Змістовний модуль:
ІСТОРІЯ РОЗВИТКУ, ФУНКЦІЇ, АРХІТЕКТУРА ТА ПРИНЦИПИ РОЗРОБКИ СУЧАСНИХ ОС

Розділ 2: ПОВТОРНЕ ВИКОРИТАННЯ КОДУ

Лекція 5

Принципи створення та використання бібліотек. Динамічні бібліотеки. Частина 2

ПИТАННЯ ДЛЯ ВИВЧЕННЯ

1. Головна функція DLL.
2. Встановлення режимів використання DLL.
3. Перевірка цілісності DLL.
4. Використання DLL в C#
5. Переваги та недоліки використання DLL в порівнянні з іншими бібліотеками

ГОЛОВНА ФУНКЦІЯ DLL. СТВОРЕННЯ

Для створення необхідно обрати тип проекту DLL(Empty при створенні бібліотеки не обирати!!!).

Перевірити, що створено файл ***dllmain.cpp***:

```
BOOL APIENTRY DllMain( HMODULE hModule, DWORD  ul_reason_for_call, LPVOID
        lpReserved)
{
        switch (ul_reason_for_call)   {
        case DLL_PROCESS_ATTACH:
        case DLL_THREAD_ATTACH:
        case DLL_THREAD_DETACH:
        case DLL_PROCESS_DETACH:
            break;
        }
        return TRUE;
} (
```

DLL_PROCESS_ATTACH : lpReserved = NULL для динамічного режиму використання)

DLL_PROCESS_DETACH: lpReserved = NULL, якщо FreeLibrary

ГОЛОВНА ФУНКЦІЯ DLL. ВИКОРИСТАННЯ. ПРИКЛАД 1

Створити DLL, яку можна використовувати тільки в динамічному режимі завантаження (Навіщо?)

```
BOOL APIENTRY DllMain( HMODULE hModule, DWORD  ul_reason_for_call,
                       LPVOID lpReserved   ){
    BOOL bRes = TRUE;
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH: if (lpReserved) bRes = FALSE;
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
      break;
    }
    return bRes;
}
```

При використанні DLL в статичному режимі – помилка запуску додатку

ГОЛОВНА ФУНКЦІЯ DLL. ВИКОРИСТАННЯ. ПРИКЛАД 2

- Забезпечити вихід із програми тільки після завершення усіх потоків, які запущені в процесі

```
BOOL b = FALSE;
BOOL APIENTRY DllMain( HMODULE hModule, DWORD  ul_reason_for_call,
                       LPVOID lpReserved   ){
    int count;
    BOOL bRes = TRUE;
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH: return TRUE;
    case DLL_THREAD_ATTACH: count++; break;
    case DLL_THREAD_DETACH: count--; b = count == 0; break;
    case DLL_PROCESS_DETACH:
      break;
    }
    return bRes;
}  В головній програмі while (!getB ());
В бібліотеку додати зовнішню функцію getB
```

ГОЛОВНА ФУНКЦІЯ DLL. ВИКОРИСТАННЯ. ПРИКЛАД 3

Створити тріальную версію DLL, яка працює тільки в одно поточному режимі

```
BOOL APIENTRY DllMain( HMODULE hModule, DWORD
    ul_reason_for_call, LPVOID lpReserved){
    switch (ul_reason_for_call){
    case DLL_PROCESS_ATTACH: return TRUE;
    case DLL_THREAD_ATTACH:{
            MessageBox (0, _T("Only 1 thread !!! “, _T(Error), 0);
            TerminateProcess (GetCurrentProcess (), 1);
            return FALSE;
     }
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH: return TRUE;
    }
} Вихід – усі потоки створити до завантаження DLL
```

ГОЛОВНА ФУНКЦІЯ DLL. ВИКОРИСТАННЯ. ПРИКЛАД 4

- Хай функції використовують загальну область пам'яті для зберігання даних (контейнер). Ця пам’ять повинна бути виділеною при першому використанні більшості функцій DLL. Може бути визволена при останньому використанні функцій DLL.
- При використанні декількох потоків кожний потік повинен мати свій контейнер

Варіант вирішення. Створити спеціальні функції для створення та знищення контейнеру, але де їх задати? Гілки

DLL_THREAD_ATTACH, DLL_THREAD_DETACH

ГОЛОВНА ФУНКЦІЯ DLL. ВИКОРИСТАННЯ. ПРИКЛАД 5

Контроль цілісності бібліотеки

1 Для чого потрібно?

