Министерство образования и науки Украины

Национальный аэрокосмический университет

им. Н.Е. Жуковского «ХАИ»

Кафедра №302

Отчет по лабораторной работе №5

**По предмету:** «ООП»

**Тема:** «Шаблоны в C#»

**Выполнил:**

студент группы 316ст

Волковский В.В.

**Проверил:**

Момот М.А.

Харьков 2014

**Постановка задачи**

Вариант №2

Реализовать на базе массивов логическую структуру для хранения данных. Демонстрационная программа должна показать работу шаблона на 2-х типах: 1) Стандартный тип; 2) Ваш тип с лабораторной работе №3.

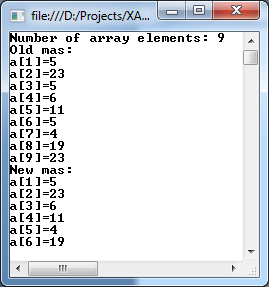
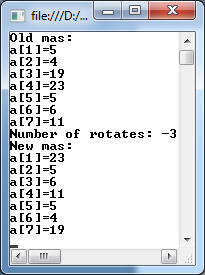
Демонстрационная программа должна показать работу каждой функции с иллюстрацией состояния до и после. Операции: input, push\_f, push\_e, pop\_f, pop\_e, sort, rotate, reverse, replace, remove, fill, unique, clear, isEmpty.

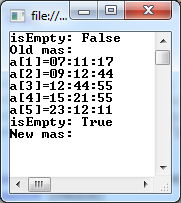
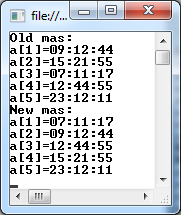
**Теоретические сведенья**

Универсальные шаблоны были добавлены в язык C# версии 2.0 и среду CLR. Универсальные шаблоны в платформе .NET Framework представляют концепцию параметров типов, которые позволяют разрабатывать классы и методы, не придерживающиеся спецификации одного или нескольких типов до тех пор, пока класс или метод не будет объявлен клиентским кодом и пока не будет создан его экземпляр. Например, используя параметр универсального типа T можно написать отдельный класс, который другой клиентский код сможет использовать без риска приведения во время выполнения или операций упаковки-преобразования.

