МИНИСТРЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ

ХАРЬКОВСКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ

РАДИОЭЛЕКТРОНИКИ

КАФЕДРА ИИ

Отчет

О выполнении лабораторной работы № 2

«Перегрузка стандартных операций и операций приведения типов»

По дисциплине «Программирование на языке С#»

Вариант № 31

Выполнил ст. гр. ІТШІ-19-2: Принял:

Меденицкий Алексей Бибичков И. Е.

2020

**Цель работы**

Изучение технологии перегрузки стандартных операций и операций явного и неявного приведения типов для пользовательских классов на языке С#. Изучение особенностей создания и использования библиотек классов в среде VS.NET

**Задание №31 «Кольцо объектов конкретного типа»**

Данные класса: указатель на текущий элемент кольца в динамическом списке элементов кольца.

Операции: считывание без извлечения элемента кольца, считывание с извлечением элемента кольца (“>”), запись элемента в кольцо (“<”), перемещение указателя по и против часовой стрелки (операции “++” и “--”).

Включить в реализацию класса конструкторы всех типов, функцию ввода, переопределить функцию ToString, вместо функций доступа и инициализации использовать свойства класса.

Для всех вариантов включить в класс функции, перегружающие операции true и false, дать интерпретацию и перегрузить операции явного и неявного приведения типов.

Для классов, владеющим списком или динамическим массивом своих элементов реализовать индексаторы.

Также для всех вариантов перегрузить операции “==” и “!=“ для сравнения объектов на равенство и на неравенство.

Указания к выполнению заданий этой группы: обязательно включить в класс конструктор, использующий в качестве одного из параметров одномерный или двумерный массив чисел, при этом конструктор преобразования реализовывать не нужно.

**Интерфейс класса**

using System;

namespace RingLib

{

public class Ring

{

private class Node

{

public int Data {get; set;}

public Node Next {get; set;}

public Node Prev {get; set;}

}

private Node Begin {get; set;}

public Ring();

public Ring(Ring copy);

public Ring(int[] a);

public static explicit operator Ring(int[] a);

public static explicit operator int[](Ring r);

public int Size();

public static int operator>>(Ring r, int n);

public static int operator<(Ring r, int n);

public static int operator>(Ring r, int n);

public static Ring operator++(Ring r);

public static Ring operator--(Ring r);

public void Input();

public override string ToString();

public override bool Equals(object obj);

public override int GetHashCode();

public int this[int i];

public static bool operator==(Ring a, Ring b);

public static bool operator!=(Ring a, Ring b);

public static bool operator true(Ring r);

public static bool operator false(Ring r);

}

} // namespace RingLib

**Реализация класса**

using System;

namespace RingLib

{

public class Ring

{

private class Node

{

public int Data {get; set;}

public Node Next {get; set;}

public Node Prev {get; set;}

}

private Node Begin {get; set;}

public Ring()

{

}

public Ring(Ring copy)

{

if (copy.Begin == null)

return;

Node t = new Node();

Node c = copy.Begin;

t.Data = c.Data;

t.Next = t;

t.Prev = t;

Begin = t;

while (c.Next != copy.Begin) {

c = c.Next;

t = new Node();

t.Data = c.Data;

t.Next = Begin;

t.Prev = Begin.Prev;

Begin.Prev.Next = t;

Begin.Prev = t;

}

}

public Ring(int[] a)

{

if (a.Length == 0)

return;

Node t = new Node();

t.Data = a[0];

t.Next = t;

t.Prev = t;

Begin = t;

for (int i = 1; i < a.Length; i++) {

t = new Node();

t.Data = a[i];

t.Next = Begin;

t.Prev = Begin.Prev;

Begin.Prev.Next = t;

Begin.Prev = t;

}

}

public static explicit operator Ring(int[] a)

{

Ring r = new Ring(a);

return r;

}

public static implicit operator int[](Ring r)

{

int N = r.Size();

int[] a = new int[N];

Node t = r.Begin;

for (int i = 0; i < N; i++) {

a[i] = t.Data;

t = t.Next;

}

return a;

}

public int Size()

{

int size = 0;

if (Begin == null)

return size;

Node t = Begin;

do {

size++;

t = t.Next;

} while (t != Begin);

return size;

}

public static int operator>>(Ring r, int n)

{

if (Begin == null)

throw new NullReferenceException();

return Begin.Data;

}

public static int operator<(Ring r, int n)

{

Node t = new Node();

t.Data = n;

if (r.Begin == null) {

t.Next = t;

t.Prev = t;

r.Begin = t;

