Лабораторна робота №6

Побудування програмної системи з множини об’єктів, керованих повідомленнями

Мета роботи: отримати вміння та навички використовувати засоби обміну інформацією та запрограмувати взаємодію незалежно працюючих програмних компонентів.

Завдання:

1. Створити у середовищі MS Visual Studio C++ проект Win32 з ім’ям **Lab6**.
2. Написати вихідні тексти усіх програм-компонентів згідно варіанту завдання.
3. Скомпілювати вихідні тексти і отримати виконувані файли програм.
4. Перевірити роботу програм. Налагодити взаємодію програм.
5. Проаналізувати та прокоментувати результати та вихідні тексти програм.
6. Оформити звіт.

Теоретичні положення

Компоненти системи можуть обмінюватися запитами-відповідями та даними

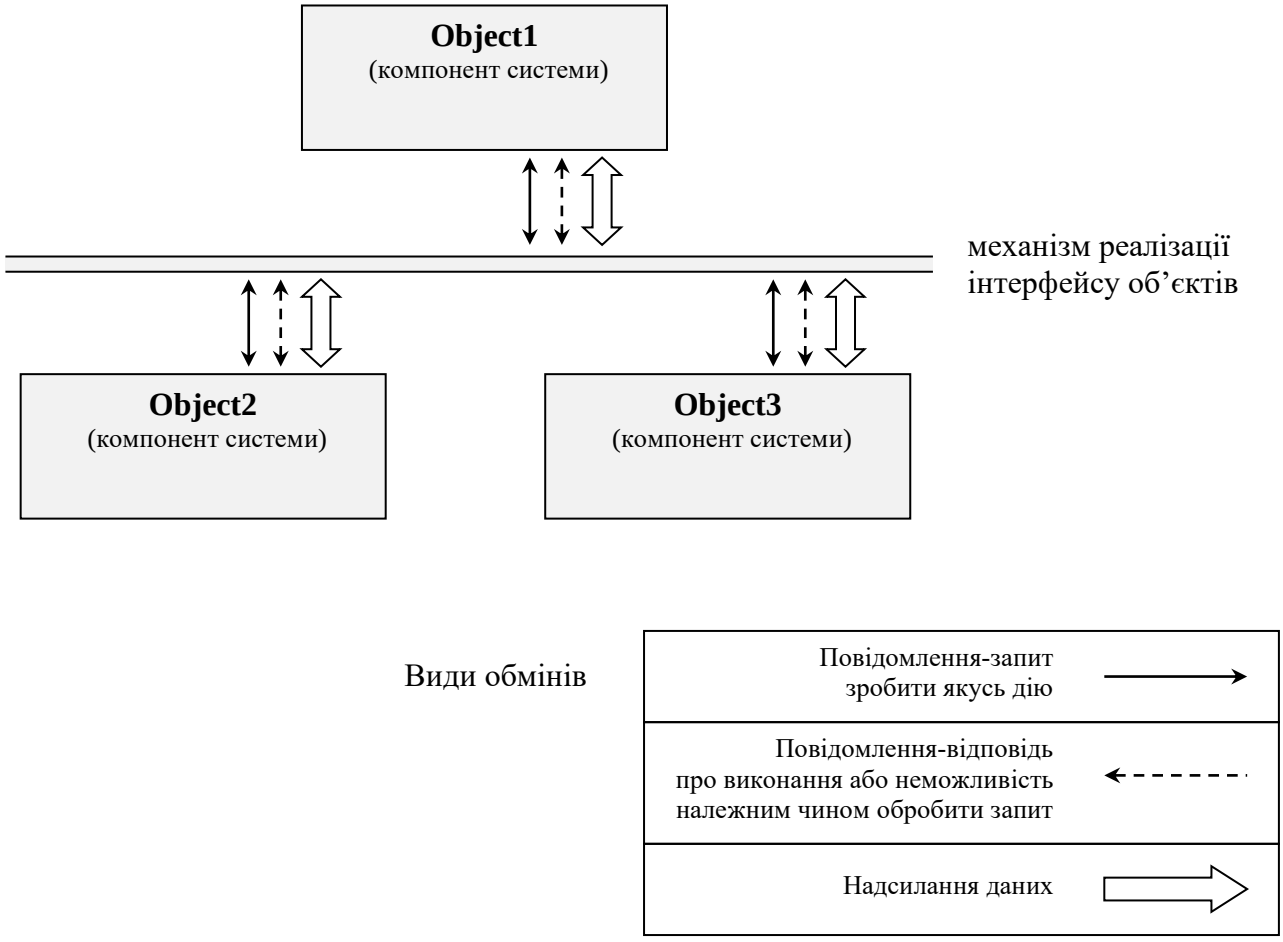


Рис. 1. Узагальнена схема взаємодії об’єктів – компонентів системи

Одним з тлумачень об'єктно-орієнтованої системи є таке: **у системі функціонують декілька об'єктів, які обмінюються повідомленнями.** Повідомлення може бути вимогою-запитом зробити щось, повідомленням-відповіддю та повідомленням надсилання даних (у вигляді самих даних, або у вигляді посилання, де ці дані треба взяти). Хтось організовує повідомлення, або об'єкти налаштовують їх самі, для виконання системою певного завдання.

Традиційна реалізація об'єктів у мовах програмування на кшалт C++, Java, C# обмін повідомленнями зводить до виклику публічних функцій-методів класів. Це достатньо примітивний спосіб взаємодії об'єктів і він породжує багато проблем реалізації ООП.

Проте, зовсім не обов'язково розуміти повідомлення тільки як виклик метода класу. Для того, щоб об'єкт виконав якісь потрібні дії, загалом, треба якимось чином надіслати йому запит-прохання. Якщо об'єкт може це зробити, то він це робить і надсилає потрібний результат, а якщо ні, наприклад, тому що він зараз зайнятий іншою роботою, або запит містить некоректну вимогу, то об'єкт надсилає відповідне повідомлення-відповідь.