**Исходный код**

CodeFile1.cs using System;  //Класс для работы с временем public class Time {  private int hours, minutes, seconds;   //Конструктор по умолчанию  public Time()  {  hours = minutes = seconds = 0;  }   //Конструктор инициализации  public Time(int hours, int minutes, int seconds)  {  if ((hours >= 0 && hours < 24) && (minutes >= 0 && minutes < 60) && (seconds >= 0 && seconds < 60))  {  this.hours = hours;  this.minutes = minutes;  this.seconds = seconds;  }  else  {  this.hours = 23;  this.minutes = 59;  this.seconds = 59;  }    }   //Конструктор копирования  public Time(Time t)  {  this.hours = t.getHours();  this.minutes = t.getMinutes();  this.seconds = t.getSeconds();  }   //Перегрузка оператора +, для сложения двух объектов типа Time  public static Time operator +(Time t1, Time t2)  {  int dH = 0, dM = 0, dS = 0;  if ((t1.getSeconds() + t2.getSeconds()) > 59)  {  dS = t1.getSeconds() + t2.getSeconds() - 60;  dM++;  }  else dS = t1.getSeconds() + t2.getSeconds();   if ((t1.getMinutes() + t2.getMinutes()) > 59)  {  dM += t1.getMinutes() + t2.getMinutes() - 60;  dH++;  }  else dM += t1.getMinutes() + t2.getMinutes();   if ((t1.getHours() + t2.getHours()) > 23)  {  dH += t1.getHours() + t2.getHours() - 24;  }  else dH += t1.getHours() + t2.getHours();   return new Time(dH, dM, dS);  }   //Перегрузка оператора +, для сложения объекта Time с int  public static Time operator +(Time t1, int s)  {  Time tR = new Time();  int dS = 0;  if ((t1.getSeconds() + s) > 59)  {  dS = t1.getSeconds() + s - 60;  tR.set(t1.getHours(), t1.getMinutes(), dS);  tR.addMinutes();  }  else  {  dS = t1.getSeconds() + s;  tR.set(t1.getHours(), t1.getMinutes(), dS);  }  return tR;  }   //Перегрузка оператора +, для сложения объекта int с Time  public static Time operator +(int s, Time t1)  {  Time tR = new Time();  int dS = 0;  if ((t1.getSeconds() + s) > 59)  {  dS = t1.getSeconds() + s - 60;  tR.set(t1.getHours(), t1.getMinutes(), dS);  tR.addMinutes();  }  else  {  dS = t1.getSeconds() + s;  tR.set(t1.getHours(), t1.getMinutes(), dS);  }  return tR;  }   //Перегрузка оператора -, для вычитания двух объектов типа Time  public static Time operator -(Time t1, Time t2)  {  int dH = 0, dM = 0, dS = 0;  if ((t1.getSeconds() - t2.getSeconds()) < 0)  {  dS = (t1.getSeconds() - t2.getSeconds()) + 60;  dM--;  }  else dS = t1.getSeconds() - t2.getSeconds();  if ((t1.getMinutes() - t2.getMinutes()) < 0)  {  dM += (t1.getMinutes() - t2.getMinutes()) + 60;  dH--;  }  else dM += t1.getMinutes() - t2.getMinutes();  if((t1.getHours() - t2.getHours()) < 0)  {  dH += (t1.getHours() - t2.getHours()) + 24;  }  else dH += t1.getHours() - t2.getHours();  return new Time(dH, dM, dS);  }   //Перегрузка оператора -, для вычитания объекта типа Time и int  public static Time operator -(Time t1, int s)  {  Time tR = new Time();  int dS = 0;  if ((t1.getSeconds() - s) < 0)  {  dS = (t1.getSeconds() - s) + 60;  tR.set(t1.getHours(), t1.getMinutes(), dS);  tR.minMinutes();  }  else  {  dS = t1.getSeconds() - s;  tR.set(t1.getHours(), t1.getMinutes(), dS);  }  return tR;  }   //Перегрузка оператора -, для вычитания int и объекта типа Time  public static Time operator -(int s, Time t1)  {  Time tR = new Time();  int dS = 0;  if ((t1.getSeconds() - s) < 0)  {  dS = (t1.getSeconds() - s) + 60;  tR.set(t1.getHours(), t1.getMinutes(), dS);  tR.minMinutes();  }  else  {  dS = t1.getSeconds() - s;  tR.set(t1.getHours(), t1.getMinutes(), dS);  }  return tR;  }   public static Time operator ++(Time t1)  {  t1.addSeconds();  return t1;  }   public static Time operator --(Time t1)  {  t1.minSeconds();  return t1;  }   public static bool operator ==(Time t1, Time t2)  {  if ((t1.getHours() == t2.getHours()) && (t1.getMinutes() == t2.getMinutes()) && (t1.getSeconds() == t2.getSeconds()))  return true;  else return false;  }   public override bool Equals(System.Object obj)  {  if (obj == null)  {  return false;  }  var t = obj as Time;  if ((System.Object)t == null)  {  return false;  }  return (this == t);  }   public static bool operator !