**Министерство образования и науки Республики Казахстан**

**НАО Евразийский национальный университет им. Л.Н.Гумилева**

Факультет Информационных технологий

Кафедра Компьютерной и программной инженерии

**КУРСОВАЯ РАБОТА**

По дисциплине: «Алгоритмы и структуры данных»

Выполнил: студент группы ВтиПО-22

Орынбай Д.Д.

**Содержание**

[**Введение.** 3](#_Toc103632911)

[**Задание 1. Анализ эффективности алгоритма.** 4](#_Toc103632912)

[**Задание 2. Структура данных стек.** 6](#_Toc103632913)

[**Задание 3. Структура данных очередь.** 8](#_Toc103632914)

[**Задание 4. Структура данных связные списки.** 10](#_Toc103632915)

[**Задание 5. Структура данных дерево.** 11](#_Toc103632916)

[**Задание 6. Сортировка массивов.** 14](#_Toc103632917)

[**Задание 7. Рекурсия с возвратом.** 15](#_Toc103632918)

[**Заключение.** 18](#_Toc103632919)

[**Список литературы.** 18](#_Toc103632920)

# **Введение.**

Структура данных – это контейнер, который хранит информацию в определенном виде. Из-за такой ‘компоновки’ она может быть эффективной в одних операциях и неэффективной в других. Цель разработчика – выбрать из существующих структур оптимальный для конкретной задачи вариант.

**Зачем нужны структуры данных?**

Данные являются самой важной сущностью в информатике, а структуры позволяют хранить их в организованной форме. Какую бы проблему вы не решали, вам приходится иметь дело с данными - будь то зарплата сотрудника, цены на акции, список покупок или даже простой телефонный справочник. В зависимости от ситуации данные должны храниться в некотором определенном формате. Структуры данных предлагают несколько вариантов таких размещений.

**Основные структуры данных.**

1) Массивы.

Массив - упорядоченный набор элементов, каждый из которых хранит одно значение, идентифицируемое с помощью одного или нескольких индексов. В простейшем случае массив имеет постоянную длину и хранит единицы данных одного и того же типа, а в качестве индексов выступают целые числа.

2) Стеки.

Стек - структура данных, напоминающая магазин автомата или стопку тарелок. Принцип работы такой структуры называется LIFO, то есть «Last In, First Out» (Последним вошёл, первым вышел).

3) Очереди.

Очередь - структура данных, из которой удаляется первым тот элемент, который был первым в очередь добавлен. То есть очередь в программировании соответствует «бытовому» понятию очереди. Очередь также называют структурой типа FIFO (first in, first out — первым пришел, первым ушел).

4) Связанные списки.

Связный список - базовая динамическая структура данных в информатике, состоящая из узлов, каждый из которых содержит как собственно данные, так и одну или две ссылки («связки») на следующий и/или предыдущий узел списка. Принципиальным преимуществом перед массивом является структурная гибкость: порядок элементов связного списка может не совпадать с порядком расположения элементов данных в памяти компьютера, а порядок обхода списка всегда явно задаётся его внутренними связями.

Виды связных списков:

1) Линейный однонаправленный список - структура данных, состоящая из элементов одного типа, связанных между собой последовательно посредством указателей. Каждый элемент списка имеет указатель на следующий элемент. Последний элемент списка указывает на NULL. Элемент, на который нет указателя, является первым (головным) элементом списка. Здесь ссылка в каждом узле указывает на следующий узел в списке. В односвязном списке можно передвигаться только в сторону конца списка. Узнать адрес предыдущего элемента, опираясь на содержимое текущего узла, невозможно.

2) Двусвязный список (двунаправленный связный список) - здесь ссылки в каждом узле указывают на предыдущий и на последующий узел в списке. Как и односвязный список, двусвязный допускает только последовательный доступ к элементам, но при этом дает возможность перемещения в обе стороны. В этом списке проще производить удаление и перестановку элементов, так как легко доступны адреса тех элементов списка, указатели которых направлены на изменяемый элемент.

3) Кольцевой связный список - разновидностью связных списков является кольцевой (циклический, замкнутый) список. Он тоже может быть односвязным или двусвязным. Последний элемент кольцевого списка содержит указатель на первый, а первый (в случае двусвязного списка) — на последний. Как правило, такая структура реализуется на базе линейного списка. С каждым кольцевым списком дополнительно хранится указатель на первый элемент. В этом списке ссылки на NULL не встречается.

5) Деревья.

Дерево – это иерархическая структура данных, состоящая из вершин (узлов) и ребер, соединяющих их. Они похожи на графы, но есть одно важное отличие: в дереве не может быть цикла. Деревья широко используются в искусственном интеллекте и сложных алгоритмах для обеспечения эффективного механизма хранения данных.

6) Графы.

Граф - набор узлов (вершин), которые соединены друг с другом в виде сети ребрами (дугами).

Виды графов:

1) Ориентированный - ребра являются направленными, т.е. существует только одно доступное направление между двумя связными вершинами.

2) Неориентированные - к каждому из ребер можно осуществлять переход в обоих направлениях.

# **Задание 1. Анализ эффективности алгоритма.**

**5. Выполнить математический анализ рекурсивного алгоритма вычисления чисел Фибоначчи.**

Рекурсия - способ организации обработки данных, при котором программа вызывает сама себя непосредственно, либо с помощью других программ.

