УДК 528.001.1

Вороніна І.Ф., к.т.н., Грідчин Д.В., студент

АДІ ДВНЗ «ДонНТУ» , м. Горлівка

ЗАСТОСУВАННЯ СИСТЕМИ ВІЗУАЛЬНОГО ПРОЕКТУВАННЯ MICROSOFT VISUAL STUDIO 2008 ЩОДО РІШЕННЯ ІНЖЕНЕРНИХ ЗАДАЧ

Розрахунок координат вершин точок теодолітного ходу - це модуль програми майбутнього розрахункового комплексу, що дозволяє швидко та ефективно виконувати складні обчислення і вирішувати завдання в області інженерної геодезії. При створенні даного модуля була використана система візуального проектування Microsoft Visual Studio 2008.

Вступ

У цей час усе гостріше проявляється необхідність застосування знань в області програмування при виконанні інженерних розрахунків у різних галузях діяльності. Досить простою в освоєнні і у той же час однією з потужних сучасних мов програмування є Visual Basic. За допомогою цієї мови можна створювати додатки практично для будь-якої області сучасних комп'ютерних технологій.

Простота й потужність Visual Basic дозволили зробити його зручною мовою для додатків Microsoft Office.

На сучасному етапі знання Visual Basic і його діалектів (VBA і VBScript) стає необхідністю для фахівця будь-якого рівня.

Постановка завдання

Геодезія - це наука про виміри земної поверхні і її зображень. У геодезії вивчаються методи й техніка виробництва вимірювальних робіт, у процесі яких визначаються фігура й розміри Землі, створюються карти, плани, цифрові, математичні, геометричні й стереоскопічні моделі місцевості. Будуються уздовж заданих напрямків вертикальні розрізи (профілі) земної поверхні, вирішуються як безпосередньо на місцевості, так і на різних геодезичних матеріалах і моделях місцевості, всілякі завдання.

Аналізуючи і застосовуючи геодезичні методи, процеси і рішення, в інженерній геодезії необхідно виконувати складні й трудомісткі розрахунки. Так, для автоматизації обчислень і рішення задач в інженерній геодезії було вирішено застосувати систему візуального проектування Visual Basic, на основі якої розробити програму комплексного розрахунку координат вершин точок теодолітного ходу.

Мета

Застосувати систему об’єктно-орієнтированого програмування Visual Basic для рішення інженерних завдань. Як об'єкт автоматизації використати методику розрахунку координат вершин теодолітного ходу. Розробити алгоритм програми, описати змінні, створити інтерфейс модуля, написати програмний код і протестувати створену програму на предмет виконання та коректності результатів.

Основна частина

1. Алгоритм розробки програми в середовищі візуального проектування Microsoft Visual Studio 2008

1.1. Створення інтерфейсу модуля програми «Розрахунок координат»

Для створення інтерфейсу програми був застосований шаблон проекту Windows Forms Application, заснований на технології .NET Framework 3.5. Даний шаблон проекту створює стандартну форму типу System.Windows.Forms. У ході розробки інтерфейсу на головну форму програми були додані такі компоненти як NumericUpDown1 (числовий перемикач кількості точок - вершин теодолітного ходу), DataGridView1 (прямокутна таблиця даних, за допомогою якої здійснюється введення/виведення інформації), Panel1 (візуальна панель із розміщеними на ній чотирма текстовими мітками Label2, Label3, Label4, Label5, призначеними для виводу значень нев'язок, причому якщо нев'язка перевищує припустиу, то текст мітки підсвічюється червоними кольорами, що сигналізує про неприйнятну точність виконаних вимірів, які необхідно зробити заново). Також унизу форми втримується рядок стану. Командна кнопка Button1, розташована поруч із числовим перемикачем, запрограмована на комплексний розрахунок, у підсумку якого досягається кінцевий результат роботи програми (рис. 1). Ідентифікація змінних, використовуваних у програмі, показана у таблиці «Змінні».