DLL – самостійний модуль, в якому тільки функції.

Кожна функція починається та закінчується стандартним набором команд:

```
push  ebp;          55
mov   ebp, esp;          8b ec
sub       esp, e0h       81 ec e0
```

та закінчується командами

```
mov      esp,ebp        8b e5
pop      ebp        5d
ret       14h       c2 14
```

Достатньо після команди sub esp, e0h вставити команду

jmp с кодом (eb)<Різниця між адресами команди mov esp,ebp та команди після команди sub esp, e0h > і функція бібліотеки нічого робити не буде!!!

ГОЛОВНА ФУНКЦІЯ DLL. ВИКОРИСТАННЯ. ПРИКЛАД 3

Вимоги до контролю цілісності

1. Перевірка під час завантаження бібліотеки. Якщо цілісність порушена – не завантажувати
2. Перевірку виконувати незалежно від способу завантаження ДЛЛ.
3. Зміна імені ДЛЛ не відміняє її перевірку

ГОЛОВНА ФУНКЦІЯ DLL. ВИКОРИСТАННЯ. ПРИКЛАД 3

Алгоритм.

- Для файлу з бібліотекою обчислити CRC (hash, цифровий підпис. Записати в кінець файлу бібліотеки (окрема програма).
- Скласти функцію для контролю цілісності файлу, додати її до складу DLL як внутрішню функцію.
- Визначити повне ім'я файлу DLL(функція GetModuleFileName)

DWORD GetModuleFileName(HMODULE hModule, LPTSTR lpFilename, DWORD nSize);

- Викликати функцію контролю цілісності файлу DLL в головній програмі DLL (гілка DLL_PROCESS_ATTACH).
- Головна програма повинна повернути FALSE у разі порушення цілісності та TRUE у разі успішного завершення функції контролю

СТВОРЕННЯ ТА ВИКОРИСТАННЯ РЕСУРСНИХ БІБЛІОТЕК

Для чого використовувати?

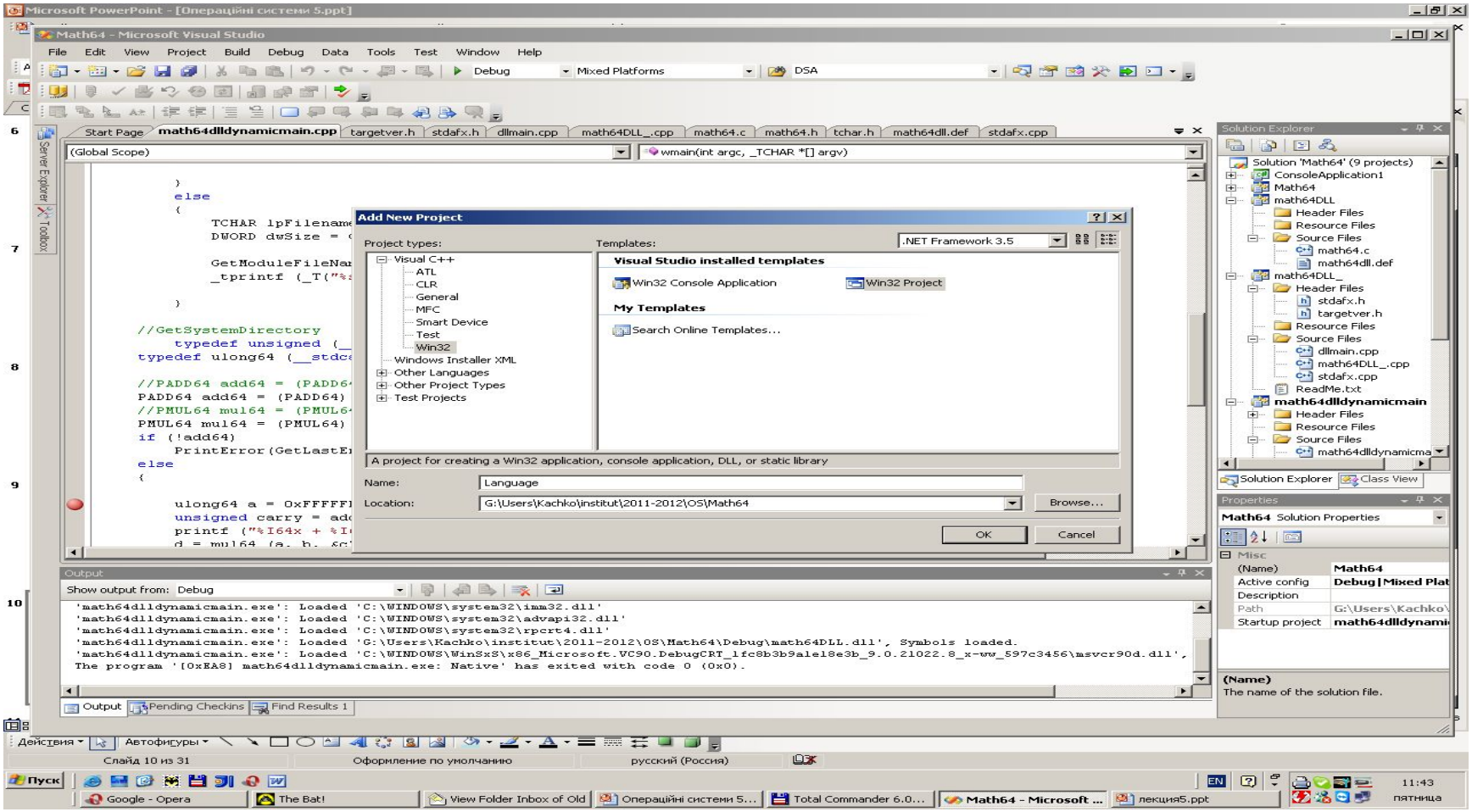
Хай необхідно створити програму з забезпеченням виведення повідомлень на різних мовах, перехід з мови на мову не потребує повторного будування програми.

