# Анализ задачи «Одномерные массивы»

1. Сформировать массив из n элементов. Предусмотреть формирование массива с клавиатуры и с помощью датчика случайных чисел (n задается пользователем с клавиатуры).
2. Распечатать массив.
3. Выполнить удаление указанных элементов из массива.
4. Выполнить добавление указанных элементов в массив.
5. Выполнить перестановку элементов в массиве.
6. Выполнить поиск указанных в массиве элементов и подсчитать количество сравнений, необходимых для поиска нужного элемента.
7. Выполнить сортировку массива указанным методом.
8. Выполнить поиск указанных элементов в отсортированном массиве и подсчитать количество сравнений, необходимых для поиска нужного элемента.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Вариант | Удаление | Добавление | Перестановка | Поиск | Сортировка |
| ХХ | K элементов, начиная с номера N | К элементов , начиная с номера N | Перевернуть массив | Первый элемента с заданным ключом | Простой обмен |

Анализ данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Описание функции** | **Входные данные** | **Классы входных данных** | **Выходные данные** |
| **1** | Сформировать массив из n элементов с помощью датчика случайных чисел (n задается пользователем с клавиатуры). | Количество элементов массива size | Целое число> 0 | Выделена память под size элементов массива |
| Целое число <=0 | Сообщение об ошибке  Массив пустой |
| Символ |
| Вещественное число |
| **2.** | Заполнение массива с помощью ДСЧ | Ссылка на массив | Память не выделена | Сообщение об ошибке |
|  | Заполненный массив |
| **3.** | Заполнение массива с клавиатуры | Ссылка массив  Число, которое пользователь вводит с клавиатуры | Память не выделена | Сообщение об ошибке |
| Целое число | Заполненный массив |
| Символ | Сообщение об ошибке, повторный ввод |
| Вещественное число |
| **4** | Распечатать массив | Ссылка на массив | Массив пустой | Сообщение об ошибке |
| Массив не пустой | Массив напечатан |
| **5** | Удалить K элементов, начиная с номера N | Ссылка на массив | Массив пустой | Сообщение об ошибке |
| Номер, с которого начинается удаление | N<0 или N>=size |
| N символ |
| N вещественное число |
| Количество удаляемых из массива элементов | N+K>size |
| K<=0 |
| К вещественное число или символ |
| N>=0 && N<size  K>0&& N+K<size | Измененный массив |
| N=0&&K=size | Пустой массив |
| **6** | Добавить К элементов , начиная с номера N | Ссылка на массив | Массив пустой | Сообщение об ошибке |
| Номер, с которого начинается добавление | N<0 или N>size |
| N символ |
| N вещественное число |
| Количество добавляемых элементов | K<=0 |
| К вещественное число или символ |
| K>0 | Измененный массив |
| **7** | Перевернуть массив | Ссылка на массив | Массив пустой | Сообщение об ошибке |
| Массив не пустой | Измененный массив |
| **8** | Выполнить поиск указанных в массиве элементов и подсчитать количество сравнений, необходимых для поиска нужного элемента. | Ссылка на массив,  Элемент Х | Массив пустой | Сообщение об ошибке |
| Х символ или вещественное число |
| Массив не пустой, число Х есть в массиве | Номер позиции Х, количество сравнений |
| Массив не пустой, числа Х нет в массиве | Нет элемента, количество сравнений |
| **9** | Выполнить сортировку массива указанным методом | Ссылка на массив | Массив пустой | Сообщение об ошибке |
| Массив не пустой | Измененный массив |
| **10** | Выполнить поиск указанных элементов в отсортированном массиве и подсчитать количество сравнений, необходимых для поиска нужного элемента. | Ссылка на массив,  Элемент Х | Массив пустой | Сообщение об ошибке |
| Массив не отсортирован |
| Х символ или вещественное число |
| Массив не пустой, число Х есть в массиве | Номер позиции Х, количество сравнений |
| Массив не пустой, числа Х нет в массиве | Нет элемента, количество сравнений |