Компоненти DataGridView1 і Panel1 прив'язані програмно до правого нижнього кута вікна форми:

```
Private Sub Form1_Resize(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Resize
DataGridView1.Width = Me.Width - 8
DataGridView1.Height = Me.Height - 175
Panel1.Width = Me.Width - 8
End Sub
```

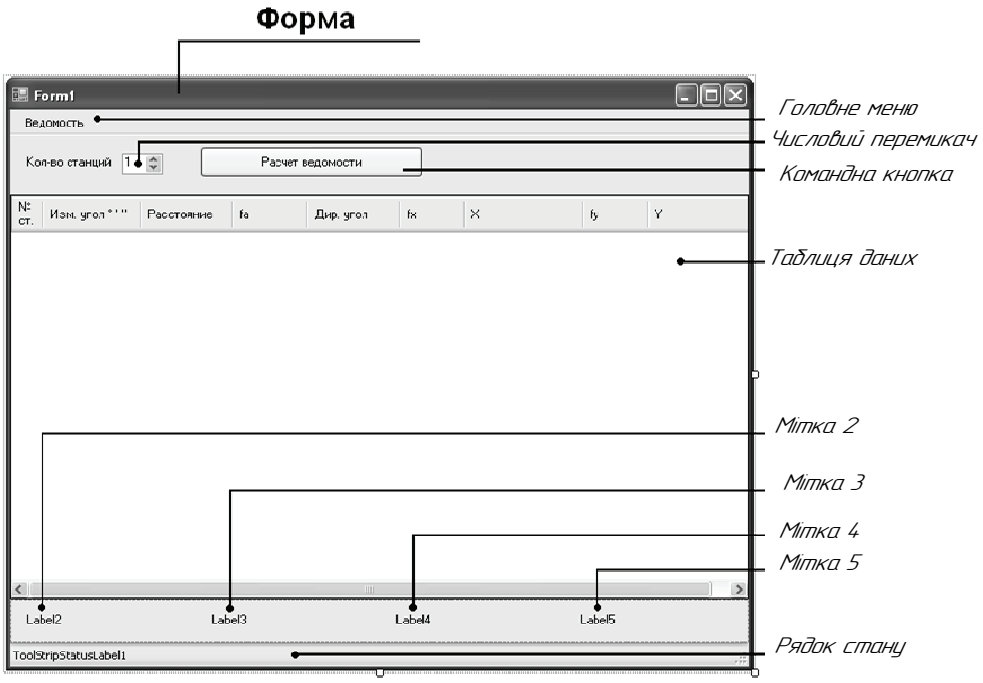

Рис. 1. Інтерфейс програми

При ініціалізації події завантаження головної форми виконується присвоєння заголовків, у першу чергу, самій формі Form1, а також компонентам Label2, Label3, Label4,

Label5 надається значення за замовчуванням числового перемикача; чарунки3 таблиці, що очікують уведення користувачем вихідних даних, офарблюються у відтінок жовтих кольорів