1 спосіб. Для виведення даних використовувати if (ukraine)…; else if (english)…

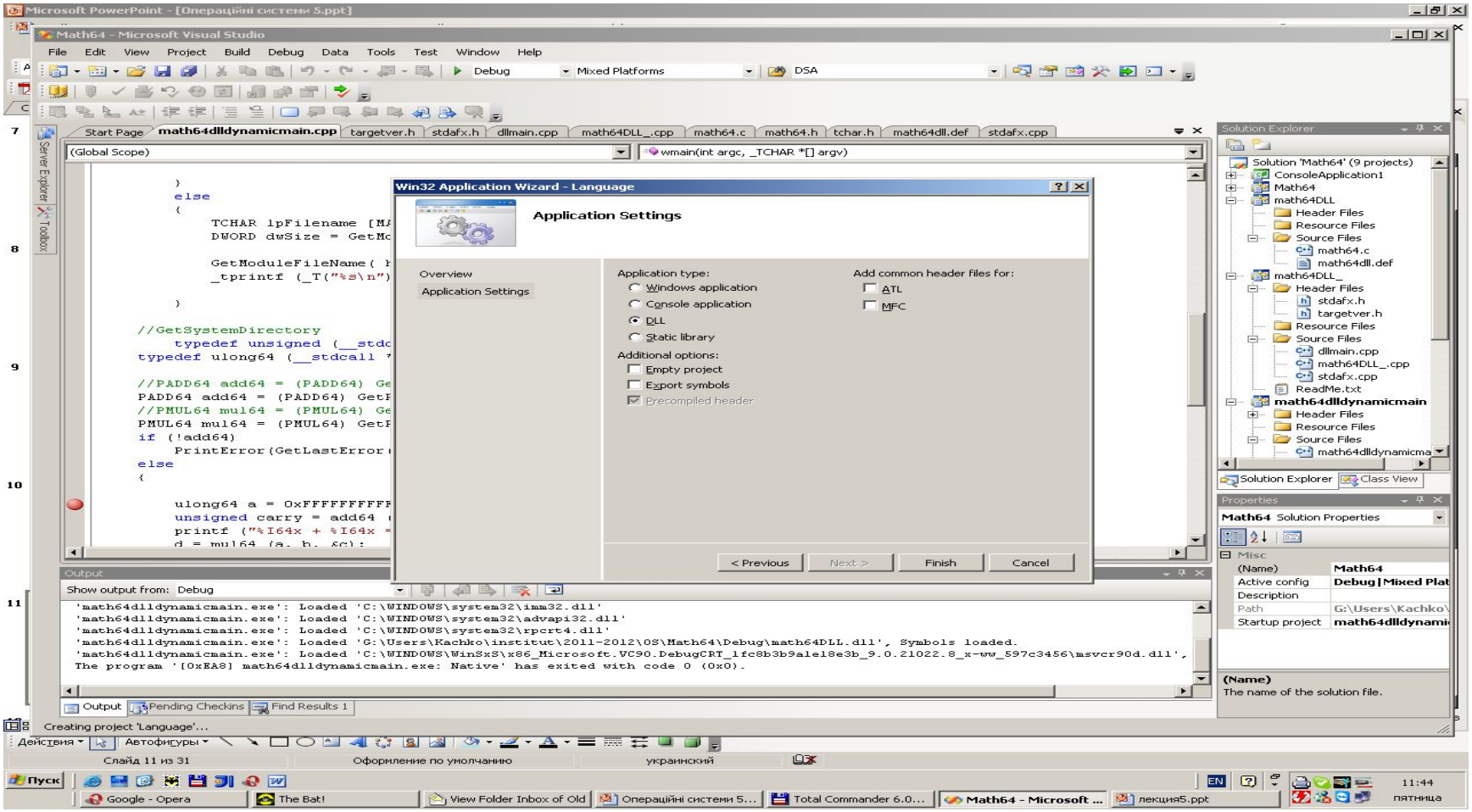
Недоліки: Не ефективно, додаток нової мови потребує перетворення додатку

