УДК 378:147:51:044.9

Дорошенко М.В.
Дрогобицький державний педагогічний університет імені Івана Франка, Дрогобич, Україна

НАВЧАННЯ ОСНОВ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ ЗАСОБАМИ VISUAL C#

DOI: 10.14308/ite000723

У статті наведено методичні рекомендації щодо вивчення освітньої дисципліни «Об'єктно-орієнтоване програмування» для підготовки фахівців першого (бакалаврського) рівня вищої освіти галузі знань 01 «Освіта» спеціальності 014 «Середня освіта (інформатика)» освітньої програми (інформатика, математика). Ця дисципліна відіграє особливо важливу роль у підготовці вчителів інформатики, тому що галузь розробки комп'ютерних програм з використанням новітніх технологій, до яких належить об'єктно-орієнтоване програмування, є важливим складником науково-технологічного прогресу.

Студентам пропонується розглянути основні властивості об'єктно-орієнтованого програмування та принципи візуального підходу до проєктування програмного забезпечення в інтегрованому середовищі Visual Studio.Net з використанням Visual C#. Вибір Visual Studio.Net зумовлений тим, що версія Visual Studio.Net Community є безкоштовною повнофункціональною версією, яка відрізняється від професійної лише незначним обмеженням функцій.

У статті розглянута методика вивчення основ та принципів об'єктно-орієнтованого програмування шляхом демонстрування на лекціях типових прикладів та закріплення вивченого матеріалу на лабораторних роботах. Приклади дібрані таким чином, щоб можна було продемонструвати особливості введення та виведення даних у середовищі візуального програмування Visual C#, програмування подій, обробку файлів, реалізацію механізмів інкапсуляції, наслідування та поліморфізму. Одночасно з демонстрацією реалізації типових прикладів вивчаються потрібні властивості елементів керування, які використовуються для реалізації проєкту.

Завдяки виконанню лабораторних робіт студенти закріплюють знання, отримані на лекціях, та набувають практичних навичок розробки проєктів засобами Visual C#. Лабораторні роботи завершуються створенням багатовіконного проєкту, який представляє комп'ютерну навчальну систему з вивчення методів обчислень. Створення такого проєкту дає можливість закріпити набуті студентами теоретичні знання та практичні навички з розробки програм і демонструє використання у освітньому процесі міжпредметних зв'язків.

Ключові слова: *учитель інформатики, об'єктно-орієнтоване програмування, середовище візуального програмування Visual C#.*

Постановка проблеми. Об'єктно-орієнтованого програмування (ООП) заслуговує на більш широке використання на уроках інформатики в закладах загальної середньої освіти. Але існують проблеми використання ООП у цих закладах, а саме: недостатня увага приділяється найбільш поширеному та дуже актуальному об'єктно-орієнтованому програмуванню, низька обізнаність учителів інформатики з основними концепціями ООП та з мовою програмування C# безпосередньо [9]. Тому в процесі підготовки майбутніх учителів

інформатики важливим є засвоєння студентами фундаментальних понять з програмування, набуття навичок практичної роботи з тими мовами програмування, які розроблялися для якомога повнішої реалізації візуального та об'єктно-орієнтованого підходу до написання програмного забезпечення. Однією з таких мов є Visual C#.

Підготовка фахівців першого (бакалаврського) рівня вищої освіти галузі знань 01 «Освіта» спеціальності 014 «Середня освіта (інформатика)» освітньої програми (інформатика, математика) потребує вивчення на першому році навчання дисципліни «Програмування», а на другому – «Об'єктно-орієнтованого програмування». Під час вивчення цих дисципліни студенти повинні сформувати базу знань, умінь та отримати навички, що будуть супроводжувати їх і під час вивчення інших дисциплін, а також у подальшій професійній діяльності.

Предметом освітньої дисципліни «Об'єктно-орієнтоване програмування» є вивчення основних концепцій ООП та принципів об'єктно-орієнтованого підходу до проєктування прикладних програм.

Завданням дисципліни є розкриття значення предмету в загальній та професійній підготовці, місце ООП в розробленні інформаційних систем; формування основ інформаційної культури майбутнього вчителя інформатики та ознайомлення студентів з основними поняттями та принципами ООП як сучасного засобу для розроблення об'єктно-орієнтованого програмного забезпечення.

Під час вивчення освітньої дисципліни «Об'єктно-орієнтоване програмування» майбутньому вчителеві інформатики пропонується ознайомитися з інтегрованим середовищем розробки програмних продуктів Visual Studio.Net. Вибір Visual Studio.Net зумовлений тим, що версія Visual Studio.Net Community є безкоштовним програмним продуктом, яку можна завантажити з офіційного сайту компанії Microsoft. Це практично повнофункціональна версія, яка відрізняється від професійної лише незначним обмеженням функцій і неповними бібліотеками класів.

Для вивчення освітньої дисципліни «Об'єктно-орієнтоване програмування» пропонується використовувати мову програмування Visual C#, яка є сучасною об'єктно-орієнтованою мовою, що створена для роботи на платформі .NET з урахуванням нових віянь в ООП і позбавлена від недоліків Visual C++.

