ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКРОНІКИ

Кафедра ЕОМ

Кваліфікаційна робота магістра

Метод організації туманих обчислень у динамічній оверлейній обчислювальній мережі на базі полінгових мереж

Виконав:
студент гр. СПм-22-6
Гунько Михайло

Керівник:
доц. каф. ЕОМ
Ткачов В.М.

1

Актуальність роботи

01. Тумані обчислення

02. Використання оверлейних мереж

03. Полінгові мережі

2

Мета роботи

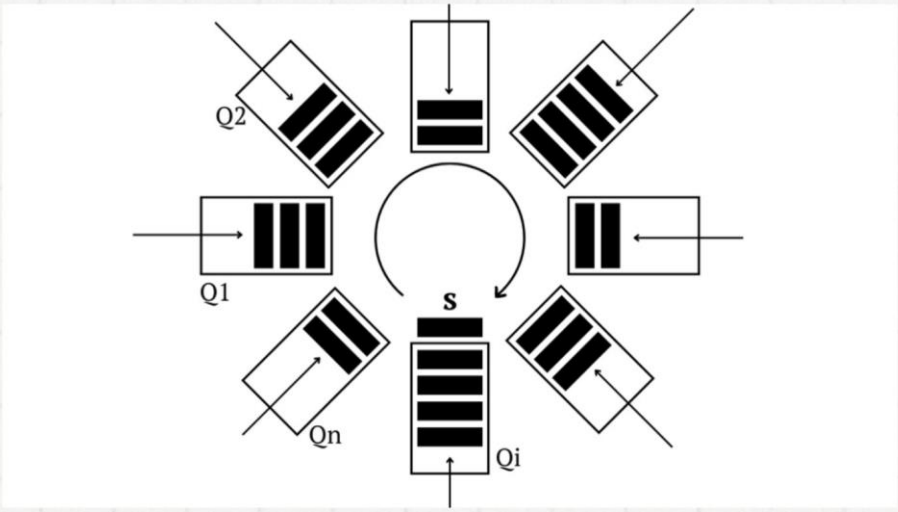
01. Розробка методу туманних обчислень на базі полінгових мереж для покращення обробки даних та зменшення часу обчислень у розподілених системах.

02. Тестування запропонованого методу для оцінки ефективності, масштабованості, гнучкості та стійкості обчислювальних процесів у різних умовах роботи системи.

03. Підвищення безпеки та ефективності управління ресурсами в обчислювальних системах через динамічні оверлейні мережі, забезпечуючи захищений канал передачі даних.

