**Лабораторная работа №1**

**Математическое обеспечение ЭВМ**

**«Работа с таблицей идентификаторов»**

Солонин Егор

А-14-19

Вариант 14

**Задание:**

Разработать функции для включения описания идентификатора в таблицу, поиска по имени и исключения описания идентификатора из таблицы. Описание идентификатора состоит из имени идентификатора и его атрибута. Разработать программу, демонстрирующую использование указанных функций. Программа должна быть разработана как консольное приложение на языке C# в среде Visual Studio.

Индивидуальный вариант (№14): Способ представления таблицы – связанный список. Метод поиска – линейный поиск

**Описание работы программы:**

При открытии приложения пользователь имеет возможность выбрать один из пунктов меню:

1. Инициализация таблицы идентификаторов
2. Вывод таблицы на экран
3. Включение описания идентификатора в таблицу
4. Поиска по имени идентификатора
5. Исключение описания идентификатора из таблицы
6. Выйти из программы

Класс Identifier имеет два свойства:

* Name – имя идентификатора
* Attr – атрибут идентификатора

Класс LinkedListNode имеет:

* \_head (закрытое поле) – указатель на начало списка
* \_tail (закрытое поле) – указатель на конец списка
* Конструкторы
* Value (свойство) – описание идентификатора, тип – Identifier
* Next (свойство) – указатель на следующий элемент списка, тип - LinkedListNode
* Count (свойство) – позволяет получить или установить размер списка
* Add(метод) – добавляет описание идентификатора в конец списка
* Contains (метод) – поиска по имени идентификатора. Если идентификатор найден – возвращает true, иначе false
* Print (метод) – вывод списка идентификаторов на экран

При инициализации очереди пользователю предлагается ввести имя идентификатора и его атрибут. Для прекращения ввода, необходимо указать пустое имя идентификатора. Алгоритм инициализации полностью основан на алгоритме добавления элемента в список.

**Алгоритмы операций на псевдокоде:**

Добавление(Add):

1. В параметрах функции передается экземпляр класса Identifier, который добавляем в список
2. Создаем новый узел списка node
3. Если список пуст, присваиваем указателям головы и хвоста списка значение node
4. Иначе у хвоста списка, указателю на следующий элемент списка присваиваем значение node
5. Обновляем хвост списка, присваивая ему значение node
6. Увеличиваем размер списка

Поиск(Contains):

1. В параметрах функции передается имя идентификатора, который хотим найти – строка name
2. Копируем в переменную node голову списка
3. Пока node != null:
   1. Если имя идентификатора узла списка == name, тогда возвращаем true – элемент найден, выход
   2. Устанавливаем новое значение текущего узла, по ссылке Next
4. Возвращаем false – элемент не найден

Удаление(Remove):

1. В параметрах функции передается имя идентификатора, который хотим удалить – строка name
2. Устанавливаем предыдущий элемент списка prev = null
3. Устанавливаем текущий элемент списка curr = \_head
4. Пока curr != null:
   1. Если имя идентификатора узла списка == name
      1. Если prev != null, т.е. текущий узел не в начале списка, то prev.Next = curr.Next, то есть исключаем элемент curr из списка. Если curr был в конце списка – обновляем указатель на конец списка
      2. Иначе(если prev == null), то есть текущий узел в начале списка, то устанавливаем новое значение головы списка – следующий элемент. Если текущий узел был единственным в списке – имеем пустой список, обновляем указатели на голову и хвост \_head=\_tail=null
      3. Уменьшаем размер списка
      4. Элемент удален – возвращаем true
   2. Иначе (если имя идентификатора узла списка != name) переходим к следующему узлу(prev=curr; curr=curr.Next;)
5. Возвращаем false – элемент не найден, либо список был пуст

**Тесты работы программы:**

|  |  |
| --- | --- |
| Инициализация списка |  |
| Вывод таблицы на экран |  |
| Добавление описания идентификатора в конец списка |  |
| Удаление элемента из начала списка |  |
| Удаление элемента из середины списка |  |
| Удаление элемента из конца списка |  |
| Попытка удалить несуществующий элемент |  |
| Поиск элемента. Два случая – когда найден, и когда не найден |  |

**Листинг программы:**

using System;

using System.Threading;

namespace Labs {

public class Identifier {

public string Name { get; set; }

public int Attr { get; set; }

}

public class LinkedListNode {

private LinkedListNode \_head;

private LinkedListNode \_tail;

public LinkedListNode() {}

public LinkedListNode(Identifier value) {

Value = value;

}

public Identifier Value { get; internal set; }

public LinkedListNode Next { get; internal set; }

public int Count {

get;

private set;