Аналіз останніх досліджень та публікацій. Мова програмування C# була розроблена протягом 1999–2000 рр. Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research на фірмі Microsoft.

Програмування мовою C# найбільш повно розглядаються в наукових публікаціях таких авторів, як К. Нейгел, Б. Ивьєн, Дж. Глинн [5], Е. Троелсен [7], Г. Шилдт [10], К. Уотсон, К.Нейгел, Я. Педерсен, Д. Рид, М. Скиннер [11].

Також у освітньому посібнику [4] викладено основи об'єктно-орієнтованого програмування на прикладі мови C#, основні принципи та засоби створення консольних додатків у середовищі Visual Studio.Net з використанням платформи .Net Framework, а в методичних рекомендаціях [1] розглядаються базовий синтаксис та основні програмні конструкції мови C# у рамках структурного підходу до розробки програм

У роботі [2] описана комп'ютерна Освітня система з методів обчислень, під час створення якої використовувалися засоби Visual C#, а в освітньому курсі [6] розглядаються принципи інтеграції СКМ Matlab з мовою програмування Visual C#.

Крім того, у деяких публікаціях багато уваги приділяється проблемам вивчення мови програмування C# у закладах загальної середньої освіти. У методичних рекомендаціях [9] розглянуто проблеми ознайомлення учнів з основними особливостями об'єктно-орієнтованої парадигми програмування, а також детально розглянуто особливості навчання мови C# у закладах загальної середньої освіти, а в процесі [8] досліджувались особливості переходу до вивчення програмування в закладах загальної середньої освіти від мови Pascal до C#.

Об'єкт дослідження – процес навчання дисципліни «Об'єктно-орієнтоване програмування» студентів спеціальності 014 «Середня освіта (інформатика)» освітньої програми (інформатика, математика) у педагогічному університеті.

Предмет дослідження – особливості вивчення об'єктно-орієнтованого програмування засобами Visual C#.

Метою дослідження – розглянути особливості реалізації основ та принципів візуального та об'єктно-орієнтованого програмування в процесі вивчення освітньої дисципліни «Об'єктно-орієнтоване програмування» для студентів спеціальності 014 «Середня освіта (інформатика)» освітньої програми (інформатика, математика).

Виклад основного матеріалу. Освітня дисципліна «Об'єктно-орієнтоване програмування» для спеціальності 014 «Середня освіта (інформатика)» освітньої програми (інформатика, математика) вивчається протягом усього другого курсу та є продовженням дисципліни «Програмування», в якій студенти знайомляться з базовими конструкціями мови програмування C++ та прикладами реалізації типових алгоритмів [3].

Під час навчання дисципліни «Об'єктно-орієнтоване програмування» передбачається вивчення основних властивостей ООП, а саме: інкапсуляції, наслідування та поліморфізму, створення Windows-додатків засобами Visual C# та формування практичних навичок роботи у середовищі Visual Studio.Net.

Середовище розробки Visual Studio.Net – це програмний продукт, у якому реалізовано принципово новий підхід до побудови каркаса середовища – Framework .Net.

Це середовище розробки додатків для розв'язування широкого класу задач, яке є в свою чергу відкритим програмним середовище, до складу якого входять такі мови програмування, як Visual C++ .Net,Visual C#.Net, F#.Net, Visual Basic.Net.

Visual C# можна використовувати для створення не тільки консольних та Windows додатків, але й додатків автоматизації обробки Word та Excel документів, додатків для мобільних платформ та комп'ютерних ігор, Web-додатків та багато інших додатків для розв'язування широкого класу наукових та бізнес-завдань. Для майбутнього вчителя інформатики важливо вміти створювати насамперед консольні, Windows-додатки та додатки автоматизації роботи з Word та Excel документами.

Зосередимо увагу на організацію процесу навчання студентів спеціальності 014 «Середня освіта (інформатика)» освітньої програми (інформатика, математика) основам та принципам об'єктно-орієнтованого програмування з використанням мови програмування Visual C#. Наведемо деякі методичні рекомендації щодо проведення лекційних та лабораторних занять.

Під час проведення лекцій із зазначеної дисципліни акцентується увага на вивченні таких тем, а саме: організація вводу та виводу даних засобами Visual C#, програмування подій, обробка файлів, створення статичних та динамічних класів, вивчення механізмів наслідування та поліморфізму, обробка графічних зображень. Оскільки середовище Visual C# інтуїтивно зрозуміле та побудоване таким чином, що містить контекстні підказки, які дозволять швидко створювати додатки, то вивчення вищеперелічених тем проводиться в комп'ютерному класі та реалізується шляхом демонстрації типових прикладів.