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
Me.Text = "Модуль " & Chr(34) & "Розрахунок координат" & _Chr(34)
NumericUpDown1.Value = 3
ToolStripStatusLabel1.Text = ""
DataGridView1.Item(4,        0).Style.BackColor        =
        System.Drawing.Color.FromArgb(255, 255, 192)
DataGridView1.Item(6,        0).Style.BackColor        =
                System.Drawing.Color.FromArgb(255, 255, 192)
DataGridView1.Item(8,        0).Style.BackColor        =
                System.Drawing.Color.FromArgb(255, 255, 192)
Label2.ForeColor = Color.Blue
Label2.Text = "Кутова нев’язка: ?"
Label3.ForeColor = Color.Blue
Label3.Text = "Нев'язка dx: ?"
Label4.ForeColor = Color.Blue
Label4.Text = "Нев'яка dy: ?"
Label5.ForeColor = Color.Blue
Label5.Text = "Відносна нев’язка: ?"
End Sub
```

Таблиця 1

Змінні

№	ПАРАМЕТРИ	ПОЗНАЧЕННЯ	ЗНАЧЕННЯ	ЗАМІТКИ
1	2	3	4	5
1	Обмірювані кути (одновимірний масив)	A	###°##'##”	виміряється
2	Обмірювані відстані (одновимірний масив)	S	###,##	виміряється
3	Фактична сума обмірюваних кутів	SumBf	Обчислюється програмою	$\sum\beta_{факт}$
4	Теоретична сума кутів	SumBt		У замкнутому багатокутнику дорівнює 180°(n-2)
5	Фактична кутова нев'язка	fBf		Розраховується як $\sum\beta_{факт}-\sum\beta_{теор}$
6	Припустима кутова нев'язка	fBd		Визначається за формулою $1'\sqrt{\sqrt{n}\,n}$
7	Виправлення до обмірюваних горизонтальних кутів	p		$P_в=\frac{-f_{факт}}{n}$
8	Сума виправлених кутів	SumBi		$\sum\beta_{испр}=\sum\beta_{факт}+nP_\beta$
9	Дирекційні кути напрямків ходу (одновимірний масив)	dir	###°##'##”	Програма запитує тільки дирекційній кут першої сторони. Інші обчислюються автоматично.
10	Косинуси дирекційних кутів (одновимірний масив)	cosd	Обчислюється програмою	Масив косинусів дирекційних кутів

Продовження табл. 1

1	2	3	4	5
11	Синуси дирекційніх кутів (одновимірний масив)	sind	Обчислюється програмою	Масив синусів дирекційніх кутів
12	Збільшення координати X (одновимірний масив)	dx		Виходячи з рішення прямого геодезичного завдання $\Delta x = S\cos\alpha,\ \Delta y = S\sin\alpha$
13	Збільшення координати Y (одновимірний масив)	dy		
14	Сума збільшень координати X	Sumdx		$\sum \Delta x$
15	Сума збільшень координати Y	Sumdy		$\sum \Delta y$
16	Сумарна нев'язка збільшень координати X	fdx		$fx = \sum \Delta x_{факт} - \sum \Delta x_{теор}$
17	Сумарна нев'язка збільшень координати Y	fdy		$fy = \sum \Delta y_{факт} - \sum \Delta y_{теор}$
18	Абсолютна нев'язка	fAbs		$f_{абс} = \sqrt{fx^2 + fy^2}$
19	Периметр замкнутого полігона	P		$P = \sum S$
20	Знаменник відносної нев'язки $f_{отн} = \frac{1}{P / f_{абс}}$	fOtnZn		$\frac{1}{f_{отн}} = \frac{P}{f_{абс}}$
21	Виправлення збільшень координати X (одномірний масив)	Px		Виправлення вводяться приблизно пропорційно довжинам ліній зі знаком, зворотним знаку нев'язки0 $P_{x_i} \cong \frac{-f_x S_i}{P}$ $P_{y_i} \cong \frac{-f_y S_i}{P}$
22	Виправлення збільшень координати Y (одновимірний масив)	Py		
23	Сума виправлень збільшень координати X	SumPx		$\sum P_x ; \sum Py$
24	Сума виправлень збільшень координати Y	SumPy		
25	Виправлені збільшення координати X	dxi		$\sum \Delta x_i + P_{x_i}$ $\sum \Delta y_i + P_{y_i}$
26	Виправлені збільшення координати Y	dyi		
27	Сума виправлених збільшень координати X	Sumdxi		$\sum \Delta x_{испр};\ \sum \Delta y_{испр}$ Теоретично повинні рівнятися 0
28	Сума виправлених збільшень координати Y	Sumdyi		
29	Координати X	X		Кінцевий результат роботи програми
30	Координати Y	Y		

1.2. Командний код

Результатом кутових вимірів геодезичними приладами є кути, представлені у форматі D°M'". Однак, у середовищі розробки Microsoft Visual Studio 2008 не передбачена робота з таким типом даних, тому раціональним буде рішення переводити міру кута з <градуси>°<хвилини>'<секунди>" у градуси з десятковими частками. Такий числовий тип можна визначити в системі візуального проектування і виконувати над ним різні операції (складати,

віднімати тощо). Для цього в модулі Module1 реалізовані дві взаємозворотні функції (підпрограми), які конвертують кутову міру з одного формату в інший.

1.2.1. Функція Convert_Decimal переводить міру кута з формату D°M'" у градуси з десятковими частками:

```
Function Convert_Decimal(ByVal Degree_Deg As String) As Double
' Оголошення змінних у форматі числа із плаваючою 'комою подвійної точності.
Dim degrees As Double
Dim minutes As Double
Dim seconds As Double
' Установити число градусів рівним частині аргументу 'функції до символу «°»
degrees = Val(Microsoft.VisualBasic.Left(Degree_Deg, InStr(1, _
                Degree_Deg, "°")))
' Установити число хвилин рівним частині аргументу 'функції між символами «°» і «'» текстового рядка Degree_Deg, 'переведеної в числовий формат за допомогою функції Val і 'поділеної на 60.
minutes = Val(Mid(Degree_Deg, InStr(Degree_Deg, "°") + 1, 2)) / 60
' Установити число хвилин рівним частини аргументу 'функції праворуч від символу «'», переведеної в числовий 'формат і поділеної на 3600.
seconds = Val(Mid(Degree_Deg, InStr(Degree_Deg, "'") + 1, 2)) / 3600
Convert_Decimal = degrees + minutes + seconds
End Function
```

1.2.2. Функція Convert_Degree переводить міру кута із градусів з десятковими частками у формат D°M'":

```
Function Convert_Degree(ByVal Decimal_Deg) As Object
     ' Оголошення змінних для значень градусів, хвилин, секунд 'окремо
Dim Degrees As Integer
Dim Minutes As Double
Dim Seconds As Double
' Передбачити два випадки: Decimal_Deg < 0 і Decimal_Deg > 0
Dim n As Integer
If Decimal_Deg < 0 Then
n = 2
Decimal_Deg = Math.Abs(Decimal_Deg)
Else
n = 1
End If
     'Установити число градусів рівним цілій частині аргументу 'функції
Degrees = Int(Decimal_Deg)
     'Установити число хвилин рівним числу правору від 'десяткового значення для змінної Decimal_Deg   '(десяткових_град), помноженому на 60
Minutes = (Decimal_Deg - Degrees) * 60
'Установити число секунд рівним числу праворуч
'від десяткового значення для змінної Minute (хвилини), 'помноженому на 60
```

```
Seconds = Format(((Minutes - Int(Minutes)) * 60), "0")

'Усунути випадки, коли кількість секунд або хвилин перевищує 60, а
градусів більше 360
While Seconds >= 60
Minutes = Minutes + 1
Seconds = Seconds - 60
End While
While Minutes >= 60
Degrees = Degrees + 1
Minutes = Minutes - 60
End While
While Degrees >= 360
Degrees = Degrees - 360
End While
'Повернути результат перекладу градусів
'залежно від знака аргументу функції
'(наприклад, 10.46 = 10°27'36")
Select Case n
Case 1
Convert_Degree = " " & Degrees & "°" & Int(Minutes) & "'" _
& Seconds & Chr(34)
Case 2
Convert_Degree = "-" & Degrees & "°" & Int(Minutes) & "'" _
& Seconds & Chr(34)
End Select
End Function
```

При зміні значення числового перемикача, що визначає кількість станцій ходу, міняється число рядків у таблиці (Minimum = 3, Maximum = 25).