Проектирование

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Имя функции** | **Входные данные** | **Выходные данные** | **Описание функции** | **Примечание** |
| **1** | MakeArray | //количество элементов в массиве  int size | // ссылка на массив целых чисел  int[] arr | Выделяет память для size элементов целого типа |  |
| **2** | MakeRandomArray | int[] arr  int size |  | Заполняет массив целыми числами с помощью ДСЧ . | Создается Random rnd.  Используется цикл с параметром.  Используется rnd.Next() |
| **3** | ReadArray | int[] arr  int size |  | Заполняет массив целыми числами, которые пользователь вводит с клавиатуры | Используется цикл с параметром.  Используется вспомогательная функция ReadIntNumber  для чтения числа из заданного диапазона |
| **4** | PrintArray | int[] arr  int size |  | Печатает массив | Используется цикл с параметром. |
| **5** | DeleteFromArray | int[] arr,  int size ,  //номер, с которого удаляем  - сколько удаляем  int firstElem –  //сколько удаляем  int howMatchElem | int[] arr  int size | Удаляет, начиная с firstElem, howMatchElem элементов | Выделяется память под вспомогательный массив из size - howMatchElem элементов.  Используется цикл с параметром.  Элементы с индексами < firstElem и больше firstElem + howMatchElem переписываются в новый массив |
| **6** | AddToArray | int[] arr,  int size ,  //номер, с которого добавляем  - сколько удаляем  int firstElem –  //сколько добавляем  int howMatchElem | int[] arr  int size | Добавляет, начиная с firstElem, howMatchElem элементов | Выделяется память под вспомогательный массив из size+ howMatchElem элементов.  Используются циклы с параметром:  1. Элементы с индексамиот 0 до firstElem переписываются в новый массив.  2. Новые элементы (howMatchElem) формируются с помощью ДСЧ и записываются в новый массив.  3. Элементы с индексами от firstElem+1 до size переписываются в новый массив. |
| **7** | Reverse | int[] arr,  int size , |  | Переворачивает массив | Используется цикл для перебора элементов от обоих концов к середине, обработка заключается в обмене элементов. |
| **8** | Sort | int[] arr,  int size |  | Сортировка методом простого обмена | Используются циклы с параметром. Внешний цикл задает количество проходов, внутренний – сравнивает пары элементов и меняет их местами, если следующий элемент меньше предыдущего. |
| **9** | FindElem | int[] arr,  int size,  int number | int index | Последовательный поиск в неотсортированном массиве | Используется цикл с параметром. Если очередной элемент совпадает с number, то цикл заканчивается, т.к. элемент найден.  Если элемент не найден, то index=-1 |
| **10** | BinarySearch | int[] arr,  int size,  int number | int index | Бинарный поиск в отсортированном массиве | Массив делится пополам. Средний элемент сравнивается с number. Если средний элемент меньше number, то левая граница переносится на средний+1, иначе правая граница переносится на средний. Деление полам продолжается до тех пор, пока границы не совпадут. Проверяется равен ли элемент, на котором границы совпали  number. Если нет, то index=-1, иначе index=левая граница. |
| **11** | ReadIntNumber | //строка, которую вводит пользователь, string buf  //разрешенный диапазон  int left, int right |  | Ввод целого числа с клавиатуры | Используется Int32.TryParse() для перевода из строки в целое число.  Используется цикл с условием.  Если число не попадает в заданный диапазон, то организуется повторный ввод.  Если введенная пользователем строка не является целым числом, то организуется повторный ввод. |
| **12** | ExistArray | int[] arr,  int size, | true/false | Проверка массива на пустоту | Проверяет, выделена ли память под массив |
| **13** | MakeMenu |  |  | Печать меню и вызов функций | Организует цикл до тех пор, пока не будет выбрана команда Выход. В цикле печатает меню и с помощью switch запускает на выполнение соответствующую выбранному пункту меню функцию. Перед вызовом функции проверяет, существует ли массив. Проверяет, выполнена ли сортировка перед вызовом бинарного поиска. |

Программа

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Sample

{

class Program

{

//ограничения для массива

const int MIN\_LENGTH = 1;

const int MAX\_LENGTH = 100;

const int MIN\_VALUE = -100;

const int MAX\_VALUE = 100;

static int ReadIntNumber(string stringForUser, int left, int right)// для ввода целого числа с проверкой границ и правильности ввода

{

bool okInput = false;

int number = MIN\_VALUE;

do

{

Console.WriteLine(stringForUser);

try

{

string buf = Console.ReadLine();

number = Convert.ToInt32(buf);// могут возникнуть исключения FormatException, OverflowException

if (number >= left && number < right) okInput = true;

else

{

Console.WriteLine("Не верно введено число!");

okInput = false;

}

}

catch (FormatException)

{

Console.WriteLine("Не верно введено число!");

okInput = false;

}

catch (OverflowException)

{

Console.WriteLine("Не верно введено число!");

okInput = false;

}

} while (!okInput);

return number;

}

static void MakeMenu(int[] arr, int sizeOfArray) //для печати меню и вызова функций

{

int check = 0;

bool sortOk = false;

do

{

Console.WriteLine("1. Формирование массива");

Console.WriteLine("2. Печать массива");

Console.WriteLine("3. Удаление элементов из массива");

Console.WriteLine("4. Добавление элементов в массив");

Console.WriteLine("5. Поиск элемента в массиве");

Console.WriteLine("6. Сортировка массива");

Console.WriteLine("7. Перестановка элементов в массиве");

Console.WriteLine("8. Выход ");

check = ReadIntNumber("", 1, 9);

switch (check)

{

case 1:

{

sizeOfArray = ReadIntNumber("Введите размер массива:", MIN\_LENGTH, MAX\_LENGTH);

arr = MakeArray(sizeOfArray);

// PrintArray(arr, sizeOfArray);

sortOk = false;

break;

}

case 2: PrintArray(arr, sizeOfArray); break;

case 3: DeleteFromArray(ref arr, ref sizeOfArray);

// PrintArray(arr, sizeOfArray);

sortOk = false;//массив изменился

break;

case 4: AddToArray(ref arr, ref sizeOfArray);

// PrintArray(arr, sizeOfArray);

sortOk = false;//массив изменился

break;

case 5:

int count=0, index=-1;

if (ExistArray(arr))

{

int number = ReadIntNumber("Введите число для поиска:", MIN\_VALUE, MAX\_VALUE);

if (!sortOk)

{

index = FindElem(arr, sizeOfArray, number, out count) + 1;//+1 т.к. нумерация с 1

if (index < 0) Console.WriteLine("Элемент {0} в массиве не найден, число сравнений: {1},\nмассив не отсортирован, выполнен линейный поиск", number, count);

else Console.WriteLine("Элемент {0} находится на позиции {1}, число сравнений: {2},\nмассив не отсортирован, выполнен линейный поиск", number, index, count);

}

else

{

index = BinarySearch(arr, sizeOfArray, number, out count)+1;//+1 т.к. нумерация с 1

if (index < 0) Console.WriteLine("Элемент {0} в массиве не найден, число сравнений: {1},\nмассив отсортирован, выполнен бинарный поиск", number, count);

else Console.WriteLine("Элемент {0} находится на позиции {1}, число сравнений: {2},\n массив отсортирован, выполнен бинарный поиск", number,index, count);

}

}

else Console.WriteLine("Массив пустой ");

break;

case 6: if (!sortOk) Sort(arr, sizeOfArray);

sortOk = true;

//PrintArray(arr, sizeOfArray);

break;

case 7: Reverse(arr, sizeOfArray);

// PrintArray(arr, sizeOfArray);

sortOk = false;

break;

}

} while (check < 8);

}

static bool ExistArray(int[] arr)

{

if (arr == null)

{

return false;

}

return true;

}

static int[] MakeArray(int sizeOfArray) //для формирования массива двумя способами

{

int[] arr = null;

int check = 0;

Console.WriteLine("1. Формирование ДСЧ");

Console.WriteLine("2. Ввод с клавиатуры");

check = ReadIntNumber("", 1, 3);

switch (check)

{

case 1: arr = MakeRandomArray(sizeOfArray);

//Console.WriteLine("Элементы сгенерированы");

break;

case 2: arr = ReadArray(sizeOfArray); break;

}

Console.WriteLine("Массив сформирован");

return arr;

}

static int[] ReadArray(int sizeOfArray)//для ввода элементов массива указанного размера с клавиатуры с проверкой правильности ввода

{

int[] arr = new int[sizeOfArray];

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

{

arr[i] = ReadIntNumber("Введите элемент массива из диапазона от -100 до 100", MIN\_VALUE,MAX\_VALUE);

}

return arr;

}

static int[] MakeRandomArray(int sizeOfArray)//для формирования элементов массива указанного размера с помощью ДСЧ

{

Random rnd = new Random();

int[] arr = new int[sizeOfArray];

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

{

arr[i] = rnd.Next(MIN\_VALUE, MAX\_VALUE);

}

return arr;

}

static void PrintArray(int[] arr, int sizeArr)

{

if (ExistArray(arr))

{

foreach (int elem in arr)

Console.Write(elem + " ");

Console.WriteLine();

}

else Console.WriteLine("Массив пустой!");

}

static void DeleteFromArray(ref int [] arr, ref int sizeOfArr)

{

if (! ExistArray(arr))

{

Console.WriteLine("Удаление невозможно!");

return ;//мссив не меняется, можно завершить функцию

}

int firstElem = ReadIntNumber("Введите номер элемента,с котрого элемента нужно удалять элементы из массива:", 0, MAX\_LENGTH);

//проверяем корректность данных для удаления

if (firstElem <= 0|| firstElem>sizeOfArr)

{

Console.WriteLine("Не правильно задан номер, с которого нужно удалять элементы массива!");

return ;//массив не меняется, можно завершить функцию

};

int howMatchElem = ReadIntNumber("Введите сколько элементов нужно удалять из массива:", 0, MAX\_LENGTH);

if (firstElem + howMatchElem > sizeOfArr+1)

{

Console.WriteLine("Не правильно задано количество удаляемых элементов массива!");

return ;//массив не меняется, можно завершить функцию

};

Console.WriteLine("Удаляем {0} элементов, начиная с {1} ",howMatchElem, firstElem );

int newSize = sizeOfArr - howMatchElem;

if (newSize == 0) //из массива удаляются все элементы

{

arr = null;

sizeOfArr = 0;

return;

