СТЕКТЕР, КЕЗЕКТЕР ЖӘНЕ ТІЗІМДЕР

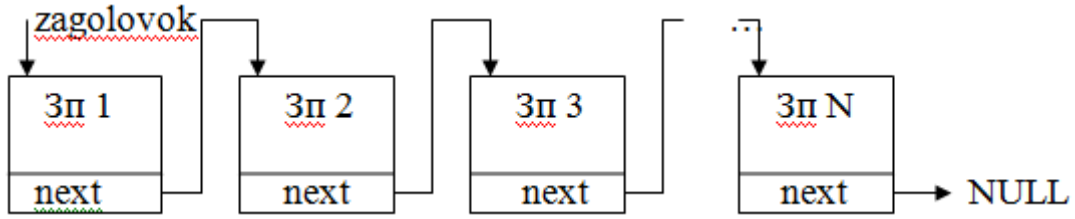
11.1 Тізімдік құрылымдар

Кез келген ақпараттық анықтама жүйесімен жұмыс жасау деректер массивін қолдануды талап етеді. Әдетте деректер магнитті дискте файлдар түрінде сақталады. Бірақ егер файл деректерін үнемі қатынасу және түрлі іздестіру операцияларын орындау керек болса, онда осы деректер файлдарын компьютердің жадына орналастыру дұрыIrimshikақ болады, өйткені магнитті дискте орналасқан файлда деректерді іздестіру уақыты компьютер жадысында орналасқан файлда іздестіру уақытынан мың есе көп.

Компьютердің жадына үлкен анықтамалық ақпарат файлдарын орналастыру мәселесі туындайды.

Деректерді файлдан компьютер жадысына, магнитті дискке енгізудің көптеген алгоритмдері бар. Мысалы, деректер файлын түрлі кесте, матрица түрінде көрсетуге болады.

Компьютердің жадына деректерді орналастырудың бір нұсқасы - түрлі тізімдік құрылымдар. Компьютердің жадына жазбалар тізімін ұйымдастыруда әрбір элемент (жазба) құрылымына қосымша өріс, яғни тізімнің келесі элементіне нұсқағыш (әдетте осы өріс next деп аталады) қосылады. Осындай құрылымның негізінде тізім элементтері біртұтас болып біріктіріледі. Құрылымда тізімге кіріс нүктесі – тізімнің басы немесе тақырыбы және тізімнің аяқталу сипаты (тізімдегі соңғы элементтің next өрісінің мәні NULL тең) немесе тізімдегі элементтер санын анықтайтын айнымалы нөлге тең болуы керек.



11.1-сурет – Қарапайым тізімнің құрылымы

Тізімдік құрылымдар әдетте мына әрекеттерді (алгоритмдерді) орындайды:

– жазбалар файлынан тізімді құруды;
– тізімді қарап шығуды;
– тізімге жаңа элементті қосуды (тізімнің басына, ортасына, соңына);
– тізімнен элементті жою (тізімнің басынан, ортасынан, соңынан);
– тізімді файлға жазу;
– бірнеше тізімдерді біріктіру;
– бір тізімді екіге немесе бірнеше бөліктерге бөлу;
– тізімдегі элементтерді сұрыптау, т.б.

Қарапайым тізімнен элементті өшіру мен қосу жолдары бойынша барлық тізімдік құрылымдарды төрт топқа бөлуге болады:

– стектер;

– кезектер;

– дектер;

– қарапайым тізімдер.

Стек – қарапайым тізім, онда элементті өшіру мен қосу бір жақ бөлігіндегі тізімнің соңында орындалады. Әдетте стекте орындалатын жұмыс принципі «Бірінші кірген соңында шығады, соңында кірген бірінші болып шығады» сөйлемімен түсіндіріледі.

Кезек – қарапайым тізім, онда элементті қосу тізімнің бір жақ бөлігінде (кезек соңы), ал өшіру немесе элементпен жұмыс істеу тізімнің екінші бөлігінде (кезектің басы) орындалады.

Дек – қарапайым тізім, онда элементті өшіру мен қосуды тізімнің кез келген соңында (ұшында) орындауға болады.