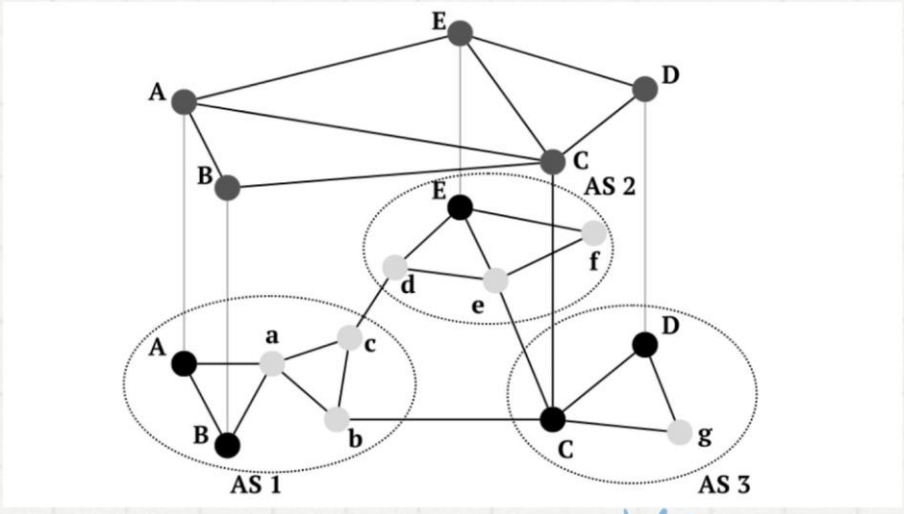
3

Полінгова мережа



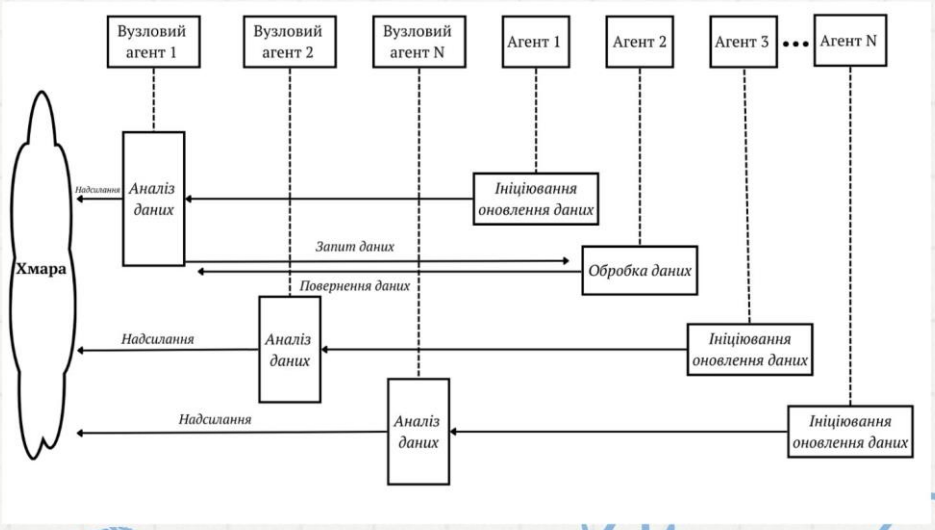
4

Оверлейна мережа



5

Архітектура рішення



6

Вираховування кількості вузлових агентів

$$N_v = N_a * \lambda$$

Визначення показника максимальної затримки

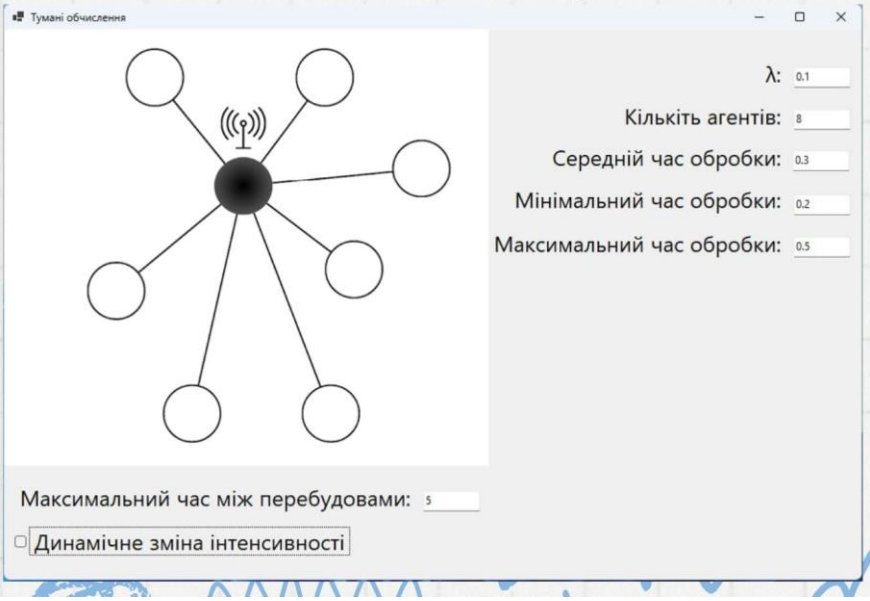
$$\varphi = (l_{aMax} + 1) * \tau_l$$

Визначення затримки на одному агенті

$$l_{ai} = \tau_{l_i} * (l_i + 1)$$