Матеріал, вивчений під час лекційних занять, закріплюється студентами на лабораторних заняттях, де пропонується ознайомитися з інтегрованим середовищем Visual Studio, зокрема із середовищем візуального програмування Visual C#, особливостями введення даних та виведення результатів, програмуванням подій мишки, клавіатури та системних подій, обробкою файлів, створенням статичних та динамічних класів, реалізацією механізмів наслідування та поліморфізму, побудовою графіків функцій, обробкою графічних зображень і створення багатовіконних додатків. Структура лабораторних робіт складається з формування вихідної задачі, побудови алгоритму її розв'язку, аналізу отриманих результатів.

Розглянемо деякі приклади, які використовуються під час проведення лекційних занять.

Приклад 1. Створити програму обчислення суми:

$$S = \sum_{i=1}^{n} \cos ix$$

Цей приклад використовується для того, щоб продемонструвати студентам різні способи вводу даних та виводу результатів виконання програми у середовищі Visual C#. Вікно програми зображене на рис. 1.

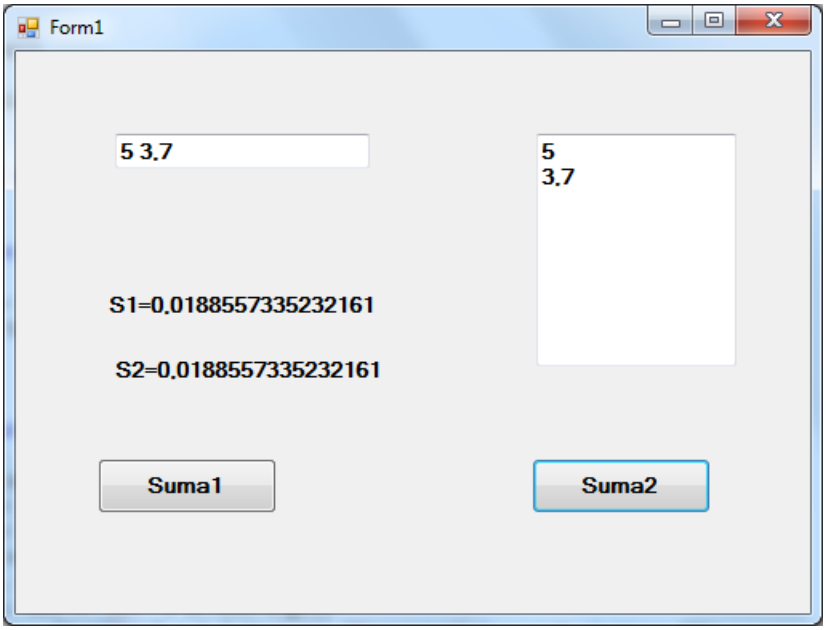

Рис. 1. Вікно програми з використанням двох способів введення даних.

Вхідими даними в програмі є ціле число *n* та дійсне – *x*. Для введення цих даних у програмі використані однорядкове текстове поле *textBox2* та багаторядкове – *textBox1*. Уведення вхідних даних *n, x* з однорядкового текстового поля *textBox2* реалізується шляхом використання методів *IndexOf* та *Substring*, які визначені в класі *String*, а обчислення суми реалізовано за допомогою функції Sum. Підпрограма, яка реалізує введення даних з однорядкового текстового поля *textBox2* та обчислення суми, має такий вигляд:

```
private void button1_Click(object sender, EventArgs e)
        {
            string str = textBox2.Text;
            int k = str.IndexOf(' ');
            int n = Convert.ToInt32(str.Substring(0, k));
            double x = Convert.ToDouble(str.Substring(k+1));
             Sum(n, x);
             label1.Text = "S=" + S.ToString();
                    }
```

Інший спосіб реалізації введення даних полягає у використанні багаторядкового поля. Для того, щоб елемент керування *textBox1* став багаторядковим текстовим полем, потрібно властивості *MultyLine* надати значення *true*. Підпрограма реалізації введення даних з багаторядкового текстового поля *textBox1* та обчислення суми має такий вигляд:

```
        private void button2_Click(object sender, EventArgs e)
        {
            int n = Convert.ToInt32(textBox1.Lines[0]);
```

```
double x = Convert.ToDouble(textBox1.Lines[1]);
Sum(n, x);
label2.Text = "S=" + S.ToString();
 }
```

Для виводу результатів виконання програми використовуються мітки *label1* та *label2*. Крім того, студентам демонструється спосіб виводу результатів за допомогою вікна повідомлень, яке можна створити, наприклад, таким чином:

```
MessageBox.Show("S1=" + S.ToString(),"Результат",MessageBoxButtons.OK);
```

У прикладах, які демонструють обробку одномірних масивів та матриць, для введення масивів використовуються елементи керування *textbox* та *dataGridView*.

У прикладі 2 демонструється програмування подій, а саме: подій мишки *MouseDown* та *MouseMove*, клавіатури *KeyDown* та системної події *Tick*. Вікно програми зображене на рис.2.