```
    Private Sub NumericUpDown1_ValueChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
NumericUpDown1.ValueChanged
DataGridView1.RowCount = NumericUpDown1.Value
    End Sub
```

Командний код щиглика кнопки Button1:

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button1.Click
```

1.3. Оголошення змінних, що беруть участь у розрахунку

```
Dim A(0 To NumericUpDown1.Value - 1) As Double
Dim S(0 To NumericUpDown1.Value - 1) As Double
Dim SumBf As Double
Dim SumBt As Double
Dim fBf As Double
Dim fBd As Double
Dim p As Double
Dim SumBi As Double
Dim dir(0 To NumericUpDown1.Value - 1) As Double
Dim Cosd(0 To NumericUpDown1.Value - 1) As Double
Dim Sind(0 To NumericUpDown1.Value - 1) As Double
Dim dx(0 To NumericUpDown1.Value - 1) As Double
```

```
Dim dy(0 To NumericUpDown1.Value - 1) As Double
Dim Sumdx As Double
Dim Sumdy As Double
Dim fdx As Double
Dim fdy As Double
Dim fAbs As Double
Dim P As Double
Dim fOtnZn As Double
Dim Px(0 To NumericUpDown1.Value - 1) As Double
Dim Py(0 To NumericUpDown1.Value - 1) As Double
Dim SumPx As Double
Dim SumPy As Double
Dim dxi(0 To NumericUpDown1.Value - 1) As Double
Dim dyi(0 To NumericUpDown1.Value - 1) As Double
Dim Sumdxi As Double
Dim Sumdyi As Double
Dim X(0 To NumericUpDown1.Value - 1) As Double
Dim Y(0 To NumericUpDown1.Value - 1) As Double
Const Pi = Math.PI
```

Обчислення припустимої кутової нев'язки в залежності від кількості сторін полігона:

```
fBd = 1/60 * Math.Sqrt(NumericUpDown1.Value)
```

Теоретична сума кутів замкнутого полігона:

```
SumBt = 180 * (NumericUpDown1.Value - 2)
```

Обнулімо фактичну суму кутів і периметр перед нагромадженням:

```
SumBf = 0
P = 0
```

Привласнюємо елементам масиву відстаней S і масиву обмірюваних кутів A (переведені в десятковий формат функцією Convert_Decimal) значення, введені у відповідні стовпчики таблиці DataGridView1, обчислюємо периметр і фактичну суму кутів замкнутого полігона

```
For i As Integer = 0 To NumericUpDown1.Value - 1
S(i) = DataGridView1.Item(2, i).Value
A(i) = Convert_Decimal(DataGridView1.Item(1, i).Value)
SumBf = SumBf + A(i)
P = P + S(i)
Next
```

1.4. Розрахунок фактичної кутової нев'язки та вивід її на екран

```
fBf = SumBf - SumBt
Label2.Text = "Кутове нев'язання: " + Convert_Degree(fBf)
```

Порівнюємо абсолютне значення фактичної кутової нев'язки із припустимим:

```
If Math.Abs(fBf) < fBd Then
Label2.ForeColor = Color.Green
Else
Label2.ForeColor = Color.Red
End If
```

Обчислюємо виправлення в обмірюваний кут:

```
p = (-1) * fBf / NumericUpDown1.Value
SumBi = 0
```

Виправлення обмірюваних кутів шляхом введення виправлень, нагромадження суми виправлених кутів для подальшого контролю:

```
For i As Integer = 0 To NumericUpDown1.Value - 1
A(i) = A(i) + p
SumBi = SumBi + A(i)
DataGridView1.Item(3, i).Value = Convert_Degree((-1) * p)
Next
```

Якщо й цього разу має місце розбіжність із теоретичною сумою, то відбулася непередбачена помилка і подальші обчислення робити не можна:

```
If SumBi <> SumBt Then
MsgBox("Внутрішня помилка введення виправлень. Додаток буде
                закрито.", _
MsgBoxStyle.Critical, "Система")
Me.Close()
End If
```

Вводимо дирекційній кут першої сторони:

```
dir(0) = Convert_Decimal(DataGridView1.Item(4, 0).Value)
```

Виходячи з першого, обчислюються наступні та виводяться в таблицю:

```
For i As Integer = 0 To NumericUpDown1.Value - 2
dir(i + 1) = dir(i) + 180 - A(i + 1)
DataGridView1.Item(4, i + 1).Value = Convert_Degree(dir(i + 1))
Next
Sumdx = 0
Sumdy = 0
```