=(Time t1, Time t2)  {  if(!(t1==t2))   return true;  else return false;  }   public static bool operator >(Time t1, Time t2)  {  if (t1.getHours() > t2.getHours()) return true;  else if (t1.getHours() == t2.getHours() && t1.getMinutes() > t2.getMinutes()) return true;  else if (t1.getHours() == t2.getHours() && t1.getMinutes() == t2.getMinutes()  && t1.getSeconds() > t2.getSeconds()) return true;  else return false;  }   public static bool operator <=(Time t1, Time t2)  {  if (!(t1 > t2)) return true;  else return false;  }   public static bool operator <(Time t1, Time t2)  {  if (t1.getHours() < t2.getHours()) return true;  else if (t1.getHours() == t2.getHours() && t1.getMinutes() < t2.getMinutes()) return true;  else if (t1.getHours() == t2.getHours() && t1.getMinutes() == t2.getMinutes()  && t1.getSeconds() < t2.getSeconds()) return true;  else return false;  }   public static bool operator >=(Time t1, Time t2)  {  if (!(t1 < t2)) return true;  else return false;  }   public int getHours()  {  return hours;  }   public int getMinutes()  {  return minutes;  }   public int getSeconds()  {  return seconds;  }   public virtual string getAll()  {  string h = hours.ToString();  string m = minutes.ToString();  string s = seconds.ToString();  if (hours / 10 == 0) h = "0" + h;  if (minutes / 10 == 0) m = "0" + m;  if (seconds / 10 == 0) s = "0" + s;  return h + ":" + m + ":" + s;  }   public void set(int h, int m, int s)  {  if ((h >= 0 && h < 24) && (m >= 0 && m < 60) && (s >= 0 && s < 60))  {  hours = h;  minutes = m;  seconds = s;  }  else  {  hours = 23;  minutes = 59;  seconds = 59;  }  }   public void input()  {  try  {  Console.Write("input hours=");  int h = Convert.ToInt32(Console.ReadLine());  Console.Write("input minutes=");  int m = Convert.ToInt32(Console.ReadLine());  Console.Write("input seconds=");  int s = Convert.ToInt32(Console.ReadLine());  set(h, m, s);  }  catch  {  Console.WriteLine("input error");  }  }   public void output()  {  Console.WriteLine("Time: " + hours + ":" + minutes + ":" + seconds);  }   public void addHours()  {  if (hours > 22)  hours = 0;  else  hours++;  }   public void minHours()  {  if (hours < 1)  hours = 23;  else  hours--;  }   public void addMinutes()  {  if (minutes > 58)  {  addHours();  minutes = 0;  }  else minutes++;  }   public void minMinutes()  {  if (minutes < 1)  {  minHours();  minutes = 59;  }  else minutes--;  }   public void addSeconds()  {  if (seconds > 58)  {  addMinutes();  seconds = 0;  }  else seconds++;  }   public void minSeconds()  {  if (seconds < 1)  {  minMinutes();  seconds = 59;  }  else seconds--;  } } Deque.cs using System; using System.Collections; using System.Collections.Generic; using System.Linq; using System.Text; using System.Threading.Tasks;  namespace DEQUE {  public class Deque<T>  {  List<T> mas;   public Deque()  {  mas = new List<T>();  }   public List<T> getList()  {  return new List<T>(mas);  }   public void push\_f(T obj)  {  mas.Insert(0, obj);  }   public void push\_e(T obj)  {  mas.Add(obj);  }   public T pop\_f()  {  T obj = mas[0];  mas.RemoveAt(0);  return obj;  }   public T pop\_e()  {  T obj = mas[mas.Count - 1];  mas.RemoveAt(mas.Count - 1);  return obj;  }   public void SortAsc()  {  if (mas.Count < 1) return;  mas.Sort(new ComparerAsc<T>());  }   public void SortDesc()  {  if (mas.Count < 1) return;  mas.Sort(new ComparerDesc<T>());  }   public void Rotate(int i)  {  if (i % mas.Count != 0)  {  if (i < 0)  {  i = Math.Abs(i);  while (i > mas.Count) i -= mas.Count;  mas = mas.Skip(i).Concat(mas.Take(i)).ToList();  }  else  {  while (i > mas.Count) i -= mas.Count;  for(int k = 0; k < i; k++)  {  T x=mas[mas.Count()-1];  