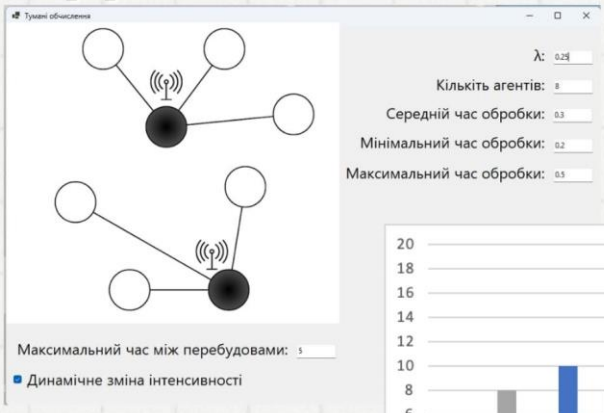
7

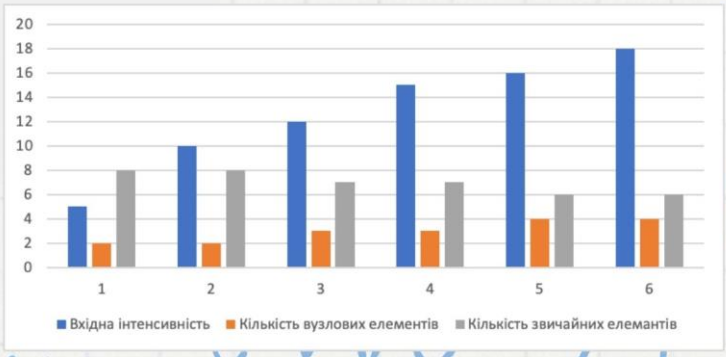
Модельний експеримент(1)



8

Модельний експеримент(2)





9

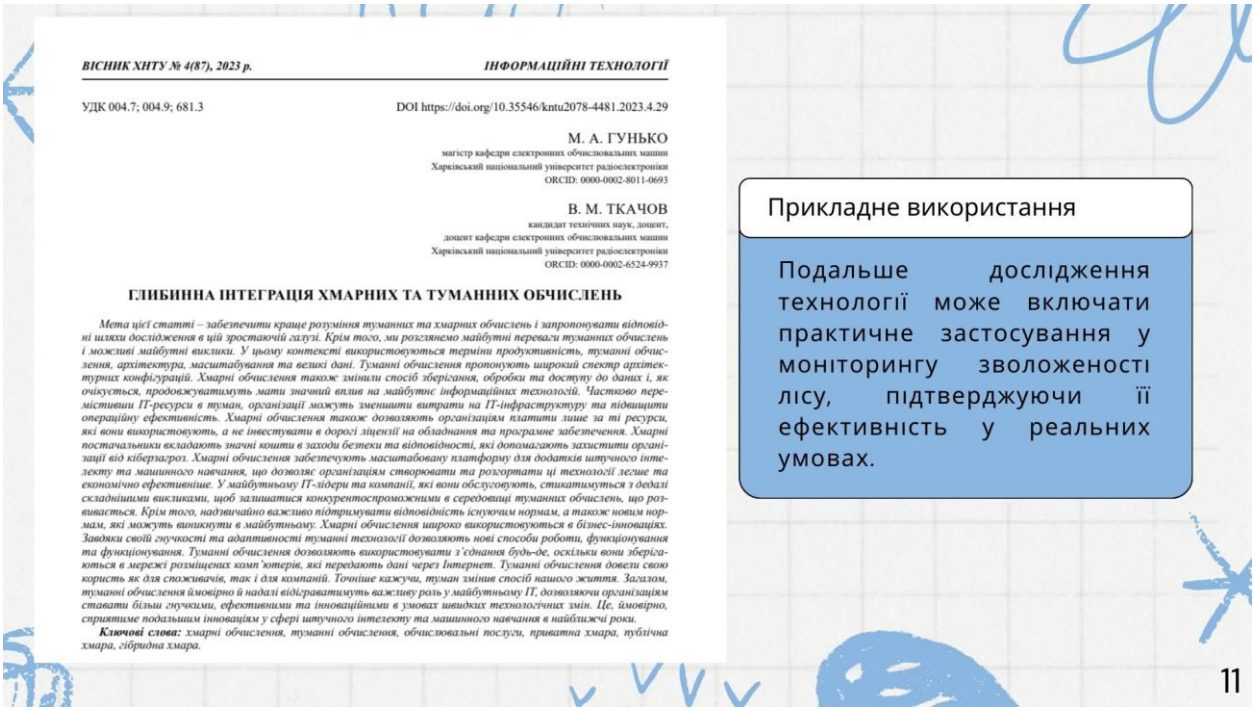
Висновки

01. Розроблений метод туманних обчислень на базі полінгових мереж

02. Проведене експериментальне моделювання

03. Запропоноване прикладне використання

10



ВІСНИК ХНТУ № 4(87), 2023 р. ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