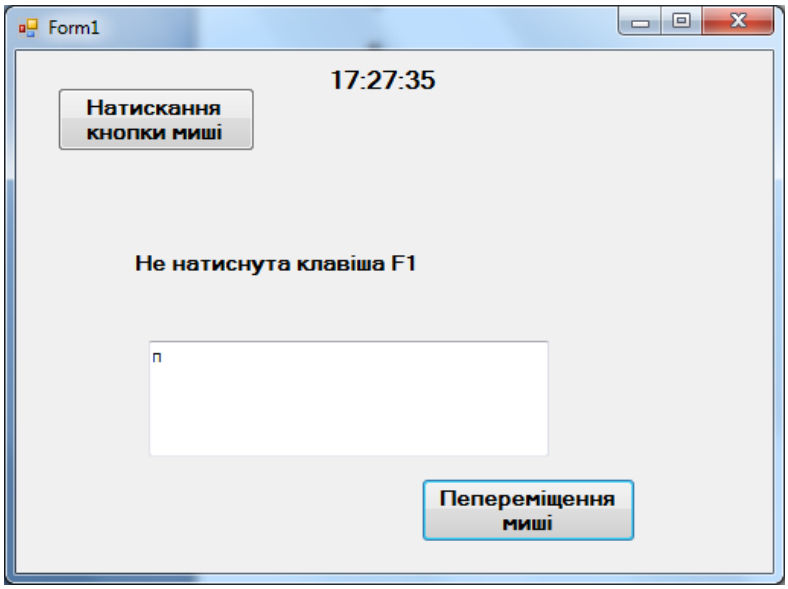

Рис. 2. Вікно програми програмування подій

Відображення рухомого часу реалізовано як підпрограму реакції на системну подію *Tick* для елемента керування таймер *timer1*, у якій записується такий оператор:

```
label2.Text = System.DateTime.Now.ToLongTimeString();
```

За допомогою підпрограми обробки реакції кнопки команд *button1* на подію *MouseDown* розпізнається, яка клавіша мишки була натиснута, а саме: ліва, права чи середня. Наприклад, розпізнавання натиснення лівої кнопки реалізується таким чином:

```
if (e.Button == MouseButtons.Left)
   MessageBox.Show("натистута ліва кнопка миші");
```

За допомогою підпрограми обробки реакції кнопки команд *button2* на подію *MouseMove* здійснюється переміщення мишки по вікні програми.

```
private void button2_MouseMove(object sender, MouseEventArgs e)
{
 if (e.Button == MouseButtons.Left)
       button2.Location = new Point(button2.Location.X + e.X,
button2.Location.Y + e.Y);
                }
```

А підпрограма *textBox1_KeyDown* розпізнає, чи була натиснута одна з функціональних клавіш: F1, F2,…, F12. Наприклад, розпізнавання натискання функціональної клавіші F1 реалізується таким чином:

```
switch (e.KeyCode)
{
    case Keys.F1: label1.Text = "натиснена клавіша F1";
```

```
                    break;
                default: label1.Text = "Не натиснута  функціональна
клавіша";
                    break;
            }
```

У мову програмування C# входить достатньо багато класів, які можна використати для роботи з файлами. До таких класів можна зарахувати, наприклад, *DirectoryInfo, FileInfo, File, StreamWriter, StreamReader.* Для того, щоб під час створення проєкту були доступні методи перелічених вище класів, потрібно використати простір імен *System.IO.*

Приклад 3. Розробити програму створення, зчитування та знищення текстового файлу з вибраної папки на диску.

Вікно програми зображене на рис. 3. При реалізації цього прикладу демонструється використання діалогових елементів керування *saveFileDialog1* та *openFileDialog1* для роботи з текстовими файлами. Підпрограма створення текстового файлу з даних, записаних у багаторядкове текстове поле *textBox1*, має такий вигляд:

```
    private void button1_Click(object sender, EventArgs e)
      {
        if (saveFileDialog1.ShowDialog() == DialogResult.OK)
          {
          StreamWriter file1 = new
StreamWriter(saveFileDialog1.FileName);
            file1.Write(textBox1.Text);
            file1.Close();
          }
      }
```

Аналогічно реалізується підпрограма зчитування текстового файлу з вибраної папки в багаторядкове текстове поле *textBox1*, а підпрограма знищення файлу має такий вигляд:

```
        private void button3_Click(object sender, EventArgs e)
        {
            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                FileInfo file = new FileInfo(openFileDialog1.FileName);
                if (file.Exists == true)
                    file.Delete();
            }
        }
```

