```
// MultThreads.cpp : Defines the entry point for the console application.
//                   Przykład tworzenia potkow i funkcji potoku
//                            Synchronizacja na podstawie sekcji krytycznych
//                   synchronizuje dostep potokow do wspolnego resursu -
//                   strumienia stdout

#include "stdafx.h"
#include <iostream>
#include "windows.h"
using namespace std;

#define MAX_THREAD_NUMB 64

//prototyp funkcji potoku
DWORD WINAPI ThreadFunc(void *lpParam);

//dane przekazywane do potoku, pakujemy w strukture
struct THREAD_DATA
{
      int ip;
};

//globalna deklaracja obiektu sekcji krytycznych
CRITICAL_SECTION cs;


int _tmain(int argc, _TCHAR* argv[])
{
      HANDLE hThread[MAX_THREAD_NUMB];
      DWORD ThreadID[MAX_THREAD_NUMB];
      THREAD_DATA tDat[MAX_THREAD_NUMB];

      int nThreads;
      int iThread;

      cout << "Input number of threads\n";
      cin >> nThreads;

      //inicjowanie obiektu sekcji krytycznych
      InitializeCriticalSection(&cs);

      for(iThread=0; iThread<nThreads; iThread++)
      {
```

```
		//wypelniamy dane, przekazywane do potoku ip
		tDat[iThread].ip = iThread;
		//tworzymy potok potomny ip
		hThread[iThread] = CreateThread(NULL, 0, ThreadFunc, (void *)&tDat[iThread], 0, &ThreadID[iThread]);
		if(hThread[iThread] == NULL)
		{
			cout << "create thread error: iThread = " << iThread << endl;
			system("pause");
			DWORD ExitCode;
			BOOL Ret = GetExitCodeProcess(GetCurrentProcess(), &ExitCode);
			ExitProcess(ExitCode);
		}
}

//Potok pierwotny czeka dokąd wszystkie potoki potomnę nie skończą działanie
//Jeśli tą instrukcje usunąć, potok pierwotny może skończyć działanie wcześniej
//od zakończenia działania potoków potomnych - PAGE FAULT
if(WaitForMultipleObjects(nThreads, hThread, TRUE, INFINITE) ==  WAIT_FAILED)
{
		cout << "WaitFor... problem "<< endl;
		system("pause");
		exit(1);
	}

	for(iThread=0; iThread<nThreads; iThread++)
		CloseHandle(hThread[iThread]);

	//deinicjowanie obiektu sekcji krytycznych
	DeleteCriticalSection(&cs);

	cout << "Hello from master thread\n";
	system("pause");

	return 0;
}

DWORD WINAPI ThreadFunc(void *lpParam)
/*=====================================================================
Funkcje potoku
=====================================================================*/
{
	THREAD_DATA *pDat = (THREAD_DATA *)lpParam;
	int ip = pDat->ip;
```

```
	//usunięcie synchronizacji powoduje niepoprawne działanie
	//funkcji biblioteki run time C\C++
	//Zakomentuj EnterCriticalSection(&cs) oraz LeaveCriticalSection(&cs)
	//i zobacz co wyjdzie
	EnterCriticalSection(&cs);    //wejscie do sekcji krytycznej
		//wyprowadzamy w strumien stdout pierwsza czesc zdania
		printf("Hello from thread ip");
		//wymuszamy odlaczenie potoku od procesora - prowokujemy blad
		Sleep(0);
		//wyprowadzamy drga czesc zdania
		printf(" = %d\n", ip);
		//system("pause");
	LeaveCriticalSection(&cs);     //wyjscie z sekcji krytycznej

	Sleep(1000);
	return 0;
}
```