- 2 спосіб. Використання ресурсів (можна замінити не тільки мову, але й картинки, наприклад виводити в якості фону красиві місця Києву, Нью-Йорку, …)

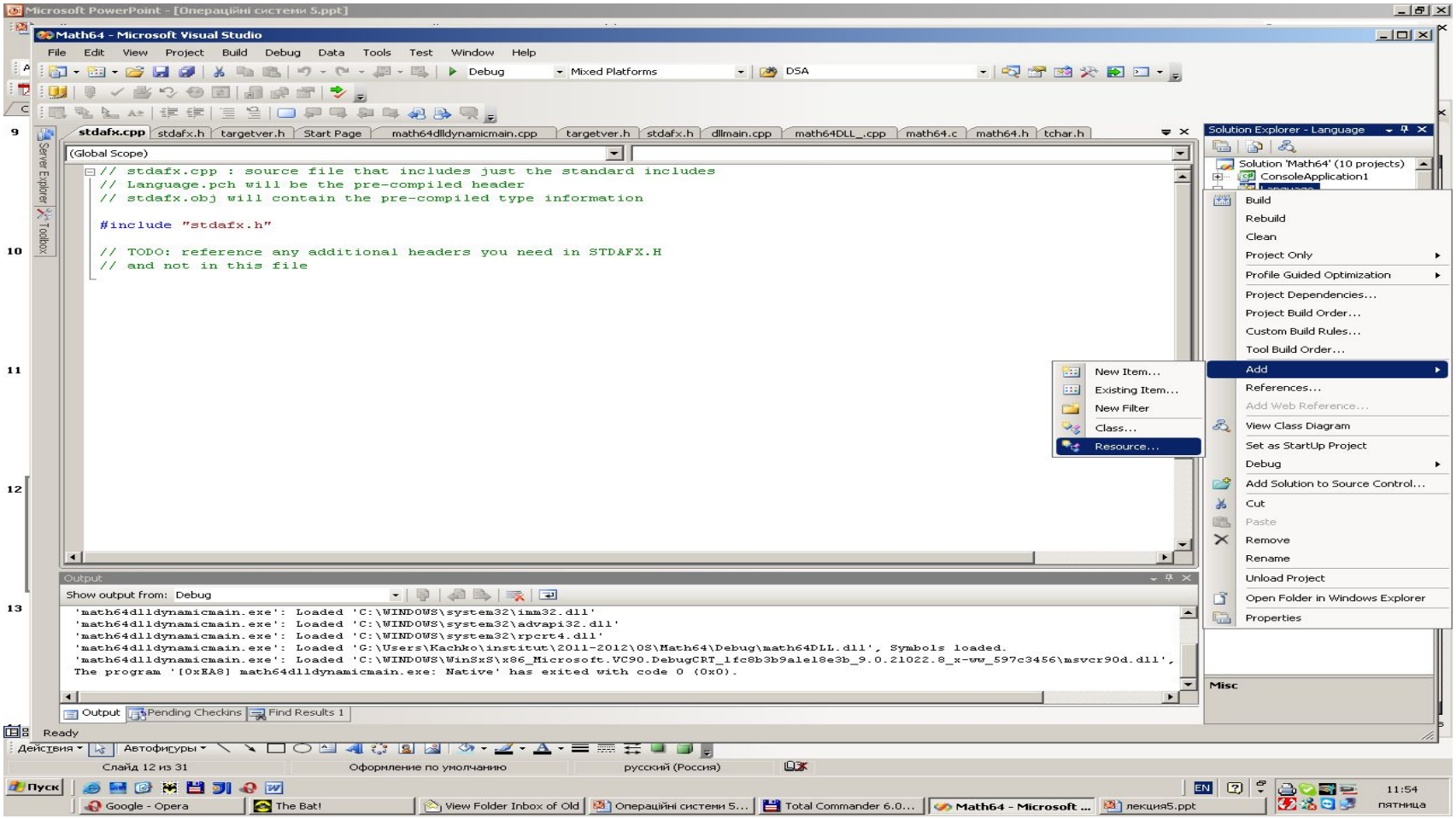
СТВОРЕННЯ РЕСУРСНИХ БІБЛІОТЕК. КРОК 1



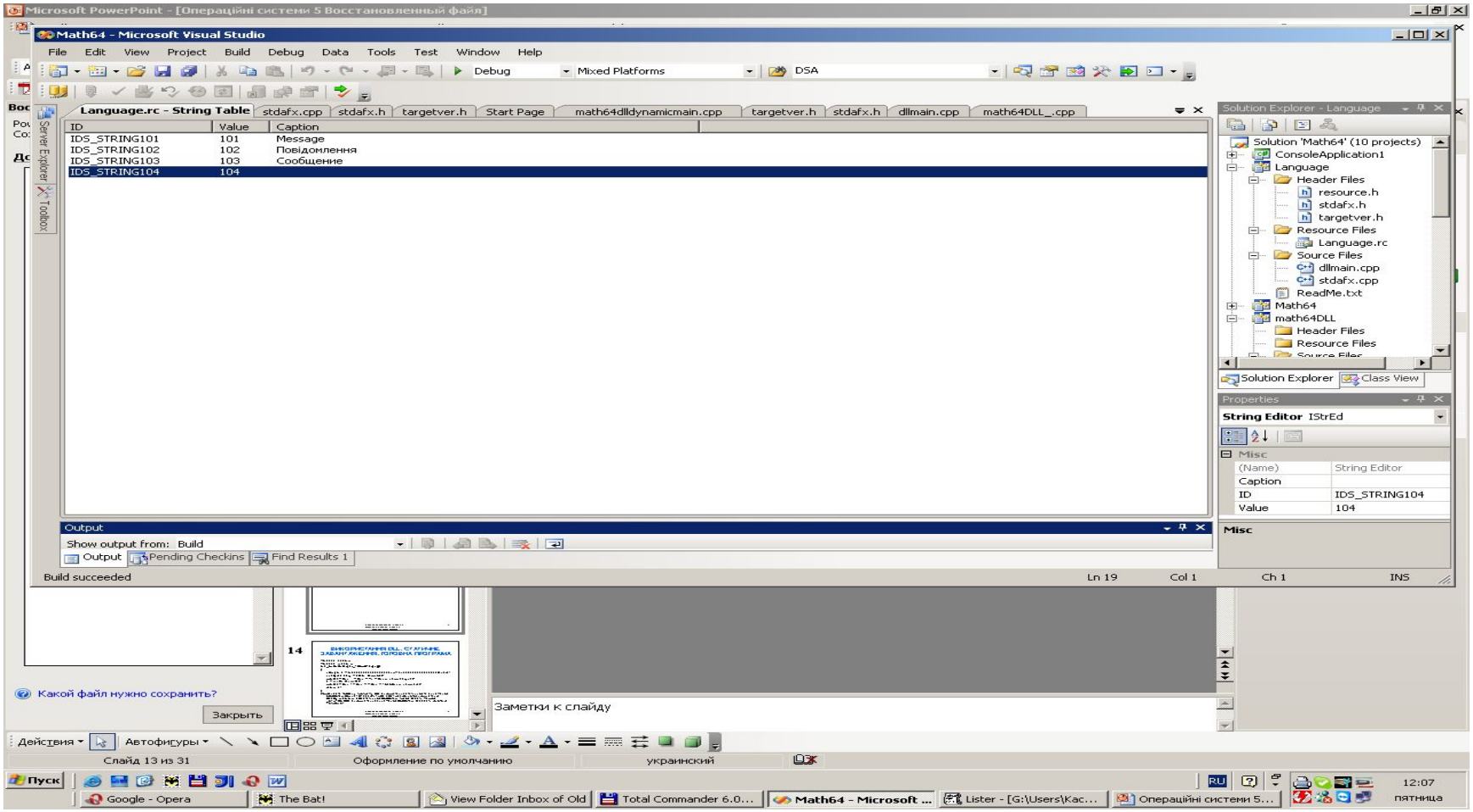
СТВОРЕННЯ РЕСУРСНИХ БІБЛІОТЕК. КРОК 2



СТВОРЕННЯ РЕСУРСНИХ БІБЛІОТЕК. КРОК 3



СТВОРЕННЯ РЕСУРСНИХ БІБЛІОТЕК. КРОК 4 (останній)



ВИКОРИСТАННЯ РЕСУРСНИХ DLL

Файл заголовків, сформований при створенні бібліотеки (resource.h) – без коментарів:

```
#define IDS_STRING101          101
#define IDS_STRING102          102
#define IDS_STRING103          103

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE      102
#define _APS_NEXT_COMMAND_VALUE       40001
#define _APS_NEXT_CONTROL_VALUE       1001
#define _APS_NEXT_SYMED_VALUE         101
#endif
#endif