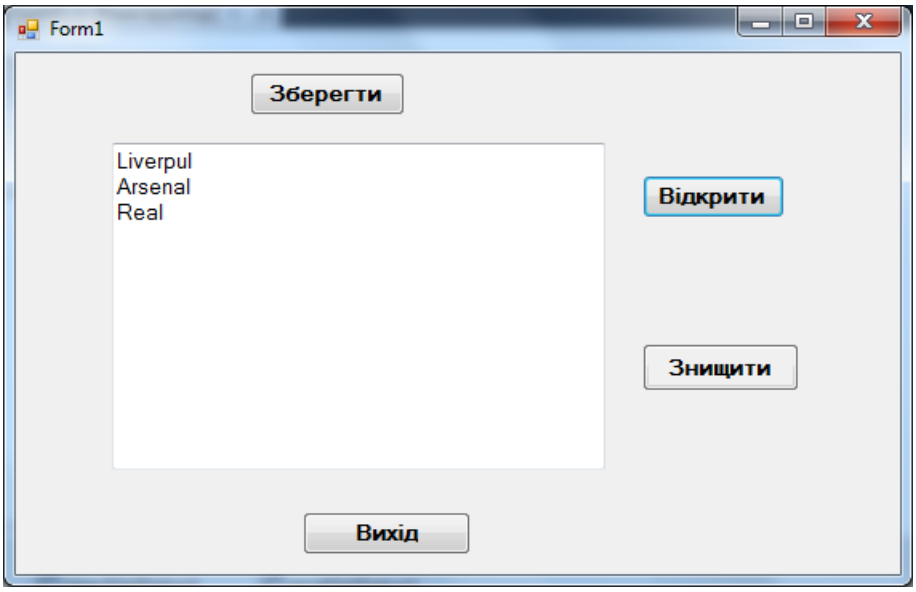

Рис. 3. Вікно програми використання діалогових елементів керування для роботи з текстовими файлами

У прикладі 4 реалізована обробка текстового файлу, яка полягає у створенні, збереженні та виводу в багаторядкове текстове поле *textBox1* рядків створеного файлу у зворотному порядку. Вікно програми зображене на рис. 4, а текст підпрограми *button1_Click* наведений нижче.

```
private void button1_Click(object sender, EventArgs e)
 {
   string[] text = File.ReadAllLines(@"D:\spycok.txt",
   Encoding.GetEncoding("Windows-1251"));
     int n = text.Length;
     int m = n - 1;
     string[] text1 = new string[n];
     for (int i = 0; i <= m; i++)
     {
         text1[i] = text[m - i];
     }
     File.WriteAllLines(@"D:\spycok1.txt", text1,
        Encoding.GetEncoding("Windows-1251"));
     textBox1.Text=File.ReadAllText(@"D:\spycok1.txt",
        Encoding.GetEncoding("Windows-1251"));
        }
```

Параметр *Windows-1251* у методі *GetEncoding* використовується для відображення кирилиці в багаторядковому текстовому полі.

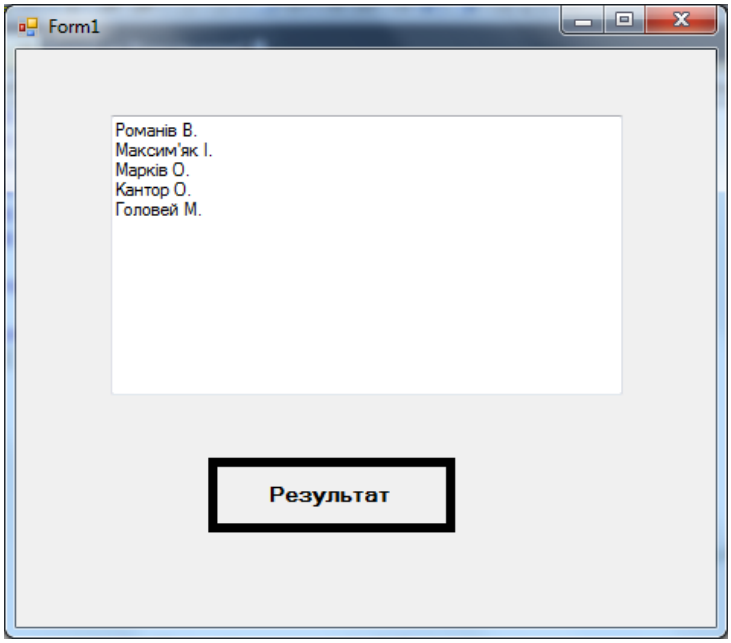

Рис. 4. Вікно програми обробки текстових файлів

Велика увага на лекційних заняттях приділяється створенню власних класів. Мовою програмування C# можна створювати статичні та динамічні класи. Також, використовуючи властивості об'єктно-орієнтованого програмування, а саме: наслідування та поліморфізм, можна будувати ієрархію класів, починаючи з деякого батьківського класу. Приклади 5, 6 та 7 демонструють принципи створення власних класів засобами мови програмування C#.

