**Kolekcije u C#**

* Kolekcije sadrže skupove nekakvih objekata u kojima je moguće pristupiti svakom elementu pojedinačno.
* Elementima je moguće pristupiti upotrebom petlje foreach.
* Za razliku od nizova kolekcije ne moraju imati unapred definisan broj elemenata.

Mogu biti:

* **Generičke**

Generičke kolekcije su parametrizovane, što znači da je pri kreiranju objekta klase koja predstavlja generičku kolekciju obevezno navesti tip podataka koje će kolecija sadržati.

Tipovi generčkih kolecija se nalaze u **System.Collections.Generic** imenskom prostoru.

U generičke kolekcije spadaju:

* **List<>**
* **LinkedList<>**
* **SortedList<TKey, TValue>**
* **Dictionary<TKey, TValue>**
* **SortedDictionary<TKey, TValue>**
* **Queue<>**
* **Stack<>**
* **Negeneričke**

Negeneričke kolekcije sarže object tip podataka, što znači da se u njima mogu naći podaci različitih tipova.

Tipovi negenerčkih kolecija se nalaze u **System.Collections** imenskom prostoru.

U negeneričke kolekcije spadaju:

* **ArrayList**
* **Queue**
* **Stack**
* **Hashtable**
* **SortedList**

**Generičke kolekcije**

**List<>**

Predstavlja listu elemenata kojima je moguće pristupiti preko indeksa, pri čemu je obezbeđeno mnoštvo metoda za manipulaciju elementima.

**Kreiranje**

  List<Student> students = new List<Student>();

**Dodavanje elemenata**

students.Add(new Student { ID = 10, Name = "Manuel Radovanovic" });

**Umetanje elementa na poziciju sa zadatim indeksom**

List<int> intList = new List<int>(){ 10, 20, 30, 40 };

intList.Insert(1, 11);// inserts 11 at 1st index: after 10.

**Pretraga**

// da li postoji element sa zadatom vrednosu u listi

bool f = list.Contains("four");

// pretraga po uslovu, da li postoji element sa vrednošću većom od 10

bool exists = list.**Exists**(element => element > 10);

// pronalaženje prve vrednosti koja zadovoljava uslov

int result = list.**Find**(item => item > 20);

// pronalaženje proslednje vrednosti koja zadovoljava uslov

int result = list.**FindLast**(item => item > 20);

**Ispis**

  foreach (Student student in students)

  {

         Console.WriteLine(student);

 }

**Sortiranje**

  students.Sort();

**Brisanje**

students.RemoveAt(2); // uklanja element sa indeksom 2

students.Remove(2); // uklanja prvu 2-ku iz liste

students.RemoveAll(x => x ==2); // uklanje svih 2 iz liste

students.RemoveRange(1, 5); // brisanje dela liste

**Odredjivanje broja elemenata**

int br = students.Count;

**Suma, minimum, maksimum**

Console.WriteLine(mojaLista.Sum());

**Dobijanje podliste**

List<int> sublist = mojaLista.GetRange(1, 2);

**Negeneričke kolekcije**

**ArrayList**

ArrayList kolekcija omogućava da dodavanje i uklanjanje elemenat, uz automatsko dinamičko alociranje memorisjkog prostora (fleksibilniji niz).

Može da sadrži podatke različitih tipova podataka.

    class Program

    {

        static void Main(string[] args)

        {

            ArrayList Employee = new ArrayList();

            Employee.Add("Manuel");

            Employee.Add("Radovanovic");

            Employee.Add("Senior Programmer");

            Employee.Add("5/2/2016");

            Employee.Add(4500.00);

            WriteLine("Information about new employees:" + Environment.NewLine);

            Print(Employee);

            WriteLine(Environment.NewLine + "Press any key to continue...");

            ReadKey();

        }

        static void Print(ArrayList arraylist)

        {

            foreach (object item in arraylist)

            {

                DateTime start;

                Double salary;

                if (DateTime.TryParse(item.ToString(), out start))

                {

                    WriteLine(start.ToString("dddd dd MMMM yyyy"));

                }

                else if (Double.TryParse(item.ToString(), out salary))

                {

                    WriteLine("{0:C}", salary);

                }

                else

                {

                    WriteLine(item);

                }

            }

        }

DODATNO

**Generičke kolekcije**

**Dictionary<TKey, TValue>**

Kolekcija sadrži parove klju/vrednost, pri čemu se može raditi pretraga i po ključu i po vrednosti.

Ključ mora imati jedinstvene vrednosti.

Kreiranje

  Dictionary<int, string> greatStudents = new Dictionary<int, string>();

Dodavanje

greatStudents.Add(student.ID, student.Name);

**Ispis i pretraga**

Dictionary<int, string> greatStudents = new Dictionary<int, string>();

greatStudents.Add(45, "Pera");

greatStudents.Add(35, "Charles Junger");

int getKey = 0;

foreach (KeyValuePair<int,string> item in greatStudents)

{

if (item.Value == "Charles Junger") getKey = item.Key;

Console.WriteLine(getKey);

}

**Brisanje celog sadržaja rečnika**

greatStudents.Clear();

**Provera da li sadrži neku vrednost ključa**

if (greatStudents.ContainsKey(35)) Console.WriteLine(greatStudents[35]);

**SortedDictionary<TKey, TValue>**

Varijacija Dictionary kolekcije koja svoje parove čuva u redosledu uređenom prema ključu. Obezbeđuje brzu pretragu.