УДК 004.7; 004.9; 681.3 DOI https://doi.org/10.35546/kntu2078-4481.2023.4.29

М. А. ГУНЬКО
магістр кафедри електронних обчислювальних машин
Харківський національний університет радіоелектроніки
ORCID: 0000-0002-8011-0693

В. М. ТКАЧОВ
кандидат технічних наук, доцент,
доцент кафедри електронних обчислювальних машин
Харківський національний університет радіоелектроніки
ORCID: 0000-0002-6524-9937

ГЛИБИННА ІНТЕГРАЦІЯ ХМАРНИХ ТА ТУМАННИХ ОБЧИСЛЕНЬ

Мета цієї статті – забезпечити краще розуміння туманних та хмарних обчислень і запропонувати відповідні шляхи дослідження в цій зростаючій галузі. Крім того, ми розглянемо майбутні переваги туманних обчислень і можливі майбутні виклики. У цьому контексті використовуються терміни продуктивність, туманні обчислення, архітектура, масштабування та великі дані. Туманні обчислення пропонують широкий спектр архітектурних конфігурацій. Хмарні обчислення також змінили спосіб зберігання, обробки та доступу до даних і, як очікується, продовжуватимуть мати значний вплив на майбутнє інформаційних технологій. Частково перемістивши IT-ресурси в туман, організації можуть зменшити витрати на IT-інфраструктуру та підвищити операційну ефективність. Хмарні обчислення також дозволяють організаціям платити лише за ті ресурси, які вони використовують, а не інвестувати в дорогі ліцензії на обладнання та програмне забезпечення. Хмарні постачальники вкладають значні кошти в заходи безпеки та відповідності, які допомагають захистити організації від кіберзагроз. Хмарні обчислення забезпечують масштабовану платформу для додатків штучного інтелекту та машинного навчання, що дозволяє організаціям створювати та розгортати ці технології легше та економічно ефективніше. У майбутньому IT-лідери та компанії, які вони обслуговують, стикатимуться з дедалі складнішими викликами, щоб залишатися конкурентоспроможними в середовищі туманних обчислень, що розвивається. Крім того, надзвичайно важливо підтримувати відповідність існуючим нормам, а також новим нормам, які можуть виникнути в майбутньому. Хмарні обчислення широко використовуються в бізнес-інноваціях. Завдяки своїй гнучкості та адаптивності туманні технології дозволяють нові способи роботи, функціонування та функціонування. Туманні обчислення дозволяють використовувати з'єднання будь-де, оскільки вони зберігаються в мережі розміщених комп'ютерів, які передають дані через Інтернет. Туманні обчислення довели свою користь як для споживачів, так і для компаній. Точніше кажучи, туман змінив спосіб нашого життя. Загалом, туманні обчислення ймовірно й надалі відіграватимуть важливу роль у майбутньому IT, дозволяючи організаціям ставати більш гнучкими, ефективними та інноваційними в умовах швидких технологічних змін. Це, ймовірно, сприятиме подальшим інноваціям у сфері штучного інтелекту та машинного навчання в найближчі роки.

Ключові слова: *хмарні обчислення, туманні обчислення, обчислювальні послуги, приватна хмара, публічна хмара, гібридна хмара.*

Прикладне використання

Подальше дослідження технології може включати практичне застосування у моніторингу зволоженості лісу, підтверджуючи її ефективність у реальних умовах.