Приклад 5 демонструє створення статичного класу *Aryf*, який містить методи *Sum*, *Dob*, *Rizn* та *Dil*, що реалізують арифметичні операції додавання, множення, віднімання та ділення двох дійсних чисел відповідно.

```
 static class Aryf
{
            public  static double Sum(double a,double b)
```

```
        {
            return a+b;
        }
        public static double Dob(double a, double b)
        {
            return a* b;
        }
        public static double Rizn(double a, double b)
        {
            return a - b;
        }
        public static double Dil(double a, double b)
        {
            return a/b;
        }
    }
```

Статичний клас не містить конструктора, тому звертання до методів класу здійснюється, наприклад, таким чином:

```
label3.Text  = "Сума="+Aryf.Sum(a, b).ToString();
```

Приклад 6 демонструє створення динамічного класу *Complex1*, який містить конструктор та методи *sum*, *mplus*, що реалізують операції додавання, множення двох комплексних чисел.

```
  class Complex1
    {
        public  double Re, Im;

        public Complex1(double x, double y)
        {
            Re = x;
            Im = y;
        }
        public Complex1 sum(Complex1 c)
        {
            return new Complex1(Re+c.Re,Im+c.Im);
        }

        public Complex1 mplus(Complex1 c)
        {
            return new Complex1(Re * c.Re - Im * c.Im, Re *
c.Im+Im*c.Re);
        }
    }
```

А приклад 7 демонструє механізм використання основних властивостей ООП: інкапсуляції, наслідування та поліморфізму при створенні класів. У цьому прикладі створюються дочірні класи *Student* та *Teacher* на основі батьківського класу *Person*.

Батьківський клас Person

```
  class Person
    {
        public string name;
        public string adress;
        public Person(string newName, string newAdress)
        {
            name = newName;
            adress = newAdress;
        }
        public string Name
```

```
        {
            get { return name; }
        }
        public string Adress
        {
            get { return adress; }
            set { adress = value; }
        }
        public virtual string GetInfo()
        {
            return name + " " + adress;
        }
    }
```

Дочірній клас Student

```
class Student:Person
    {
         public int gr;
              public Student(string newName, string newAdress,int newgr)
                  :base(newName,newAdress)
              {
                 gr=newgr;
              }
                public override string GetInfo()
                {
                    return name +" "+ adress+" "+"ІН-"+gr.ToString()+"Б";
                }
    }
```

Дочірній клас Teacher

```
    class Teacher:Person
    {
         public string work;
         public Teacher(string newName, string newAdress,  string
newwork)
                  :base(newName,newAdress)
              {
                 work=newwork;
              }
                public override string GetInfo()
                {
                    return name +" "+ adress+" "+work;
                }
}
```

Механізм інкапсуляції продемонстровано в описі батьківського класу *Person* шляхом використання властивостей *Name* та *Adress.* Для властивості *Adress* визначені методи *get* і *set*, а властивість *Name* має тільки метод *get*, що забезпечує неможливість несанкціонованої зміни значення змінної *name.*

Механізм наслідування продемонстровано шляхом використання конструкторів *Teacher* та *Student* у дочірних класах, а механізм поліморфізму – шляхом реалізації віртуальних методів з однаковим іменем *GetInfo,* які в кожному класі мають інший вигляд.

Крім того, велика увага на лекційних заняттях приділяється вивченню властивостей елементів керування за допомогою демонстрації реалізації типових прикладів. Наприклад, під час проєктування дизайну вікна програми авторизації користувача (рис. 5) вивчаються такі властивості, як *Dock* (розміщення елемента керування на формі), *Flat* (установлення типу елемента керування), *FlatStyle, FlatApptearance* (зміна стилю елемента керування),

установлення кольору *BackColor*, *MinimixeBox* та *MaximizeBox*, які визначені для вікна та властивості *PasswordChar*, яка використовується для створення пароля.

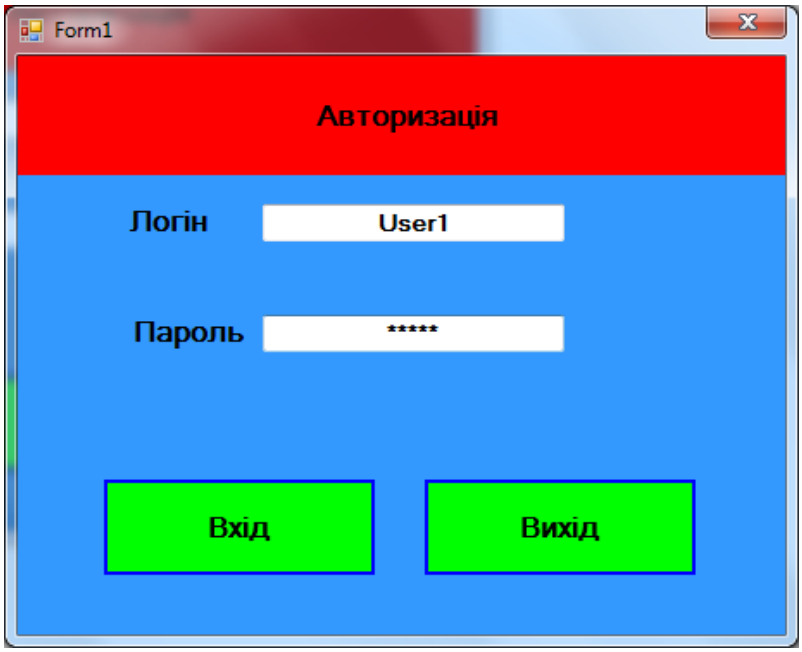

Рис. 5. Вікно програми авторизації.

