ЛАБОРАТОРНА РОБОТА №11

Тема: Динамічне розміщення даних.

Мета роботи

Навчитися використовувати динамічні структури мови C#.

Постановка завдання

Розробити проект з інтерфейсом користувача, використовуючи необхідні компоненти та оброблювач подій мовою C#.

Теоретичні відомості

1. Динамічне розміщення даних

Динамічний розподіл пам'яті широко використовується для економії обчислювальних ресурсів: ті об'єкти, які стають непотрібними, програма знищує, а звільнене місце використовує для нових об'єктів.

2. Змінення розміру масиву під час виконання програми

Для змінення розміру масиву використовують функцію Array.Resize:

```
char[] a = new char[6] { '0', '1', '2', '3', '4', '5' };
Array.Resize(ref a, 10);
a[6] = '6';
a[7] = '7';
a[8] = '8';
a[9] = '9';
string t = "";
foreach (char c in a) t += c;
// Тепер значення t == "0123456789"
```

3. List (список)

Список дозволяє пов'язано зберігати однотипні елементи, динамічно виділяючи пам'ять для наступних елементів. Це означає, що вам не потрібно задавати розмір списку при його ініціалізації на відміну від масивів. Доступ до елемента можливий за індексом, але зазвичай елементи списку обробляються послідовно.

```
List<byte> item = new List<byte>(new byte[] {
    5,9,4,8,2,6,5,7,1,9,5,4,3,0,1,5,4,9,2,3,0,5 });

item.RemoveAll(n => n >= 4 && n <= 7);
// Видалення усіх елементів, які відповідають заданій
умові
// Результат: 9, 8, 2, 1, 9, 3, 0, 1, 9, 2, 3, 0
```

```
item.Sort();
// Сортування елементів
// Результат: 0, 0, 1, 1, 2, 2, 3, 3, 8, 9, 9, 9

int i = item.IndexOf(3);
// Пошук першого елемента, рівного заданому

item.Reverse(i, item.Count - i);
// Змінення порядку елементів на протилежний для усіх
// елементів, починаючи з елемента 3
// Результат: 0, 0, 1, 1, 2, 2, 9, 9, 9, 8, 3, 3

item.Clear();
// Очищення списку від усіх елементів

item.Add(100);
// Результат: 100
```

4. Dictionary (словник)

Словник – це колекція для зберігання і подальшого пошуку однотипних елементів. У словнику елементи зберігаються у вигляді пари «ключ-значення». Це означає, що ви можете зберегти значення, і надалі використовуючи ключ швидко отримати значення зі словника. Тип ключа задається вручну і може бути будь-яким об'єктом.

```
// Структура для зберігання данних про місто
public struct City
{
    public string Name;
    public uint Population;
};

public Form1()
{
    InitializeComponent();
    Dictionary<string, City> LargestCities =
        new Dictionary<string, City>
    {
        { "KV", new City {
            Name="Київ", Population=2797553 } },
        { "KH", new City {
            Name="Харків", Population=1430885 } },
        { "DP", new City
```

```
            Name="Дніпропетровськ", Population=1032822 }
}
    };
    textBox1.Text = "У місті " +
        LargestCities["DP"].Name + " населення складає "
      + LargestCities["DP"].Population.ToString();
}
```