У середовищі Windows є багато різноманітних можливостей побудувати інтерфейс повідомленнями між об'єктами-компонентами програмних систем. Одною з таких можливостей є механізм повідомлень Windows (*Windows messages*).

Пошук об'єкта-компонента для співпраці

Уявимо собі, що програма хоче надіслати повідомлення вікну іншої програми. А для цього треба знати хендл її вікна. У складі API Windows є функція **FindWindow**, яка дозволяє отримати hWnd будь-якої програми, якщо та зараз працює. Рекомендується шукати іншу програму по імені класу її вікна (який потрібно заздалегідь знати). Наприклад, якщо клас головного вікна програми зветься "DATACREATOR" то

```
hWndDataCreator = FindWindow("DATACREATOR", NULL);
```

Де знайти назву класу головного вікна програми? У головному cpp-файлі (якщо він спочатку був автоматично згенерований при створенні нового проекту Win32) є перемінна szWindowClass. Значення цієї перемінної – рядок тексту записується у файлі ресурсів у розділі STRINGTABLE

```
STRINGTABLE
BEGIN
   IDC_DATACREATOR   "DATACREATOR"
   IDS_APP_TITLE     "DataCreator"

END
```

Таким чином, на початку обміну повідомленнями з іншою програмою треба виконати наступне

```
hWndOther = FindWindow(classNameOther, NULL);
if (hWndOther)
   {     //потрібна програма вже функціонує
   . . . //надсилаємо повідомлення на адресу вікна hWndOther
   }
else     //потрібної програми немає
   {
   . . . //виклик на виконання потрібної програми
   }
```

Запуск програми на виконання іншою програмою

Уявимо собі, що зараз працює програма, яка хоче звернутися (наприклад, надіслати повідомлення) до іншої програми – а тої просто немає, вона ще не завантажена – її потрібно спочатку викликати. Для виклику будь-якої програми можна скористатися функцією API Windows **WinExec**

```
WinExec(LPCSTR lpCmdLine, UINT uCmdShow);
```

Рядок тексту **lpCmdLine** є командним рядком виклику програми. Цей рядок повинен містити ім’я виконуваного файлу потрібної програми. Наприклад:

```
WinExec("otherprog.exe", SW_SHOW);
```