} else {

t.Next = r.Begin;

t.Prev = r.Begin.Prev;

r.Begin.Prev.Next = t;

r.Begin.Prev = t;

}

return n;

}

public static int operator>(Ring r, int n)

{

if (r.Begin == null)

throw new NullReferenceException();

int res = r.Begin.Data;

if (r.Begin == r.Begin.Next) {

r.Begin = null;

} else {

r.Begin.Prev.Next = r.Begin.Next;

r.Begin.Next.Prev = r.Begin.Prev;

r.Begin = r.Begin.Next;

}

return res;

}

public static Ring operator++(Ring r)

{

if (r.Begin == null)

throw new NullReferenceException();

r.Begin = r.Begin.Next;

return r;

}

public static Ring operator--(Ring r)

{

if (r.Begin == null)

throw new NullReferenceException();

r.Begin = r.Begin.Prev;

return r;

}

public void Input()

{

int N = int.Parse(Console.ReadLine());

int n;

for (int i = 0; i < N; i++) {

n = (int)Convert.ChangeType(Console.ReadLine(), typeof(int));

n = this < n;

}

}

public override string ToString()

{

string s = "{ ";

Node t = Begin;

do {

s += $"{t.Data}, ";

t = t.Next;

} while (t != Begin);

s = s.Remove(s.Length-2);

s += " }";

return s;

}

public override bool Equals(object obj) {

if (obj == null || GetType() != obj.GetType())

return false;

Ring r = (Ring)obj;

Node tnode = Begin;

Node rnode = r.Begin;

if (tnode == null || rnode == null)

return false;

do {

if (!tnode.Data.Equals(rnode.Data))

return false;

tnode = tnode.Next;

rnode = rnode.Next;

} while (tnode != Begin && rnode != r.Begin);

if (tnode != Begin || rnode != r.Begin)

return false;

return true;

}

public override int GetHashCode()

{

return base.GetHashCode();

}

public int this[int i]

{

get

{

if (Begin == null)

throw new NullReferenceException();

Node t = Begin;

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

t = t.Next;

return t.Data;

}

set

{

if (Begin == null)

throw new NullReferenceException();

Node t = Begin;

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

t = t.Next;

t.Data = value;

}

}

public static bool operator==(Ring a, Ring b)

{

return Equals(a, b);

}

public static bool operator!=(Ring a, Ring b)

{

return !Equals(a, b);

}

public static bool operator true(Ring r)

{

return r.Begin != null;

}

public static bool operator false(Ring r)

{

return r.Begin == null;

}

}

} // namespace RingLib

**Исходник функции main консольного приложения**

using System;

using RingLib;

namespace RingConsole

{

class Program

{

static void Main(string[] args)

{

Ring a = new Ring();

int n;

n = a < 3;

n = a < 4;

n = a < 5;

a++;

a--;

n = a > n;

Console.WriteLine(n);

Ring b = new Ring(a);

n = a > n;

n = b > n;

Console.WriteLine(n);

n = b >> n;

Console.WriteLine(n);

Ring c = new Ring(new int[6]{1, 2, 3, 4, 5, 6});

n = c > n;

Console.WriteLine(n);

Ring d = new Ring();

d.Input();

Console.WriteLine(d);

Ring e = new Ring(d);

Console.WriteLine(e);

Console.WriteLine(e == d);

Console.WriteLine(a != e);

if (e)

Console.WriteLine("E is full");

else

Console.WriteLine("E is empty");

e[1] = 148;

for (int i = 0; i < e.Size(); i++)

Console.Write("{0} ", e[i]);

Console.WriteLine();

Console.WriteLine("size of e is {0}", e.Size());

int[] arr = e;

foreach (var i in arr)

Console.Write("{0} ", i);

Console.WriteLine(" ");

int[] arr2 = {13, 12, 16};

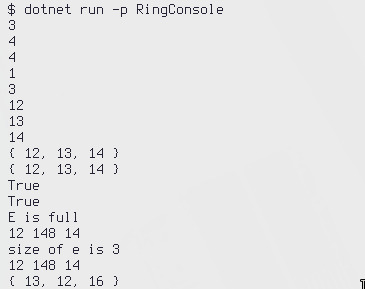
Console.WriteLine((Ring)arr2);

}

}

}

**Скрины результатов работы программы**

****

**Выводы по работе**

В результате выполнения лабораторной работы были изучены технологии перегрузки стандартных операций и операций явного и неявного приведения типов для пользовательских классов на языке С#, а также особенности создания и использования библиотек классов в среде VS.NET Была разработана библиотека RingLib с классом Ring реализующим кольцо чисел связанных двусвязным списком.