Опрацьований лекційний матеріал студенти закріплюють шляхом виконання лабораторних робіт та освоєння матеріалу, який винесений на самостійне опрацювання.

Основним завданням лабораторних робіт є формування в майбутніх учителів інформатики практичних навичок програмування засобами Visual C#. Кожна лабораторна робота містить теоретичні відомості, перелік індивідуальних завдань для кожного студента та супроводжується списком запитань для самоперевірки. Завдання для кожного студента дібрані таким чином, щоб вони були однакового рівня складності. Теми розташовані в такому порядку, щоб кожна наступна не тільки надавала нові знання, але й закріплювала і розширювала знання з попередньої теми. Лабораторні роботи завершуються створенням багатовіконного проєкту на тему «Розробка освітньої системи вивчення методів наближеного розв'язування математичних задач».

Вибір такої теми базується на використанні міжпредметних зв'язків, тому що паралельно з освітньою дисципліною «Об'єктно-орієнтоване програмування» студенти вивчають дисципліну «Методи обчислень».

Освітня система повинна складатися з таких вікон: ***Авторизація, Титульна сторінка, Головне меню, Вивід освітнього матеріалу, Реалізація чисельного методу*** *та* ***Контроль знань.***

Наприклад, для реалізації контролю знань потрібно створити файл тесту, у якому містяться контрольні запитання та варіанти відповідей, і програму виводу та обробки відповідей студентів на тести.

Під час створення багатовіконного проєкту закріплюються знання студентів, які отримані в процесі виконання попередніх лабораторних робіт, а також реалізується принцип використання міжпредметних зв'язків, що сприяє підвищенню прикладної, практичної та науково-теоретичної підготовки студентів.

Висновки. У статті розглянута методика вивчення основ та принципів об'єктно-орієнтованого програмування шляхом демонстрування на лекціях типових прикладів та закріплення вивченого матеріалу на лабораторних заняттях. Приклади дібрані таким чином, щоб можна було продемонструвати особливості введення та виведення даних у системі візуального програмування Visual C#, програмування подій, обробку файлів, реалізації механізмів інкапсуляції, наслідування та поліморфізму. Одночасно з демонстрацією

реалізації типових прикладів вивчаються потрібні властивості елементів керування, які використовуються для надання потрібного дизайну вікна програми. Шляхом виконання лабораторних робіт студенти закріплюють знання, отримані на лекціях, та набувають практичних навичок розробки програм засобами Visual C#. Лабораторні роботи завершуються створенням багатовіконного проєкту, який представляє собою комп'ютерну освітню систему з вивчення методів обчислень. Створення такого проєкту дає можливість закріпити набуті студентами теоретичні знання та практичні навички з розробки програм у середовищі візуального програмування Visual C# та демонструє використання в освітньому процесі міжпредметних зв'язків. Тому що педагогічна практика свідчить, що міжпредметність відіграє важливу роль освітньому процесі педагогічного університету, зокрема, сприяє підвищенню прикладної, практичної і науково-теоретичної підготовки студентів, а ООП надає додаткові можливості для тісної інтеграції з багатьма освітніми дисциплінами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Брила, А. Ю., Антосяк, П. П., Глебена, М. І., Чупов, С. В. & Семйон, І. В. (2014). Основи програмування у C#. Методичні вказівки до лабораторних робіт для студентів І-го курсу математичного факультету спеціальності «Прикладна математика». Ужгород: Видавничий відділ Ужгородського національного університету. Відновлено з https://www.uzhnu.edu.ua/uk/infocentre/get/5868
2. Дорошенко, М. В. & Драб, Т. І. (2016) Комп'ютерна система вивчення та реалізації наближених методів алгебри та аналізу з використанням інтегрованого середовища Word та Matlab. *Вісник Національного університету* «ХПІ». 49(1221), 36–42.
3. Лазурчак, Л. В. & Вдовичин, Т. Я. (2017). Інформатика. Програмування мовою C++: методичні вказівки до виконання лабораторних робіт. Дрогобич: Видавничий відділ Дрогобицького державного педагогічного університету імені Івана Франка.
4. Настенко, Д. В. & Нестерко, А. Б. (2016). Об'єктно-орієнтоване програмування. Частина 1-2. Основи об'єктно-орієнтованого програмування на мові C#. Навчальний посібник. Київ:НТУУ«КПІ». Відновлено з https://ela.kpi.ua/bitstream/123456789/16671
5. Нейгел, К., Ивьен, Б. & Глинн, Дж. (2006). C# 2005 для профессионалов. Москва: Вильямс.
6. Смоленцев, Н. К. (2009). Matlab. Программирование на Visual C#, JBuilder: Учебный курс. Москва: Диалектика.
7. Троелсен, Э. (2007). C# и платформа .Net . Санкт-Петербург :СПб. Питер.
8. Шевчук, П. Г. (2016). Від Pascal до C#. Комп'ютер у закладах загальної середньої освіти та сім'ї. *Науково-методичний журнал*. 5, 40-45. Відновлено з http://nbuv.gov.ua/UJRN/komp_2011_5_12.
9. Шевчук, П. Г. (2012). Навчання програмування в класах технологічного профілю загальноосвітніх освітніх закладів на основі використання мови C#. Методичні рекомендації для учитеів інформатики. Київ: Інститут інформаційних технологій і засобів навчання НАПН України. Відновлено з http://lib.iitta.gov.ua/896/4/Metodrekom_Shevchuk_.pdf
10. Шилдт, Г. (2011). Объектно-ориентированное программирование на C#. Полное руководство. Москва : Вильямс.
11. Уотсон, К, Нейгел, К, Педерсен, Я., …, Скиннер, М. (2011). Visual C#. Полный курс. Москва: Диалектика.