Можна вказувати як повне ім’я виконуваного файлу, так і скорочене – буде шукатися вказана програма, у першу чергу, у поточній директорії. Якщо ім’я програми містить проміжки, то це ім’я треба записувати у лапки, наприклад:

```
WinExec("\"c:\\Program files\\otherprog.exe\"", SW_SHOW);
```

Окрім імені програми, яка викликається, у командному рядку можуть записуватися параметри – слова, розділені проміжками. Часто у якості параметра вказується ім’я файлу, який ця програма повинна обробити одразу після виклику, наприклад

```
WinExec("otherprog.exe myfile.txt", SW_SHOW);
```

otherprog.exe – ім’я виконуваного файлу програми
myfile.txt – файл, з яким ця програма повинна щось зробити, наприклад, завантажити дані з цього файлу, або, навпаки, записати дані у вказаний файл.

Крім того, у командному рядку можна можна вказувати й інші параметри. Скажімо, для того, щоб інша програма знала, яка програма її викликала – щоб потім надсилати їй повідомлення, ніхто не заважає у командному рядку записати також ідентифікатор вікна програми-викликача. Наприклад, якщо hWnd викликача дорівнює 3471, то

```
WinExec("otherprog.exe myfile.txt 3471", SW_SHOW);
```

Коли програма, яка буде викликана, розшифрує значення hWnd = 3471, то тепер вона, у свою чергу, може повідомити програмі, яка її викликала, значення hWnd свого вікна.

Як програма може узнати командний рядок, з яким її викликали? Це дуже просто – така можливість є стандартною для мов програмування C/C++. Командний рядок є параметром-аргументом функції **`main(arg. . .)`**. У середовищі Visual Studio проект програми у стилі Win API містить головний файл .cpp, у якому така функція є, проте, вона зветься трохи інакше (це ще залежіть від версії Visual Studio), наприклад

```
int APIENTRY _tWinMain(HINSTANCE hInstance,
                       HINSTANCE hPrevInstance,
                       LPTSTR    lpCmdLine,      //це командний рядок
                       int       nCmdShow)
{
. . .
}
```

Третій параметр цієї функції – **`lpCmdLine`**, є вказівником на командний рядок виклику. Потрібно запрограмувати розбір (парсінг) вмісту цього рядка.

Як програмі найпростіше передати іншій програмі одне числове значення?

Щоб надіслати іншій програмі 32-бітове значення **`value`**, можна порекомендувати надіслати повідомлення, наприклад, WM_COMMAND, у якому в якості параметра `lParam` буде записане потрібне значення:

```
PostMessage(hWndOther, WM_COMMAND, (WPARAM)wParam, (LPARAM)value);
```

Потрібно враховувати, що тип LPARAM для Win32 означає тип **long**.
Для третього параметра – wParam потрібно вказати значення, наприклад, 10000, яке не співпаде з жодним ідентифікатором пункта меню цієї програми.

Чому **PostMessage**, а не **SendMessage?** Функція PostMessage записує повідомлення у чергу повідомлень, а SendMessage надсилає позачергово. Порада: завжди користуйтесь PostMessage – тоді менша вірогідність конфіліктів у системі при обробці довготривалих процедур обробки повідомлень. Використовуйте SendMessage тільки для окремих випадків, як вимагає документація по API Windows, наприклад, для WM_COPYDATA.

Коли надсилати іншій програмі (вікну hWndOther) значення hWnd свого вікна? Звичайно, можна будь-коли, але якщо потрібно надіслати це одразу після розшифрування командного рядка у функції WinMain, то можна порекомендувати зробити це впродовж обробки повідомлення WM_CREATE

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
switch (message)
   {
   . . .
   case WM_CREATE:
     if (hWndOther) PostMessage(hWndOther, WM_COMMAND, 10000, (long)hWnd);
     break;
   . . .
```

Повідомлення WM_COPYDATA

Щоб надіслати масив даних іншій програмі, програма може зробити це за допомогою Windows-повідомлення WM_COPYDATA. Таке повідомлення надсилається функцією API Windows **SendMessage**

