Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Кафедра инженерной психологии и эргономики

­­­­­

Конструирование программного обеспечения

Отчет по лабораторной работе №5

«Обобщения и шаблоны»

Выполнил: Усов А.М.

Студент группы 310901

Преподаватель: Давыдович К. И.

Минск 2023

Цель: получить навыки создания обобщённых типов. Изучить шаблоны.

Задание 1. Построить шаблонный класс, который будет описывать элемент хранимых данных, доступ к ним, сравнение элементов и т.п. по необходимости.

Построить контейнерный шаблонный класс операций над элементами данных, включающий операции:

* + добавления;
  + удаления;
  + поиска;
  + просмотра;
  + сортировки элементов;
  + перестановки элементов в обратном порядке;
  + замены всех подобных элементов по заданному ключу;
  + поиска максимального элемента;
  + остальные функции добавлять по необходимости.

Для данного контейнерного класса предусмотреть при формировании элемента задание режима уникальных элементов (т.е. проверку на дублирование значений элементов).

Для обработки всех ошибочных ситуаций использовать конструкцию try…catch().

Дополнительно к контейнеру рекомендуется реализовать класс-итератор.

В Main создать три экземпляра шаблонного класса-контейнера для разных типов данных. Работа с этими объектами должна демонстрироваться на следующих операциях: добавить – просмотреть – найти – удалить – найти – просмотреть.

Отладить и выполнить полученную программу. Проверить обработку исключительных ситуаций (например, чтение из пустого стека, дублирование объектов и т.п.).

-Кольцо двусвязанное

Код программы 1:

using System;  
  
class Program  
{  
 public static void Main(string[] args)  
 {  
 Ring<int> ringInt = new Ring<int>();  
 Ring<string> ringString = new Ring<string>();  
 Ring<double> ringDouble = new Ring<double>();  
  
  
 Console.WriteLine("Work with Ring int model:\n add - veiw - find - remove - find - veiw");  
 *// add* ringInt.push(1);  
 ringInt.push(6);  
 ringInt.push(2);  
 *// view* ringInt.Print();  
 *// find* Console.WriteLine("Find:");  
 ringInt.Get(6).Print();  
 *// remove* ringInt.Delete(6);  
 *// find* Console.WriteLine("Find: ");  
 ringInt.Get(6).Print();  
 *// view* Console.WriteLine("");  
 ringInt.Print();  
   
 Console.WriteLine("Work with Ring string model:\n add - veiw - find - remove - find - veiw");  
  
 *// add* ringString.push("Hello");  
 ringString.push("World");  
 ringString.push("!!");  
 *// view* ringString.Print();  
 *// find* Console.WriteLine("Find:");  
 ringString.Get("World")?.Print();  
 ringString.Get(1).Print();  
 *// remove* ringString.Delete("World");  
 *// find* Console.WriteLine("Find: ");  
 ringString.Get("World")?.Print();  
 *// view* Console.WriteLine("");  
 ringString.Print();  
  
 Console.WriteLine("Work with Ring double model:\n add - veiw - find - remove - find - veiw");  
  
 *// add* ringDouble.push(1.1);  
 ringDouble.push(2.2);  
 ringDouble.push(3.3);  
 *// view* ringDouble.Print();  
 *// find* Console.WriteLine("Find:");  
 ringDouble.Get(2.2).Print();  
 *// remove* ringDouble.Delete(2.2);  
 *// find* Console.WriteLine("Find: ");  
  
 ringDouble.Get(2.2)?.Print();  
 *// view* Console.WriteLine("");  
 ringDouble.Print();  
  
  
  
  
 Console.WriteLine("===============================================");  
 ringInt.Print();  
 Console.WriteLine("Sort");  
 ringInt.Sort();  
 ringInt.Print();  
 Console.WriteLine("Reverse");  
 ringInt.Reverse();  
 ringInt.Print();  
 Console.WriteLine("Max");  
 ringInt.Max().Print();  
 Console.WriteLine("Replace");  
 ringInt.Replace(1, 10);  
 ringInt.Print();  
 Console.WriteLine("Delete");  
 ringInt.Delete(10);  
 ringInt.Print();  
  
  
  
  
  
 Console.WriteLine("===============================================");  
 ringDouble.Print();  
 Console.WriteLine("Sort");  
 ringDouble.Sort();  
 ringDouble.Print();  
 Console.WriteLine("Reverse");  
 ringDouble.Reverse();  
 ringDouble.Print();  
 Console.WriteLine("Max");  
 ringDouble.Max().Print();  
 Console.WriteLine("Replace");  
 ringDouble.Replace(1, 10);  
 ringDouble.Print();  
 Console.WriteLine("Delete");  
 ringDouble.Delete(10);  
 ringDouble.Print();  
  