Обчислення збільшень координат та їхніх сум:

```
For i As Integer = 0 To NumericUpDown1.Value - 1
dx(i) = S(i) * Math.Cos(dir(i) * Pi / 180)
dy(i) = S(i) * Math.Sin(dir(i) * Pi / 180)
Sumdx = Sumdx + dx(i)
Sumdy = Sumdy + dy(i)
Next
```

Визначення нев'язок збільшень, вивід їх на екран, розрахунок абсолютної та відносної нев'язки:

```
fdx = Sumdx
fdy = Sumdy
Label3.Text = "Нев'язання dx: ~ " & Math.Round(fdx, 2)
Label4.Text = "Нев'язання dy: ~ " & Math.Round(fdy, 2)
Label3.ForeColor = Color.Green
Label4.ForeColor = Color.Green
fAbs = Math.Sqrt(fdx ^ 2 + fdy ^ 2)
fOtnZn = P / fAbs
```

Якщо знаменник відносної нев'язки перевищує 2000, то нев'язка вважається прийнятою:

```
If fOtnZn < 2000 Then
Label5.ForeColor = Color.Red
Else
Label5.ForeColor = Color.Green
End If
Label5.Text = "Отн. нев'язання: 1/" & Math.Round(fOtnZn)
SumPx = 0
SumPy = 0
```

Обчислення виправлень у збільшення координат, та їхніх сум:

```
For i As Integer = 0 To NumericUpDown1.Value - 1
```

```
Px(i) = (-1 * fdx * S(i)) / P
Py(i) = (-1 * fdy * S(i)) / P
DataGridView1.Item(5, i).Value = Math.Round(Px(i), 2)
DataGridView1.Item(7, i).Value = Math.Round(Py(i), 2)
SumPx = SumPx + Px(i)
SumPy = SumPy + Py(i)
Next
```

Сума виправлень повинна рівнятися нев'язці, взятій із протилежним знаком:

```
If (SumPx <> -1 * fdx) Or (SumPy <> -1 * fdy) Then
MsgBox("Внутрішня помилка обчислення виправлень. Додаток буде закрито.", _
MsgBoxStyle.Critical, "Система")
Me.Close()
End If
Sumdxi = 0
Sumdyi = 0
```

Введення виправлень у відповідні збільшення, нагромадження суми виправлених збільшень:

```
For i As Integer = 0 To NumericUpDown1.Value - 1
dxi(i) = dx(i) + Px(i)
dyi(i) = dy(i) + Py(i)
Sumdxi = Sumdxi + dxi(i)
Sumdyi = Sumdyi + dyi(i)
Next
```

Одержання координат першої точки ходу:

```
X(0) = DataGridView1.Item(6, 0).Value
Y(0) = DataGridView1.Item(8, 0).Value
```

Обчислення наступних координат і відображення їх у таблиці:

```
For i As Integer = 0 To NumericUpDown1.Value - 2
X(i + 1) = X(i) + dxi(i)
Y(i + 1) = Y(i) + dyi(i)
DataGridView1.Item(6, i + 1).Value = X(i + 1)
DataGridView1.Item(8, i + 1).Value = Y(i + 1)
Next
End Sub
```

Якщо необхідно виконати розрахунок знову, то в головному меню **Відомість** створений пункт **Новий розрахунок**, що обновить поле таблиці, видаливши з неї усі дані:

```
Private Sub НовыйРасчетToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
НовыйРасчетToolStripMenuItem.Click
For i As Integer = 0 To DataGridView1.ColumnCount - 1
For j As Integer = 0 To DataGridView1.RowCount - 1
DataGridView1.Item(i, j).Value = ""
Next
Next
End Sub
```

Примітка: якщо аргумент функції Convert_Decimal(Degree_Deg) є рядком, то необхідно вводити вихідні кути суворо у форматі <градуси>°<хвилини>’<секунди>” без пропусків (Значок «°» можна одержати натисканням клавіш ALT+0176).