```
SendMessage(hWndDest, WM_COPYDATA, (WPARAM)hWndSrc, (LPARAM)&cds);
```

де:

hWndDest – HWND вікна призначення (програми-отримувача)

hWndSrc – HWND вікна, яке надсилає (програми-передавача)

cds – структура типу COPYDATASTRUCT, яка містить дані, які надсилаються

Структура COPYDATASTRUCT має такі поля:

```
COPYDATASTRUCT
{
ULONG_PTR dwData;    //будь-яке ціле (32-бітове) число
DWORD cbData;        //кількість байтів даних, на які вказує член lpData
LPVOID lpData;       //адреса даних.
};
```

Таким чином можна надіслати одне числове значення (член **dwData**) плюс масив даних (його адреса – член **lpData**). Дані, які надсилаються не можуть містити вказівники або посилання на об’єкти, які є недоступними для іншої програми. Вказівник **lpData** можна використовувати тільки упродовж обробки повідомлення WM_COPYDATA. Дані тільки для читання. Якщо програмі-приймачу потрібно потім обробляти прийняті дані, то треба скопіювати їх у якійсь буфер.

Розглянемо як надіслати іншій програмі п’ять числових значень, наприклад, значень перемінних **nPoints,xMin,xMax,yMin,yMax**

```
long nPoints,xMin,xMax,yMin,yMax;
. . .

//--це наша власна функція--
int SendCopyData(HWND hWndDest, HWND hWndSrc, void *lp, long cb)
{
COPYDATASTRUCT cds;

cds.dwData = 1;     //а можна і будь-яке інше значення
cds.cbData = cb;
cds.lpData = lp;
return SendMessage(hWndDest, WM_COPYDATA, (WPARAM)hWndSrc, (LPARAM)&cds);
}
. . .

//сформуємо дані як суцільний масив, наприклад, так
long params[5] = {nPoints, xMin, xMax, yMin, yMax};

//а тепер відправимо масив params вікну hWndOther іншої програми
SendCopyData(hWndOther, hWnd, params, sizeof(params));
```

Як приймати дані? Потрібно у функції вікна програми-приймача зробити обробник повідомлення WM_COPYDATA

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
switch (message)
   {
   . . .
   case WM_COPYDATA:
      OnCopyData(hWnd,wParam,lParam);     //це наша власна функція-обробник
      break;
   . . .
```

Якщо OnCopyData для прийому тільки п’яти параметрів, то можна запрограмувати так:

```
void OnCopyData(HWND hWnd,WPARAM wParam,LPARAM lParam)
{
COPYDATASTRUCT *cds;
cds = (COPYDATASTRUCT *)lParam;
long *p = (long *)cds->lpData;
nPoints = p[0];
xMin = p[1];
xMax = p[2];
yMin = p[3];
yMax = p[4];
}
```

А якщо програма через повідомлення WM_COPYDATA повинна приймати різноманітні порції даних, то можна використати поле **cds->dwData** для ідентифікації кількості та типу даних, що надсилаються-отримуються.

Буфер обміну – Clipboard Windows

Усі користувачі Windows знайомі з цим буфером – майже будь-яка програма копіює у буфер якісь дані після натискування клавіш Ctrl-C. Потім можна вставити у потрібне місце (у тому числі у середовищі іншої програмі) дані з буфера натискуванням клавіш Ctrl-V. Такі натискування клавіш не є обов'язковими атрибутами Windows – програмісти повинні спеціально запрограмувати такі можливості для своїх програм (зауваження: це не зовсім так – для дочірніх вікон вводу стрічок, редагування тексту та у деяких інших випадках копіювання та вставка вже реалізовані на рівні операційної системи).
Буфер – Clipboard Windows призначений для обміну даними між програмами. У цей буфер можна записувати тексти, зображення, та, взагалі, будь-що.