}

public void Add(Identifier item) {

LinkedListNode node = new LinkedListNode(item);

if(\_head == null) {

\_head = node;

\_tail = node;

} else {

\_tail.Next = node;

\_tail = node;

}

Count++;

}

public bool Contains(string name) {

LinkedListNode node = \_head;

while(node != null) {

if(node.Value.Name == name)

return true;

node = node.Next;

}

return false;

}

public bool Remove(string name) {

LinkedListNode prev = null;

LinkedListNode curr = \_head;

while(curr != null) {

if(curr.Value.Name == name) {

if(prev != null) {

prev.Next = curr.Next;

if (curr.Next == null)

\_tail = prev;

} else {

\_head = \_head.Next;

if(\_head == null)

\_tail = null;

}

Count--;

return true;

}

prev = curr;

curr = curr.Next;

}

return false;

}

public void Print()

{

LinkedListNode node = \_head;

while(node != null) {

Console.WriteLine($"{node.Value.Name}, {node.Value.Attr}");

node = node.Next;

}

}

}

public class Menu {

public void ShowMenu() {

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("\n\*\*MENU\*\*\n");

Console.WriteLine("1. init new table of identifiers");

Console.WriteLine("2. print table");

Console.WriteLine("3. add item");

Console.WriteLine("4. find item");

Console.WriteLine("5. remove item");

Console.WriteLine("6. exit\n");

Console.ResetColor();

}

public void Init(LinkedListNode list) {

Console.ForegroundColor = ConsoleColor.DarkGreen;

Console.WriteLine("\n\*\*CREATING NEW TABLE OF IDENTIFIERS\*\*\n");

Console.WriteLine("Input items. To finish - input empty name of identifier \n");

while(true)

{

Console.Write("Name: ");

var name = Console.ReadLine();

if(name == "")

break;

Console.Write("Attribute(int): ");

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

Identifier id = new Identifier

{

Name = name,

Attr = attr

};

list.Add(id);

Console.WriteLine();

}

Console.WriteLine("\nNew table has been successfully created\n");

Thread.Sleep(3000);

Console.ResetColor();

}

public void Print(LinkedListNode list) {

Console.ForegroundColor = ConsoleColor.DarkGreen;

Console.WriteLine("\n\*\*PRINT TABLE\*\*");

if(list.Count == 0)

Console.WriteLine("\n\*---EMPTY---\*");

else

list.Print();

Thread.Sleep(3000);

Console.ResetColor();

}

public void Add(LinkedListNode list) {

Console.ForegroundColor = ConsoleColor.DarkGreen;

Console.WriteLine("\n\*\*ADD ITEM\*\*\n");

Console.WriteLine("Input item you wanna add\n");

Console.Write("Name: ");

var name = Console.ReadLine();

Console.Write("Attribute(int): ");

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

Identifier id = new Identifier {

Name = name,

Attr = attr

};

list.Add(id);

Console.WriteLine("\n\*\*SUCCESS\*\*\n");

Thread.Sleep(3000);

Console.ResetColor();

}

public void Remove(LinkedListNode list) {

Console.ForegroundColor = ConsoleColor.DarkGreen;

Console.WriteLine("\n\*\*REMOVE ITEM\*\*\n");

Console.WriteLine("Input name of item you wanna remove\n");

Console.Write("Name: ");

var name = Console.ReadLine();

Console.WriteLine(list.Remove(name) ? "\n\*\*SUCCESS\*\*\n" : "\n\*\*FAILED\*\*\n");

Thread.Sleep(3000);

Console.ResetColor();

}

public void Find(LinkedListNode list) {

Console.ForegroundColor = ConsoleColor.DarkGreen;

Console.WriteLine("\n\*\*FIND ITEM\*\*\n");

Console.WriteLine("Input name of item you wanna find\n");

Console.Write("Name: ");

var name = Console.ReadLine();

Console.WriteLine(list.Contains(name) ? "\n\*\*TABLE CONTAINS THIS ITEM\*\*\n" : "\n\*\*ITEM HAS NOT BEEN FOUND\*\*\n");

Thread.Sleep(3000);

Console.ResetColor();

}

}

internal class Program {

private static void Main() {

LinkedListNode list = new LinkedListNode();

var running = true;

while(running) {

var menu = new Menu();

menu.ShowMenu();

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

switch(option) {

case 1:

menu.Init(list);

break;

case 2:

menu.Print(list);

break;

case 3:

menu.Add(list);

break;

case 4:

menu.Find(list);

break;

case 5:

menu.Remove(list);

break;

case 6:

running = false;

break;

default:

Console.WriteLine("ERROR - INCORRECT OPTION");

break;

}

}

}

}

}