}

//из массива удаляются не все элементы

int[] temp = new int[sizeOfArr - howMatchElem];//для переноса элементов, которые останутся после удаления

int j = 0, i = 0;

for (; i < sizeOfArr; i++)

if (i < firstElem-1 || i >= (firstElem + howMatchElem)-1)//номера элементов считаются с 1, а не с 0 (как в машинном массиве)

{

temp[j] = arr[i];

j++;

}

sizeOfArr = sizeOfArr - howMatchElem;

arr = temp;

}

static void AddToArray( ref int [] arr, ref int sizeOfArr)

{

int firstElem = ReadIntNumber("Введите номер, с котрого нужно добавлять элементы в массив:", 0, MAX\_LENGTH);

//проверяем корректность данных для добавления

if (firstElem <= 0 || firstElem>sizeOfArr+1)// +1 т.к. можно добавить элементы после последнего элемента массива

{

Console.WriteLine("Не правильно задан номер, с которого нужно добавлять элементы массива!");

return;//массив не меняется, можно завершить функцию

};

int howMatchElem = ReadIntNumber("Введите сколько элементов нужно добавлять в массив:", 0, MAX\_LENGTH);

if (sizeOfArr + howMatchElem > MAX\_LENGTH)

{

Console.WriteLine("Не правильно задано количество добавляемых элементов массива!");

return; //массив не меняется, можно завершить функцию

};

int[] temp;

if (ExistArray (arr)) temp = new int[sizeOfArr + howMatchElem];//для переноса элементов, которые будут в массиве после добавления

else temp = new int[howMatchElem];// в новом массиве будут только новые элементы

int[]elemForAdding=MakeArray(howMatchElem);//для формирования массива с элементами для добавления

if(!ExistArray (elemForAdding)) //пользователь пропустил фомирование массива и вернулся

{

Console.WriteLine("Элементы для добавления не сформированы!");

return; //массив не меняется, можно завершить функцию

}

int j = 0, i1 = 0, i2=0;

for (; i1 < firstElem-1; i1++) //номера элементов считаются с 1, а не с 0 (как в машинном массиве)

{

temp[j] = arr[i1];

j++;

}

for (; i2 < howMatchElem; i2++)

{

temp[j] = elemForAdding[i2];

j++;

}

for (i1 = firstElem-1; i1 < sizeOfArr; i1++) //номера элементов считаются с 1, а не с 0 (как в машинном массиве)

{

temp[j] = arr[i1];

j++;

}

Console.WriteLine("Элементы в массив добавлены");

sizeOfArr = sizeOfArr + howMatchElem;

arr=temp;

}

static int FindElem(int[] arr, int sizeOfArray, int number, out int count)

{

int index = -1; //начальное значение != индексу в массиве

count = 0;

if (ExistArray(arr))

{

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

{

count++;

if (arr[i] == number)

{

index = i; break;

}

}

Console.WriteLine("Поиск элемента выполнен");

}

else

Console.WriteLine("Поиск элемента не выполнен, массив пустой");

return index; //отрицетельное значение, если число не найдено

}

static int BinarySearch(int[]arr, int sizeOfArray, int number, out int count)

{

int left = 0, right = sizeOfArray - 1, middle = 0;

count = 0;

if (ExistArray(arr))

{

do

{

middle = (left + right) / 2;

if (arr[middle] < number) left = middle + 1;

else right = middle;

count++;

} while (left != right);

Console.WriteLine("Поиск элемента выполнен");

if (number == arr[left]) return left;

else return -1;

}

else

{

Console.WriteLine("Поиск элемента не выполнен, массив пустой");

return -1;

}

}

static void Sort(int[] arr, int sizeOfArray)

{

if (ExistArray(arr))

{

for (int i = 1; i < sizeOfArray; i++)

for (int j = sizeOfArray - 1; j >= i; j--)

if (arr[j] < arr[j - 1])

{

int temp = arr[j];

arr[j] = arr[j - 1];

arr[j - 1] = temp;

}

Console.WriteLine("Сортировка выполнена");

}

else

Console.WriteLine("Сортировка не выполнена, массив пустой");

}

static void Reverse(int[] arr, int sizeOfArray)

{

if (ExistArray(arr))

{

int left = 0, right = sizeOfArray - 1;

while (left < right)

{

int temp = arr[left];

arr[left] = arr[right];

arr[right] = temp;

left++; right--;

}

Console.WriteLine("Перестановка элементв выполнена");

}

else

Console.WriteLine("Перестановка элементв не выполнена, массив пустой");

}

static void Main(string[] args)

{

int sizeOfArray=0;

int[] arr = null;

MakeMenu(arr, sizeOfArray);

}

}

}