Для реалізації Clipboard Windows використовується глобальна динамічна віртуальна пам'ять. Рекомендується використовувати Clipboard для обміну даними обсягом не більше декількох десятків мегабайтів.
Для того, щоб програма записала дані у Clipboard, потрібно викликати функцію API Windows **SetClipboardData**. Ця функція має у якості параметрів ідентифікатор формату та *handle* блоку пам'яті, відкритого за допомогою функцій **GlobalAlloc**. Нижче наведений код функції, яку можна рекомендувати для запису ANSI-текстів у Clipboard

```
int PutTextToClipboard_POREV(HWND hWnd, char *src)
{
HGLOBAL hglbCopy;
BYTE *pTmp;
long len;

if (src == NULL) return 0;
if (src[0] == 0) return 0;
len = strlen(src);
hglbCopy = GlobalAlloc(GHND, len+1);
if (hglbCopy == NULL) return FALSE;
pTmp = (BYTE *)GlobalLock(hglbCopy);
memcpy(pTmp, src, len+1);
GlobalUnlock(hglbCopy);
if (!OpenClipboard(hWnd))
   {
   GlobalFree(hglbCopy);
   return 0;
   }
EmptyClipboard();
SetClipboardData(CF_TEXT, hglbCopy);
CloseClipboard();
return 1;
}
```

Приклад запису в Clipboard декількох рядків тексту – пар числових значень, розділених табуляцією

```
char text[] =
"x    y\r\n"
"1.0  3.12\r\n"
"2.0  5.3\r\n"
"3.0  6.11\r\n"
"4.0  7.1\r\n"
"5.0  8.05\r\n";

PutTextToClipboard_POREV(hWnd, text);
```

Якщо програма це зробить, то можна відкрити Блокнот Windows або інший текстовий редактор і вставити вміст Clipboard – натиснути Ctrl-V. У результаті повинно бути:

```
x     y
1.0   3.12
2.0   5.3
3.0   6.11
4.0   7.1
5.0   8.05
```

