УДК 004.43
Бойченко П. В., студент гр. ОМ-02
Поліщук В.О., ст. викл.

Національний технічний університет “Київський політехнічний інститут”
Інститут енергозбереження та енергоменеджменту
кафедра електромеханічного обладнання енергоємний виробництв

Інтеграція C++ Builder і Microsoft Excel для обробки даних

Організація передачі даних між програмою Microsoft Excel і додатками C++ Builder.
Организация передачи данных между программой Microsoft Excel и приложениями C++ Builder.
Data transmission between program Microsoft Excel and program C++ Builder.

Microsoft Excel широко застосовується для проведення однотипних розрахунків для великих діапазонів даних, автоматизації результуючих обчислень, розв’язку задач шляхом підбору значень параметрів, обробки результатів експерименту, проведення пошуку оптимальних значень параметрів і т. ін., разом з тим можливості по створення зручного інтерфейсу користувача досить обмежені. C++ Builder дозволяє будувати потужні додатки будь-якого призначення. Тому в реальних системах, як правило, слід об’єднувати власні програми з програмами інших додатків, наприклад, додатками Microsoft Office.

Найбільш вдалими способами інтеграції додатків C++ Builder з Excel, є можливість використання Excel як сервер COM чи як сервер автоматизації OLE. Ці сервера надають множину об’єктів. властивостей, методів, якими можна керувати. Таким чином, програма на C++ Builder отримує доступ до всіх можливостей сервера. Вона може формувати і форматувати будь які документи, зберігати їх, друкувати, відсилати по пошті і т. ін.

Програми для роботи з сервером автоматизації OLE суттєво відрізняється від програм для роботи з сервером COM. Порівняно з використанням серверів COM сервери автоматизації мають наступні переваги:

- властивості і методи сервера автоматизації повністю відповідають описам в вбудованих довідках Microsoft Office. В серверах COM в деяких випадках є суттєві розбіжності;
- програмний код додатку зрозумілий і зручний у використанні;
- додатки не залежать від версії C++ Builder;
- додатки легко переносяться з однієї версії Microsoft Office в іншу.

До недоліків слід віднести:

- в процесі розробки коду не з’являється підказка правильного написання методу чи властивості;
- оператори, при всій своїй наочності, досить довгі, порівняно з аналогічними операторами сервера COM.

При роботі з сервером автоматизації не потрібно підключати до додатку спеціальні додаткові модулі. Якщо додаток вміщує великі переліки констант, структур і т. ін., то стандартно використовуються заголовочні файли користувача.

Розглянемо варіант передачі даних між Microsoft Excel та C++ Builder. Для створення нового екземпляру сервера і з’єднання з ним використовується метод CreateObject класу Variant. Якщо потрібно приєднатися до уже існуючого екземпляру сервера, то використовується функція GetActiveObject класу Variant. В будь-яку з цих функцій

передається ім’я серверу під яким він зареєстрований у системі. Для з’єднання з сервером Excel можна використовувати ім’я “Excel.Application”. Можуть також використовуватися аналогічні імена з уточненням версії, наприклад, “Excel.Application.10”. Функції CreateObject і GetActiveObject повертають об’єкт типу Variant, пов’язаний з вказаним сервером. Наприклад,

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
// перевірка існування сервера та створення нового екземляру
try
{
 ExcelApplication1=Variant::GetActiveObject("Excel.Application");
}
catch(...)
 {
  try
  {
   ExcelApplication1=Variant::CreateObject("Excel.Application");