```

ВИКОРИСТАННЯ РЕСУРСНИХ DLL

Функція *LoadString*

int LoadString(HINSTANCE hInstance,
UINT uID, LPTSTR lpBuffer, int nBufferMax
);

Де:

hInstance – дескриптор модуля. Де визначено ресурс (у нас це дескриптор DLL, який повертається функцією LoadLibrary);

uID – номер ресурсу, який визначено в файлі resource.h, у нас ці номери **IDS_STRING101, IDS_STRING102, IDS_STRING103;**

lpBuffer – буфер для запису результати (ресурси – рядки в універсальному форматі);

nBufferMax – розмір буферу (символів).

Функція повертає кількість символів, які записано в буфер (без нульового завершувача). Якщо розмір буферу недостатній, ресурс усікається. Якщо ресурсу не знайдено – повертає 0.

ВИКОРИСТАННЯ РЕСУРСНИХ DLL

```
#include "stdafx.h"
#include <windows.h>
#include "resource.h"
int _tmain(int argc, _TCHAR* argv[])
{
        HMODULE h = LoadLibrary (_T("Language.dll"));
        if (h)
        {
            TCHAR tBuf [MAX_PATH];
            LoadString (h,  IDS_STRING101, tBuf, MAX_PATH);
            MessageBox (0, tBuf, tBuf, 0);
            LoadString (h,  IDS_STRING102, tBuf, MAX_PATH);
            MessageBox (0, tBuf, tBuf, 0);
            LoadString (h,  IDS_STRING103, tBuf, MAX_PATH);
            MessageBox (0, tBuf, tBuf, 0);
        }
        return 0;
}
```

ВИКОРИСТАННЯ РЕСУРСНИХ DLL

1. Створюємо ресурсні бібліотеки для заданих мов, наприклад, Ukraine.dll, English.dll, Russian.dll
2. При інсталяції програми по запиту на вибір мови змінюємо імя обраної бібліотеки на language.dll
3. В програмі завантажуємо language.dll.
4. Якщо треба додати нову мову – додаємо відповідну dll.
5. По аналогії можна використовувати іконки, малюнки та інші ресурси.