Для читання вмісту Clipboard у форматі ANSI-тексту можна скористатися функцією

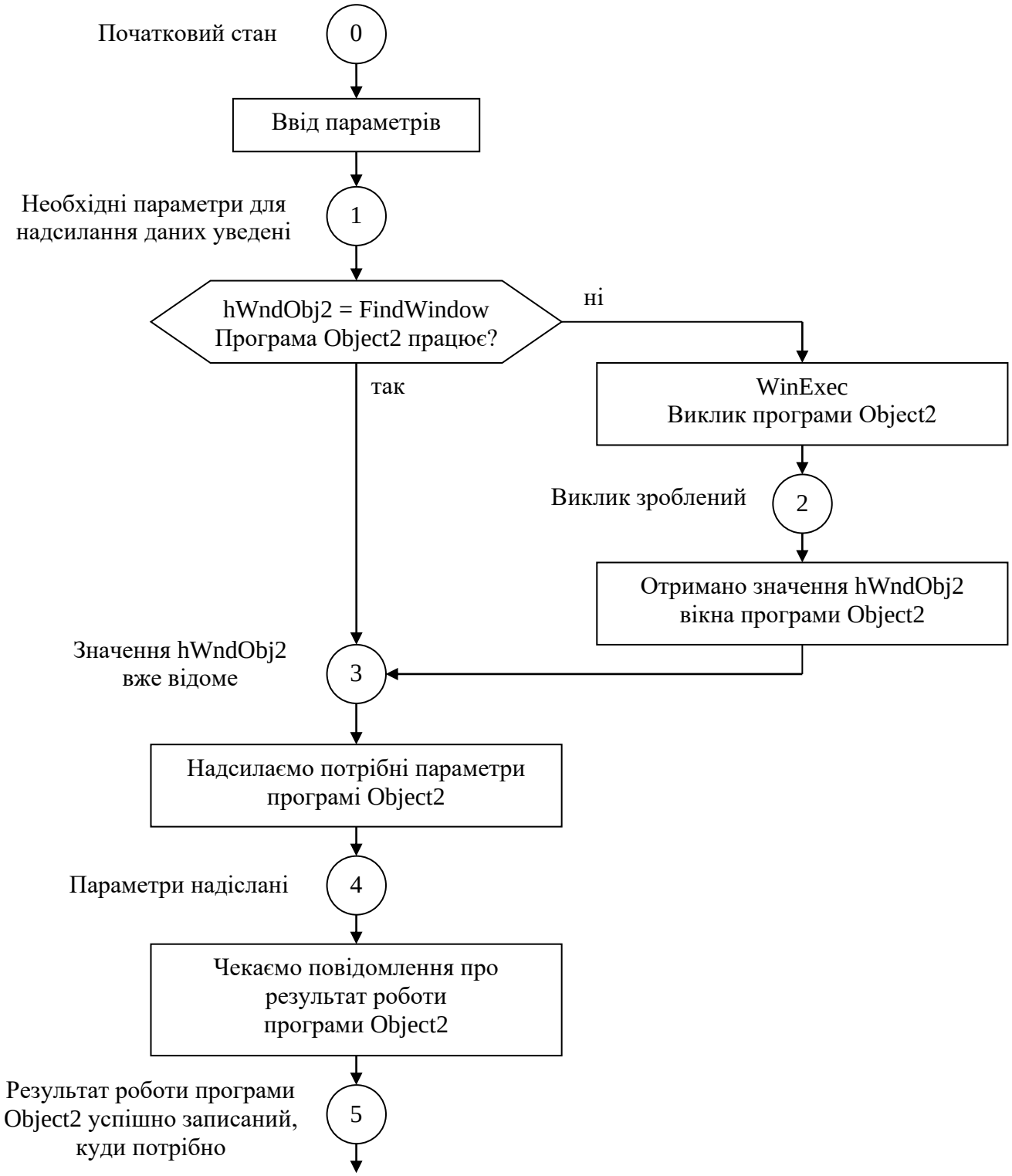
```
long GetTextFromClipboard_POREV(HWND hWnd, char *dest, long maxsize)
{
HGLOBAL hglb;
LPTSTR lptstr;
long size, res;

res = 0;
if (!IsClipboardFormatAvailable(CF_TEXT)) return 0;
if (!OpenClipboard(hWnd)) return 0;
hglb = GetClipboardData(CF_TEXT);
if (hglb != NULL)
   {
   lptstr = (char *)GlobalLock(hglb);
   if (lptstr != NULL)
      {
      size = strlen(lptstr);
      if (size > maxsize)
         {
         lptstr[maxsize] = 0;
         size = strlen(lptstr);
         }
      strcpy(dest, lptstr);
      res = size;
      GlobalUnlock(hglb);
      }
   }
CloseClipboard();
return res;
}
```

Ця функція копіює вміст Clipboard у буфер, вказаний параметром dest.