for(int j=mas.Count()-1; j > 0; j--)  mas[j]=mas[j-1];  mas[0]=x;  }  }  }  }   public void Replace(T obj, int i)  {  mas[i] = obj;  }   public void Reverse()  {  mas.Reverse();  }   public void Remove(int i)  {  mas.RemoveAt(i);  }   public void Fill(T value)  {  for (int i = 0; i < mas.Count(); i++) mas[i] = value;  }   public int Count()  {  return mas.Count();  }    public void Unique()  {  if (mas.Count < 2) return;  var newStack = new List<T>();  foreach (var item in mas)  {  if (!newStack.Contains(item))  {  newStack.Add(item);  }  }  mas = newStack;  }   public void Clear()  {  mas.Clear();  }   public bool isEmpty()  {  return mas.Count() == 0;  }  } }  public class ComparerAsc<T> : IComparer<T> {  int IComparer<T>.Compare(T inX, T inY)  {  dynamic dx = inX, dy = inY;  if (dx < dy)  {  return 1;  }  if (dx > dy)  {  return -1;  }  return 0;  } }  public class ComparerDesc<T> : IComparer<T> {  int IComparer<T>.Compare(T inX, T inY)  {  dynamic dx = inX, dy = inY;  if (dx < dy)  {  return -1;  }  if (dx > dy)  {  return 1;  }  return 0;  } } Program.cs using System; using System.Collections.Generic; using System.Linq; using System.Text; using System.Threading.Tasks;  namespace DEQUE {  class Program  {  //Template for integer value  static Deque<int> deque = new Deque<int>();   static void Input()  {  Console.Clear();  Console.Write("Count = ");  int n = Convert.ToInt32(Console.ReadLine());  for(int i = 0; i < n; i++)  {  Console.Write("mas[" + (i + 1) + "]=");  deque.push\_e(Convert.ToInt32(Console.ReadLine()));  }  Console.WriteLine("Ok!");  Console.ReadLine();  }   static void Push\_f()  {  List<int> a = deque.getList();  Console.Clear();  Console.WriteLine("Old mas: ");  for(int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i]);  Console.Write("Push input element: ");  deque.push\_f(Convert.ToInt32(Console.ReadLine()));  a = deque.getList();  Console.WriteLine("New mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i]);  Console.ReadLine();  }   static void Push\_e()  {  List<int> a = deque.getList();  Console.Clear();  Console.WriteLine("Old mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i]);  Console.Write("Push input element: ");  deque.push\_e(Convert.ToInt32(Console.ReadLine()));  a = deque.getList();  Console.WriteLine("New mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i]);  Console.ReadLine();  }   static void Pop\_e()  {  List<int> a = deque.getList();  Console.Clear();  Console.WriteLine("Old mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i]);  Console.WriteLine("Poped last element " + deque.pop\_e());  a = deque.getList();  Console.WriteLine("New mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i]);  Console.ReadLine();  }   static void Pop\_f()  {  List<int> a = deque.getList();  Console.Clear();  Console.WriteLine("Old mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i]);  Console.WriteLine("Poped first element " + deque.pop\_f());  a = deque.getList();  Console.WriteLine("New mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i]);  Console.ReadLine();  }   static void Sort()  {  List<int> a = deque.getList();  Console.Clear();  Console.WriteLine("Old mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i]);  deque.SortDesc();  a = deque.getList();  Console.WriteLine("New mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i]);  Console.ReadLine();  }   static void Rotate()  {  List<int> a = deque.getList();  Console.Clear();  Console.WriteLine("Old mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i]);  Console.Write("Number of rotates: ");  deque.Rotate(Convert.ToInt32(Console.ReadLine()));  a = deque.getList();  Console.WriteLine("New mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i]);  Console.ReadLine();  }   static void Replace()  {  List<int> a = deque.getList();  Console.Clear();  Console.WriteLine("Old mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i]);  Console.Write("Position replacement: ");  int k = Convert.ToInt32(Console.ReadLine());  k--;  Console.Write("Replace number: ");  deque.Replace(Convert.ToInt32(Console.ReadLine()), k);  a = deque.getList();  Console.WriteLine("New mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i]);  Console.ReadLine();  }   static void Reverse()  {  List<int> a = deque.getList();  Console.Clear();  Console.WriteLine("Old mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i]);  deque.Reverse();  a = deque.getList();  Console.WriteLine("New mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i]);  Console.ReadLine();  }   static void Remove()  {  List<int> a = deque.getList();  Console.Clear();  Console.WriteLine("Old mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i]);  Console.Write("Remove position: ");  deque.Remove(Convert.ToInt32(Console.ReadLine()) - 1);  a = deque.getList();  Console.WriteLine("New mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i]);  Console.ReadLine();  }   static void Fill()  {  List<int> a = deque.getList();  Console.Clear();  Console.WriteLine("Old mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i]);  Console.Write("Fill value: ");  deque.Fill(Convert.ToInt32(Console.ReadLine()));  a = deque.getList();  Console.WriteLine("New mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i]);  Console.ReadLine();  }   static void Unique()  {  List<int> a = deque.getList();  Console.Clear();  Console.WriteLine("Number of array elements: " + deque.Count());  Console.WriteLine("Old mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i]);  deque.Unique();  a = deque.getList();  Console.WriteLine("New mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i]);  Console.ReadLine();  }   static void isEmpty()  {  List<int> a = deque.getList();  Console.Clear();  Console.WriteLine("isEmpty: " + deque.isEmpty());  Console.WriteLine("Old mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i]);  deque.Clear();  a = deque.getList();  Console.WriteLine("isEmpty: " + deque.isEmpty());  Console.WriteLine("New mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i]);  Console.ReadLine();  }   //Template for Time value  static Deque<Time> dequeTime = new Deque<Time>();   static void TInput()  {  Console.Clear();  Console.Write("Count = ");  int n = Convert.ToInt32(Console.ReadLine());  Console.WriteLine("Enter the time in the format (hh:mm:ss):");  for (int i = 0; i < n; i++)  {  Console.Write("mas[" + (i + 1) + "]=");  string sTime = Console.ReadLine();  dequeTime.push\_e(new Time(  Convert.ToInt32(sTime.Substring(0, 2)),  Convert.ToInt32(sTime.Substring(3, 2)),  Convert.ToInt32(sTime.Substring(6, 2))  ));  }  Console.WriteLine("Ok!");  Console.ReadLine();  }   static void TPush\_f()  {  List<Time> a = dequeTime.getList();  Console.Clear();  Console.WriteLine("Old mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i].getAll());  Console.Write("Push input element: ");  string sTime = Console.ReadLine();  dequeTime.push\_f(new Time(  Convert.ToInt32(sTime.Substring(0, 2)),  Convert.ToInt32(sTime.Substring(3, 2)),  Convert.ToInt32(sTime.Substring(6, 2))  ));  a = dequeTime.getList();  Console.WriteLine("New mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i].getAll());  Console.ReadLine();  }   static void TPush\_e()  {  List<Time> a = dequeTime.getList();  Console.Clear();  Console.WriteLine("Old mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i].getAll());  Console.Write("Push input element: ");  string sTime = Console.ReadLine();  dequeTime.push\_e(new Time(  Convert.ToInt32(sTime.Substring(0, 2)),  Convert.ToInt32(sTime.Substring(3, 2)),  Convert.ToInt32(sTime.Substring(6, 2))  ));  a = dequeTime.getList();  Console.WriteLine("New mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i].getAll());  Console.ReadLine();  }   static void TPop\_e()  {  List<Time> a = dequeTime.getList();  Console.Clear();  Console.WriteLine("Old mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i].getAll());  Console.WriteLine("Poped last element " + dequeTime.pop\_e().getAll());  a = dequeTime.getList();  Console.WriteLine("New mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i].getAll());  Console.ReadLine();  }   static void TPop\_f()  {  List<Time> a = dequeTime.getList();  Console.Clear();  Console.WriteLine("Old mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i].getAll());  Console.WriteLine("Poped first element " + dequeTime.pop\_f().getAll());  a = dequeTime.getList();  Console.WriteLine("New mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i].getAll());  Console.ReadLine();  }   static void TSort()  {  List<Time> a = dequeTime.getList();  Console.Clear();  Console.WriteLine("Old mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i].getAll());  dequeTime.SortDesc();  a = dequeTime.getList();  Console.WriteLine("New mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i].getAll());  Console.ReadLine();  }   static void TRotate()  {  List<Time> a = dequeTime.getList();  Console.Clear();  Console.WriteLine("Old mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i].getAll());  Console.Write("Number of rotates: ");  dequeTime.Rotate(Convert.ToInt32(Console.ReadLine()));  a = dequeTime.getList();  Console.WriteLine("New mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i].getAll());  Console.ReadLine();  }    static void TReplace()  {  List<Time> a = dequeTime.getList();  Console.Clear();  Console.WriteLine("Old mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i].getAll());  Console.Write("Position replacement: ");  int k = Convert.ToInt32(Console.ReadLine());  k--;  Console.Write("Replace time: ");  string sTime = Console.ReadLine();  dequeTime.Replace(new Time(  Convert.ToInt32(sTime.Substring(0, 2)),  Convert.ToInt32(sTime.Substring(3, 2)),  Convert.ToInt32(sTime.Substring(6, 2))  ), k);  a = dequeTime.getList();  Console.WriteLine("New mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i].getAll());  Console.ReadLine();  }   static void TReverse()  {  List<Time> a = dequeTime.getList();  Console.Clear();  Console.WriteLine("Old mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i].getAll());  dequeTime.Reverse();  a = dequeTime.getList();  Console.WriteLine("New mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i].getAll());  Console.ReadLine();  }   static void TRemove()  {  List<Time> a = dequeTime.getList();  Console.Clear();  Console.WriteLine("Old mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i].getAll());  Console.Write("Remove position: ");  dequeTime.Remove(Convert.ToInt32(Console.ReadLine()) - 1);  a = dequeTime.getList();  Console.WriteLine("New mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i].getAll());  Console.ReadLine();  }   static void TFill()  {  List<Time> a = dequeTime.getList();  Console.Clear();  Console.WriteLine("Old mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i].getAll());  Console.Write("Fill time: ");  string sTime = Console.ReadLine();  dequeTime.Fill(new Time(  Convert.ToInt32(sTime.Substring(0, 2)),  Convert.ToInt32(sTime.Substring(3, 2)),  Convert.ToInt32(sTime.Substring(6, 2))  ));  a = dequeTime.getList();  Console.WriteLine("New mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i].getAll());  Console.ReadLine();  }   static void TUnique()  {  List<Time> a = dequeTime.getList();  Console.Clear();  Console.WriteLine("Number of array elements: " + dequeTime.Count());  Console.WriteLine("Old mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i].getAll());  dequeTime.Unique();  a = dequeTime.getList();  Console.WriteLine("New mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i].getAll());  Console.ReadLine();  }   