Ministerul Educației al Republicii Moldova

Universitatea Tehnică a Moldovei
Facultatea CIM

Departamentul Ingineria Software și Automatică

RAPORT

Lucrare de laborator Nr. 3
La APPOO
Tema: Biblioteca standartă a șabloanelor.

A efectuat: st. Gr. TI-142
Cojucari Dan

A verificat: Gavrișco Alexandru
Andrei Postaru

Chișinău 2017

Scopul lucrării: *Studierea tehnologiei de programare folosind Standard Template Library(STL) a limbajului C#.*

1. Sarcina lucrării

De elaborat 3 programe. Primul program demostrează folisrea contaierilor și a tipurilor de date standart. Programul 2 demonstrează folosirea containerilor și a claselor definite de utilizator. Programul 3 demonstrează folosirea algoritmilor STL.

Programul 1 va efectua:

1. Să se creeze contanier conform variantei și să se umple cu date conform variantei.
2. Să se vizualizeze containerul.
3. Să se modifice containerul, prin ștergerea unui element și inserarea altui.
4. Să se vizualizeze containerul utilizînd iteratorul.
5. Să se creeze al 2-lea container cu date de același tip.
6. Să se modifice primul container, ștergînd elemente de la al n-lea element și să se adauge toate elementele containerului 2.
7. Să se vizualizeze primul și al 2-lea container.

Programul 2 va efectua aceleași funcții ca și î primul program, dar cu date de tip utilizator.

Programul 3 va efectua:

1. Să se creeze contanier conform variantei și să se umple cu date de tip utilizator.
2. Să se sorteze containerul descrescător.
3. Să se vizualizeze containerul.
4. Să se găsească un element în container conform condiției căutate.
5. Elementele care satisfac condiția să fie copiete în container de al 2-lea tip.
6. Să se vizualizeze containerul 2.
7. Să se sorteze containerele descrescător.
8. Să se vizualizeze ambele containere.
9. Să se obține al 3-lea container prin fuziunea primelor două.
10. Să se afișeze containerul 3.
11 .Să se afișeze cîte elemente satisfac condiția.
12.Să se determine dacă contanerul coține elementul căutat.

Varianta 5:

5	queue	vector	char

3. Realizarea sarcinii

Codul sursă este anexat în anexă.

În cadrul acestei lucrări de laborator am creat același program ca și în laboratorul precedent doar că utilizînd Java. În cadrul acestei lucrări am creat mia multe clase deoarece Java este un lombaj mai mult orientat pe obiecte, și fiecare subprogram reprezintă o clasă aparte.

Pentru al dolea și al treilea program am definit o clasă Carte ce conține o variabilă integer cu numărul paginilor, dar și numele cărții. Am definit în cadrul acestei clase constructorii dar și am creat metodele necesare pentru citirea și afișarea datelor acestei clase, deoarece în Java operatorii nu pot fi supraîncărcați.

```
public class Carte:IComparable<Carte>
{
    private int pagesNumber;
    private string name;
    public Carte()
    {
        this.pagesNumber = 0;
        this.name = "Fara_Nume";
    }
    public Carte(int pages, string newName)
    {
        this.name = newName;
        this.pagesNumber = pages;
    }
    public void setPagesNumber(int nr)
    {
        this.pagesNumber = nr;
    }
    public int getPagesNumber()
    {
        return this.pagesNumber;
    }
    public void setName(string name)
    {
        this.name = name;
    }
    public string getName()
    {
        return this.name;
    }
    public void WriteData()
    {
        Console.WriteLine("Name: " + name);
        Console.WriteLine("Pages number: " + pagesNumber);
    }
    public void ReadData()
    {
        Console.Write("Name: ");
        name = Console.ReadLine();
        Console.Write("Pages number: ");
        pagesNumber = Convert.ToInt32(Console.ReadLine());
    }
    public int CompareTo(Carte other)
    {
        return this.pagesNumber - other.pagesNumber;
    }
}
```