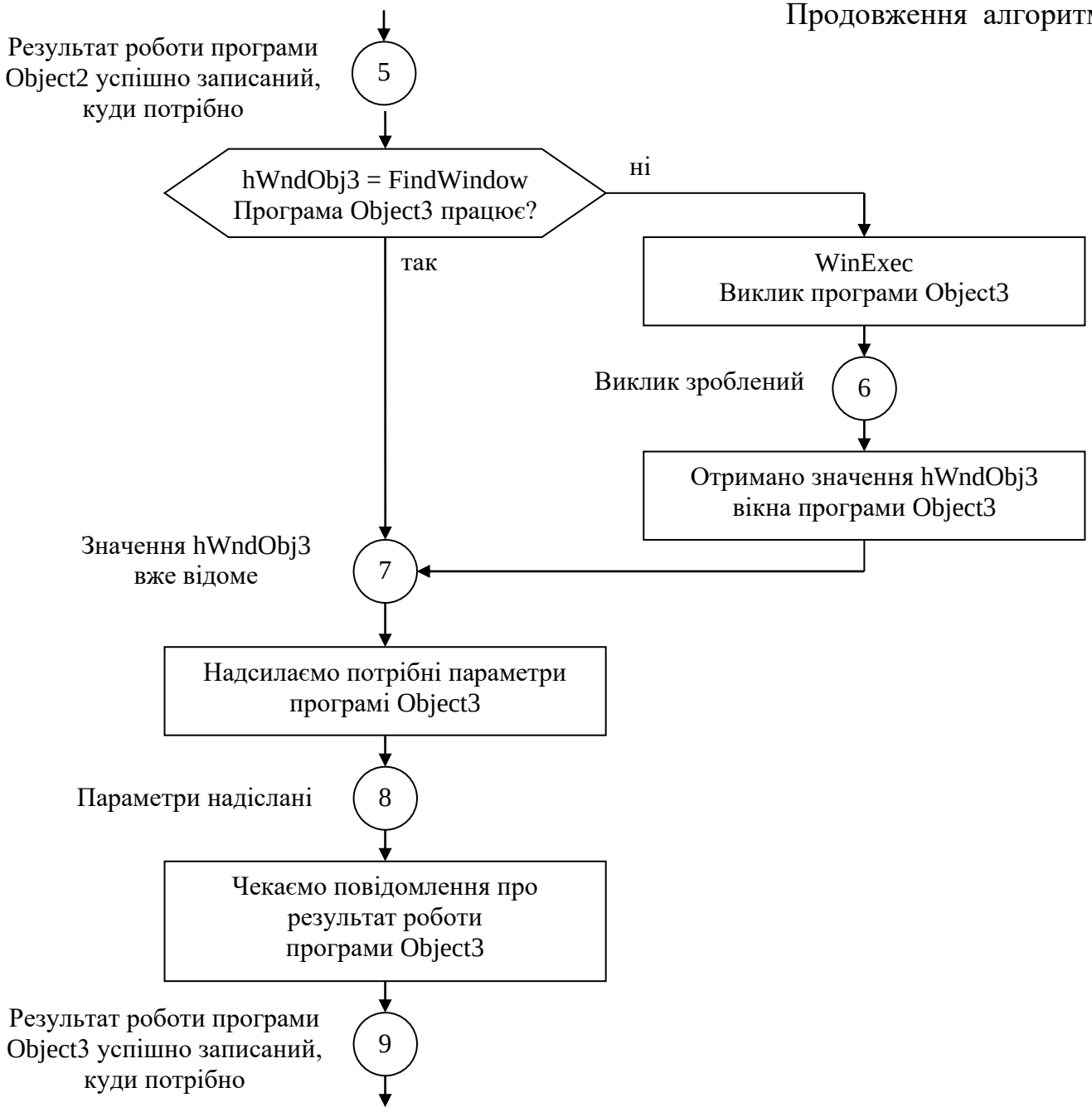
Алгоритм обміну повідомленнями об’єктів-компонентів системи

Програма **Object1** спочатку просить щось зробити програму **Object2**



Далі потрібно доручити програмі Object3 обробити дані, створені програмою Object2

Продовження алгоритму



Роботу завершено.
Перехід на початковий стан 0.

Методичні рекомендації

Для виконання лабораторної роботи потрібно створити три незалежні програми, для чого можна створити три проекта у одному рішенні (Solution) Microsoft Visual Studio C++. У цьому випадку усі виконувані файли будуть знаходитися у спільній папці \Debug або \Release.

Головний проект – програма **Lab6** є менеджером, який керує двома іншими програмами – **Object2** та **Object3**. Програма **Lab6** автоматично, без участі користувача, виконує обмін повідомленнями з програмами **Object2** та **Object3** для виконання потрібного завдання згідно наведеному вище алгоритму.

У кожній програмі передбачити автоматичну обробку потрібних повідомлень і переходів з одного стану в інший.

Головні вимоги

1. Для початку роботи користувач програми вибирає потрібний пункт меню програми **Lab6**. Далі з'являється вікно діалогу, у якому потрібно ввести параметри згідно варіанту завдання. У вікні діалогу користувач натискує кнопку "Так" (або "Виконати") і на цьому місія користувача закінчується – далі він тільки спостерігає, як програма сама автоматично виконає усе, що потрібно для отримання результату. Виклик інших програм – **Object2** та **Object3** головна програма **Lab6** повинна робити без участі користувача.

2. Обмін повідомленнями та масивами даних між програмами **Lab6, Object2** та **Object3** повинен відбуватися **автоматично, без участі користувача**.