ДОДАТОК Б

Фрагмент програмного коду

```
public partial class Form1 : Form
    {

        private BindingSource BindingSource1 = new
BindingSource();
        private BindingSource BindingSource2 = new
BindingSource();



        List<Person> list;
        List<Subject> list2;

        public Form1()
        {
            list2 = new List<Subject>();
            list = new List<Person>();
            BindingSource1.DataSource = list;
            BindingSource2.DataSource = list2;
            InitializeComponent();
            teachers.DataSource = BindingSource1;
            subjectGrid.DataSource = BindingSource2;
        }

        private void add_Click(object sender, EventArgs e)
        {

            string teacher = Teacher.Text;

            if (string.IsNullOrEmpty(teacher))
            {
                MessageBox.Show("Ошибка. Правильно заполните
поля!");
            }
            else
            {
                list.Add(new Person(teacher));

            }
            BindingSource1.ResetBindings(true);
        }

        private void TeacherOn_Click(object sender, EventArgs e)
        {

            string s = textSubject.Text;
```

```
            if (string.IsNullOrEmpty(s))
            {
                return;
            }

            var selectedUsers = (from user in list where
user.name == s select user.subject).ToList();
            string result = "";
            foreach (string user in selectedUsers)
            {
                result += user + "\n";
            }

            if (result.Equals(""))
            {
                MessageBox.Show("Предметов не найдено!");
            }
            else
            {
                MessageBox.Show(result);
            }

        }

        private void SubjectOn_Click(object sender, EventArgs e)
        {
            string s = textTeacher.Text;
            if (string.IsNullOrEmpty(s))
            {
                return;
            }

            var selectedUsers = (from user in list where
user.subject == s select user.name).ToList();
            string result = "";
            foreach (string user in selectedUsers)
            {
                result += user + "\n";
            }

            if (result.Equals(""))
            {
                MessageBox.Show("Преподавателей не найдено!");
            }
            else
            {
                MessageBox.Show(result);
            }
        }

        private void Save_Click(object sender, EventArgs e)
        {
            XmlSerializer formatter = new
```

```
XmlSerializer(typeof(List<Person>));
            using (FileStream fs = new
FileStream(@"D:\lab5.xml", FileMode.Truncate))
            {
                formatter.Serialize(fs, list);
            }
            XmlSerializer formatter2 = new
XmlSerializer(typeof(List<Subject>));
            using (FileStream fs = new
FileStream(@"D:\lab51.xml", FileMode.Truncate))
            {
                formatter2.Serialize(fs, list2);
            }
        }

        private void Remove_Click(object sender, EventArgs e)
        {
            try
            {
                int index = Convert.ToInt32(textIndex.Text) - 1;
                string teacher =
teachers.Rows[index].Cells["name"].Value.ToString();
                string subject =
teachers.Rows[index].Cells["subject"].Value.ToString();

               var indexOf = list.FindIndex(x => x.name ==
teacher && x.subject == subject);
                list.RemoveAt(indexOf);
                BindingSource1.ResetBindings(true);
             }

        private void Start_Click(object sender, EventArgs e)
        {
            XmlSerializer formatter = new
XmlSerializer(typeof(List<Person>));
            using (FileStream fs = new
FileStream(@"D:\lab5.xml", FileMode.Open))
            {
                list = (List<Person>)formatter.Deserialize(fs);
            }
            BindingSource1.DataSource = list;
            BindingSource1.ResetBindings(true);

            XmlSerializer formatter2 = new
XmlSerializer(typeof(List<Subject>));
            using (FileStream fs = new
FileStream(@"D:\lab51.xml", FileMode.Open))
            {
                list2 =
(List<Subject>)formatter2.Deserialize(fs);
            }
            BindingSource2.DataSource = list2;
            BindingSource2.ResetBindings(true);
```

```
        }

        private void sortUp_Click(object sender, EventArgs
e)//done
        {
            var newList = list.OrderBy(p => p.name).ToList();
//.Sort((Person x, Person y) => x.name.CompareTo(y.name));
            list = newList;
            BindingSource1.DataSource = list;
            BindingSource1.ResetBindings(false);

        }

        private void sortDown_Click(object sender, EventArgs
e)//done
        {
            var newList = list.OrderByDescending(p =>
p.name).ToList(); //.Sort((Person x, Person y) =>