Concluzie

În urma efectuării aceste lucrări de laborator am făcut un program cu meniu, care îndeplinește cele 3 condiții. Am lucrat cu biblioteca STL a limbajului C#. Am observat că limbajul C# nu are mari diferențe față de limbajul Java. În C# operatorii pot fi supraîncărcați, dar nu avem nevoie în cazul nostru. Pentru sortare am folosit ca și în Java CompareTo, o metoda predefinită in Icomparable. Am lucrat cu containerele Queue și List, deoarece în C# vector nu este, iar List este o alternativa la Vector. De asemenea, am observat ca coada nu are iterator, de aceea folosim un vector temporar, in care efectuam parcurgerea, apoi copiem inapoi in coada dupa finisarea parcurgerii.

Anexa

Codul sursă:

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace ConsoleApp1
{
    public class Carte:IComparable<Carte>
    {
        private int pagesNumber;
        private string name;

        public Carte()
        {
            this.pagesNumber = 0;
            this.name = "Fara_Nume";
        }
        public Carte(int pages, string newName)
        {
            this.name = newName;
            this.pagesNumber = pages;
        }
        public void setPagesNumber(int nr)
        {
            this.pagesNumber = nr;
        }
        public int getPagesNumber()
        {
            return this.pagesNumber;
        }
        public void setName(string name)
        {
            this.name = name;
        }
        public string getName()
        {
            return this.name;
        }
        public void WriteData()
        {
            Console.WriteLine("Name: " + name);
            Console.WriteLine("Pages number: " + pagesNumber);
        }
        public void ReadData()
        {
            Console.Write("Name: ");
            name = Console.ReadLine();
            Console.Write("Pages number: ");
            pagesNumber = Convert.ToInt32(Console.ReadLine());
        }

        public int CompareTo(Carte other)
        {
            return this.pagesNumber - other.pagesNumber;
        }
    }
```

```
    class Program
    {
        //Program 1
        private static void pushDataInQueue(Queue<char> data)
        {
            String tempString;
            Console.WriteLine("Introduceti un sir de caractere:");
            tempString = Console.ReadLine();
            char[] arr = tempString.ToCharArray();
            foreach(char c in arr)
            {
                data.Enqueue(c);
            }
        }
        private static void showQueue(Queue<char> data)
        {
            Queue<char> tempData = new Queue<char>(data);
            while(tempData.Count != 0)
            {
                Console.Write(tempData.First() + " ");
                tempData.Dequeue();
            }
            Console.WriteLine();
        }
        private static void program1()
        {
            int elementsNr, elementsForDelete;
            Queue<char> queue1 = new Queue<char>();

            pushDataInQueue(queue1);
            elementsNr = queue1.Count();

            Console.WriteLine("Coada initiala este:");
            showQueue(queue1);
            Console.WriteLine();

            queue1.Dequeue();
            queue1.Enqueue('s');
            Console.WriteLine("(A fost sters primul element si a fost adaugat altul la
sfirsitul cozii)");
            Console.WriteLine("Coada modificata este:");
            showQueue(queue1);
            Console.WriteLine();

            Queue<char> queue2 = new Queue<char>(queue1);
            Console.WriteLine("(S-au copiat elementele din coada 1 in coada 2)");
            Console.WriteLine("Coada 2:");
            showQueue(queue2);
            Console.WriteLine();

            Console.WriteLine("Introduceti numarul de elemente ce vor fi sterse (nr < " +
elementsNr + ") : ");
            elementsForDelete = Convert.ToInt32(Console.ReadLine());
            for(int i = 0; i<elementsForDelete; i++)
            {
                queue1.Dequeue();
            }
            while(queue2.Count!=0)
            {
                queue1.Enqueue(queue2.First());
                queue2.Dequeue();
            }
            Console.WriteLine("(S-au copiat elementele din coada 2 in coada 1)");
            Console.WriteLine("Coada 1: ");
            showQueue(queue1);
```

```
            Console.WriteLine("Coada 2: ");
            showQueue(queue2);
            Console.WriteLine();