3. У результаті одного сеансу роботи користувач повинен бачити головні вікна програм **Object2** та **Object3**, у яких відображатимуться потрібні результати відповідно варіанту завдань. Для цього вікна програм повинні автоматично розташуватися так, щоб усі результати було видно. Програма **Lab6** повинна залишатися у активному стані, щоб користувач мав можливість повторно виконати роботу.

4. Передбачити варіанти успішної роботи у випадках, коли програми **Object2** та **Object3** (одна або обидві) до того вже були викликані.

5. По завершенні роботи програми **Lab6** повинні **автоматично завершуватися** і програми **Object2** та **Object3.**

Вибір варіанту завдання

Номер варіанту завдання згідно формулі

Номер варіанту = Ж mod 4,

де: Ж – номер в списку студентів в журналі. Зміст завдань у таблиці нижче

Таблиця варіантів завдань

Номер варіанту	Програма Lab6	Програма Object2	Програма Object3
0	1. Користувач вводить значення параметрів **nPoint, xMin, xMax, yMin, yMax** у діалоговому вікні. 2. Програма викликає програми Object2, 3 і виконує обмін повідомленнями з ними для передавання, отримання інформації.	1. Створює **nPoint** пар цілих (**int**) значень (x,y) в діапазонах **xMin** – **xMax**, **yMin** – **yMax**. 2. Показує числові значення у власному головному вікні 3. Записує дані в Clipboard Windows у текстовому форматі	1. Зчитує дані з Clipboard Windows 2. Відображає графік **y=f(x)** у власному головному вікні. Графік, як в математиці – лінія, що проходить через точки; осі координат з підписами числових значень x та y.
1	1. Користувач вводить значення **n, Min, Max** у діалоговому вікні. 2. Програма викликає програми Object2, 3 і виконує обмін повідомленнями з ними для передавання, отримання інформації.	1. Створює матрицю **n×n** цілих (**int**) чисел у діапазоні **Min – Max** 2. Показує числові значення у власному головному вікні 3. Записує дані в Clipboard Windows у текстовому форматі	1. Зчитує дані з Clipboard Windows 2. Відображає значення детермінанту матриці у власному головному вікні
2	1. Користувач вводить значення **n, Min, Max** у діалоговому вікні. 2. Програма викликає програми Object2, 3 і виконує обмін повідомленнями з ними для передавання, отримання інформації.	1. Створює вектор **n** дробових (**double**) чисел у діапазоні **Min – Max** 2. Показує числові значення у декількох стовпчиках та рядках у власному головному вікні 3. Записує дані в Clipboard Windows у текстовому форматі	1. Зчитує дані з Clipboard Windows 2. Виконує сортування масиву чисел і відображає його у декількох стовпчиках та рядках у власному головному вікні
3	1. Користувач вводить значення **n, Min, Max** у діалоговому вікні. 2. Програма викликає програми Object2, 3 і виконує обмін повідомленнями з ними для передавання, отримання інформації.	1. Створює вектор **n** дробових (**double**) чисел у діапазоні **Min – Max** 2. Показує числові значення у декількох стовпчиках та рядках у власному головному вікні 3. Записує дані в Clipboard Windows у текстовому форматі	1. Зчитує дані з Clipboard Windows 2. Відображає графік **y=f(x)** у власному головному вікні Значення **y** – це значення вектора, **x** – індекси елементів. Графік, як в математиці – лінія, що проходить через точки; осі координат з підписами числових значень x та y.

Зміст звіту

1. Титульний аркуш
2. Варіант завдання
3. Вихідний текст файлів .cpp трьох програм
4. Схема послідовності надсилання-обробки повідомлень
5. Ілюстрації (скріншоти)
6. Висновки

Контрольні запитання

1. Як запрограмувати виклик у програмі іншої програми?
2. Як отримати hWnd іншої програми?
3. Як надіслати іншій програмі повідомлення?
4. Як надіслати іншій програмі одне число?
5. Як надіслати іншій програмі масив числових значень?
6. Як запрограмувати обмін інформацією з Clipboard Windows?