 Console.WriteLine("===============================================");  
 ringString.Print();  
 Console.WriteLine("Sort");  
 ringString.Sort();  
 ringString.Print();  
 Console.WriteLine("Reverse");  
 ringString.Reverse();  
 ringString.Print();  
 Console.WriteLine("Max");  
 ringString.Max().Print();  
 Console.WriteLine("Replace");  
 ringString.Replace("Hello", "Hi");  
 ringString.Print();  
 Console.WriteLine("Delete");  
 ringString.Delete("Hi");  
 ringString.Print();  
  
  
  
  
  
 }  
}  
  
class Ring<T> where T: IComparable<T>  
{  
 internal class Node  
 {  
 private T data;  
 private Node Next;  
 private Node Prev;  
  
 public Node(T value)  
 {  
 this.data = value;  
 this.Next = null;  
 this.Prev = null;  
 }  
  
 public Node(T value, Node next, Node prev)  
 {  
 this.data = value;  
 this.Next = next;  
 this.Prev = prev;  
 }  
  
 public T getData()  
 {  
 return data;  
 }  
  
 public Node getNext()  
 {  
 return Next;  
 }  
  
 public Node getPrev()  
 {  
 return Prev;  
 }  
  
 public void setNext(Node Next)  
 {  
 this.Next = Next;  
 }  
  
 public void setPrev(Node Prev)  
 {  
 this.Prev = Prev;  
 }  
 public void setData(T data)  
 {  
 this.data = data;  
 }  
 public void Print()  
 {  
 Console.WriteLine(data);  
 }  
 }  
  
 private Node Head;  
 private Node Tail;  
 private int size;  
  
 public Ring()  
 {  
 this.size = 0;  
 this.Head = null;  
 this.Tail = null;  
 }  
  
 public int getSize()  
 {  
 return this.size;  
 }  
  
 public void push(T value)  
 {  
 bool hasDuplicates = FindDuplicates(value);  
  
 if (hasDuplicates)  
 return;  
  
 if (size == 0)  
 {  
 Head = new Node(value);  
 Tail = Head;  
 Head.setNext(Tail);  
 Head.setPrev(Tail);  
 }  
 else  
 {  
 Node newNode = new Node(value, Head, Tail);  
 Tail.setNext(newNode);  
 Head.setPrev(newNode);  
 Tail = newNode;  
 }  
 size++;  
 }  
  
 private bool FindDuplicates(T value)  
 {  
   
 Node current = Head;  
 for (int i = 0; i < size; i++)  
 {  
 if (current.getData().CompareTo(value) == 0)  
 return true;  
  
 current = current.getNext();  
 }  
 return false;  
   
 }  
  
 public void Print()  
 {  
 Node current = Head;  
 for(int i = 0; i < size; i++)  
 {  
 Console.WriteLine(current.getData());  
 current = current.getNext();  
   
 }  
 }  
  
 public void Sort()  
 {  
   
 for (int i = 0; i < size; i++)  
 {  
 Node current = Head;  
  
 for (int j = 0; j < size - i - 1; j++)  
 {  
 if (current.getData().CompareTo(current.getNext().getData()) > 0)  
 {  
 T temp = current.getData();  
 current.setData(current.getNext().getData());  
 current.getNext().setData(temp);  
 }  
 current = current.getNext();  
 }  
  
 }  
 }  
  
  
 public void Reverse()  
 {  
 Node current = Head;  
 for(int i = 0; i < size; i++)  
 {  
 Node temp = current.getPrev();  
 current.setPrev(temp.getNext().getNext());  
 current.setNext(temp);  
 temp = null;  
 current = current.getNext();  
 }  
  
 Node temp1 = Head;  
 Head = Tail;  
 Tail = temp1;  
 temp1 = null;  
 }  
 public void Delete(T value)  
 {  
   