            Console.ReadLine();
        }

        //Program 2
        private static void pushDataInQueue(Queue<Carte> data, int size)
        {
            Console.WriteLine("Introduceti" + size + " elemente: ");
            for(int i=0; i<size; i++)
            {
                Carte tmp = new Carte();
                Console.WriteLine("Introduceti cartea " + (i + 1) + " :");
                tmp.ReadData();
                data.Enqueue(tmp);
            }
        }
        private static void showQueue(Queue<Carte> data)
        {
            Queue<Carte> tempData = new Queue<Carte>(data);
            while (tempData.Count != 0)
            {
                tempData.First().WriteData();
                tempData.Dequeue();
            }
            Console.WriteLine();
        }
        private static void program2()
        {
            int elementsNr, elementsForDelete;
            Queue<Carte> queue1 = new Queue<Carte>();

            Console.Write("Introduceti numarul de elemente: ");
            elementsNr = Convert.ToInt32(Console.ReadLine());
            pushDataInQueue(queue1, elementsNr);

            Console.WriteLine("Coada initiala este:");
            showQueue(queue1);
            Console.WriteLine();

            Carte temp = new Carte(295, "Totul despre C/C++");

            queue1.Dequeue();
            queue1.Enqueue(temp);
            Console.WriteLine("(A fost sters primul element si a fost adaugat altul la
sfirsitul cozii)");
            Console.WriteLine("Coada modificata este:");
            showQueue(queue1);
            Console.WriteLine();

            Queue<Carte> queue2 = new Queue<Carte>(queue1);
            Console.WriteLine("(S-au copiat elementele din coada 1 in coada 2)");
            Console.WriteLine("Coada 2:");
            showQueue(queue2);
            Console.WriteLine();

            Console.Write("Introduceti numarul de elemente ce vor fi sterse (nr < " +
elementsNr + ") : ");
            elementsForDelete = Convert.ToInt32(Console.ReadLine());
            for (int i = 0; i < elementsForDelete; i++)
            {
                queue1.Dequeue();
            }
```

```
    while (queue2.Count != 0)
    {
        queue1.Enqueue(queue2.First());
        queue2.Dequeue();
    }
    Console.WriteLine("(S-au copiat elementele din coada 2 in coada 1)");
    Console.WriteLine("Coada 1: ");
    showQueue(queue1);
    Console.WriteLine("Coada 2: ");
    showQueue(queue2);
    Console.WriteLine();

    Console.ReadLine();
}

//Program 3
private static void sortQueue(Queue<Carte> data)
{
    List<Carte> tmp = new List<Carte>();

    while (data.Count() != 0)
    {
        tmp.Add(data.First());
        data.Dequeue();
    }

    tmp.Sort();

    for (int i = tmp.Count() - 1; i >= 0; i--)
    {
        data.Enqueue(tmp[i]);
    }
}
private static Boolean numarPar(Carte itm)
{
    return ((itm.getPagesNumber() % 2) == 0);
}
private static Carte findCarte(Queue<Carte> data)
{
    List<Carte> tmp = new List<Carte>();
    Queue<Carte> tempData = new Queue<Carte>(data);

    while (tempData.Count() != 0)
    {
        tmp.Add(tempData.First());
        tempData.Dequeue();
    }

    for (int i = 0; i < tmp.Count(); i++)
    {
        if (numarPar(tmp[i]))
        {
            return tmp[i];
        }
    }
    return null;
}
private static void copyElements(Queue<Carte> data, List<Carte> vct)
{
    Queue<Carte> tempData = new Queue<Carte>(data);
    while (tempData.Count() != 0)
    {
        if (numarPar(tempData.First()))
        {
            vct.Add(tempData.First());
```