ВИКОРИСТАННЯ DLL В C#

1 Додаємо необхідність використання системного InteropServices:

2. Для визначення DLL, що використовується, та адреси функції. яка потрібна, використовуються оператори:

*[DllImport(*Ім'я DLL*)]*

public static extern заголовок потрібної функції.

Приклад

```
[DllImport("Math64Dynamic.dll")]
public static extern ulong Add64 (ulong a,ulong b,out ulong c);
[DllImport("Math64Dynamic.dll")]
public static extern ulong Mul64(ulong a, ulong b, out ulong c);
```

Далі функції використовуються, як звичайні

ВИКОРИСТАННЯ DLL В C#

```
…
using System.Runtime.InteropServices;

namespace ConsoleApplication1{
   class Program {
      [DllImport("math64DLL.dll")]
      public static extern uint  add64(ulong a, ulong b, out ulong c);
      [DllImport("math64DLL.dll")]
      public static extern ulong mul64(ulong a, ulong b, out ulong c);
      static void Main(string[] args) {
         ulong a1 = 0xFFFFFFFFFFFFFFFF, b1 = 0xFFFFFFFFFFFFFFFF, c1, carry1=0;
         ulong a2 = 0xFFFFFFFFFFFFFFFF, b2 = 0xFFFFFFFFFFFFFFFF, c2, carry2=0;
         carry1 = add64(a1, b1, out c1);
         carry2 = mul64(a2, b2, out c2);
      …
      }
   }
}
```

ПОРІВНЯННЯ ШВИДКОДІЇ ПРИ ВИКОРИСТАННІ C, C#

Експеримент.

1. Створити DLL на C++ (множення квадратних матриць з елементами типу double).

2 Використати цю DLL в C++ для матриць розміром 512 * 512 елементів

3 Виміряти час виконання операції

4 Створити відповідну функцію в C# та використати її в C# з тими ж даними і розміром. Виміряти час

5 Використати DLL на C++ в C# для тих же даних виміряти час

6 Під час виміру часу використовувати режим Release

ФУНКЦІЯ МНОЖЕННЯ МАТРИЦЬ ДЛЯ C++

```
MATH64_API  void MulMatr(double a[], double b[], double
   c[], ulong64 n) {
        int i, j, k;
   for (i = 0; i < n; ++i)
            for ( j = 0; j < n; ++j) {
                c[i * n + j] = 0;
                for ( k = 0; k < n; ++k)
                    c [i * n + j] += a[i * n + j] * b [j * n + k];
            }
     }
```

ФУНКЦІЯ МНОЖЕННЯ МАТРИЦЬ ДЛЯ C#

```
static void MulMatr1(double[] a, double[] b, double[] c, long
   n) {
        for (int i = 0; i < n; ++i)
            for (int j = 0; j < n; ++j) {
                c[i * n + j] = 0;
                for (int k = 0; k < n; ++k)
                    c [i * n + j] += a[i * n + j] * b [j * n + k];
            }
    }
```

ВИЗНАЧЕННЯ ТА ІНІЦІАЛІЗАЦІЯ МАТРИЦЬ (C++ ТА C#)

```
const int  n = 512;                    //C++
static double a [n * n], b [n * n], c [n * n];
long n = 512;
double[] a = new double[n * n];          // C#
double[] b = new double[n * n];
double[] c = new double[n * n];
for (int i = 0; i < n; ++i )
    for (int j = 0; j < n; ++j) {
        a[i * n + j] = (double)((i + j)%10);
         b[i * n + j] = (double)((i + j)%10);
       }
```

ВАРІАНТ 1. ВИЗНАЧЕННЯ ЧАСУ ВИКОНАННЯ. С++

```
MulMatr (a, b , c, n);
printf ("c [0][0] = %lg c [n-1][n-1]= %lg\n", c [0], c[(n
   - 1) * n + n - 1]);
DWORD start, finish;
start = GetTickCount ();
MulMatr (a, b , c, n);
finish = GetTickCount ();
printf ("time = %d\n", finish - start);
printf ("c [0][0] = %lg c [n-1][n-1]= %lg\n", c [0], c[(n
   - 1) * n + n - 1]);
```