 Node current = Head;  
 for (int i = 0; i < size; i++)  
 {  
 if (current.getData().CompareTo(value) == 0)  
 {  
 current.getPrev().setNext(current.getNext());  
 current.getNext().setPrev(current.getPrev());  
 current = null;  
  
  
 if (current == Head)  
 {  
 Head = current.getNext();  
 }  
 if (current == Tail)  
 {  
 Tail = current.getPrev();  
 }  
  
 size--;  
 return;  
 }  
 current = current.getNext();  
 }  
   
 }  
 public void Delete(int index)  
 {  
 Node node = Head;  
 for(int i = 0; i < index; i++)  
 {  
 node = node.getNext();  
 }  
  
 node.getPrev().setNext(node.getNext());  
 node.getNext().setPrev(node.getPrev());  
   
 if (node == Head)  
 {  
 Head = node.getNext();  
 }  
 if (node == Tail)  
 {  
 Tail = node.getPrev();  
 }  
 node = null;   
  
 size--;  
  
 }  
  
  
 public Node Get(T value)  
 {  
   
 Node current = Head;  
 for (int i = 0; i < size; i++)  
 {  
 if (current.getData().CompareTo(value) == 0)  
 {  
 return current;  
 }  
 current = current.getNext();  
 }  
   
 return null;  
 }  
  
 public Node Get(int index)  
 {  
 Node current = Head;  
 for(int i = 0; i < index; i++)  
 {  
 current = current.getNext();  
 }  
 return current;  
  
 }  
  
 public Node Max()  
 {  
 Node current = Head;  
 Node max = current;  
 for (int i = 0; i < size; i++)  
 {  
 if(current.getData().CompareTo(max.getData()) > 0)  
 {  
 max = current;  
 }  
 current = current.getNext();  
  
 }  
 return max;  
 }  
  
  
 public void Replace(T valueOfReplace, T valueToReplace)  
 {  
 try  
 {  
 Node current = Head;  
 for(int i = 0; i < size; i++)  
 {  
 if(current.getData().CompareTo(valueOfReplace) == 0)  
 {  
 current.setData(valueToReplace);  
 }  
  
 current = current.getNext();  
  
 }  
  
 }catch(Exception e)  
 {  
 Console.WriteLine("May be invalid type\n" + e.Message);  
 }  
 }  
}

Результат работы программы представлен на рисунках 1.5 – 1.6

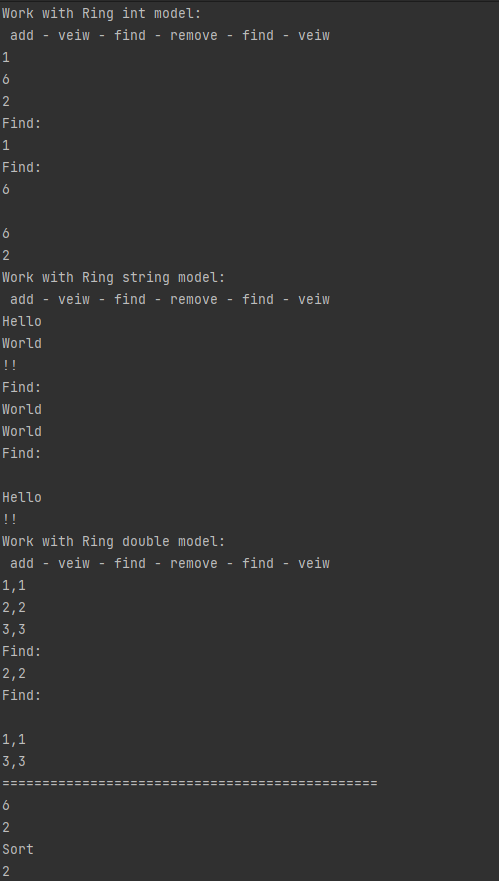


Рисунок 1.5 – Результат выполнения программы 1

# 

Рисунок 1.6 – Результат работы программы 1

Вывод: В результате выполнения пятой лабораторной работы по теме "Обобщения и шаблоны" были достигнуты ключевые цели, включающие в себя получение навыков создания обобщённых типов и изучение шаблонов. Эта работа позволила мне углубиться в концепцию обобщённых типов данных и их значимость в разработке программного обеспечения.

В ходе работы я освоил применение шаблонов, что стало важным шагом в моём профессиональном развитии. Использование обобщённых типов помогло мне создавать более универсальные и переиспользуемые компоненты, что, в свою очередь, повысило производительность и упростило поддержку моего кода.

Понимание различных типов шаблонов и их применение в различных ситуациях разработки также оказалось весьма ценным. Это позволило мне не только ознакомиться с новыми концепциями, но и успешно применить их на практике, создавая функциональные и эффективные программы.

Таким образом, выполнение этой лабораторной работы позволило мне расширить мои знания в области обобщений и шаблонов, а также приобрести практические навыки, которые будут полезны в моей дальнейшей профессиональной деятельности в области разработки программного обеспечения.