```
        }
        tempData.Dequeue();
    }
}
private static void showVector(List<Carte> data)
{
    for (int i = 0; i < data.Count(); i++)
    {
        data[i].WriteData();
    }
    Console.WriteLine();
}
private static void sortAsc(Queue<Carte> queue1, List<Carte> vector1)
{
    List<Carte> tmp = new List<Carte>();

    while (queue1.Count() != 0)
    {
        tmp.Add(queue1.First());
        queue1.Dequeue();
    }

    tmp.Sort();
    vector1.Sort();

    for (int i = 0; i < tmp.Count(); i++)
    {
        queue1.Enqueue(tmp[i]);
    }
}
private static Queue<Carte> concat(Queue<Carte> queue1, List<Carte> vct)
{
    Queue<Carte> queue2 = new Queue<Carte>(queue1);
    for (int i = 0; i < vct.Count(); i++)
    {
        queue2.Enqueue(vct[i]);
    }
    return queue2;
}
private static void program3()
{
    int elementsNr;
    Queue<Carte> queue1 = new Queue<Carte>(); ;

    Console.Write("Introduceti numarul de elemente: ");
    elementsNr = Convert.ToInt32(Console.ReadLine());
    pushDataInQueue(queue1, elementsNr);

    Console.WriteLine("Coada initiala este:");
    showQueue(queue1);

    sortQueue(queue1);
    Console.WriteLine("Coada sortata:");
    showQueue(queue1);

    Carte tmpCarte = findCarte(queue1);
    if (tmpCarte != null)
    {
        Console.WriteLine("Elementul par este:");
        tmpCarte.WriteData();
    }
    Console.WriteLine();

    List<Carte> vct = new List<Carte>();
    copyElements(queue1, vct);
```

```
    Console.WriteLine("Vectorul initial:");
    showVector(vct);
    Console.WriteLine();

    sortAsc(queue1, vct);
    Console.WriteLine("Coada sortata:");
    showQueue(queue1);
    Console.WriteLine("Vectorul sortat:");
    showVector(vct);

    Queue<Carte> queue2 = concat(queue1, vct);
    Console.WriteLine("Coada + Vector :");
    showQueue(queue2);

    int length = 0;
    while (queue2.Count() != 0)
    {
        if (numarPar(queue2.First()))
        {
            length++;
        }
        queue2.Dequeue();
    }
    if (length == 0)
    {
        Console.WriteLine("Nu sunt valori pare in coada");
    }
    else
    {
        Console.WriteLine("Sunt " + length + " valori pare");
    }
    Console.WriteLine();

    Console.ReadLine();
}

//Main
static void Main(string[] args)
{
    int command;
    while(true)
    {
        Console.Clear();
        Console.WriteLine("     Menu:");
        Console.WriteLine("(1) Program 1");
        Console.WriteLine("(2) Program 2");
        Console.WriteLine("(3) Program 3");
        Console.WriteLine("(0) Iesire");
        Console.Write("Commanda: ");
        command = Convert.ToInt32(Console.ReadLine());
        Console.Clear();
        switch (command)
        {
            case 0:
                Environment.Exit(0);
                break;
            case 1:
                program1();
                break;
            case 2:
                program2();
                break;
            case 3:
                program3();
                break;
```

```
                    default:
                        Console.WriteLine("Comanda gresita! Incearca din nou!");
                        Console.ReadLine();
                        break;
                }
            }
        }
    }
}
```

Bibliografie:

1. Standard Template Library (STL) [Resursă electronică]. – Regim de access: http://www.infoarena.ro/stl
2. Standard Template Library (STL) [Resursă electronică]. – Regim de access: http://vega.unitbv.ro/~cataron/Courses/PCLPII/PCLP2_Capitolul9.pdf
3. Queue C# [Resursă electronică]. – Regim de access: https://msdn.microsoft.com/en-us/library/system.collections.queue(v=vs.110).aspx
4. List C# [Resursă electronică]. – Regim de access: https://msdn.microsoft.com/en-us/library/6sh2ey19(v=vs.110).aspx
5. "C# equivalent of C++ vector, with contiguous memory?" Stackoverflow [Resursă electronică]. – Regim de access: http://stackoverflow.com/questions/6943229/c-sharp-equivalent-of-c-vector-with-contiguous-memory