x.name.CompareTo(y.name));
            list = newList;
            BindingSource1.DataSource = list;
            BindingSource1.ResetBindings(false);
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }

        private void addSubjectR_Click(object sender, EventArgs
e)
        {
            bool allRight = true;
            string subject = textSubj.Text;
            foreach (Subject sb in list2)
            {
                if (sb.subject.Equals(subject))
                {
                    MessageBox.Show("Помилка.)
                    allRight = false;
                }
            }
            if (string.IsNullOrEmpty(subject))
            {
                MessageBox.Show("Помилка!");
            }
            else if(allRight)
            {

                list2.Add(new Subject(subject));

            }
            BindingSource2.ResetBindings(true);
```

```
        }

        private void teachers_DoubleClick(object sender,
EventArgs e)
        {
            list[teachers.CurrentCell.RowIndex].subject =
list2[subjectGrid.CurrentCell.RowIndex].subject;
            BindingSource1.ResetBindings(true);
        }
    }

[Serializable]
    public class Person
    {

        public string name { get; set; }
        public string subject { get; set; }

        public Person()
        {
        }

        public Person(string name)
        {
            this.name = name;
        }

        public void setSubject(string subject)
        {
            this.subject = subject;
        }
    }

[Serializable]
    public class Subject
    {

        public string subject { get; set; }

        public Subject(string subject)
        {
            this.subject = subject;
        }

        public Subject()
        {
        }
    }
public class University
    {

        public string name {  get; set; }
        public int faculty { get; set; }
```

```
        public int laboratoriesNumber { get; set; }
        public int lecturesNumber { get; set; }
        public int numberOfStudent { get; set; }
        public List<string> students { get; set; }
        public int staffNumber { get; set; }
        public List<string> staff { get; set; }
        public int laborantNumbers { get; set; }

        //инициализирующий
        public University(string name, int faculty, int
laboratoriesNumber, int lecturesNumber, List<string> students,
List<string> staff)
        {
            this.name = name;
            this.faculty = faculty;
            this.laboratoriesNumber = laboratoriesNumber;
            this.lecturesNumber = lecturesNumber;
            numberOfStudent = students.Count;
            this.students = students;
            this.students.Sort();
            staffNumber = staff.Count;
            this.staff = staff;
            this.staff.Sort();
            laborantNumbers = checkLaborantNumbers();
        }
        //копирующий
        public University(University university)
        {
            name = university.name;
            faculty = university.faculty;
            laboratoriesNumber = university.laboratoriesNumber;
            lecturesNumber = university.lecturesNumber;
            numberOfStudent = university.numberOfStudent;
            students = university.students;
            staffNumber = university.staffNumber;
            int temp = (university.lecturesNumber +
university.laboratoriesNumber) / 2;
            if (temp % 2 == 1)
            {
                laborantNumbers = temp + 1;
            }
            else
            {
                laborantNumbers = temp;
            }
        }

        public void deleteStud(String nameStud)
        {
            foreach (string student in students)
            {
                if (student.Equals(nameStud))
                {
```

```
                students.Remove(student);
                numberOfStudent--;
                break;
            }
        }
    }

    public void addStud(String name)
    {
        students.Add(name);
        numberOfStudent--;
    }

    public void addLec()
    {
        lecturesNumber++;
    }

    public void addLab()
    {
        laboratoriesNumber++;
    }
    public void addStuff(string nameStaff)
    {
        staff.Add(nameStaff);
        staffNumber++;
    }

    public void deleteStaff(string nameStaff)
    {
        foreach (string st in staff)
        {
            if (st.Equals(nameStaff))
            {
                staff.Remove(nameStaff);
                staffNumber--;
                break;
            }
        }
    }

    public void updateLaborantNumbers()
    {
        laborantNumbers = checkLaborantNumbers();
    }

    public int checkLaborantNumbers()
    {
        int temp = (lecturesNumber + laboratoriesNumber) /
2;
        if ((lecturesNumber + laboratoriesNumber) % 2 == 1)
        {
            return (temp + 1);
