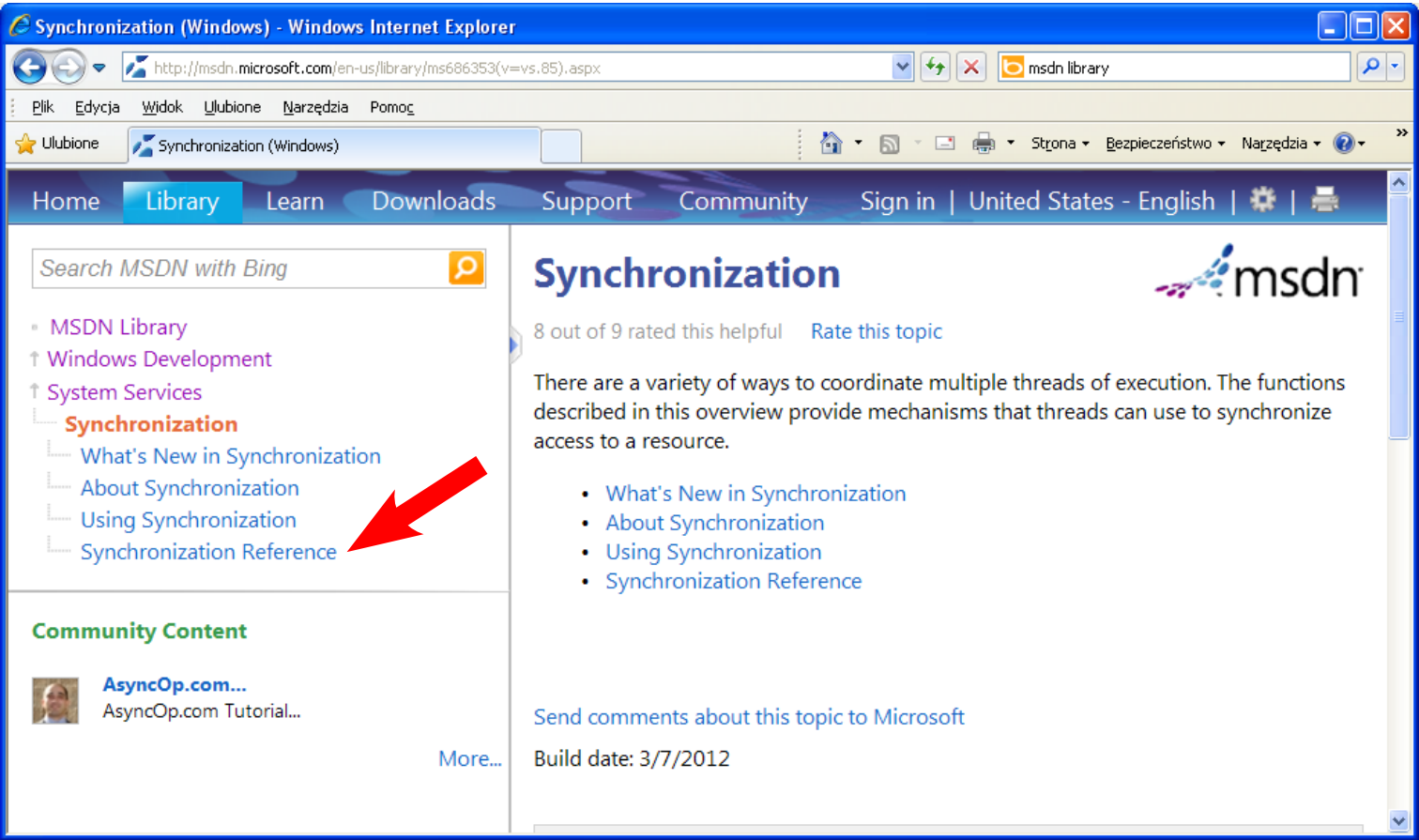
Źródła informacji



Źródła informacji



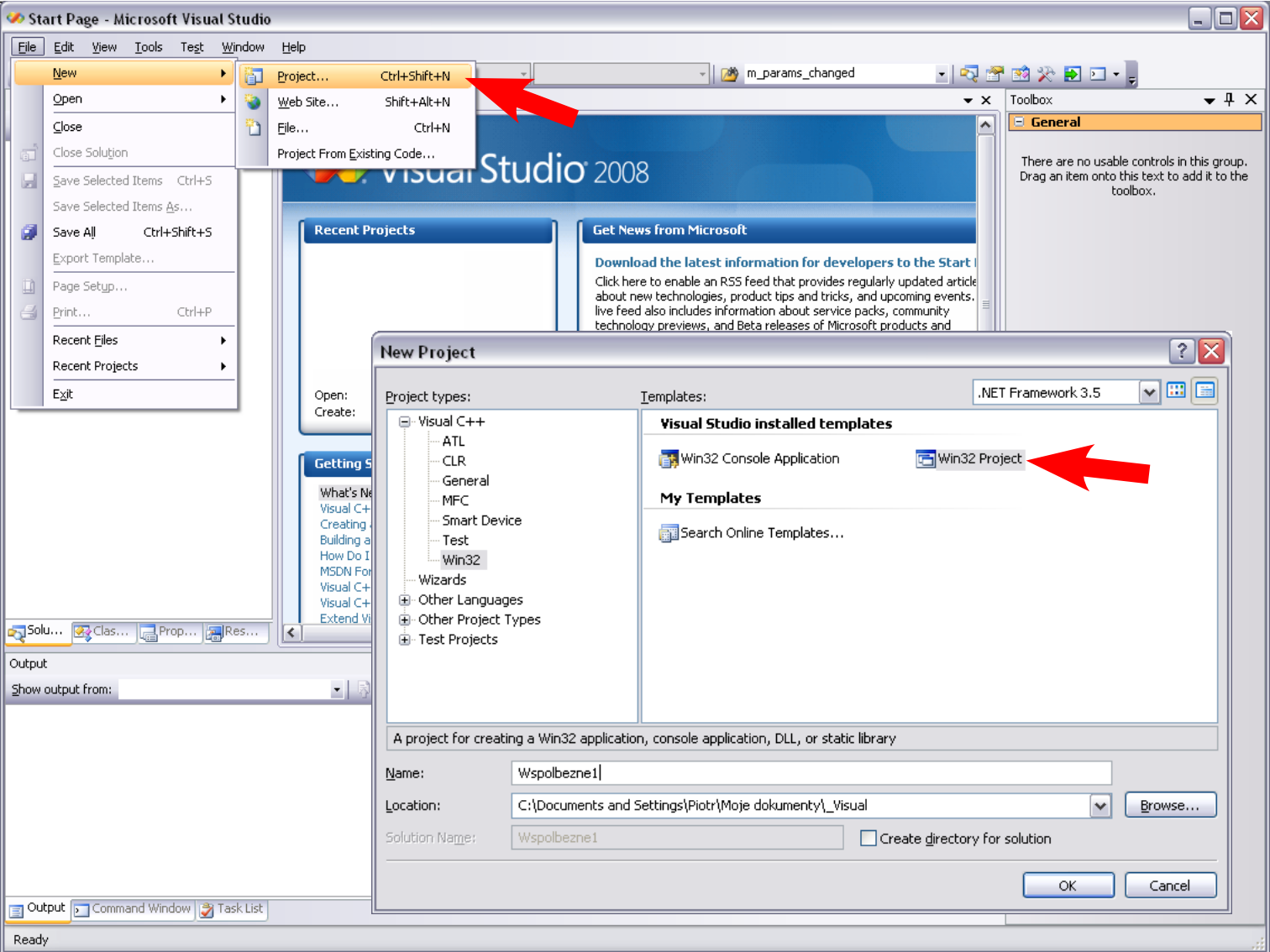
Źródła informacji



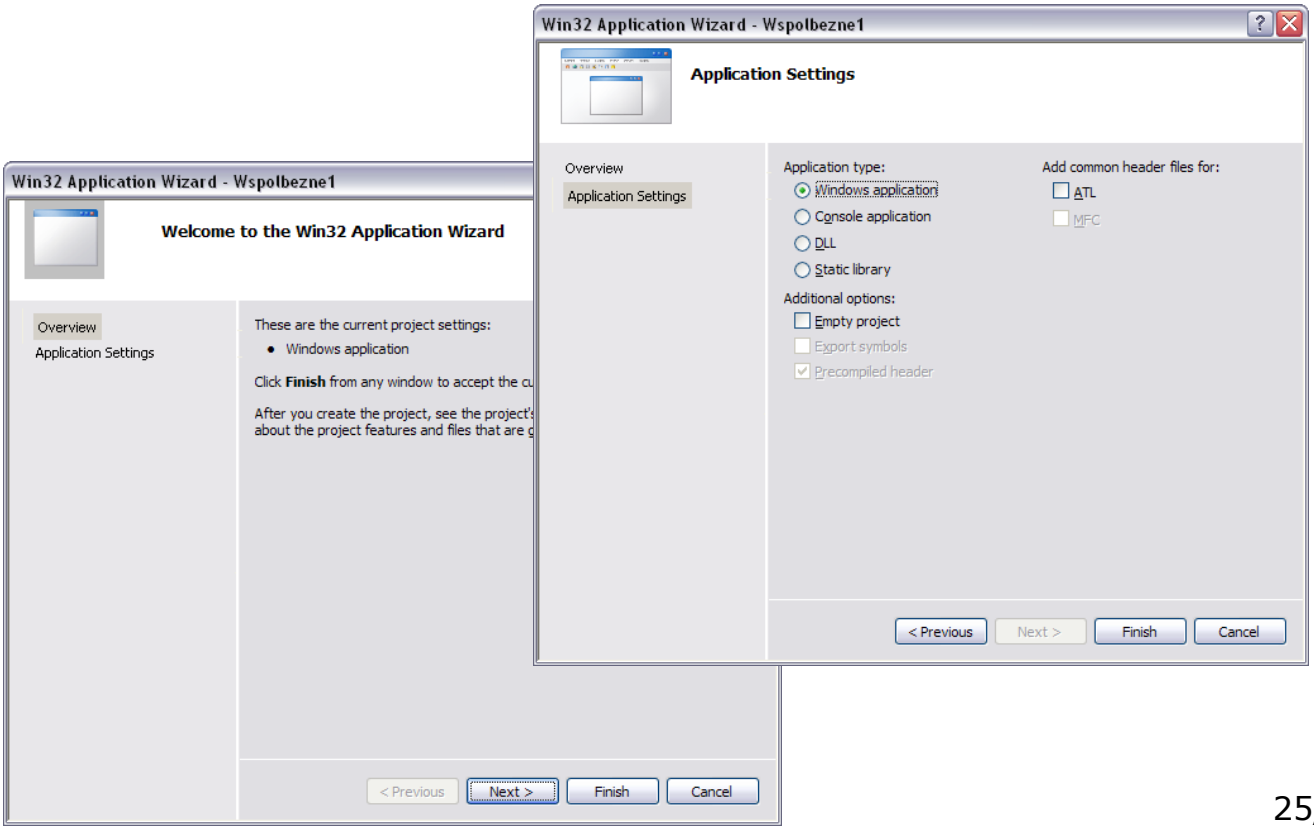
Klasyczny program Windows

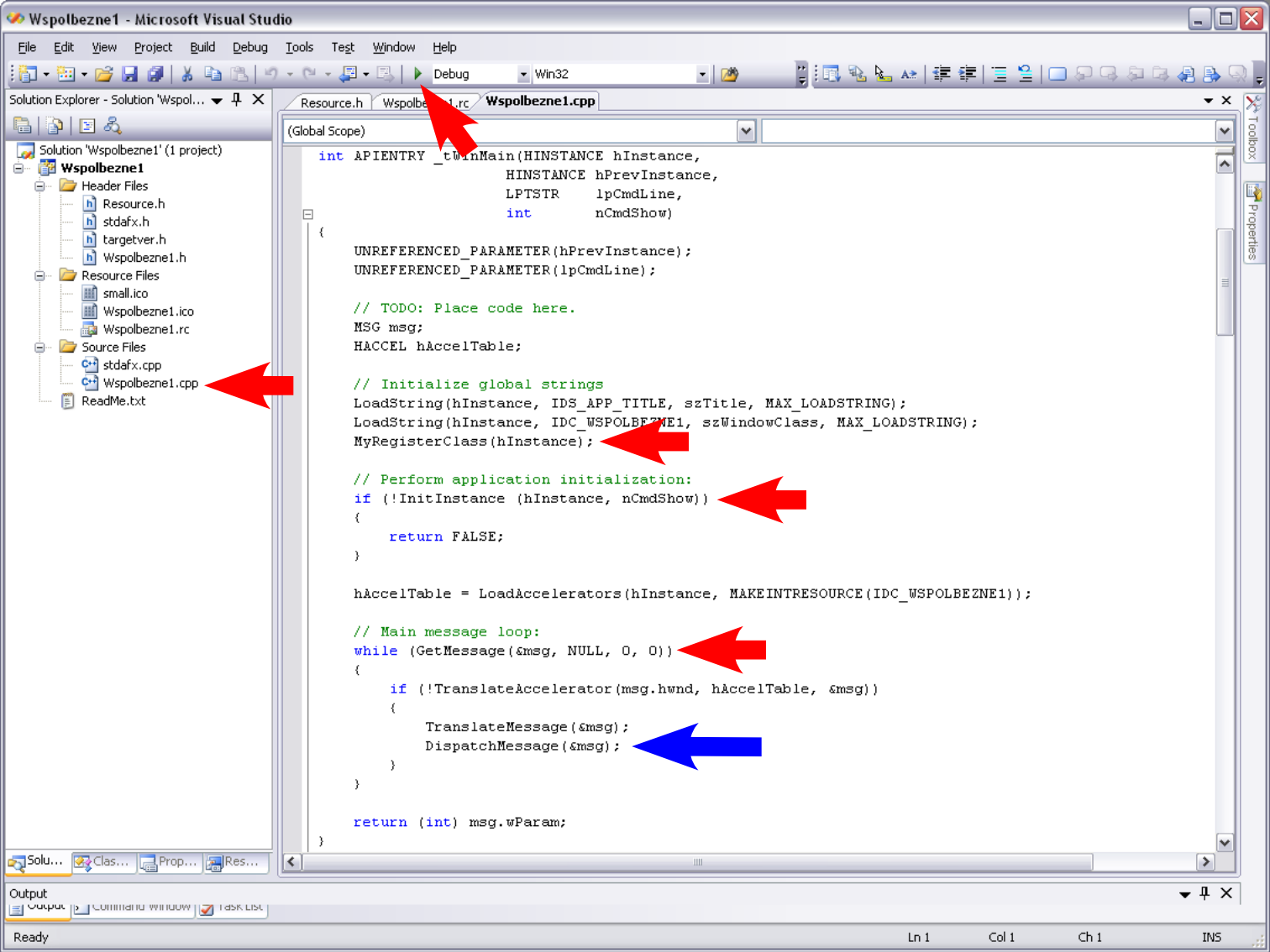
Zagadnienia:

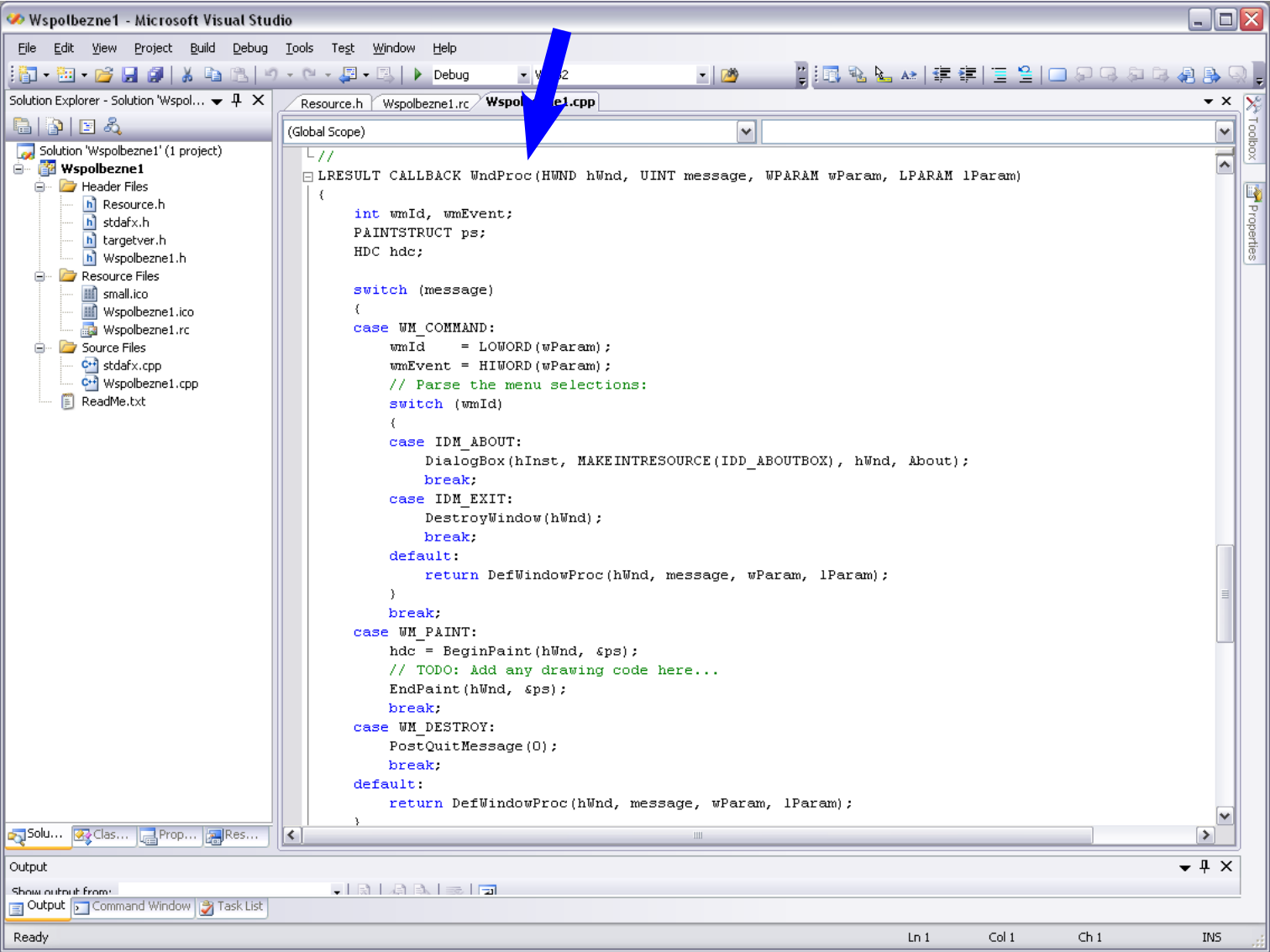
- Projekt programu w Visual Studio C++
- Tworzenie interfejsu graficznego (okna) programu
- Zdarzenia i pętla obsługi zdarzeń (jeden wątek wiele procesów!)
- Efekt blokowania się interfejsu graficznego
- Dodanie wątku obliczeniowego
- Komunikowanie się wątków za pomocą kolejki zdarzeń (Skończyłem!)
- Wywoływanie programów zewnętrznych i oczekiwanie na ich zakończenie


Start Page - Microsoft Visual Studio
File Edit View Tools Test Window Help
New
Open
Close
Close Solution
Save Selected Items Ctrl+S
Save Selected Items As...
Save All Ctrl+Shift+S
Export Template...
Page Setup...
Print... Ctrl+P
Recent Files
Recent Projects
Exit
Project... Ctrl+Shift+N
Web Site... Shift+Alt+N
File... Ctrl+N
Project From Existing Code...
m_params_changed
Toolbox
General
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.
Visual Studio 2008
Recent Projects
Get News from Microsoft
Download the latest information for developers to the Start
Click here to enable an RSS feed that provides regularly updated article
about new technologies, product tips and tricks, and upcoming events.
live feed also includes information about service packs, community
technology previews, and Beta releases of Microsoft products and
Open:
Create:
Getting S
What's Ne
Visual C+
Creating
Building a
How Do I
MSDN For
Visual C+
Visual C+
Extend Vi
Solu...
Clas...
Prop...
Res...
Output
Show output from:
Output
Command Window
Task List
Ready
New Project
Project types:
Templates:
.NET Framework 3.5
Visual C++
ATL
CLR
General
MFC
Smart Device
Test
Win32
Wizards
Other Languages
Other Project Types
Test Projects
Visual Studio installed templates
Win32 Console Application
Win32 Project
My Templates
Search Online Templates...
A project for creating a Win32 application, console application, DLL, or static library
Name: Wspolbezne1
Location: C:\Documents and Settings\Piotr\Moje dokumenty_Visual
Browse...
Solution Name: Wspolbezne1
Create directory for solution
OK
Cancel