REFERENCES (TRANSLATED AND TRANSLITERATED)

1. Brila, A. Yu., Antosyak, P. P, Glebena, M. I., Chupov, S. V. & Semyon, I. V. (2014). Basics of programming in C #. Methodical instructions for laboratory works for first-year students of the

Faculty of Mathematics, specialty "Applied Mathematics". Uzhhorod: Publishing Department of Uzhhorod National University. Updated from https://www.uzhnu.edu.ua/uk/infocentre/get/5868

2. Doroshenko, M. V. & Drab, T. I. (2016) Computer system for studying and implementing approximate methods of algebra and analysis using the integrated environment of Word and Matlab. *Bulletin of the National University "KhPI"*. 49 (1221), 36–42.
3. Lazurchak, L.V. & Vdovichin, T. Ya. (2017). Informatics. C ++ programming: guidelines for laboratory work. Drohobych: Publishing Department of Drohobych Ivan Franko State Pedagogical University.
4. Nastenko, D.V. & Nesterko, A.B. (2016). Object-oriented programming. Part 1-2. Basics of object-oriented programming in C #. Tutorial. Kyiv: NTUU "KPI". Updated from https://ela.kpi.ua/bitstream/123456789/16671
5. Neigel, K., Ivien, B. & Glynn, J. (2006). C # 2005 for professionals. Moscow: Williams.
6. Smolentsev, N. K. (2009). Matlab Visual C # Programming, JBuilder: Tutorial. Moscow: Dialectics.
7. Troelsen, E. (2007). C # and .Net platform. St. Petersburg: St. Petersburg Peter.
8. Shevchuk, P. G. (2016). From Pascal to C #. Computer at school and family. *Scientific and methodical journal*. 5, 40-45. Updated from http://nbuv.gov.ua/UJRN/komp_2011_5_12.
9. Shevchuk, P. G. (2012). Learning programming in technological profile classes of secondary schools based on the use of C # language. Methodical recommendations for computer science teachers. Kyiv: Institute of Information Technologies and Teaching Aids of the National Academy of Pedagogical Sciences of Ukraine. Updated from http://lib.iitta.gov.ua/896/4/Metodrekom_Shevchuk_.pdf
10. Schildt, G. (2011). Object Oriented Programming in C #. Complete guide. Moscow: Williams.
11. Watson, K., Neigel, K., Pedersen, J.,…, Skinner, M. (2011). Visual C #. Full course. Moscow: Dialectics.

Стаття надійшла до редакції 20.05.2020.
The article was received 20 May 2020.

Mykola Doroshenko
Drohobych Ivan Franko State Pedagogical University, Drohobych, Ukraine

LEARNING THE FUNDAMENTALS OF OBJECT-ORIENTED PROGRAMMING WITH VISUAL C #

The article provides guidelines for studying the discipline "Object-Oriented Programming" for the training of specialists of the first (bachelor's) level of higher education in the field of knowledge 01 "Education" specialty 014 "Secondary education (computer science)" educational program (computer science, mathematics). This discipline plays a particularly important role in the training of computer science teachers, because the field of computer program development using the latest technologies, which includes object-oriented programming, is an important component of scientific and technological progress.

Students are invited to consider the basic features of object-oriented programming and the principles of a visual approach to software design in an integrated environment Visual Studio.Net using Visual C#. The choice of Visual Studio.Net is due to the fact that the version of Visual Studio.Net Community is a free full-featured version, which differs from the professional only slightly limited features.

The article considers the method of studying the basics and principles of object-oriented programming by demonstrating typical examples in lectures and consolidating the studied material in laboratory work. The examples are selected in such a way that it is possible to demonstrate the features of data input and output in the Visual C# visual programming environment, event programming, file processing, implementation of encapsulation, inheritance and polymorphism mechanisms. Simultaneously with the demonstration of the implementation of typical examples, the required properties of the control elements used in the project implementation are studied.

By performing laboratory work, students consolidate the knowledge gained in lectures and acquire practical skills of project development using Visual C#. The laboratory work ends with the creation of a multi-window project, which represents a computer training system for studying computational methods. The creation of such a project allows students to consolidate the theoretical knowledge and practical skills of program development and demonstrates the use of interdisciplinary links in the educational process.

Key words: computer science teacher, object-oriented programming, Visual C# visual programming environment.