Қарапайым тізімде элементті өшіру мен қосу тізімнің кез келген жерінде орындауға болады, сонымен қатар тізімнің ортасында.

11.2 Стек типіндегі құрылыммен жұмысты ұйымдастыру

Стек типіндегі құрылым әртүрлі есептеу үдерісінде кең қолданылады. Компьютер процессоры үзуді өңдеу барысында стекті қолданады. Алгоритмдердің көпшілігі алгоритмді алдыңғы қалпына «қайтаруды» орындау үшін стекттерді қолданады, мысалы, лабиринтті, графты немесе бұтақтарды жүріп өткен кезде. Көптеген көліктер туралы есептерді шешу алгоритмдерінде оңтайлы маршруттарды табу үшін стек пайдаланылады, т.б.

Әлбетте, «Алгоритмизация және бағдарламалау» пәнінде стек сияқты құрылымды қарастырмай өте алмаймыз, C# тілінде оған әдістер жиыны бар арнайы класс (`Stack`) бөлінген.

Стектермен жұмысты орындау барысында негізгі алгоритмдер: стекке элементті қосу алгоритмі, стектен элементті жою алгоритмі, стек мазмұнын қарау.

Стек типіндегі құрылымға қол жеткізу әдетте стек төбесі деп аталатын оның тек бір жақ шетінде ғана мүмкін.

Стек класы динамикалық коллекция – бір типтегі деректердің бірігуі түрінде бола алады. Стекті кеңейткен кезде («толтырылған» стекке элементті қосу барысында) стек сыйымдылығы динамикалық түрде екі есе үлкейеді.

`Stack` класында келесі үш конструктор анықталған:

Public Stack();

Public Stack(int capacity);

Public Stack(ICollection n);

Бірінші конструктор 10 элементтен тұратын «бос» стекті құрады. Екінші конструктор сыйымдылығы `capacity` болатын «бос» стекті құрады. Үшінші конструктор n элементті стекті құрады.

11.1-кестеде `Stack` класының негізгі әдістері ұсынылған.

11.1-кесте – `Stack` класының негізгі әдістері

Әдіс	Сипатама

public virtual bool Contains(object v)	True мәнін қайтарылады, егер v объектісі стекте болса, әйтпесе false мәні қайтарылады.
public virtual void Clear()	Стекті тазартады (Count қасиеті – стек элементтерінің саны нөлге теңестіріледі).
public virtual object Peek()	Стек төбесінің элементін қайтарады, бірақ оны жоймайды.
public virtual object Pop()	Стек төбесінің элементін қайтарады және оны жояды.
public virtual void Push(object v)	v элементін стекке қосады - стек төбесіне

Стек ішіндегісін көру үшін `in` операциясы жиі қолданылады, мысалы:

```
Console.Write("Стек ішіндегісі = ");
foreach (int i in vst)
Console.Write(i + " ");
Console.WriteLine();
```

Кодтың көрсетілген жолдары vst төбесінен және бүтін сандардан тұратын стекті қарап шығуға мүмкінді береді.

Стектермен жұмыс жасайтын алгоритмнің мысалы ретінде келесі есепті қарастырайық:

11.1-есебі. 5 бүтін саннан тұратын стек үшін санды қосу және шығару ықтималдығы кездейсоқ құрылады. Бұл орайда санды стекке қосу ықтималдығы 70% тең, ал стектен шығару 30% тең. Сандар кездейсоқ құрылады және минус 50 мен 50 аралығында болады.

Стекті толтыру барасында және стекке жаңа санды қосу кезінде, шығару ықтималдықтары қарама-қарсы мәндерге өзгереді. Бағдарлама жұмысының соңы санды «бос» стектен шығару мүмкіндігімен анықталады.