2. Тестування програми

Перед початком уведення даних у таблицю необхідно визначитися із кількістю вершин теодолітного ходу. У нашому випадку їх буде 6.

У першому рядку стовпчика «Изм. Кут ° ‘ “» необхідно ввести кут при першій вершині теодолітного ходу, у сусідній чарунці стовпця «Відстань» вказується відстань від першої точки до другої точки ходу по годинній стрілці. Інші кути й відстані вводяться аналогічно. Потім у чарунці дирекційний кут вказується дирекційний кут першої сторони ходу, а в першому рядку стовпчиків «X» і «Y» - відповідно координата X и Y першої вершини теодолітного ходу.

Усі чарунки в стовпцях 2 і 3, а також чарунки 5 і 7, у першому рядку, повинні бути заповнені, інакше програма не виведе результат або відобразить невірний.

На рисунку 2 показаний підсумок роботи програми.

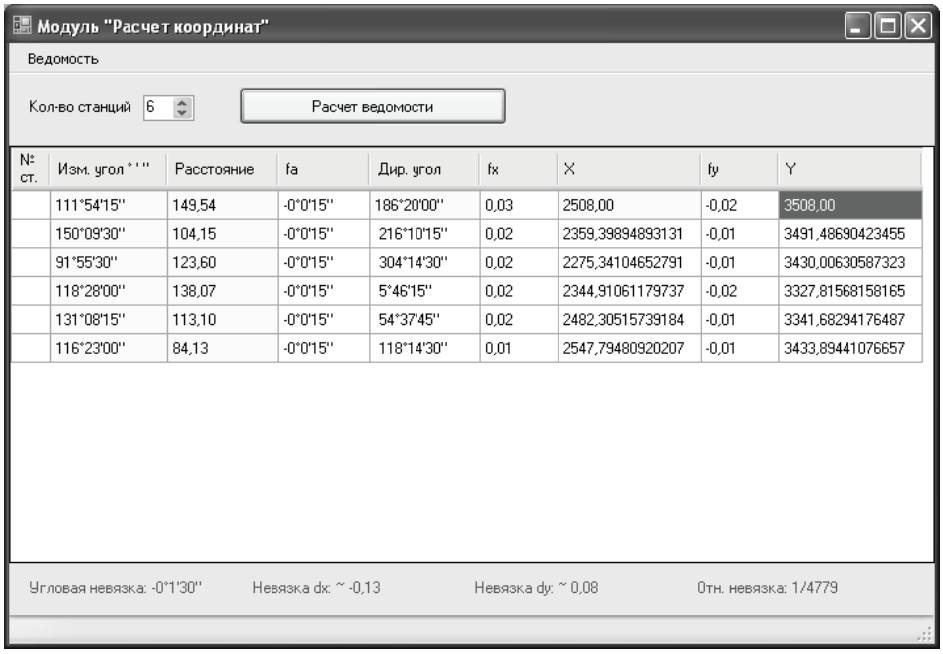

Рис. 2. Кінцевий результат роботи програми

Результат обчислень коректний, тому що аналогічні дані були отримані у ході трудомісткого розрахунку вручну.

Висновок

Розроблений модуль програми доводить, що автоматизація процесу передачі координат від однієї вершини полігона до всіх інших вершин за допомогою прямого геодезичного завдання, а також обчислення координат в абсолютній системі доцільно. У ході тестування програми був отриманий коректний результат.

Програма може бути використана як у геодезичній практиці студентів так і як база для подальшої розробки розрахункового комплексу з геодезії.

Список літератури

1. Федоров В.И., Шилов П.І. Інженерна геодезія. — М.: Надра, 1982. — 359 с.
2. Кондратов Ю.Н. Програмування на Visual Basic для Windows. — Webmaster (c), 2001. — 127 с.
3. Коляда М.Г. Інформатика та комп'ютерні технології: Навчальний посібник. — Донецьк: Батьківщина, 1999. — 608 с.

Стаття надійшла до редакції 24.04.08