```

```
            }
            else
            {
                return temp;
            }

        }
        public int indexLec()
        {
            return 0;
        }
        public int indexLab()
        {
            return 0;
        }


        public static University operator +(University obj,
University obj2)
        {
            List<string> stud = obj.students;
            foreach(string str in obj2.students.ToArray())
            {
                stud.Add(str);
            }

            List<string> staff = obj.staff;
            foreach (string str in obj2.staff.ToArray())
            {
                staff.Add(str);
            }
            University result = new University(obj.name + "+" +
obj2.name, obj.faculty + obj2.faculty, obj.laboratoriesNumber +
obj2.laboratoriesNumber, obj.lecturesNumber +
obj2.lecturesNumber, stud, staff);
            return result;
        }

        public int this[int index]
        {
            get
            {
                if(index == 1)
                {
                    return laboratoriesNumber;
                }
                else if(index == 2)
                {
                    return lecturesNumber;
                }
                else
                {
                    return -1;
```

```
                }
            }
            set
            {
                if (index == 1)
                {
                   laboratoriesNumber = value;
                }
                else if (index == 2)
                {
                    lecturesNumber = value;
                }
                else
                {

                }
            }
        }
    }

public partial class Form1 : Form
    {

        static public University selectedUniversity;
        Logics logics;
        public Form1()
        {
            InitializeComponent();
             logics = new Logics();
            logics.inizialaze();
            iniz();

        }

        private void iniz()
        {
            dropUniv.Items.Clear();
                 for (int i = 0; i < logics.universities.Count;
i++)
                            {

dropUniv.Items.Add(logics.universities[i].name);
                            }
            dropUniv.Text = logics.universities[0].name;
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }

        private void deleteUniv_Click(object sender, EventArgs
e)
```

```
        {
            if (dropUniv.SelectedIndex > -1)
            {

logics.universities.Remove(logics.universities[dropUniv.Selected
Index]);
                dropUniv.Items.Remove(dropUniv.SelectedItem);
                facultyAmount.Text = "0";
                lecAmount.Text = "0";
                labAmount.Text = "0";
                dropUniv.Text = logics.universities[0].name;
            }
            else
            {
                messageBox();
            }
        }

        private void dropUniv_SelectedIndexChanged(object
sender, EventArgs e)
        {
            update();
        }

        private void addLab_Click(object sender, EventArgs e)
        {
             selectedUniversity.laboratoriesNumber++;
            labAmount.Text =
selectedUniversity.laboratoriesNumber.ToString();
            selectedUniversity.updateLaborantNumbers();
            laborantAmount.Text =
selectedUniversity.laborantNumbers.ToString();
        }

        private void addLec_Click(object sender, EventArgs e)
        {
            selectedUniversity.lecturesNumber++;
            lecAmount.Text =
selectedUniversity.lecturesNumber.ToString();
            selectedUniversity.updateLaborantNumbers();
            laborantAmount.Text =
selectedUniversity.laborantNumbers.ToString();
        }

        private void addStud_Click(object sender, EventArgs e)
        {
            Form2 enterData = new Form2(true);
            enterData.ShowDialog();

            update();
            // laborantAmount.Text =  enterData.Name;
        }
```

```
        private void deleteStud_Click(object sender, EventArgs
e)
        {
            if (dropStud.SelectedIndex > -1)
            {

selectedUniversity.students.Remove(selectedUniversity.students[d
ropStud.SelectedIndex]);
                dropStud.Items.Remove(dropStud.SelectedItem);
            }
            else
            {
                messageBox();
            }

        }

        private void deleteTeach_Click(object sender, EventArgs
e)
        {
            if (dropTeach.SelectedIndex > -1)
            {

selectedUniversity.staff.Remove(selectedUniversity.staff[dropTea
ch.SelectedIndex]);
                dropTeach.Items.Remove(dropTeach.SelectedItem);
            }
            else
            {
                messageBox();
            }
        }
        private void messageBox()
        {
        }

        private void update()
        {