Бағдарламаның коды:

```
using System;
using System.Collections;
namespace ConsoleApplication1
{
 class Program
 {
 static void vkl(Stack vst, int n)
 {
  vst.Push(n);
  Console.Write("Stekke element zhazildi - {0}", n);
  Console.Write("Stekte zhazildi = ");
  foreach (int i in vst)
  Console.Write(i + " ");
  Console.WriteLine();
 }
 static void iskl(Stack vst)
```

```
{
 if (vst == null) Console.WriteLine("Stek bos!");
 else
 {
 int n = (int)vst.Pop();
 Console.Write("Stekten element zhoildi {0}", n);
 Console.Write("Stekte zhazildi = ");
 foreach (int i in vst)
  Console.Write(i + " ");
 Console.WriteLine();
 }
}
static void Main()
{
 Stack vstek = new Stack();
 int i, k, n;
 Random rnd = new Random();
 i = 0;
 while (i < 5)
 {
 k = rnd.Next() % 101;
 if (k <= 70)
 {
  i++;
  n = rnd.Next() % 101 - 50;
  vkl(vstek, n);
 }
 else
  if (i > 0)
  {
  i--;
  iskl(vstek);
  }
 }
 Console.WriteLine("Stek tolik!");
 while (i > 0)
 {
 k = rnd.Next() % 101;
 if (k <= 30)
 {
  i++;
  n = rnd.Next() % 101 - 50;
  vkl(vstek, n);
 }
 else
  if (i > 0)
```

```
   {
   i--;
   iskl(vstek);
   }
  }
  Console.WriteLine("Stek bos!");
  Console.WriteLine("Enter pernesin basiniz");
  Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы:

```
    Stekke element zhazildi - -6Stekte zhazildi = -6
    Stekke element zhazildi - 12Stekte zhazildi = 12 -6
    Stekke element zhazildi - 11Stekte zhazildi = 11 12 -6
    Stekten element zhazildi 11Stekte zhazildi = 12 -6
    Stekke element zhazildi - -8Stekte zhazildi = -8 12 -6
    Stekke element zhazildi - -24Stekte zhazildi = -24 -8 12
-6
    Stekke element zhazildi - 14Stekte zhazildi = 14 -24 -8
12 -6
    Stek tolik!
    Stekten element zhoildi 14Stekte zhazildi = -24 -8 12 -6
    Stekten element zhoildi -24Stekte zhazildi = -8 12 -6
    Stekten element zhoildi -8Stekte zhazildi = 12 -6
    Stekten element zhoildi 12Stekte zhazildi = -6
    Stekke element zhazildi - -27Stekte zhazildi = -27 -6
    Stekke element zhazildi - -13Stekte zhazildi = -13 -27 -
6
    Stekten element zhoildi -13Stekte zhazildi = -27 -6
    Stekten element zhoildi -27Stekte zhazildi = -6
    Stekten element zhoildi -6Stekte zhazildi =
    Stek bos!
    Enter pernesin basiniz
```

Бағдарламада стекке элементті қосу және одан элементті шығару операцияларының әрқайсысына түсініктеме қосылған.

11.3 Кезек типіндегі құрылыммен жұмыстарды ұйымдастыру

Кезек типіндегі құрылымдар «қайтаруы бар іздеу» алгоритмдерінде кең қолданылады. Графтардың, бұтақтардың төбелерінен өту алгоритмдері «ішіне қарай жүру» алгоритімін жүзеге aIrimshiky кезінде – стектерді немесе «көлденеңінен жүру» алгоритмін жүзеге aIrimshiky кезінде – кезектерді қолданады. Жаппай қызмет

көрсету теориясының көптеген есептері оңтайлы нұсқаларды табу үшін кезектерді қолданады.

Кезек типіндегі құрылымның екі тақырыбы болады: ол арқылы кезек элементімен жұмысты орындайтын (элементті жою) кезектің басы және кезекке жаңа элементтерді қосу үшін қолданылатын кезектің соңы. Кезектермен жұмыс жасау үшін C# тілінде арнайы класс (`Queue`) бөлінген. Бұл кластың әдістер жиыны бар

`Queue` класында келесі үш конструктор бар:

Public Queue ();

Public Queue (int capacity);

Public Queue (ICollection c);

Бірінші конструктор 10 элементке «бос» кезекті құрайды. Екінші конструктор сиымдылығы `capacity` элементіне тең «бос» кезекті құрайды. Үшінші конструктор ICollection элементтері арқылы жұмыс жасайтын, n элементке арналған жаңа кезекті құрайды.

`Queue` класының негізгі әдістері 11.2-кестесінде ұсынылған.

11.2-кесте – `Queue` класының негізгі әдістері.

Әдіс	Сипатама
public virtual bool Contains(object v)	True мәнін қайтарылады, егер v объектісі кезекте болса, әйтпесе false мәні қайтарылады.
public virtual void Clear()	Кезекті тазартады (Count қасиеті – кезек элементтерінің саны нөлге теңестіріледі).
public virtual object Peek()	Объектті кезектің басынан қайтарады, бірақ оны жоймайды.
public virtual object Dequeue()	Объектті кезектің басынан қайтарады және оны жояды.
public virtual void Enqueue(object v)	v объектісін кезек соңына қосады.

Кезек әдістерінің жұмысын түсіндіретін мысал келесі есепте қарастырылады:

11.2-есебі. Дүкенге бір күнде 250 адам келеді. Сатып алушыларға қызмет көрсету ретінде мыналар сатылады: нан – ықтималдығы 25%; ірімшік – ықтималдығы 20%; печенье– ықтималдығы 20%; сыра – ықтималдығы 10% және балмұздақ – ықтималдығы 25%. Әрбір сатып алушы тек бір өнімді ғана сатып алады деп есептейік. Барлық оқиғалардың ықтималдықтары 0 – 100 аралығына тең. Сатып алушылар кезегін ұйымдастыру керек, ал оларға қызмет көрсету кезінде әртүрдегі сатылған өнімдердің санын және жиі сатып алынатын өнімді анықтап басып шығару керек.

Есептің шарты бойынша барлық оқиғалардың ықтималдықтары бірдей, сондықтан, егер 0 мен 100 аралығында кездейсоқ санды құрсақ, онда оқиғаның орындалу ықтималдығы берілген диапозон арқылы анықталады. Мысалы, нанды сатып алу ықтималдылығы 1-ден 25-ке дейінгі аралықта анықталады, ал ірімшікті – 26 мен 45, печеньені – 46 мен 65, сыраны – 66 мен 75 және балмұздақты –76 мен 100 аралықтарында анықтайды.

Өнімнің әрбір түрін 0 мен 4-ке дейінгі бүтін сандармен анықтауға болады. Сонымен, сатып алушылар кезегін олар сатып алатын өнімдер кезегімен алмастыруға болады, олар бүтін санмен белгіленеді.

Бағдарлама коды:

```
using System;
using System.Collections;
using System.Text;
namespace ConsoleApplication1
{
 class Program
 {
 public static int n;
 static void vkl(Queue ocer, int n)
 {
  ocer.Enqueue(n);
 }
 static void iskl(Queue ocer)
 {
  n = (int)ocer.Dequeue();
 }
 public static void Main()
 {
  Queue zagol = new Queue();
  int[] masi = new int[5];
  char[] masc = new char[5] { 'N', 'I', 'P', 'S', 'B' };
  string[] mass = new string[5] { "Nan", "Irimshik",
"Печенье", "Sira", "Balmuzdak" };
  int i, j, p, k;
  Random rnd = new Random();

  for (i = 0; i < 4; i++) masi[i] = 0;
  for (i = 0; i < 250; i++)
  {
  p = rnd.Next() % 100 + 1;
  if (p <= 25) vkl(zagol, 0);
  if (25 < p && p <= 45) vkl(zagol, 1);
  if (45 < p && p <= 65) vkl(zagol, 2);
  if (65 < p && p <= 75) vkl(zagol, 3);
  if (p > 75) vkl(zagol, 4);
  };
  for (i = 0; i < 250; i++)
  {
  iskl(zagol);
  masi[n]++;
  Console.Write(masc[n]);
  if ((i + 1) % 25 == 0) Console.WriteLine();
```

```
   };
   k = 0; j = 0;
   for (i = 0; i < 5; i++)
   {
   Console.WriteLine(" {0} = {1}", mass[i], masi[i]);
   if (k < masi[i]) { k = masi[i]; j = i; }
   }
   Console.WriteLine("En kop satilgan {0} = {1}", mass[j],
masi[j]);
   Console.WriteLine("Enter pernesin basiniz");
   Console.ReadLine();
  }
  }
}
```

Бағдарлама жұмысы:

```
IBIINSBIINNINPINPPNPBSIIP
INNPBPSNNNNNPBSPINSBPSNPP
BBBNNNNIBNNNBINISPBSPPNNB
PPNPNPBSBNNININPBSINIPIPB
IPBBPBBBNBNBINPNPBNPNPBSS
NBBPNBNPSINNBBSNBPPPPPNBB
BPNPBSIPPINPBPBBBPIBNNPBI
PPISPSIPPNBIIIPNNBBSNNBSP
PBSINIPSSPNIIPSBISPINNSSN
NSNIIPNPNBBNIININSIPINSNI
 Nan = 68
 Irimshik = 45
 Pechenie= 58
 Sira = 29
 Balmuzdak = 50
En kop satilgan Nan = 68
Enter pernesin basiniz
```

11.4 Тізім типіндегі құрылыммен жұмыстарды ұйымдастыру

Тізім типіндегі әр түрлі, құрылымдар болады. C# тілінде олар үшін арнайы `List<T>`, `ArrayList`, `LinkedList<T>` , т.б. кластар дайындалған. Объекттердің динамикалық массивтері болып келетін `List<T>` және `ArrayList` кластары жиі қолданылады. (11.3, 11.4 кестелері).

11.3-кесте –ArrayList класының негізгі қасиеттері

Кластың қасиеті	Сипаттамасы
public virtual int Capacity {get;}	Осы қасиет оқу үшін ғана арналған, коллекция ағымдағы сыйымдылығын сақтайды.
public virtual int Count	Осы қасиет оқу үшін ғана арналған,

{get;};	коллекцияның ағымдағы ұзындығын сақтайды.

11.4-кесте – ArrayList класының негізгі әдістері

Кластың әдістері	Сипаттамасы
public static ArrayList (IList: List);	List коллекциясы негізінде ArrayList объектісі құрылады
public virtual int Add(Oblect Value);	Тізім соңына жаңа объектті қосады, оның индексін қайтарады.
public virtual viod AddRange (ICollection coll);	Тізім соңына бірнеше объекттерді қосады
public virtual int BinsrySearch(Qbject Value);	Сұрыпталған тізімде Value объектісін табады және оның индексін қайтарады. Егер табылмаса, теріс санды қайтарады.
public virtual bool Containse (Object Value);	Егер коллекцияда Value элементі бар болса, true қайтарылады.
public static ArrayList FixedSize(ArrayList AL);	Әдіс элементтерін өзгертуге болатын, бірақ қосуға немесе жоюға болмайтын объекттерді қайтарады,
public virtual IEnumerator GetEnumerator();	Объект үшін итераторды қайтарады.
public virtual ArrayList GetRange(int Indx, int Count);	Элементтер диапазонын қайтарады.
public virtual int IndexOf(Object Value);	Value мәні бар элемент индексін қайтарады.
public virtual void Insert (int Indx, Object Value);	Коллекцияның керекті Indx орнына Value элементін қосады.
public virtual void InsertRange(int Indx, IColliction col);	Элементтер диапазонын қосады
public virtual bool IsFixedSize{get;}	Егер объекттің белгіленген өлшемі болса, true қайтарылады.
public virtual bool IsReadOnly{get;}	Объектте оқуға ғана арналған элементтер болса , true қайтарылады.
public virtual int LastIndexOf(Object Value);	Коллекцияда соңғы рет кездесетін Valiue мәнінің индексін қайтарады.
public static ArrayList ReadOnly(ArrayList AL);	Коллекция элементтеріне оқуға ғана рұқсат беретін режімді орнатады.
public virtual void Remove (Object Value);	Коллекциядан мәні Value болатын бірінші элементті жояды.
public virtual void RemoveAt (int Indx);	Коллекциядан индексі Indx болатын элементті жояды.
public virtual void RemoveRange(int Indx, int count);	Коллекциядан индексі Indx болатын элементтен бастап count элементтерін жояды.

11.4-кестесінің жалғасы

Кластың әдістері	Сипаттамасы
public virtual Object this[int Indx]{set; get;}	Осы қасиет элементтерге индексі бойынша қатынасуға мүмкіндік береді
public static ArrayList Repeat(Object Value, int count);	Value элементі count рет қайталанатын коллекцияны қайтарады.
public virtual void Reverse();	Элементтердің ретін кері бағытқа өзгертеді.
public virtual void SetRange (int Indx, ICollection col);	Indx индексінен бастап col коллекциясын қосады.
public virtual void Sort();	Коллекцияны сұрыптайды.
public virtual void TrimToSize();	Коллекцияға элементтерінің санына тең болатын сыйымдылықты орнатады.
public virtual void Clear();	Коллекцияның барлық элементтерін жояды.

`ArrayList` коллекциясының алғашқы сыйымдылығы 16-ты элементке тең. Коллекцияны кеңейту кезінде оның сыйымдылығы екі есе үлкейеді және 32, 64, 128, т.б. құрайды. `TrimToSize()` әдісі қолданылмайтын элементтерді өшіреді.

Бағдарлама кодының мысалы:

```
using System;
using System.Collections;
using System.Text;
namespace ConsoleApplication1
{
 struct Avto
 {
 public String Marka; //автомобильдің үлгісі
 public String Voditel; // жүргізушінің аты-жөні
 public int Stoim; // автомобильдің бағасы
 public int God;  // шығарылған жылы
 }
 class Program
 {
 public static int kol = 0;
 public static Avto avto = new Avto();
 static ArrayList Garaj = new ArrayList();
 public static void addcpi()
 {
  int b;
  int kol;
  string buf;
  Console.WriteLine("Avtomobil sani");
  buf = Console.ReadLine();
  kol = Convert.ToInt32(buf);
  for (int i = 0; i < kol; i++)
  {
```

```
  Console.WriteLine("{0}-Avtomobildin ylgisin engiziniz", i
+ 1);
  buf = Console.ReadLine();
  avto.Marka = buf;
  Console.WriteLine("{0}-Avtomobil zhyrgizyshinin ati-
zhoni", i + 1);
  buf = Console.ReadLine();
  avto.Voditel = buf;
  Console.WriteLine("{0}-Avtomobildin bagasi", i + 1);
  buf = Console.ReadLine();
  b = Convert.ToInt32(buf);
  avto.Stoim = b;
  Console.WriteLine("{0}-Avtomobildin shikkan zhili", i +
1);
  buf = Console.ReadLine();
  b = Convert.ToInt32(buf);
  avto.God = b;
  Garaj.Add(avto);
  }
 }
 public static void printcpi()
 {
  Console.WriteLine("{0,20}, {1, 20}, {2, 15}, {3, 15}",
  "Ylgisi", "Zhyrgizyshi ati-zhoni", "Avtom. bagasi",
"Avtom. shikkan zhili");
  foreach (Avto T in Garaj)
  Console.WriteLine("{0,20}, {1, 20}, {2, 15}, {3, 15}",
   T.Marka, T.Voditel, T.Stoim, T.God);
 }
 public static void ydalelemcpi()
 {
  int n;
  string buf;
  Console.WriteLine("Zhoilatin zhazba nomeri?");
  buf = Console.ReadLine();
  n = Convert.ToInt32(buf);
  // элементті номері бойынша жоямыз
  Garaj.RemoveAt(n);
 }
 static void Main()
 {
  // тізімді құрамыз және тоолтырамыз
  addcpi();
  // тізімді шығарамыз
  printcpi();
  // элементті номері бойынша жоямыз
```

```
  ydalelemcpi();
  // тізімді шығарамыз
  printcpi();
  Console.ReadLine();
 }
 }
}
```

Бағдарлама жұмысы:

```
Avtomobil sani
3
1-Avtomobildin ylgisin engiziniz
Лада
1-Avtomobil zhyrgizyshinin ati-zhoni
Петров
1-Avtomobildin bagasi
2000
1-Avtomobildin shikkan zhili
2007
2-Avtomobildin ylgisin engiziniz
Фиат
2-Avtomobil zhyrgizyshinin ati-zhoni
Иванов
2-Avtomobildin bagasi
3000
2-Avtomobildin shikkan zhili
2008
3-Avtomobildin ylgisin engiziniz
БМВ
3-Avtomobil zhyrgizyshinin ati-zhoni
Сидоров
3-Avtomobildin bagasi
4000
3-Avtomobildin shikkan zhili
2006
 Ylgisi, Zhyrgizyshi ati-zhoni, Avtom. bagasi, Avtom. shikkan zhili
 Лада, Петров, 2000, 2007
 Фиат, Иванов, 3000, 2008
 БМВ, Сидоров, 4000, 2006
Zhoilatin zhazba nomeri?
1
 Ylgisi, Zhyrgizyshi ati-zhoni, Avtom. bagasi, Avtom. shikkan zhili
 Лада, Петров, 2000, 2007
 БМВ, Сидоров, 4000, 2006
```

11.5 Кеңейтілген іздеу мүмкіндіктері бар тізімдер

C# тілінде бір бағыттық тізімдердің іздестіру мүмкіндіктерін кеңейту үшін қос бағыттық тізімдер қосылды, мысалы, `LinkedList<T>`.

Қос бағыттық тізімдер түйіндердің тізбектерінен тұрады, онда келесі (әдете `Next` өрісі) және алдыңғы (әдете `Old` өрісі) түйіндерге сілтемелер болады.

Кеңейтілген іздестіру мүмкіндіктері тізім бойынша кез келген бағытта көшу мүмкіндігімен байланысты. Қос бағыттық тізімдердің қосымша артықшылығы - тізім элементерінің қосу, алу операцияларының қарапайымдылығы. Осы тәртіп араларында операция орындалатын тізім элементтерінің бірнеше сілтемелерін жаңартады.

Қос бағыттық тізімнің келесі модификациясы циклдік тізімдерді құру болып табылады. Циклдік тізімде соңғы элемент `Next` нұсқағышы арқылы бірінші элементпен, ал тізімнің бірінші элементі `Old` нұсқағышы көмегімен соңғы элементпен байланыстырылады.

Циклдік тізімде тақырып кез келген элементте орналаса алады. Мысалы, іздеу операциясын орындағаннан кейін біз тақырыпты табылған элементке орната аламыз. Келесі іздеу операциясы кезінде тақырып келесі табылған элементке көшірілуі мүмкін.

Тізімде екі бағытта орнын ауыстыру мүмкіндігін ескере отырып іздеу алгоритмін жақсартуға болады.

Алайда стандартты іздеу алгоритмдерінің көпшілігі тізімдерді өңдеуге бағытталған. Бұл орайда тізім тақырыбы белгілі жерде орналасуы керек. Осы мақсатта тақырыпты өрістерінде мағыналық мәні жоқ элементке орналастыру ұсынылды, ал элемент циклдік тізімде бастапқы нүкте ретінде керек.

Тізімді осылай ұйымдастыру тақырыбы бөлінген тізім деген атқа ие болды.

Екілік іздеу алгоритмін орындау үшін бұтақтар тәрізді құрылым қолданылады, элементтерінде екі нұсқағыштары бар (оң және сол).

11.6 Өзін-өзі тексеру сұрақтары

1 Элементтерді қосу мен жою тізімнің бір жақ ұшында орындалады. Осы қарапайым тізім қалай аталады?

2 Стектің басы қалай аталады?

3 Public Stack(ICollection n); жазбасы нені білдіреді?

4 Бағдарламаның келесі үзіндісі нені орындайды

```
 int n = (int) vst.Pop();,
```

мұнда `vst` – стек төбесі?

5 Бағдарламаның келесі үзіндісі нені орындайды:

```
i = 0;
while (i < 5)
  {
   i++;
   n = rnd.Next() % 101 - 50;
```

```
   vstek.Push(n);
  } ,
```

мұнда `vstek` – стек төбесі?

6 элементтерді қосу тізімнің бір жақ ұшында, ал элементті жою екінші жақ ұшында орындалатын қарапайым тізім қалай аталады?

7 `Public Queue()` жазбасы нені білдіреді?

8 `object queue.Peek()` жазбасы нені білдіреді?

9 Бағдарламаның келесі үзіндісі нені орындайды:

```
foreach (int i in ocer)
 Console.Write(i + " ");
 Console.WriteLine();,
```

мұнда `ocer` – кезектің тақырыбы?

10 Бағдарламаның келесі үзіндісі нені орындайды:

```
ocer.Clear();,
```

мұнда `ocer` – кезектің тақырыбы?