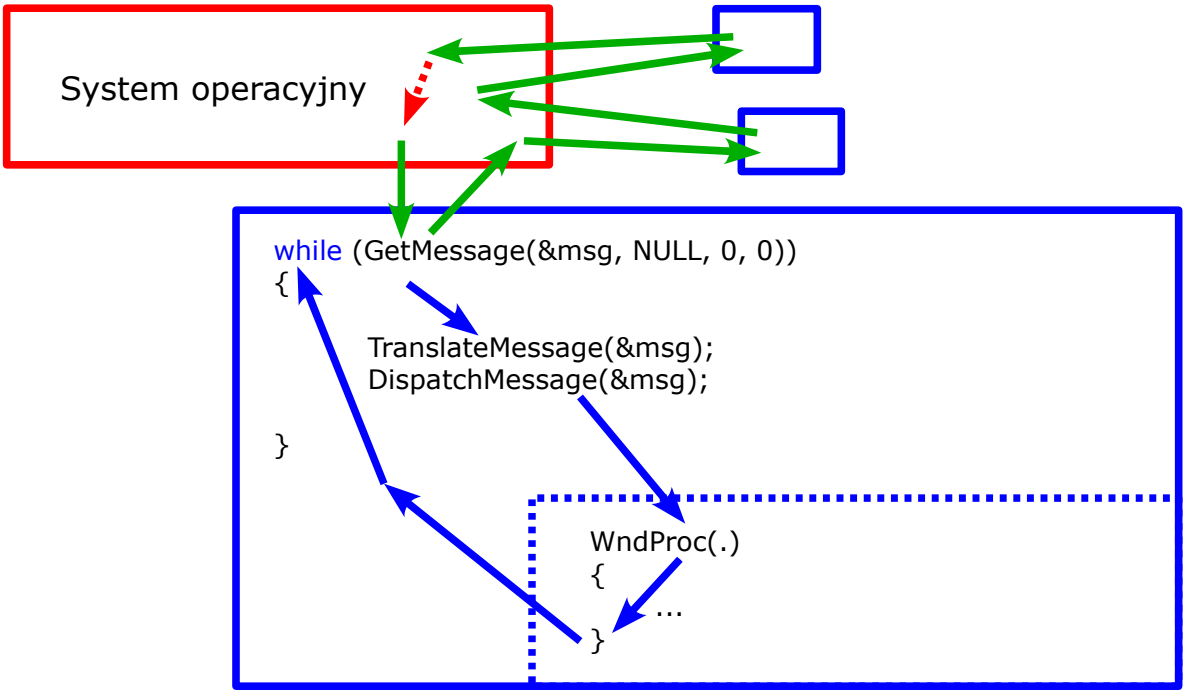
Projekt programu w Visual Studio

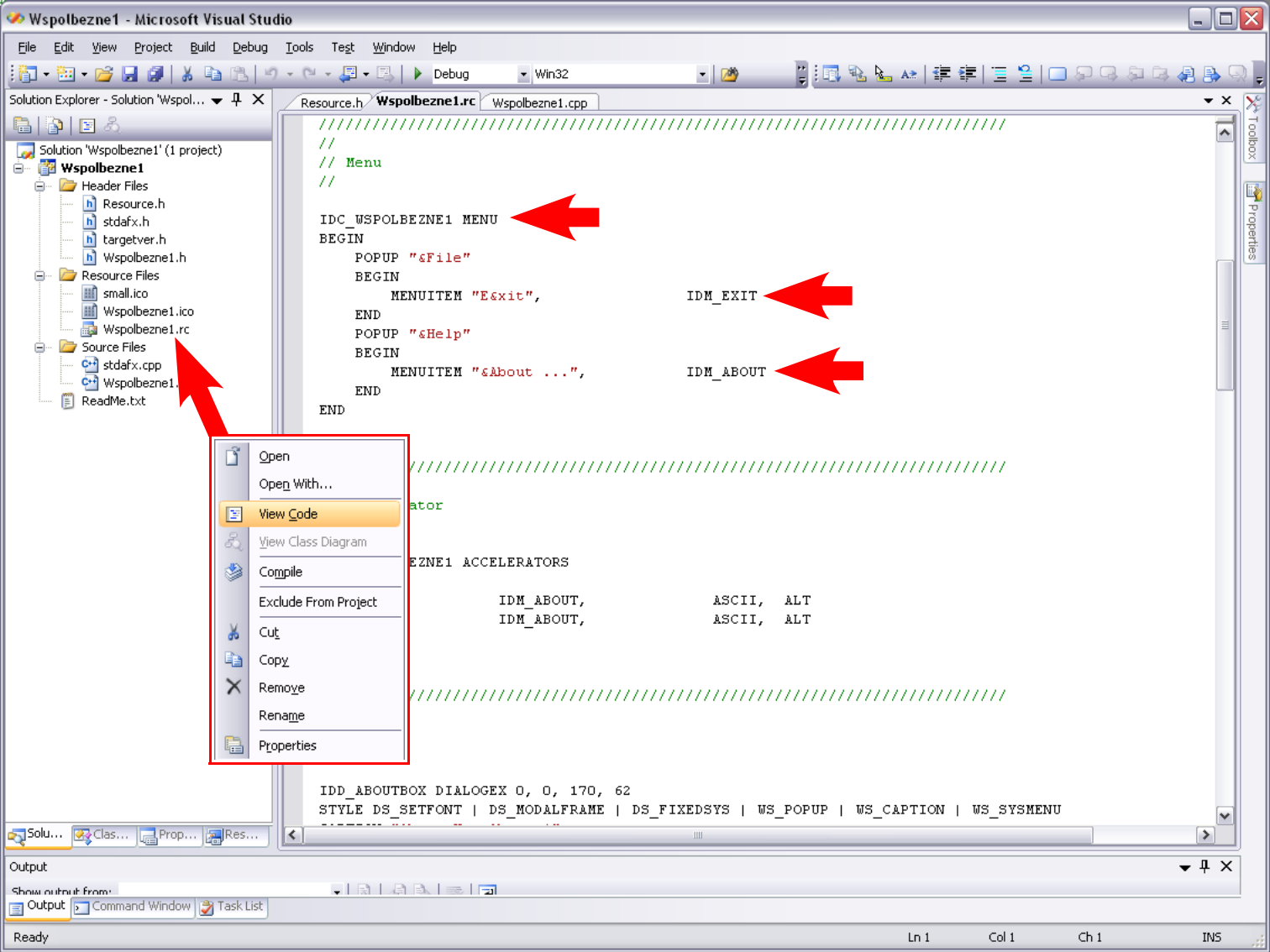


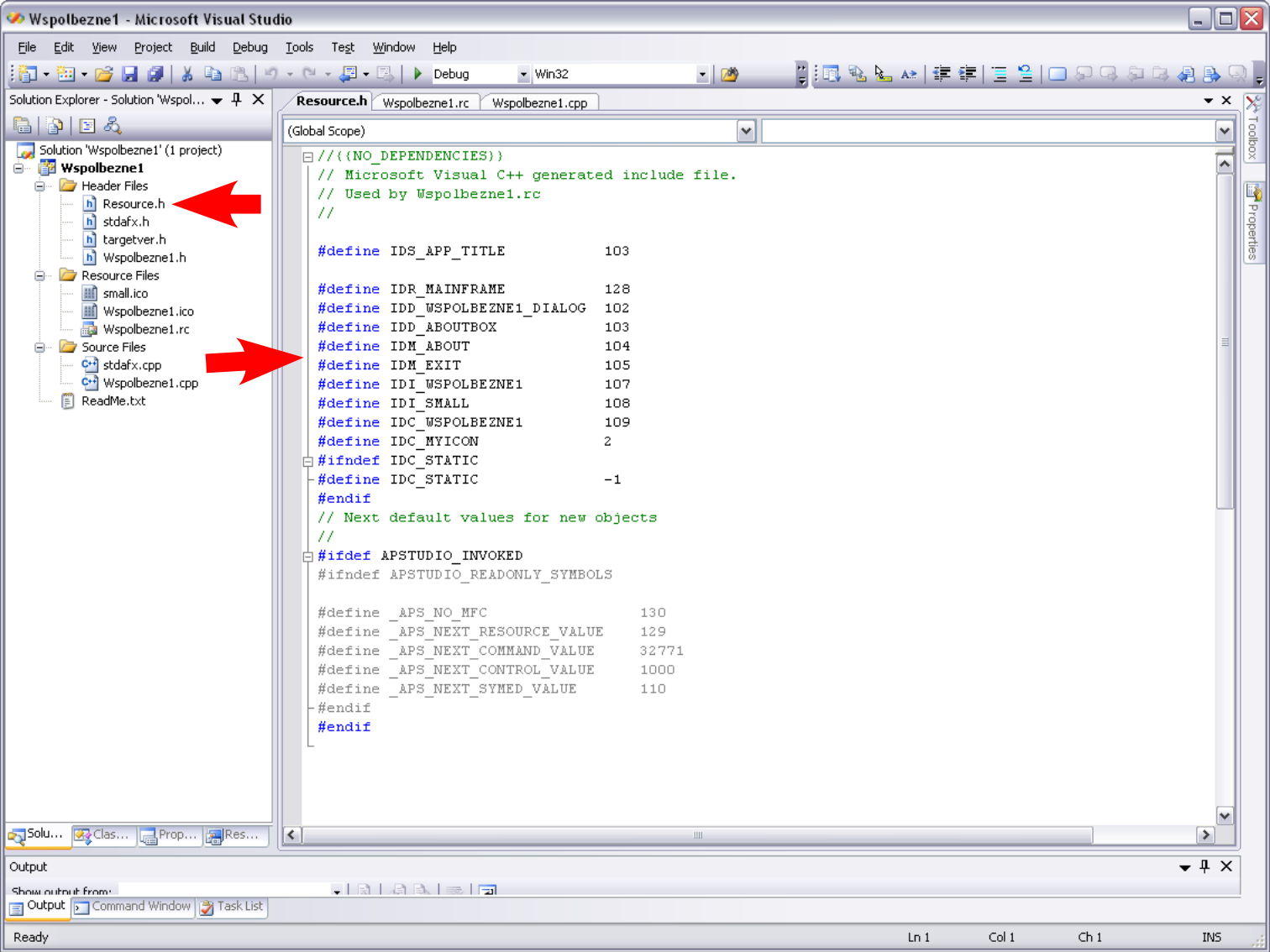

Wspolbezne1 - Microsoft Visual Studio
File Edit View Project Build Debug Tools Test Window Help
Debug
Win32
Solution Explorer - Solution 'Wspol...
Solution 'Wspolbezne1' (1 project)
Wspolbezne1
Header Files
Resource.h
stdafx.h
targetver.h
Wspolbezne1.h
Resource Files
small.ico
Wspolbezne1.ico
Wspolbezne1.rc
Source Files
stdafx.cpp
Wspolbezne1.cpp
ReadMe.txt
Resource.h
Wspolbezne1.cpp
(Global Scope)
int APIENTRY _tWinMain(HINSTANCE hInstance,
HINSTANCE hPrevInstance,
LPTSTR lpCmdLine,
int nCmdShow)
{
UNREFERENCED_PARAMETER(hPrevInstance);
UNREFERENCED_PARAMETER(lpCmdLine);
// TODO: Place code here.
MSG msg;
HACCEL hAccelTable;
// Initialize global strings
LoadString(hInstance, IDS_APP_TITLE, szTitle, MAX_LOADSTRING);
LoadString(hInstance, IDC_WSPOLBEZNE1, szWindowClass, MAX_LOADSTRING);
MyRegisterClass(hInstance);
// Perform application initialization:
if (!InitInstance (hInstance, nCmdShow))
{
return FALSE;
}
hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC_WSPOLBEZNE1));
// Main message loop:
while (GetMessage(&msg, NULL, 0, 0))
{
if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))
{
TranslateMessage(&msg);
DispatchMessage(&msg);
}
}
return (int) msg.wParam;
}
Toolbox
Properties
Output
Ready
Ln 1
Col 1
Ch 1
INS


Wspolbezne1 - Microsoft Visual Studio
File Edit View Project Build Debug Tools Test Window Help
Debug
Solution Explorer - Solution 'Wspol...
Solution 'Wspolbezne1' (1 project)
Wspolbezne1
Header Files
Resource.h
stdafx.h
targetver.h
Wspolbezne1.h
Resource Files
small.ico
Wspolbezne1.ico
Wspolbezne1.rc
Source Files
stdafx.cpp
Wspolbezne1.cpp
ReadMe.txt
Resource.h
Wspolbezne1.rc
(Global Scope)
//
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
int wmId, wmEvent;
PAINTSTRUCT ps;
HDC hdc;
switch (message)
{
case WM_COMMAND:
wmId = LOWORD(wParam);
wmEvent = HIWORD(wParam);
// Parse the menu selections:
switch (wmId)
{
case IDM_ABOUT:
DialogBox(hInst, MAKEINTRESOURCE(IDD_ABOUTBOX), hWnd, About);
break;
case IDM_EXIT:
DestroyWindow(hWnd);
break;
default:
return DefWindowProc(hWnd, message, wParam, lParam);
}
break;
case WM_PAINT:
hdc = BeginPaint(hWnd, &ps);
// TODO: Add any drawing code here...
EndPaint(hWnd, &ps);
break;
case WM_DESTROY:
PostQuitMessage(0);
break;
default:
return DefWindowProc(hWnd, message, wParam, lParam);
}
Output
Output
Command Window
Task List
Ready
Ln 1
Col 1
Ch 1
INS
Toolbox
Properties

Pętla zdarzeń (jeden wątek wiele procesów!)