            dropStud.Items.Clear();
            dropTeach.Items.Clear();
            for (int i = 0; i < logics.universities.Count; i++)
            {
                if
((logics.universities[i].name).Equals(dropUniv.SelectedItem))
                {
                    selectedUniversity = logics.universities[i];
                }
            }
            facultyAmount.Text =
selectedUniversity.faculty.ToString();
            lecAmount.Text =
selectedUniversity.lecturesNumber.ToString();
            labAmount.Text =
```

```
selectedUniversity.laboratoriesNumber.ToString();
            laborantAmount.Text =
selectedUniversity.laborantNumbers.ToString();
            for (int i = 0; i <
selectedUniversity.students.Count; i++)
            {

dropStud.Items.Add(selectedUniversity.students[i]);
            }
            for (int i = 0; i < selectedUniversity.staff.Count;
i++)
            {

dropTeach.Items.Add(selectedUniversity.staff[i]);
            }
        }

        private void addTeach_Click(object sender, EventArgs e)
        {
            Form2 enterData = new Form2(false);
            enterData.ShowDialog();
            update();
        }

        private void toOne_Click(object sender, EventArgs e)
        {
            Random rnd = new Random();
            int rnd1 = rnd.Next(0, logics.universities.Count -
1);
            int rnd2 = rnd.Next(0, logics.universities.Count -
1);
            logics.universities[rnd1] +=
logics.universities[rnd2];

logics.universities.Remove(logics.universities[rnd2]);
            iniz();
            update();
        }

        private void indexRoom_Click(object sender, EventArgs e)
        {
            int lab = selectedUniversity[1];
            int lec = selectedUniversity[2];
        }




    }


    class BUniversityEqualityComparer :
IEqualityComparer<University>
```

```
    {

        bool IEqualityComparer<University>.Equals(University x,
University y)
        {
            bool equals = x.name.Equals(y.name) &&
x.laborantNumbers == y.laborantNumbers && x.lecturesNumber ==
y.lecturesNumber && x.laboratoriesNumber == y.laboratoriesNumber
                && x.faculty == y.faculty && x.numberOfStudent
== y.numberOfStudent && x.staffNumber == y.staffNumber &&
x.staff.Equals(y.staff) && x.students.Equals(y.students);
            if (x == null && y == null)
                return true;
            else if (x == null || y == null)
                return false;
            else if (equals)
                return true;
            else
                return false;
        }

        int IEqualityComparer<University>.GetHashCode(University
obj)
        {
            int hashCode = 2096631019;
            hashCode = hashCode * -1521134295 +
EqualityComparer<string>.Default.GetHashCode(obj.name);
            hashCode = hashCode * -1521134295 +
obj.faculty.GetHashCode();
            hashCode = hashCode * -1521134295 +
obj.laboratoriesNumber.GetHashCode();
            hashCode = hashCode * -1521134295 +
obj.lecturesNumber.GetHashCode();
            hashCode = hashCode * -1521134295 +
obj.numberOfStudent.GetHashCode();
            hashCode = hashCode * -1521134295 +
EqualityComparer<List<string>>.Default.GetHashCode(obj.students)
;
            hashCode = hashCode * -1521134295 +
obj.staffNumber.GetHashCode();
            hashCode = hashCode * -1521134295 +
EqualityComparer<List<string>>.Default.GetHashCode(obj.staff);
            hashCode = hashCode * -1521134295 +
obj.laborantNumbers.GetHashCode();
            return hashCode;
        }
    }

public partial class Form2 : Form
    {

        Boolean staff;
        public Form2(Boolean stud)
```

```
        {
            staff = !stud;
            InitializeComponent();
        }

        private void textBox1_TextChanged(object sender,
EventArgs e)
        {

        }

        private void addUser_Click(object sender, EventArgs e)
        {
            string text = textBox1.Text;
            if (checkString(text))
            {
                if (staff)
                {
                    Form1.selectedUniversity.staff.Add(text);
                }
                else
                {
                    Form1.selectedUniversity.students.Add(text);
                }

                this.Close();
            }
            else
            {
            MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
            }


        }

        private Boolean checkString(string A)
        {
            for (int i = 0; i < A.Length; i++)
            {
                if (A[i] >= 'A' && A[i] <= 'Я' || A[i] >= 'a' &&
A[i] <= 'я' || A[i].Equals(' ') )
                {
                    continue;
                }
                else
                {
                    return false;
                }
            }
            return true;
        }
    }
```