  }
  catch(...)
  {
   ShowMessage("Помилка, можливо на ПК не установлено Excel");
  }
 }
}
```

Подальша робота з сервером автоматизації може бути організована двома групами методів класу Variant: функцією Exec з аргументами класів Procedure (якщо метод не повертає значення), Function (якщо метод повертає значення), PropertyGet (повертає властивість сервера), PropertySet (задає значення властивості для сервера), або аналогічними функціями OleProcedure, OleFunction, OlePropertyGet, OlePropertySet.

Щоб при закритті додатку закривався і сервер слід використовувати оператор Quit, наприклад

ExcelApplication1.Exec(Procedure(“Quit”));

чи

ExcelApplication1.OleProcedure(Quit);

в залежності від методу, який було застосовано.

Після закінчення роботи зі змінними типу Variant, які були пов’язані з роботою сервера чи окремими його об’єктами, пам’ять слід очистити методом Clear:

ExcelApplication1.Clear();

чи присвоїти значення Unassigned:

ExcelApplication1= Unassigned;

Основні можливості роботи між додатками Microsoft Excel і C++ Builder демонструє інтерфейс користувача (рис.1), а також наведені тексти відповідних функцій.

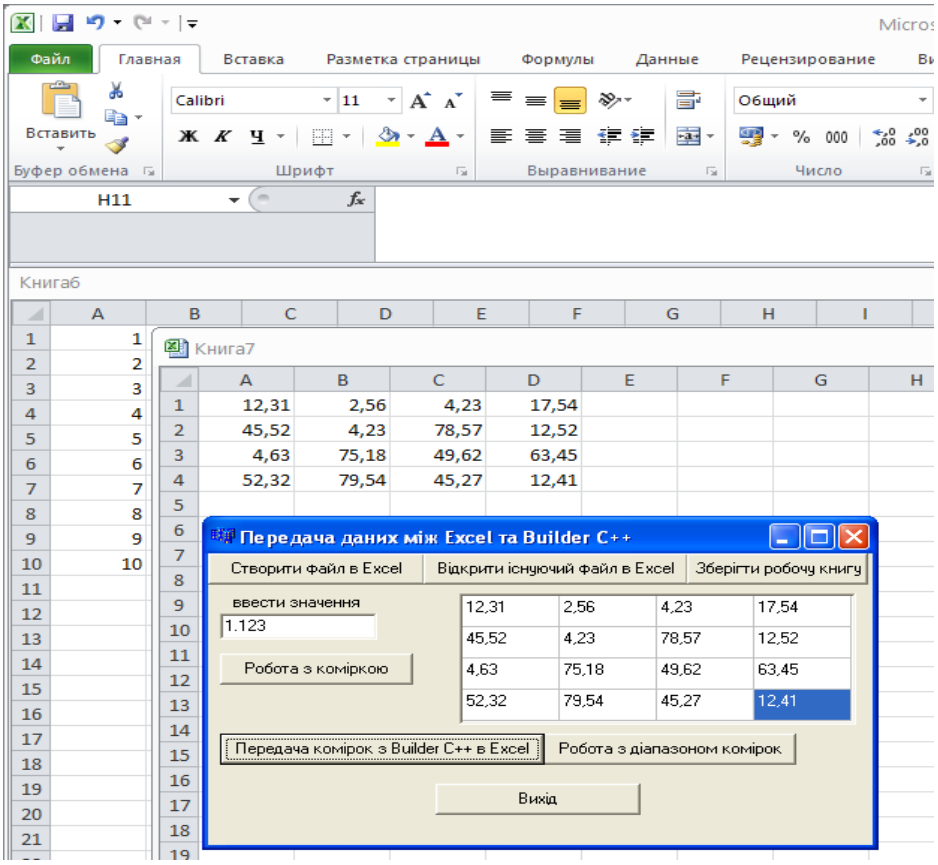

Рис. 1. Інтерфейс користувача передачі даних

Текст функції “Створити файл в Excel”

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
// перевірка існування сервера та створення нового екземляру
try
{
 ExcelApplication1=Variant::GetActiveObject("Excel.Application");
}
catch(...)
 {
  try
  {
   ExcelApplication1=Variant::CreateObject("Excel.Application");
  }
  catch(...)
  {
   ShowMessage("Помилка, можливо на ПК не установлено Excel");
  }
 }
}
```

Текст функції “Відкрити існуючий файл в Excel”

```
void __fastcall TForm1::Button3Click(TObject *Sender)
{
// відкрити існуючий файлу в Excel
if(OpenDialog1->Execute())
  try
  {
   ExcelApplication1.OlePropertyGet("Workbooks")
                .OleFunction("Add",WideString(OpenDialog1->FileName));
  }
  catch(...)
  {
   ShowMessage("Шаблон не можна відкрити");
  }
}
```

Текст функції “Зберігти робочу книгу”

```
void __fastcall TForm1::Button4Click(TObject *Sender)
{
// збереження відкритого .xls файлу
ExcelApplication1.OlePropertyGet("ActiveWorkbook")
            .OleProcedure("Save");
}
```

Текст функції “Робота з коміркою”

```
void __fastcall TForm1::Button5Click(TObject *Sender)
{
// читання значення  з комірки B3 і заміна його на значення з Edit поля
// 1 спосіб
//Variant MyRange=ExcelApplication1.OlePropertyGet("Range",WideString("B3"));
//Variant V=MyRange.OlePropertyGet("Value");
//ShowMessage(V);
//MyRange.OlePropertySet("Value",WideString(Edit1->Text));
// 2 спосіб
ExcelApplication1.OlePropertyGet("Range",WideString("B3"))
            .OlePropertySet("Value",WideString(Edit1->Text));
}
```

Текст функції “Передача даних з Builder C++ в Excel”

```
void __fastcall TForm1::Button7Click(TObject *Sender)
{
// передача комірок з Builder C++ в Excel
  ExcelApplication1.OlePropertyGet("WorkBooks").OleFunction("Add");
 for(int i=1; i<5; i++)
   for(int j=1; j<5; j++)
```

```
   ExcelApplication1.OlePropertyGet("Cells",i,j)
                .OlePropertySet("Value",WideString(StringGrid1->Cells[j-1][i-1]));
}
```

Текст функції “Робота з діапазоном комірок”

```
void __fastcall TForm1::Button6Click(TObject *Sender)
{
// обробка діапазону комірок
  ExcelApplication1.OlePropertyGet("WorkBooks").OleFunction("Add");
//  заповнення комірок в циклі
for(int i=1; i<=10; i++)
   ExcelApplication1.OlePropertyGet("Cells",i,1)
                .OlePropertySet("Value",i);
}
```

Таким чином, використовуючи призначення Microsoft Excel та C++ Builder можна створювати потужні додатки обробки інформації. На основі даної технології розроблена лабораторна робота, яку виконують студенти кафедри в курсі “Сучасне програмне забезпечення у вирішенні інженерно-математичних задач”.