Wspolbezne1 - Microsoft Visual Studio
File Edit View Project Build Debug Tools Test Window Help
Debug
Win32
Solution Explorer - Solution 'Wspol...
Solution 'Wspolbezne1' (1 project)
Wspolbezne1
Header Files
Resource.h
stdafx.h
targetver.h
Wspolbezne1.h
Resource Files
small.ico
Wspolbezne1.ico
Wspolbezne1.rc
Source Files
stdafx.cpp
Wspolbezne1.
ReadMe.txt
Resource.h
Wspolbezne1.rc
Wspolbezne1.cpp
// Menu
//
IDC_WSPOLBEZNE1 MENU
BEGIN
POPUP "&File"
BEGIN
MENUITEM "E&xit", IDM_EXIT
END
POPUP "&Help"
BEGIN
MENUITEM "&About ...", IDM_ABOUT
END
END
ator
EZNE1 ACCELERATORS
IDM_ABOUT, ASCII, ALT
IDM_ABOUT, ASCII, ALT
IDD_ABOUTBOX DIALOGEX 0, 0, 170, 62
STYLE DS_SETFONT | DS_MODALFRAME | DS_FIXEDSYS | WS_POPUP | WS_CAPTION | WS_SYSMENU
Open
Open With...
View Code
View Class Diagram
Compile
Exclude From Project
Cut
Copy
Remove
Rename
Properties
Output
Output
Command Window
Task List
Ready
Ln 1
Col 1
Ch 1
INS


Wspolbezne1 - Microsoft Visual Studio
File Edit View Project Build Debug Tools Test Window Help
Debug
Win32
Solution Explorer - Solution 'Wspol...
Solution 'Wspolbezne1' (1 project)
Wspolbezne1
Header Files
Resource.h
stdafx.h
targetver.h
Wspolbezne1.h
Resource Files
small.ico
Wspolbezne1.ico
Wspolbezne1.rc
Source Files
stdafx.cpp
Wspolbezne1.cpp
ReadMe.txt
Resource.h
Wspolbezne1.rc
Wspolbezne1.cpp
(Global Scope)
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by Wspolbezne1.rc
//
#define IDS_APP_TITLE 103
#define IDR_MAINFRAME 128
#define IDD_WSPOLBEZNE1_DIALOG 102
#define IDD_ABOUTBOX 103
#define IDM_ABOUT 104
#define IDM_EXIT 105
#define IDI_WSPOLBEZNE1 107
#define IDI_SMALL 108
#define IDC_WSPOLBEZNE1 109
#define IDC_MYICON 2
#ifndef IDC_STATIC
#define IDC_STATIC -1
#endif
// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NO_MFC 130
#define _APS_NEXT_RESOURCE_VALUE 129
#define _APS_NEXT_COMMAND_VALUE 32771
#define _APS_NEXT_CONTROL_VALUE 1000
#define _APS_NEXT_SYMED_VALUE 110
#endif
#endif
Output
Output
Command Window
Task List
Ready
Ln 1
Col 1
Ch 1
INS
Toolbox
Properties

Projekt programu w Visual Studio

Resource.h

```
#define  IDM_OBLICZ      155
```

Wspolbiezne1.rc

```
IDC_WSPOLBEZNE1 MENU
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "E&xit", IDM_EXIT
    END

    MENUITEM "&Oblicz", IDM_OBLICZ

    POPUP "&Help"
    BEGIN
        MENUITEM "&About ...", IDM_ABOUT
    END
END
```

Projekt programu w Visual Studio

Wspolbiezne1.cpp

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
        ...

        switch (message)
        {
        case WM_COMMAND:
                ...

                switch (wmId)
                {
                case IDM_OBLICZ:
                        MessageBox(hWnd, TEXT("Rozpoczynam"), TEXT("Obliczenia"), MB_OK);
                        Sleep(10000);  //Symuluje 10.sekundowe obliczenia
                        MessageBox(hWnd, TEXT("Koncze"), TEXT("Obliczenia"), MB_OK);
                        break;
                ...
```

Search MSDN with Bing

- MSDN Library
- Windows Development
- System Services
- Processes and Threads
- Process and Thread Reference
  - Process and Thread Functions
    - AssignProcessToJobObject
    - AttachThreadInput
    - AvQuerySystemResponsiveness
    - AvRevertMmThreadCharacteristics
    - AvRtCreateThreadOrderingGroup
    - AvRtCreateThreadOrderingGroupEx
    - AvRtDeleteThreadOrderingGroup
    - AvRtJoinThreadOrderingGroup
    - AvRtLeaveThreadOrderingGroup
    - AvRtWaitOnThreadOrderingGroup
    - AvSetMmMaxThreadCharacteristics
    - AvSetMmThreadCharacteristics
    - AvSetMmThreadPriority
    - BindIoCompletionCallback
    - CallbackMayRunLong
    - CancelThreadpoolIo
    - CleanupGroupCancelCallback
    - CloseThreadpool
    - CloseThreadpoolCleanupGroup
    - CloseThreadpoolCleanupGroupMembers
    - CloseThreadpoolIo
    - CloseThreadpoolTimer
    - CloseThreadpoolWait
    - CloseThreadpoolWork
    - ConvertFiberToThread
    - ConvertThreadToFiber
    - ConvertThreadToFiberEx
    - CreateFiber
    - CreateFiberEx
    - CreateJobObject
    - CreateProcess
    - CreateProcessAsUser

Sleep function

Applies to: desktop apps only

Suspends the execution of the current thread until the time-out interval elapses.

To enter an alertable wait state, use the **SleepEx** function.

Syntax

Copy

```
VOID WINAPI Sleep(
  __in  DWORD dwMilliseconds
);
```

Parameters

dwMilliseconds [in]

The time interval for which execution is to be suspended, in milliseconds.

A value of zero causes the thread to relinquish the remainder of its time slice to any other thread that is ready to run. If there are no other threads ready to run, the function returns immediately, and the thread continues execution.

Windows XP: A value of zero causes the thread to relinquish the remainder of its time slice to any other thread of equal priority that is ready to run. If there are no other threads of equal priority ready to run, the function returns immediately, and the thread continues execution. This behavior changed starting with Windows Server 2003.

A value of INFINITE indicates that the suspension should not time out.

Return value

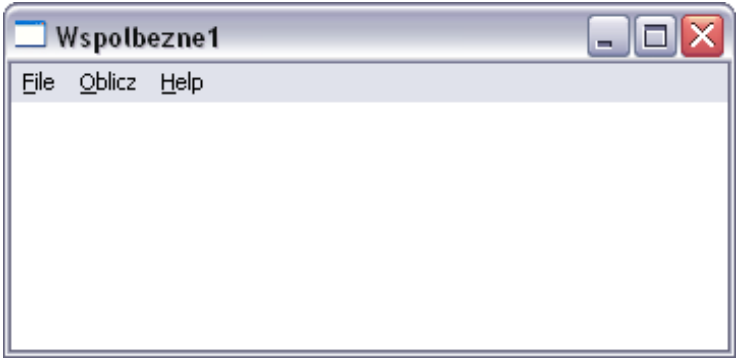
This function does not return a value.

Remarks

This function causes a thread to relinquish the remainder of its time slice and become unrunnable for an interval based on the value of *dwMilliseconds*. The system clock "ticks" at a constant rate. If *dwMilliseconds* is less than the resolution of the system clock, the thread may sleep for less than the specified length of time. If *dwMilliseconds* is greater than one tick but less than two, the wait can be anywhere between one and two ticks, and so on. To increase the accuracy of the sleep interval, call the **timeGetDevCaps** function to determine the supported minimum timer resolution and the **timeBeginPeriod** function to set the timer resolution to its minimum. Use caution when

Efekt blokowania interfejsu

1. Uruchomić program (zielony trójkąt) 



2. Wybrać opcję “Oblicz”
3. Pojawi się okno z informacją o rozpoczęciu obliczeń – wcisnąć “OK”

   i...

   Program jest martwy przez 10 sekund!

Dodanie wątku obliczeniowego

```
HWND okno;
DWORD watek_id;
HANDLE watek;

DWORD WatekObliczen(LPVOID param)
{
        MessageBox(okno, TEXT("Rozpoczynam"), TEXT("Obliczenia"), MB_OK);
        Sleep(10000);
        MessageBox(okno, TEXT("Koncze"), TEXT("Obliczenia"), MB_OK);
        return 0;
}

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
        ...

                case IDM_OBLICZ:
                        okno = hWnd;
                        watek = CreateThread(NULL, 0,
                                        (LPTHREAD_START_ROUTINE)WatekObliczen,
                                         NULL, 0, &watek_id);
                        break;
```

Search MSDN with Bing

- MSDN Library
- Windows Development
- System Services
- Processes and Threads
- Process and Thread Reference
  - Process and Thread Functions
    - AssignProcessToJobObject
    - AttachThreadInput
    - AvQuerySystemResponsiveness
    - AvRevertMmThreadCharacteristics
    - AvRtCreateThreadOrderingGroup
    - AvRtCreateThreadOrderingGroupEx
    - AvRtDeleteThreadOrderingGroup
    - AvRtJoinThreadOrderingGroup
    - AvRtLeaveThreadOrderingGroup
    - AvRtWaitOnThreadOrderingGroup
    - AvSetMmMaxThreadCharacteristics
    - AvSetMmThreadCharacteristics
    - AvSetMmThreadPriority
    - BindIoCompletionCallback
    - CallbackMayRunLong
    - CancelThreadpoolIo
    - CleanupGroupCancelCallback
    - CloseThreadpool
    - CloseThreadpoolCleanupGroup
    - CloseThreadpoolCleanupGroupMembers
    - CloseThreadpoolIo
    - CloseThreadpoolTimer
    - CloseThreadpoolWait
    - CloseThreadpoolWork
    - ConvertFiberToThread
    - ConvertThreadToFiber
    - ConvertThreadToFiberEx
    - CreateFiber
    - CreateFiberEx
    - CreateJobObject
    - CreateProcess
    - CreateProcessAsUser

CreateThread function

Applies to: desktop apps only

Creates a thread to execute within the virtual address space of the calling process.

To create a thread that runs in the virtual address space of another process, use the **CreateRemoteThread** function.

Syntax

Copy

```
HANDLE WINAPI CreateThread(
  __in_opt   LPSECURITY_ATTRIBUTES lpThreadAttributes,
  __in       SIZE_T dwStackSize,
  __in       LPTHREAD_START_ROUTINE lpStartAddress,
  __in_opt   LPVOID lpParameter,
  __in       DWORD dwCreationFlags,
  __out_opt  LPDWORD lpThreadId
);
```

Parameters

lpThreadAttributes [in, optional]

A pointer to a **SECURITY_ATTRIBUTES** structure that determines whether the returned handle can be inherited by child processes. If *lpThreadAttributes* is NULL, the handle cannot be inherited.

The **lpSecurityDescriptor** member of the structure specifies a security descriptor for the new thread. If *lpThreadAttributes* is NULL, the thread gets a default security descriptor. The ACLs in the default security descriptor for a thread come from the primary token of the creator.

Windows XP: The ACLs in the default security descriptor for a thread come from the primary or impersonation token of the creator. This behavior changed with Windows XP with SP2 and Windows Server 2003. For more information, see Remarks.

dwStackSize [in]

The initial size of the stack, in bytes. The system rounds this value to the nearest page. If this parameter is zero, the new thread uses the default size for the executable. For more information, see Thread Stack Size.

lpStartAddress [in]

Dodanie wątku obliczeniowego

1. Uruchomić program (zielony trójkąt)
2. Wybrać opcję “Oblicz”
3. Pojawi się okno z informacją o rozpoczęciu obliczeń – wcisnąć “OK”

   i...
4. Wybrać opcję “Oblicz”
5. Pojawi się okno z informacją o rozpoczęciu obliczeń – wcisnąć “OK”

Uruchomiliśmy dwa niezależne wątki obliczeniowe!

Interfejs graficzny się nie blokuje!

Dodanie wątku obliczeniowego

Zablokowanie wielokrotnego uruchamiania wątku przez wykorzystanie uchwytu wątku jako „mutexa”.

```
HWND okno;
DWORD watek_id;
HANDLE watek = NULL;

DWORD WatekObliczen(LPVOID param)
{
//   MessageBox(okno, TEXT("Rozpoczynam"), TEXT("Obliczenia"), MB_OK);
     Sleep(10000);
     MessageBox(okno, TEXT("Koncze"), TEXT("Obliczenia"), MB_OK);
     watek = NULL;
     return 0;
}
```

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
      ...

            case IDM_OBLICZ:
                  okno = hWnd;
                  if(watek != NULL)
                  {
                        MessageBox(hWnd, TEXT("Juz dziala"),
                                    TEXT("Obliczenia"), MB_OK);
                  }
                  else
                  {
                      MessageBox(hWnd, TEXT("Rozpoczynam"),
                                  TEXT("Obliczenia"), MB_OK);

                      watek = CreateThread(NULL, 0,
                                    (LPTHREAD_START_ROUTINE)WatekObliczen,
                                    NULL, 0, &watek_id);
                  }
                  break;
```

Komunikowanie się wątków

Wątek obliczeniowy może poinformować wątek GUI o zakończeniu działania generując zdarzenie. Umieszcza to zdarzenie w kolejce zdarzeń programu za pomocą funkcji PostMessage(). Wykorzystano kod zdarzenia „użytkownika” WM_USER.

```
DWORD WatekObliczen(LPVOID param)
{
      Sleep(10000);
      watek = NULL;
      PostMessage(okno, WM_USER, 0, 0);
      return 0;
}
```

Search MSDN with Bing

- MSDN Library
- Windows Development
- Windows Application UI Development
- Windows and Messages
- Messages and Message Queues
- Message Reference
  - Message Functions
    - BroadcastSystemMessage
    - BroadcastSystemMessageEx
    - DispatchMessage
    - GetInputState
    - GetMessage
    - GetMessageExtraInfo
    - GetMessagePos
    - GetMessageTime
    - GetQueueStatus
    - InSendMessage
    - InSendMessageEx
    - PeekMessage
    - **PostMessage**
    - PostQuitMessage
    - PostThreadMessage
    - RegisterWindowMessage
    - ReplyMessage
    - SendAsyncProc
    - SendMessage
    - SendMessageCallback
    - SendMessageTimeout
    - SendNotifyMessage
    - SetMessageExtraInfo
    - TranslateMessage
    - WaitMessage

Community Content

SIGN IN PROBLEM...