ВАРІАНТ 2. ВИЗНАЧЕННЯ ЧАСУ ВИКОНАННЯ. C#. ФУНКЦІЯ В C#

```
MulMatr1 (a, b , c, n);
Console.Write("c[0][0] = "); Console.Write(c[0]);
Console.Write(" c[n-1][n-1] = ");
Console.Write(c[(n - 1) * n + n - 1]);
Console.Write("\n");
Stopwatch sWatch = new Stopwatch();
sWatch.Start();   MulMatr1 (a, b , c, n); sWatch.Stop();
Console.WriteLine(sWatch.ElapsedMilliseconds.ToString());
Console.Write ("c[0][0] = "); Console.Write (c[0]);
Console.Write(" c[n-1][n-1] = ");
Console.Write ( c [(n - 1) * n + n - 1]);
Console.Write("\n");
```

ВАРІАНТ 3. ВИЗНАЧЕННЯ ЧАСУ ВИКОНАННЯ. C#. ФУНКЦІЯ З DLL (C++)

```
MulMatr (a, b , c, n);
Console.Write("c[0][0] = "); Console.Write(c[0]);
…
Console.Write(" c[n-1][n-1] = ");
Console.Write(c[(n - 1) * n + n - 1]);
Console.Write("\n");
Stopwatch sWatch = new Stopwatch();
sWatch.Start();   MulMatr (a, b , c, n); sWatch.Stop();
Console.WriteLine(sWatch.ElapsedMilliseconds.ToString());
Console.Write ("c[0][0] = "); Console.Write (c[0]);
Console.Write(" c[n-1][n-1] = ");
Console.Write ( c [(n - 1) * n + n - 1]);
Console.Write("\n");
```

ПОРІВНЯННЯ РЕЗУЛЬТАТІВ

Варіант	C++→ C++	C#→ C#	C++→ C#
Час виконання (мс), n = 512	1625	3534	4237

ПОРІВНЯННЯ РІЗНИХ СПОСОБІВ ВИКОРИСТАННЯ DLL

Статичне завантаження. Переваги

1. Простота використання функцій
2. Програма не почне виконуватись, якщо немає усіх необхідних DLL

Динамічне завантаження. Переваги

1. Можливість визволення пам'яті, коли бібліотека більш не потрібна
2. Можливість використання однієї DLL в різних платформах
3. Можливість вказати місце знаходження DLL

ВИСНОВКИ

- Бібліотеки застосовуються для повторного використання коду.
- Динамічна бібліотека має формат файлів для виконання, тому може використовуватися разом з програмою, яка її використовує.
- Треба уважно слідкувати за тим, яка саме бібліотека завантажується при умові наявності кількох різних екземплярів бібліотек.
- Середовище для розробки динамічних бібліотек та для програм, які використовують ці бібліотеки, можуть не співпадати, якщо використовується динамічний метод використання бібліотеки заголовків функцій бібліотеки.
- Бібліотека для використання може не завантажуватись, якщо попередній додаток вже її завантажив.
- Завантажувач повторно використовує бібліотеку тільки в тому разі, якщо повні імена бібліотек співпадають, тобто еквівалентність встановлюється не за змістом файлу а за його іменем.
- При використанні динамічних бібліотек в динамічному режимі необхідно обов'язково перевіряти успішність завантаження бібліотеки та визначення адрес функцій!!!

ПИТАННЯ ДЛЯ САМОСТІЙНОГО ВИВЧЕННЯ

- Спосіб передачі в списку параметрів класів для функцій, що експортуються, для статичного завантаження бібліотеки
- Спосіб передачі в списку параметрів класів для функцій, що експортуються, для динамічного завантаження бібліотеки
- Особливості використання структур та класів при організації зв'язку C++ та C#

МАТЕРІАЛИ ДЛЯ ЕКСПРЕС-КОНТРОЛЮ

- Яки типи бібліотек Ви знаєте?
- Для чого використовуються бібліотеки
- Коли має сенс застосовувати динамічні бібліотеки?
- Які питання ви будете задавати постановнику задачі для вирішення питання необхідності розробляти статичну, динамічну бібліотеки, або не звичайний додаток?
- Які узгодження по виклику Ви знаєте, які узгодження рекомендується використовувати для бібліотек?
- В якому разі узгодження __stdcall не можна використовувати для бібліотеки?
- Коли використовується статичне або динамічне завантаження DLL?
- Що дає використання .def файлу для DLL?
- Коли рекомендується використовувати бібліотеки C++ в програмах на C#?