static void TisEmpty()  {  List<Time> a = dequeTime.getList();  Console.Clear();  Console.WriteLine("isEmpty: " + dequeTime.isEmpty());  Console.WriteLine("Old mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i].getAll());  dequeTime.Clear();  a = dequeTime.getList();  Console.WriteLine("isEmpty: " + dequeTime.isEmpty());  Console.WriteLine("New mas: ");  for (int i = 0; i < a.Count(); i++) Console.WriteLine("a[" + (i + 1) + "]=" + a[i].getAll());  Console.ReadLine();  }   static void menuInt()  {  bool b = true;  while (b)  {  Console.Clear();  Console.WriteLine("Int");  Console.WriteLine("1 - Input elements");  Console.WriteLine("2 - push\_f()");  Console.WriteLine("3 - push\_e()");  Console.WriteLine("4 - pop\_f()");  Console.WriteLine("5 - pop\_e()");  Console.WriteLine("6 - sort()");  Console.WriteLine("7 - rotate()");  Console.WriteLine("8 - replace()");  Console.WriteLine("9 - reverse()");  Console.WriteLine("10 - remove()");  Console.WriteLine("11 - fill()");  Console.WriteLine("12 - unique(), count()");  Console.WriteLine("13 - is\_empty(), clear()");  Console.WriteLine("0 - Exit");  Console.Write("Your choise: ");  try  {  int choice = Convert.ToInt32(Console.ReadLine());  switch (choice)  {  case 1: Input(); break;  case 2: Push\_f(); break;  case 3: Push\_e(); break;  case 4: Pop\_f(); break;  case 5: Pop\_e(); break;  case 6: Sort(); break;  case 7: Rotate(); break;  case 8: Replace(); break;  case 9: Reverse(); break;  case 10: Remove(); break;  case 11: Fill(); break;  case 12: Unique(); break;  case 13: isEmpty(); break;  case 0: b = false; break;  default:  Console.Clear();  Console.WriteLine("Wrong operation!");  Console.ReadLine();  break;  }  }  catch  {  Console.Clear();  Console.WriteLine("Wrong operation!");  Console.ReadLine();  }  }  }   static void menuTime()  {  bool b = true;  while (b)  {  Console.Clear();  Console.WriteLine("Time");  Console.WriteLine("1 - Input elements");  Console.WriteLine("2 - push\_f()");  Console.WriteLine("3 - push\_e()");  Console.WriteLine("4 - pop\_f()");  Console.WriteLine("5 - pop\_e()");  Console.WriteLine("6 - sort()");  Console.WriteLine("7 - rotate()");  Console.WriteLine("8 - replace()");  Console.WriteLine("9 - reverse()");  Console.WriteLine("10 - remove()");  Console.WriteLine("11 - fill()");  Console.WriteLine("12 - unique(), count()");  Console.WriteLine("13 - is\_empty(), clear()");  Console.WriteLine("0 - Exit");  Console.Write("Your choise: ");  try  { int choice = Convert.ToInt32(Console.ReadLine());  switch (choice)  { case 1: TInput(); break;  case 2: TPush\_f(); break;  case 3: TPush\_e(); break;  case 4: TPop\_f(); break;  case 5: TPop\_e(); break;  case 6: TSort(); break;  case 7: TRotate(); break;  case 8: TReplace(); break;  case 9: TReverse(); break;  case 10: TRemove(); break;  case 11: TFill(); break;  case 12: TUnique(); break;  case 13: TisEmpty(); break;  case 0: b = false; break;  default:  Console.Clear();  Console.WriteLine("Wrong operation!");  Console.ReadLine();  break;  } } catch  { Console.Clear();  Console.WriteLine("Wrong operation!");  Console.ReadLine();  } } } static void Main(string[] args)  { bool b = true;  while(b)  { Console.Clear();  Console.WriteLine("1 - Int");  Console.WriteLine("2 - Time");  Console.WriteLine("0 - Exit");  Console.Write("Your choise: ");  try {  int choice = Convert.ToInt32(Console.ReadLine());  switch (choice)  { case 1: menuInt(); break;  case 2: menuTime(); break;  case 0: b = false; break;  default:  Console.Clear();  Console.WriteLine("Wrong operation!");  Console.ReadLine();  break;  } } catch {  Console.Clear();  Console.WriteLine("Wrong operation!");  Console.ReadLine();  } } } }} **Скриншоты работы программы**





**Вывод**

На данной лабораторной работе я закрепил теоретические знания, а также приобрел практические навыки по реализации шаблонов в программировании.