PostMessage function

Applies to: desktop apps only

Places (posts) a message in the message queue associated with the thread that created the specified window and returns without waiting for the thread to process the message.

To post a message in the message queue associated with a thread, use the **PostThreadMessage** function.

Syntax

```
BOOL WINAPI PostMessage(
  __in_opt  HWND hWnd,
  __in      UINT Msg,
  __in      WPARAM wParam,
  __in      LPARAM lParam
);
```

Parameters

hWnd [in, optional]

Type: **HWND**

A handle to the window whose window procedure is to receive the message. The following values have special meanings.

Value	Meaning
HWND_BROADCAST ((HWND)0xffff)	The message is posted to all top-level windows in the system, including disabled or invisible unowned windows, overlapped windows, and pop-up windows. The message is not posted to child windows.
NULL	The function behaves like a call to **PostThreadMessage** with the *dwThreadId* parameter set to the identifier of the current thread.

Starting with Windows Vista, message posting is subject to UIPI. The thread of a process can post messages only to message

Komunikowanie się wątków

Wątek GUI obsługuje zdarzenie WM_USER w funkcji WndProc.

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
	...
	switch (message)
	{
	case WM_USER:
		MessageBox(hWnd, TEXT("Koncze"), TEXT("Obliczenia"), MB_OK);
		break;
		...
```

Search MSDN with Bing

- MSDN Library
- Windows Development
- Windows Application UI Development
- Windows and Messages
- Messages and Message Queues
- Message Reference
  - Message Constants
    - OCM_BASE
    - WM_APP
    - WM_USER

WM_USER

Applies to: desktop apps only

Used to define private messages for use by private window classes, usually of the form **WM_USER**+*x*, where *x* is an integer value.

```
#define WM_USER                         0x0400
```

Remarks

The following are the ranges of message numbers.

Range	Meaning
0 through **WM_USER** –1	Messages reserved for use by the system.
WM_USER through 0x7FFF	Integer messages for use by private window classes.
WM_APP (0x8000) through 0xBFFF	Messages available for use by applications.
0xC000 through 0xFFFF	String messages for use by applications.
Greater than 0xFFFF	Reserved by the system.

Message numbers in the first range (0 through **WM_USER** –1) are defined by the system. Values in this range that are not explicitly defined are reserved by the system.

Message numbers in the second range (**WM_USER** through 0x7FFF) can be defined and used by an application to send messages within a private window class. These values cannot be used to define messages that are meaningful throughout an application because some predefined window classes already define values in this range. For example, predefined control classes such as **BUTTON**, **EDIT**, **LISTBOX**, and **COMBOBOX** may use these values. Messages in this range should not be sent to other applications unless the applications have been designed to exchange messages and to attach the same meaning to the message numbers.

Message numbers in the third range (0x8000 through 0xBFFF) are available for applications to use as private messages. Messages in

Wywołanie programu zewnętrznego

Tworzenie wątku – CreateThread()

Tworzenie procesu... – CreateProcess()

Search MSDN with Bing

- MSDN Library
- Windows Development
- System Services
- Processes and Threads
- Process and Thread Reference
  - Process and Thread Functions
    - AssignProcessToJobObject
    - AttachThreadInput
    - AvQuerySystemResponsiveness
    - AvRevertMmThreadCharacteristics
    - AvRtCreateThreadOrderingGroup
    - AvRtCreateThreadOrderingGroupEx
    - AvRtDeleteThreadOrderingGroup
    - AvRtJoinThreadOrderingGroup
    - AvRtLeaveThreadOrderingGroup
    - AvRtWaitOnThreadOrderingGroup
    - AvSetMmMaxThreadCharacteristics
    - AvSetMmThreadCharacteristics
    - AvSetMmThreadPriority
    - BindIoCompletionCallback
    - CallbackMayRunLong
    - CancelThreadpoolIo
    - CleanupGroupCancelCallback
    - CloseThreadpool
    - CloseThreadpoolCleanupGroup
    - CloseThreadpoolCleanupGroupMembers
    - CloseThreadpoolIo
    - CloseThreadpoolTimer
    - CloseThreadpoolWait
    - CloseThreadpoolWork
    - ConvertFiberToThread
    - ConvertThreadToFiber
    - ConvertThreadToFiberEx
    - CreateFiber
    - CreateFiberEx
    - CreateJobObject
    - CreateProcess

CreateProcess function

Applies to: desktop apps only

Creates a new process and its primary thread. The new process runs in the security context of the calling process.

If the calling process is impersonating another user, the new process uses the token for the calling process, not the impersonation token. To run the new process in the security context of the user represented by the impersonation token, use the **CreateProcessAsUser** or **CreateProcessWithLogonW** function.

Syntax

```
BOOL WINAPI CreateProcess(
  __in_opt     LPCTSTR lpApplicationName,
  __inout_opt  LPTSTR lpCommandLine,
  __in_opt     LPSECURITY_ATTRIBUTES lpProcessAttributes,
  __in_opt     LPSECURITY_ATTRIBUTES lpThreadAttributes,
  __in         BOOL bInheritHandles,
  __in         DWORD dwCreationFlags,
  __in_opt     LPVOID lpEnvironment,
  __in_opt     LPCTSTR lpCurrentDirectory,
  __in         LPSTARTUPINFO lpStartupInfo,
  __out        LPPROCESS_INFORMATION lpProcessInformation
);
```

Parameters

lpApplicationName [in, optional]

The name of the module to be executed. This module can be a Windows-based application. It can be some other type of module (for example, MS-DOS or OS/2) if the appropriate subsystem is available on the local computer.

The string can specify the full path and file name of the module to execute or it can specify a partial name. In the case of a partial name, the function uses the current drive and current directory to complete the specification. The function will not use the search path. This parameter must include the file name extension; no default extension is assumed.

The *lpApplicationName* parameter can be **NULL**. In that case, the module name must be the first white space–delimited token in the *lpCommandLine* string. If you are using a long file name that contains a space, use quoted strings to indicate where the file

Wywołanie programu zewnętrznego

Nowy proces (program) uruchomimy wewnątrz wątku obliczeniowego.

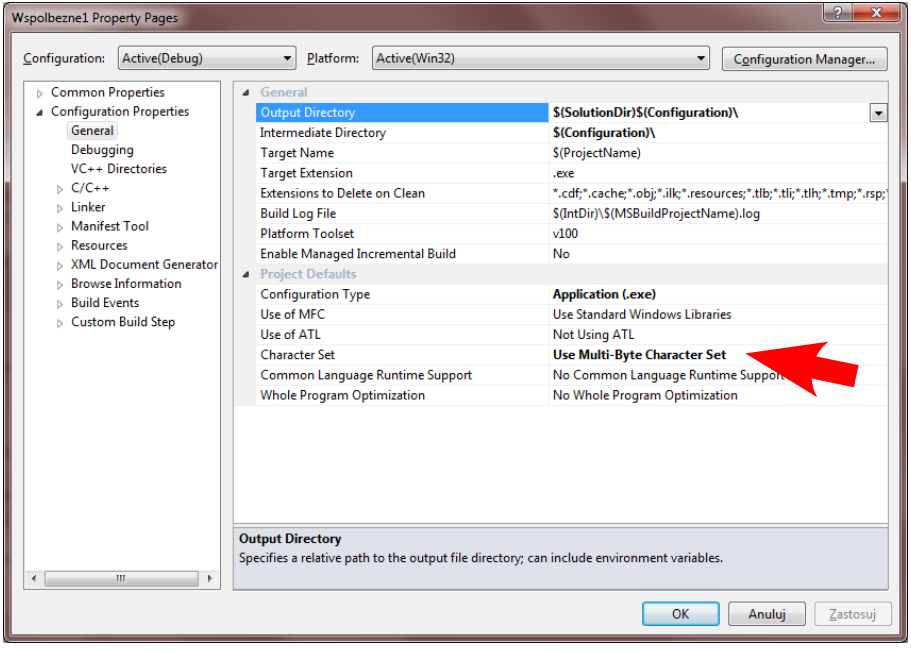
Wątek obliczeń kończy się zanim zostanie zamknięte okno notatnika!

```
DWORD WatekObliczen(LPVOID param)
{
    STARTUPINFO si;
    PROCESS_INFORMATION pi;
    ZeroMemory( &si, sizeof(si) );
    si.cb = sizeof(si);
    ZeroMemory( &pi, sizeof(pi) );

    //Sleep(10000);
    CreateProcess(NULL, TEXT("notepad.exe"), NULL, NULL,
                  FALSE, 0, NULL, NULL, &si, &pi);
    watek = NULL;
    PostMessage(okno, WM_USER, 0, 0);
    return 0;
}
```

Jeśli są problemy

Zmienić właściwości projektu na *Use Multi-Byte Character Set*



Wywołanie programu zewnętrznego

Chcemy zakończyć wątek obliczeń dopiero po zamknięciu okna notatnika.

Wykorzystujemy synchronizację – wstrzymujemy zamknięcie wątku do momentu zakończenia procesu notatnika.

```
DWORD WatekObliczen(LPVOID param)
{
    ...

    CreateProcess(NULL, TEXT("notepad.exe"), NULL, NULL,
                  FALSE, 0, NULL, NULL, &si, &pi);

    WaitForSingleObject(pi.hProcess, INFINITE);
    CloseHandle(pi.hProcess);
    CloseHandle(pi.hThread);

    watek = NULL;
    PostMessage(okno, WM_USER, 0, 0);
    return 0;
}
```

Search MSDN with Bing

- MSDN Library
- Windows Development
- System Services
- Synchronization
- Synchronization Reference
  - Synchronization Functions
    - AcquireSRWLockExclusive
    - AcquireSRWLockShared
    - AddSIDToBoundaryDescriptor
    - AddIntegrityLabelToBoundaryDescriptor
    - APCProc
    - CancelWaitableTimer
    - ChangeTimerQueueTimer
    - ClosePrivateNamespace
    - CreateBoundaryDescriptor
    - CreateEvent
    - CreateEventEx
    - CreateMutex
    - CreateMutexEx
    - CreatePrivateNamespace
    - CreateSemaphore
    - CreateSemaphoreEx
    - CreateTimerQueue
    - CreateTimerQueueTimer
    - CreateWaitableTimer
    - CreateWaitableTimerEx
    - DeleteBoundaryDescriptor
    - DeleteCriticalSection
    - DeleteSynchronizationBarrier
    - DeleteTimerQueue
    - DeleteTimerQueueEx
    - DeleteTimerQueueTimer
    - EnterCriticalSection
    - EnterSynchronizationBarrier
    - GetOverlappedResult
    - GetOverlappedResultEx
    - InitializeConditionVariable

WaitForSingleObject function

Applies to: desktop apps only

Waits until the specified object is in the signaled state or the time-out interval elapses.

To enter an alertable wait state, use the **WaitForSingleObjectEx** function. To wait for multiple objects, use the **WaitForMultipleObjects**.

Syntax

```
DWORD WINAPI WaitForSingleObject(
  __in  HANDLE hHandle,
  __in  DWORD dwMilliseconds
);
```

Parameters

hHandle [in]

A handle to the object. For a list of the object types whose handles can be specified, see the following Remarks section.

If this handle is closed while the wait is still pending, the function's behavior is undefined.

The handle must have the **SYNCHRONIZE** access right. For more information, see Standard Access Rights.

dwMilliseconds [in]

The time-out interval, in milliseconds. If a nonzero value is specified, the function waits until the object is signaled or the interval elapses. If *dwMilliseconds* is zero, the function does not enter a wait state if the object is not signaled; it always returns immediately. If *dwMilliseconds* is **INFINITE**, the function will return only when the object is signaled.

Return value

If the function succeeds, the return value indicates the event that caused the function to return. It can be one of the following values.

Return code/value	Description

Sekcja krytyczna

```
EnterCriticalSection(&CriticalSection);

    Tutaj jest fragment kodu, który może być wykonywany tylko
    przez pojedynczy wątek w tym samym czasie.