**LinkedList<>**

Elemente koje sadrži čuva u obliku povezane liste. Ukoliko će elementi liste često menjati pozicijiju u nizu, onda je ova klasa dobar izbor.

Kreiranje

            LinkedList<string> students = new LinkedList<string>();

Dodavanje novog podatka na kraj

            students.AddLast("Manuel Radovanovic");

Uklanjaje

            students.Remove("Ella Napolis");

Dobijanje reference na čvor u listi

            LinkedListNode<string> node = students.Find("Manuel Radovanovic");

Umetanje novog čvora zadavanjem podatka koji treba da sadrži ispred nekog odabranog čvora koji postoji u listi

            students.AddBefore(node, "Ella Napolis");

Dodavanje nog čvora na početak

            students.AddFirst("Antoinete Amorros");

Pronalaženje čvora koji sadrži određeni podatak

            node = students.Find("Antoinete Amorros");

            students.AddAfter(node,"Charles Junger");

**Queue<>**

Elemente koje sadrži čuva u obliku povezane liste, ali uređene u FIFO redosledu (First In Fist Out).

Queue je stukture su specfi

Kreiranje

 Queue<string> numbers = new Queue<string>();

Dodavanje

            numbers.Enqueue("one");

Izbacivanje (prvog)

            numbers.Dequeue();

**Stack<>**

Elemente koje sadrži čuva u obliku povezane liste, ali uređene u LIFO redosledu (Last In Fist Out).

Kreiranje

          Stack<string> numbers = new Stack<string>();

Dodavanje

            numbers.Push("one");

Izbacivanje (poslednjeg)

            numbers.Pop();

**SortedList<TKey, TValue>**

* Sortirana generička lista koja sadrži par key/value (ključ/vrednost).
* Sortira podatke isključivo prema ključu.
* Ključ mora imati jedinstvene vrednosti.

Kreiranje

 SortedList<string, int> students = new SortedList<string, int>();

Dodavanje novog para

  students.Add("Manuel Radovanovic", 10);

students.Add("Miroslav Markov", 40);

Ispis

foreach (KeyValuePair<string, int> x in students)

{

Console.WriteLine("{0}:{1}",x.Key,x.Value);

}

**Negeneričke kolekcije**

**Queue**

Kolekcija koj može sadržati elemente različitih tipova i koja ih pakuje kao FIFIO strukturu.

Count – predstavlja broj elemenata u kolekciji (obratiti pažnju nije metod)

Clear() – metoda koja briše sve elemente u kolekciji

Contains() – prepoznaje zadati elemenat ukoliko ga imate u kolekciji

CopyTo() – kopira sve elemente iz kolekcije u zadati niz

Dequeue() – uklanja i vraća prvi elemenat u kolekciji

Enqueue() – dodaje elemenat i stavlja ga zadnjeg u kolekciju

GetEnumerator() – vraća enumerator za redove

Peek() – vraća prvi elemenat iz kolekcije ali ga ne uklanja

ToArray() – kopira sve elemente iz kolekcije u novi niz

Kreiranje

            Queue queue = new Queue();

Dodavanje

            queue.Enqueue("Hello");

Ispis

            int count = queue.Count;

            for (int i = 0; i < count; i++)

            {

                Write(queue.Dequeue() + " ");

            }

**Stack**

Kolekcija koj može sadržati elemente različitih tipova i predstavlja LIFO strukturu.

Zahtev da se skine jedan element sa steka će dati element koji je poslednji dodat.

Kreiranje

            Stack stack = new Stack();

Dodavanje

            stack.Push("First");

Skidanje i ispis poslednjeg elemeta koji je dodat

            WriteLine(stack.Pop());

**Hashtable**

Vrsta rečnika, poznata pod nazivom mapa ili heš tabela.

Count –broj elemenata u heš tabeli

Add(Object, Object) – dodaje elemenat sa specifičnim ključem i vrednošću

Clear() – briše sve elemente iz heš tabele

Clone() – kreira površnu kopiju heš tabele

Contains(Object) – određuje da li heš tabela sadrži specifični ključ

ContainsKey(Object) – isto kao i prethodna metoda

ContainsValue(Object) – određuje da li heš tabela sadrži specifičnu vrednost

CopyTo(Array, Int32) – kopira elemente iz heš tabele u jednodimenzionalan niz instanciran na specifičnom indeksu

Equals(Object) – određuje da li je specifičan objekat isti kao trenutni objekat

GetHash(Object) – vraća heš kod za specifičan ključ

Remove(Object) - uklanja elemenat iz heš tabele na osnovu zadatog ključa

Kreiranje

           Hashtable employee = new Hashtable();

Dodavanje

            employee.Add("M0127", "Amanda Scott");

Pretraga i uklanjaje

            foreach (DictionaryEntry entry in hashtable)

            {

                WriteLine($"Code: {entry.Key} - {entry.Value}");

            }

if (key != null) hashtable.Remove(key);

**SortedList**

Lista sa sortiranim elementima.

Kreiranje

           SortedList employee = new SortedList();

Dodavanje

            employee.Add(1, "Amanda Scott");

Ispis

employee.Add(1, "Amanda Scott");

foreach (DictionaryEntry entry in employee)

{

Console.WriteLine(entry.Key + " " + entry.Value);

}