LeaveCriticalSection(&CriticalSection);
```

Using Critical Section Objects

The following example shows how a thread initializes, enters, and releases a critical section. It uses the **InitializeCriticalSectionAndSpinCount**, **EnterCriticalSection**, **LeaveCriticalSection**, and **DeleteCriticalSection** functions.

```
// Global variable
CRITICAL_SECTION CriticalSection;

int main( void )
{
    ...

    // Initialize the critical section one time only.
    if (!InitializeCriticalSectionAndSpinCount(&CriticalSection,
        0x00000400) )
        return;
    ...

    // Release resources used by the critical section object.
    DeleteCriticalSection(&CriticalSection);
}

DWORD WINAPI ThreadProc( LPVOID lpParameter )
{
    ...

    // Request ownership of the critical section.
    EnterCriticalSection(&CriticalSection);

    // Access the shared resource.

    // Release ownership of the critical section.
    LeaveCriticalSection(&CriticalSection);

    ...
return 1;
}
```

InitializeCriticalSectionAndSpinCount function

Applies to: desktop apps only

Initializes a critical section object and sets the spin count for the critical section. When a thread tries to acquire a critical section that is locked, the thread *spins*: it enters a loop which iterates spin count times, checking to see if the lock is released. If the lock is not released before the loop finishes, the thread goes to sleep to wait for the lock to be released.

Syntax

```
BOOL WINAPI InitializeCriticalSectionAndSpinCount(
  __out  LPCRITICAL_SECTION lpCriticalSection,
  __in   DWORD dwSpinCount
);
```

Parameters

lpCriticalSection [out]

A pointer to the critical section object.

dwSpinCount [in]

The spin count for the critical section object. On single-processor systems, the spin count is ignored and the critical section spin count is set to 0 (zero). On multiprocessor systems, if the critical section is unavailable, the calling thread spins *dwSpinCount* times before performing a wait operation on a semaphore associated with the critical section. If the critical section becomes free during the spin operation, the calling thread avoids the wait operation.

Return value

This function always returns a nonzero value.

Windows Server 2003 and Windows XP: If the function succeeds, the return value is nonzero. If the function fails, the return value is zero (0). To get extended error information, call **GetLastError**. This behavior was changed starting with Windows Vista.

Sekcja krytyczna

Problemy:

1. Konieczność inicjowania i usuwania sekcji krytycznej.
2. Konieczność wywoływania funkcji *LeaveCriticvalSection*().

Zakończenie programu oraz wyjście z funkcji może zachodzić w różnych miejscach.

Trzeba z uwagą analizować kod i we wszystkich miejscach, w których program może się zakończyć wywołać funkcję *DeleteCriticalSection*(). We wszystkich miejscach, w których wychodzi się z sekcji krytycznej wywołać *LeaveCriticvalSection*(). Takich miejsc może być dużo i łatwo któreś przeoczyć!

Sekcja krytyczna

W C++ można zdefiniować dwie klasy: sekcji krytycznej i blokowania sekcji.

W konstruktorach klas umieszcza się elementy inicjujące a w destruktorach funkcje:

DeleteCriticalSection

oraz

LeaveCriticalSection.

Kompilator C++ zapewni wywołanie destruktora w odpowiednim czasie podczas opuszczania sekcji krytycznej oraz podczas zamykania programu. Przykładem implementacji takich klas są klasy (Microsoft plik wxutil.h):

CcritSec – wykorzystuje strukturę CRITICAL_SECTION

oraz

CautoLock – wywołuje *EnterCriticalSection* w konstruktorze i *LeaveCriticalSection* w destruktorze.

Sekcja krytyczna

```
class CAutoLock
{
   CAutoLock(const CAutoLock &refAutoLock);
   ...
protected:
   CCritSec * m_pLock;

public:
   CAutoLock(CCritSec * plock)
   {
      m_pLock = plock;
      m_pLock->Lock();
   };

   ~CAutoLock() {
      m_pLock->Unlock();
   };
};
```

```
class CCritSec {

   CCritSec(const CCritSec &refCritSec);
   ...
   CRITICAL_SECTION m_CritSec;
public:
   CCritSec() {
      InitializeCriticalSection(&m_CritSec);
   };
   ~CCritSec() {
      DeleteCriticalSection(&m_CritSec);
   };
   void Lock() {
      EnterCriticalSection(&m_CritSec);
   };
   void Unlock() {
      LeaveCriticalSection(&m_CritSec);
   };
};
```

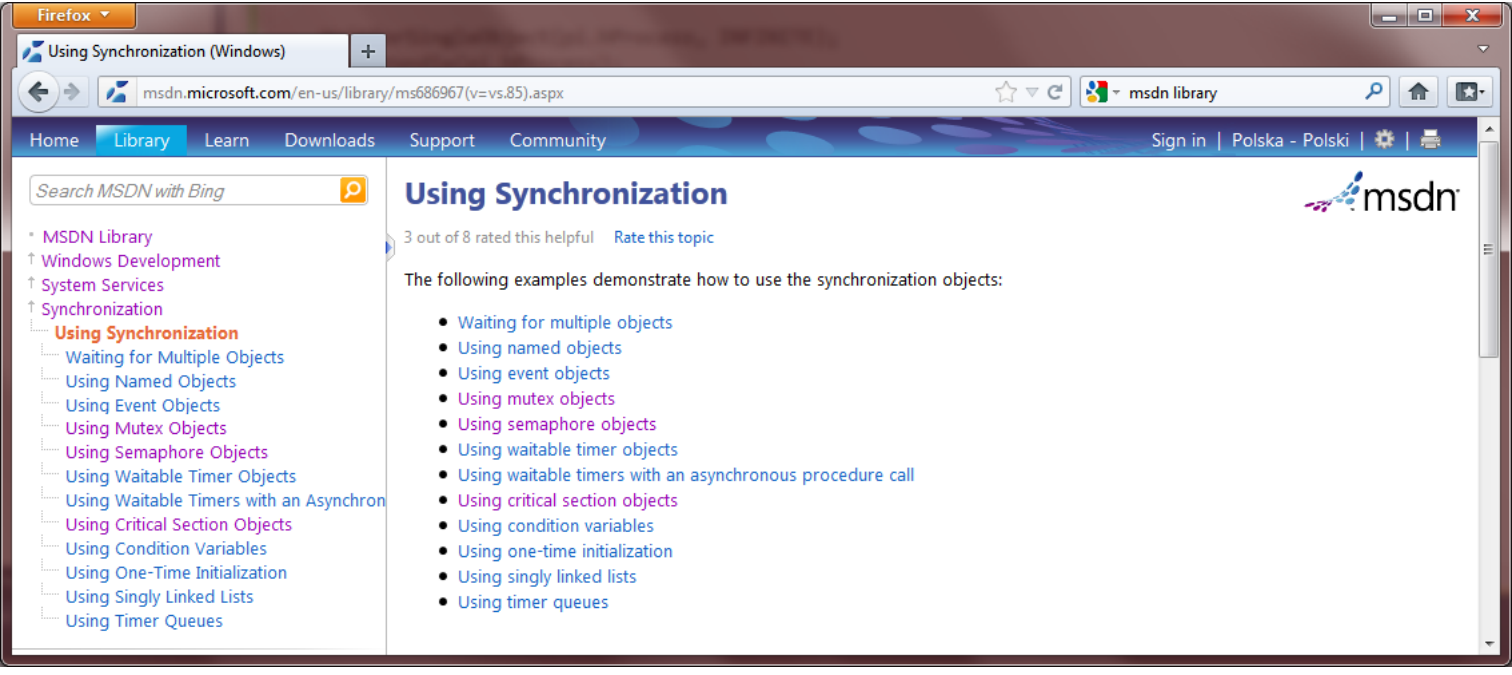
Sekcja krytyczna

Przykład użycia:

```
class KlasaZSekcjaKrytyczna
{
   public:
      int JakasFunkcja(void)
      {
         CAutoLock zablokuj_1(&blokada);
         ...
      }
      int InnaFunkcja(void)
      {...
         {
            CAutoLock zablokuj_2(&blokada);
            ...
         }...
      }

   private:
      CCritSec blokada;
};
```

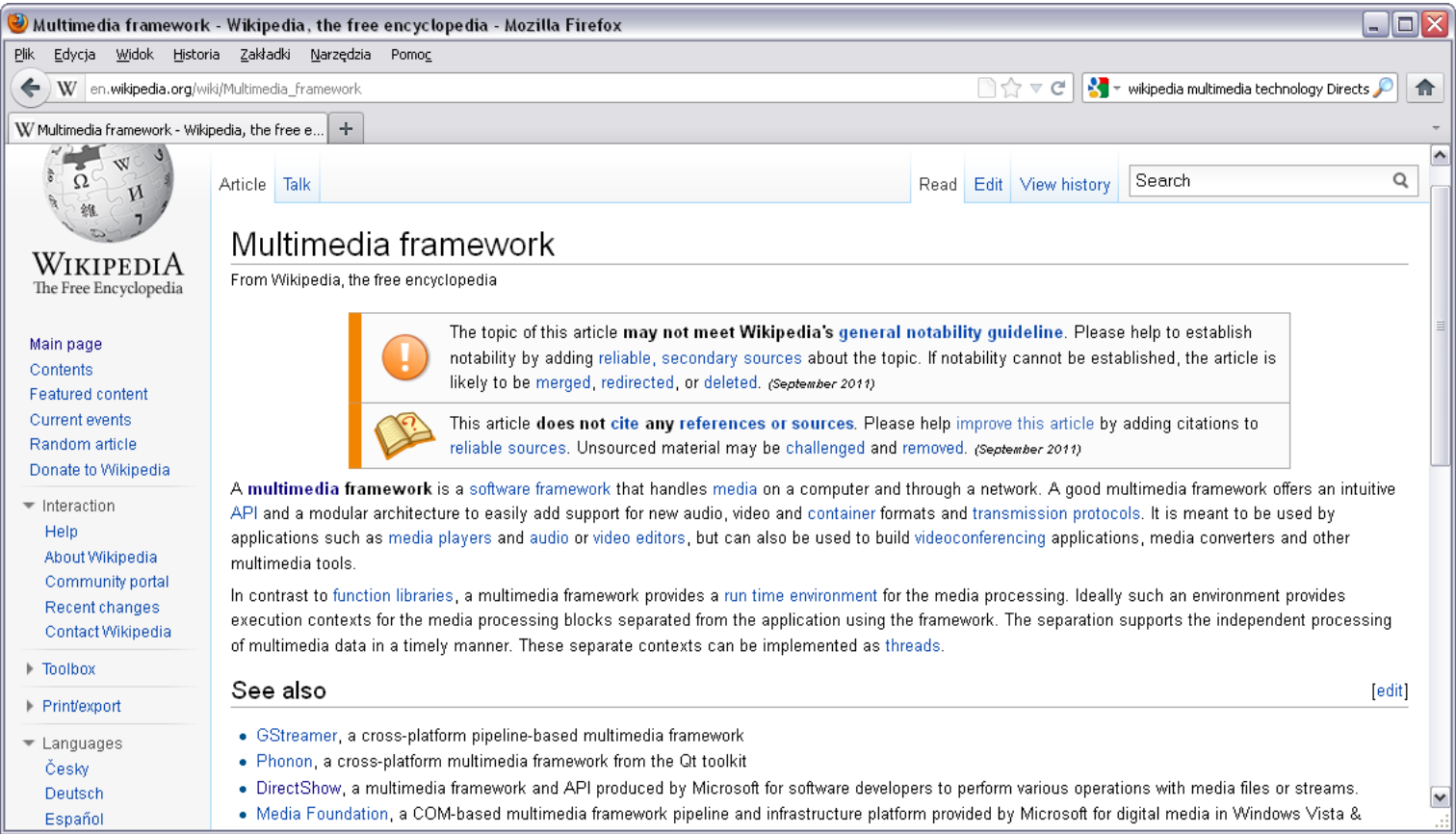
Inne narzędzia synchronizacji



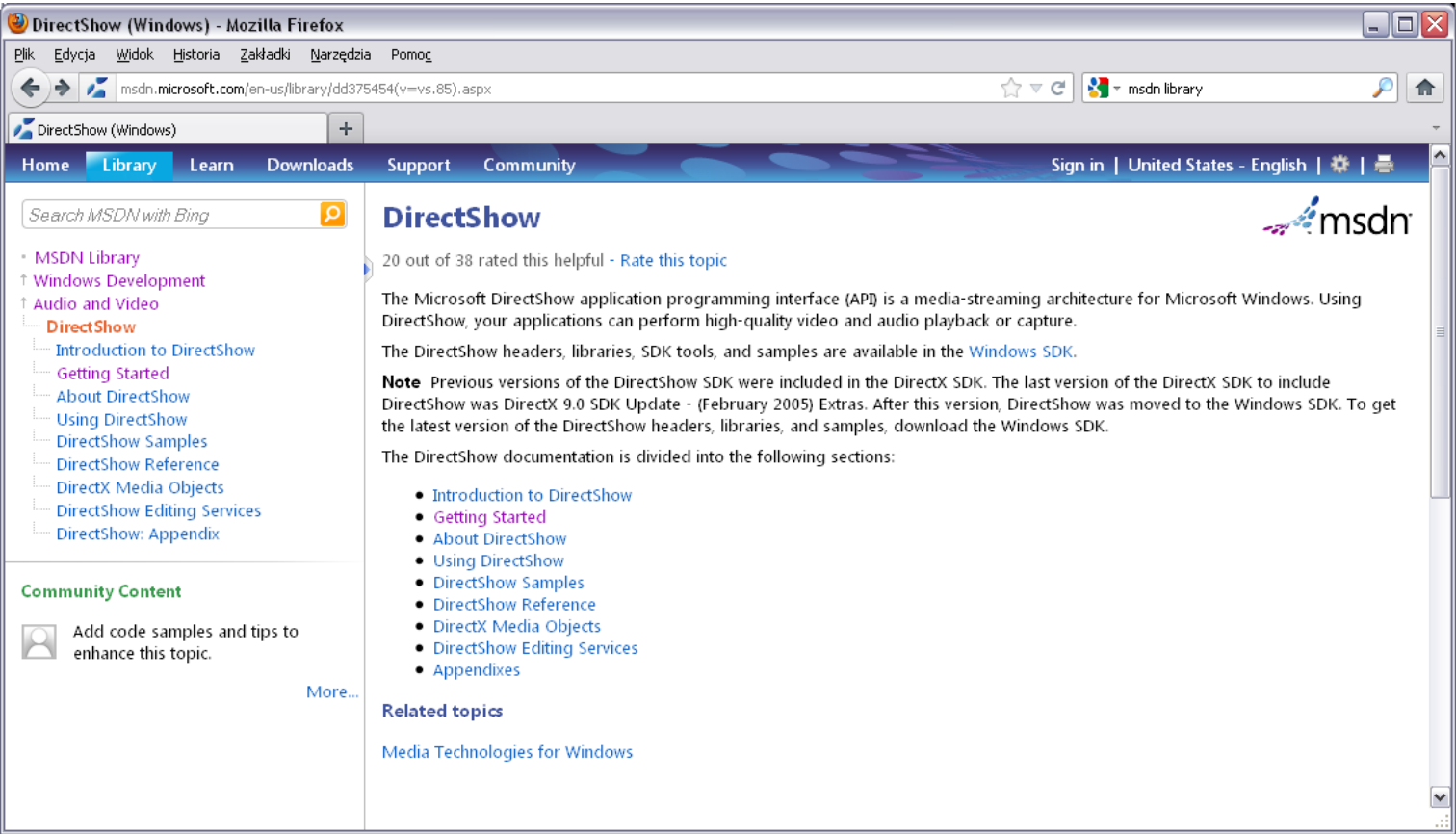
Synchronizacja w multimediach

1. Przetwarzanie potokowe (dodanie wątku rozdzielającego ścieżkę przetwarzania danych),
2. Rozplatanie danych wideo i audio (dodawanie oddzielnych wątków przetwarzania obrazu i dźwięku),
3. Łączenie potoków danych multimedialnych z różnych źródeł (splatanie danych wideo i audio, porównywanie danych),

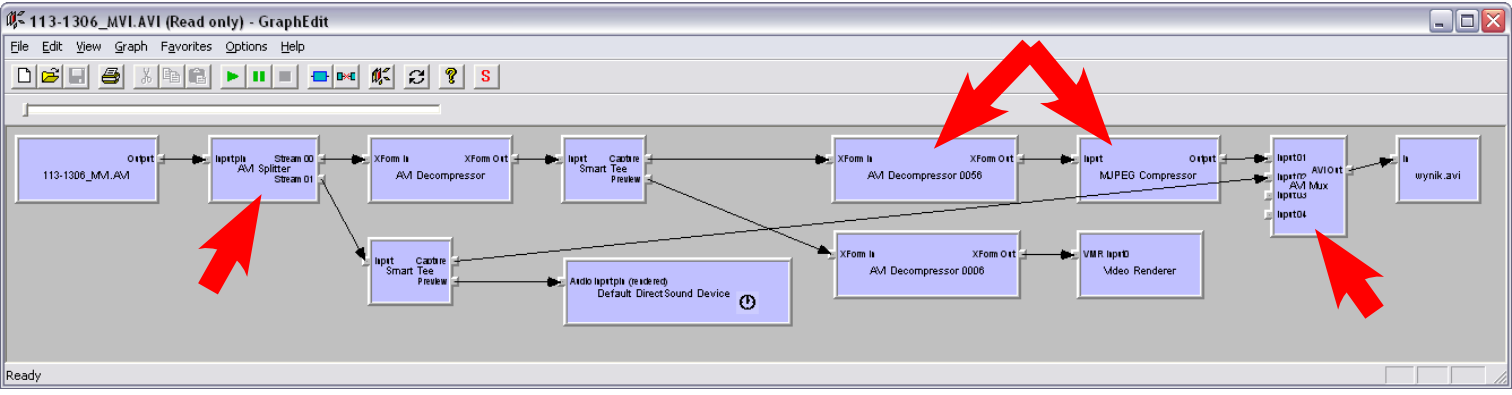
Technologia multimedialna



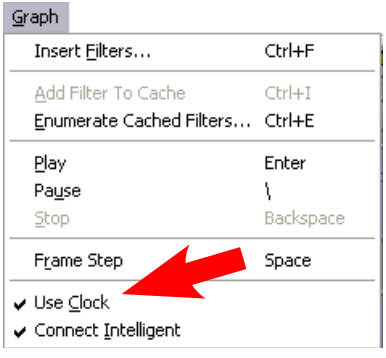
DirectShow


DirectShow (Windows) - Mozilla Firefox
Plik Edycja Widok Historia Zakładki Narzędzia Pomoc
msdn.microsoft.com/en-us/library/dd375454(v=vs.85).aspx
msdn library
DirectShow (Windows)
Home Library Learn Downloads Support Community
Sign in | United States - English
Search MSDN with Bing
MSDN Library
Windows Development
Audio and Video
DirectShow
Introduction to DirectShow
Getting Started
About DirectShow
Using DirectShow
DirectShow Samples
DirectShow Reference
DirectX Media Objects
DirectShow Editing Services
DirectShow: Appendix
Community Content
Add code samples and tips to enhance this topic.
More...
DirectShow
msdn
20 out of 38 rated this helpful - Rate this topic
The Microsoft DirectShow application programming interface (API) is a media-streaming architecture for Microsoft Windows. Using DirectShow, your applications can perform high-quality video and audio playback or capture.
The DirectShow headers, libraries, SDK tools, and samples are available in the Windows SDK.
Note Previous versions of the DirectShow SDK were included in the DirectX SDK. The last version of the DirectX SDK to include DirectShow was DirectX 9.0 SDK Update - (February 2005) Extras. After this version, DirectShow was moved to the Windows SDK. To get the latest version of the DirectShow headers, libraries, and samples, download the Windows SDK.
The DirectShow documentation is divided into the following sections:
Introduction to DirectShow
Getting Started
About DirectShow
Using DirectShow
DirectShow Samples
DirectShow Reference
DirectX Media Objects
DirectShow Editing Services
Appendixes
Related topics
Media Technologies for Windows

Grafy i filtry



1. Graf przetwarzania (przykład grafu rekompresji danych z podglądem) – program GraphEdit,
2. Filtr (moduł przetwarzania) – interfejsy.
3. Wątki: GUI, sterujący, przetwarzania danych (wiele),
4. Synchronizacja (pakietów danych oraz w czasie).



Interfejsy

CBaseInputPin class (Windows) - Mozilla Firefox

Plik Edycja Widok Historia Zakładki Narzędzia Pomoc

msdn.microsoft.com/en-us/library/dd318897(v=vs.85).aspx

graph edit

CBaseInputPin class (Windows)

Home | Library | Learn | Downloads | Support | Community | Sign in | United States - English

Search MSDN with Bing

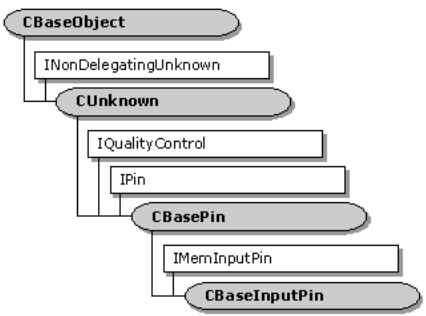
- MSDN Library
- Windows Development
- Audio and Video
- DirectShow
- DirectShow Reference
- DirectShow Base Classes
- DirectShow Base Class Reference
  - CBaseInputPin
    - m_pAllocator
    - m_bReadOnly
    - m_bFlushing
    - m_SampleProps
    - BeginFlush
    - BreakConnect
    - CBaseInputPin
    - ~CBaseInputPin
    - CheckStreaming
    - EndFlush
    - GetAllocator
    - GetAllocatorRequirements
    - Inactive
    - IsFlushing
    - IsReadOnly
    - Notify
    - NotifyAllocator

CBaseInputPin class

msdn

1 out of 1 rated this helpful - Rate this topic

Applies to: desktop apps only




```
HRESULT Receive(
  IMediaSample *pSample
);
```

The `CBaseInputPin` class is an abstract base class for implementing input pins. This class adds support for the **IMemInputPin** interface, in addition to the **IPin** interface support provided by **CBasePin**.

To use this class, derive a new class and override at least the following methods:

- **CBaseInputPin::BeginFlush**
- **CBaseInputPin::EndFlush**
- **CBaseInputPin::Receive**
- **CBasePin::CheckMediaType**
- **CBasePin::GetMediaType**

Interfejsy

CBaseOutputPin class (Windows) - Mozilla Firefox

Plik Edycja Widok Historia Zakładki Narzędzia Pomoc

msdn.microsoft.com/en-us/library/dd319032(v=vs.85).aspx

graph edit

CBaseOutputPin class (Windows)

Home | Library | Learn | Downloads | Support | Community | Sign in | United States - English

Search MSDN with Bing

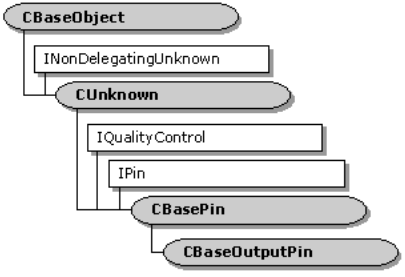
- MSDN Library
- Windows Development
- Audio and Video
- DirectShow
- DirectShow Reference
- DirectShow Base Classes
- DirectShow Base Class Reference
  - CBaseOutputPin
    - m_pAllocator
    - m_pInputPin
    - Active
    - BeginFlush
    - BreakConnect
    - CBaseOutputPin
    - CheckConnect
    - CompleteConnect
    - DecideAllocator
    - DecideBuffer
    - Deliver
    - DeliverBeginFlush
    - DeliverEndFlush
    - DeliverEndOfStream
    - DeliverNewSegment
    - EndFlush
    - EndOfStream

CBaseOutputPin class

msdn

This topic has not yet been rated - Rate this topic

Applies to: desktop apps only




```
virtual HRESULT Deliver(
  IMediaSample *pSample
);
```

The `CBaseOutputPin` class is an abstract base class that implements an output pin.

This class derives from **CBasePin**. It differs from **CBasePin** in the following respects:

- It connects only to input pins that support the **IMemInputPin** interface.
- It supports local memory transport through the **IMemAllocator** interface.
- It rejects end-of-stream, flush, and new-segment notifications. (These should not be sent to an output pin.)
- It provides methods for delivering samples downstream.

When the pin connects, it requests a memory allocator from the input pin. Failing that, it creates a new allocator object. The output pin is responsible for setting the allocator properties. It does this through the pure virtual method **CBaseOutputPin::DecideBufferSize**. Override this method in your derived class. If the input pin has any buffer requirements, they are passed to the **DecideBufferSize**

Przeczytaj

Introduction to DirectShow Application Programming (Windows) - Mozilla Firefox

Plik Edycja Widok Historia Zakładki Narzędzia Pomoc

msdn.microsoft.com/en-us/library/dd390352(v=vs.85).aspx

msdn library

Introduction to DirectShow Application Progr...

Home | Library | Learn | Downloads | Support | Community | Sign in | United States - English

Search MSDN with Bing

- MSDN Library
- Windows Development
- Audio and Video
- DirectShow
  - Getting Started
  - Building DirectShow Applications
  - Introduction to DirectShow Application
  - How To Play a File

Community Content

Add code samples and tips to enhance this topic.

More...

Introduction to DirectShow Application Programming

msdn

9 out of 10 rated this helpful - Rate this topic

This article introduces the basic terminology and concepts that are used in DirectShow. After reading this section, you will be ready to write your first DirectShow application.

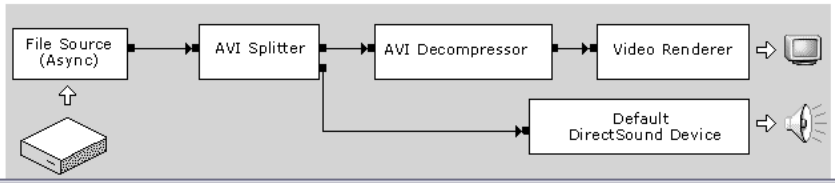
Filters and Filter Graphs

The building block of DirectShow is a software component called a *filter*. A filter is a software component that performs some operation on a multimedia stream. For example, DirectShow filters can

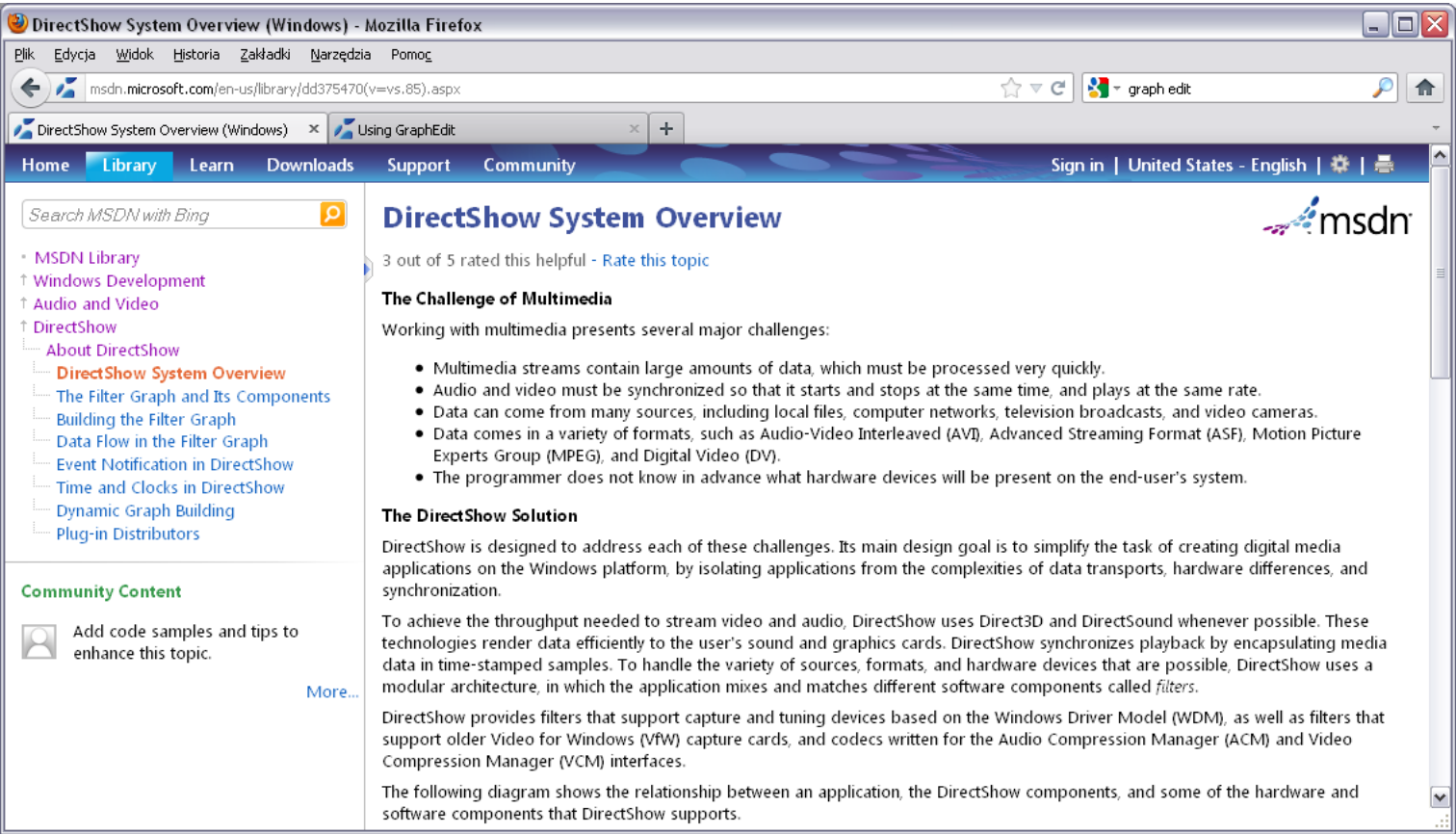
- read files
- get video from a video capture device
- decode various stream formats, such as MPEG-1 video
- pass data to the graphics or sound card

Filters receive input and produce output. For example, if a filter decodes MPEG-1 video, the input is the MPEG-encoded stream and the output is a series of uncompressed video frames.

In DirectShow, an application performs any task by connecting chains of filters together, so that the output from one filter becomes the input for another. A set of connected filters is called a *filter graph*. For example, the following diagram shows a filter graph for playing an AVI file.



Przeczytaj


DirectShow System Overview (Windows) - Mozilla Firefox
Plik Edycja Widok Historia Zakładki Narzędzia Pomoc
msdn.microsoft.com/en-us/library/dd375470(v=vs.85).aspx
graph edit
DirectShow System Overview (Windows)
Using GraphEdit
Home Library Learn Downloads Support Community
Sign in | United States - English
Search MSDN with Bing
MSDN Library
Windows Development
Audio and Video
DirectShow
About DirectShow
DirectShow System Overview
The Filter Graph and Its Components
Building the Filter Graph
Data Flow in the Filter Graph
Event Notification in DirectShow
Time and Clocks in DirectShow
Dynamic Graph Building
Plug-in Distributors
Community Content
Add code samples and tips to enhance this topic.
More...
DirectShow System Overview
msdn
3 out of 5 rated this helpful - Rate this topic
The Challenge of Multimedia
Working with multimedia presents several major challenges:
Multimedia streams contain large amounts of data, which must be processed very quickly.
Audio and video must be synchronized so that it starts and stops at the same time, and plays at the same rate.
Data can come from many sources, including local files, computer networks, television broadcasts, and video cameras.
Data comes in a variety of formats, such as Audio-Video Interleaved (AVI), Advanced Streaming Format (ASF), Motion Picture Experts Group (MPEG), and Digital Video (DV).
The programmer does not know in advance what hardware devices will be present on the end-user's system.
The DirectShow Solution
DirectShow is designed to address each of these challenges. Its main design goal is to simplify the task of creating digital media applications on the Windows platform, by isolating applications from the complexities of data transports, hardware differences, and synchronization.
To achieve the throughput needed to stream video and audio, DirectShow uses Direct3D and DirectSound whenever possible. These technologies render data efficiently to the user's sound and graphics cards. DirectShow synchronizes playback by encapsulating media data in time-stamped samples. To handle the variety of sources, formats, and hardware devices that are possible, DirectShow uses a modular architecture, in which the application mixes and matches different software components called filters.
DirectShow provides filters that support capture and tuning devices based on the Windows Driver Model (WDM), as well as filters that support older Video for Windows (VfW) capture cards, and codecs written for the Audio Compression Manager (ACM) and Video Compression Manager (VCM) interfaces.
The following diagram shows the relationship between an application, the DirectShow components, and some of the hardware and software components that DirectShow supports.

Przeczytaj

Simulating Graph Building with GraphEdit (Windows) - Mozilla Firefox

Plik Edycja Widok Historia Zakładki Narzędzia Pomoc

msdn.microsoft.com/en-us/library/dd377601(v=vs.85).aspx

graph edit

Simulating Graph Building with GraphEdit (Wi...

Home Library Learn Downloads Support Community Sign in | United States - English |

Search MSDN with Bing

- MSDN Library
- Windows Development
- Audio and Video
- DirectShow
- Using DirectShow
  - Simulating Graph Building with GraphEd
  - Using GraphEdit
  - Loading a Graph From an External Proces
  - Saving a Filter Graph to a GraphEdit File
  - Loading a GraphEdit File Programmatical
  - GraphEdit File Format

Community Content

Just make a download link for gr...
I hate installing 2-4 GB of stuf...

location...
NB that at least with SDK v7.1 t...

More...

Simulating Graph Building with GraphEdit

msdn

3 out of 3 rated this helpful - Rate this topic

DirectShow provides a debugging utility called GraphEdit, which you can use to create and test filter graphs.

GraphEdit is a visual tool for building filter graphs. With GraphEdit, you can experiment with a filter graph before you write any application code. You can also load a filter graph that your application creates, to verify that your application is building the correct graph. If you develop a custom filter, GraphEdit provides a quick way to test it: Simply load a graph with your custom filter and try running the graph. If you are new to DirectShow, GraphEdit is a good way to become familiar with filter graphs and the DirectShow architecture.

The following illustration shows how GraphEdit represents a simple filter graph.

Each filter is represented as a rectangle. Smaller squares along the edges of the filters represent pins. Input pins are on the left side of the filter, and output pins are on the right side. The arrows represent the connections between pins.

With GraphEdit, you can:

- Create and modify filter graphs using a visual, drag-and-drop interface.
- Simulate programmatic calls to build a graph.
- Run, pause, stop, and seek a graph.
- See what filters are registered on your computer, and view registry information for each filter.
- View filter property pages.
- View the media types of pin connections.