Математический анализ рекурсивных алгоритмов:

1) Выберите параметр (или параметры), по которому будет оцениваться размер входных данных алгоритма.

2) Определите основную операцию алгоритма.

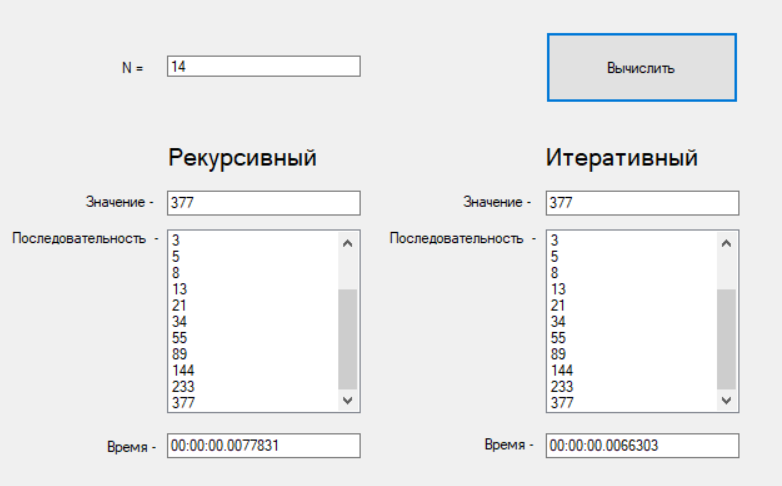
3) Проверьте, зависит ли число выполняемых основных операций только от размера входных данных. Если оно зависит и от других факторов, рассмотрите при необходимости, как меняется эффективность алгоритма для наихудшего, среднего и наилучшего случаев.

4) Составьте рекуррентное уравнение, выражающее количество выполняемых основных операций алгоритма, и укажите соответствующие начальные условия.

5) Найдите решение рекуррентного уравнения или, если это невозможно, определите хотя бы его порядок роста.

Главным инструментом анализа временной эффективности рекурсивного алгоритма является построение рекуррентного соотношения для количества выполнений его основной операции и выяснение порядка его роста.

Числа Фибоначчи - это числа такой последовательности, в которой первые два элемента - 0 и 1, а каждый последующий элемент равен сумме двух предшествующих.



Итеративный метод решения:

int f(int n)

{

int a = 0;

int b = 1;

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

{

a = a + b;

b = a - b;

}

return a;

}

Рекурсивный метод решения:

int fibonacci(int number)

{

if (number == 0)

return 0;

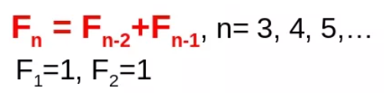
if (number == 1)

return 1;

return fibonacci(number - 1) + fibonacci(number - 2);

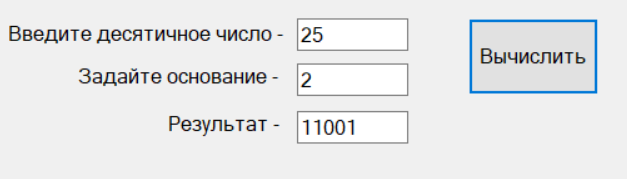
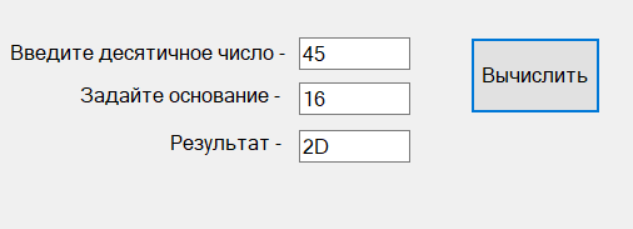
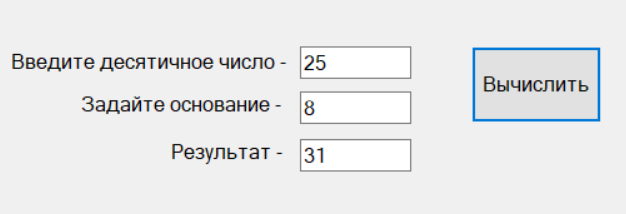
}

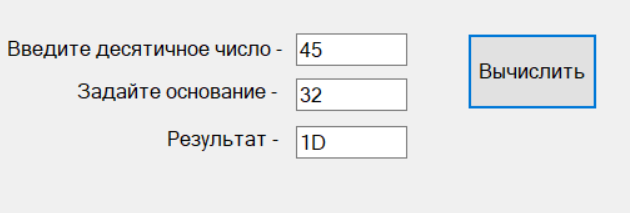
Рекуррентная формула:



# **Задание 2. Структура данных стек.**

2. C помощью стека решить задачу перевода десятичного числа в систему с заданным основанием.





string res = "";

if (num\_system >= 2 && num\_system <= 36)

{

string digs = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";

var my\_stack = new Stack<int>();

while (dec != 0)

{

my\_stack.Push(dec % num\_system);

dec /= num\_system;

}

while (my\_stack.Count != 0)

{

res += digs[my\_stack.Pop()];

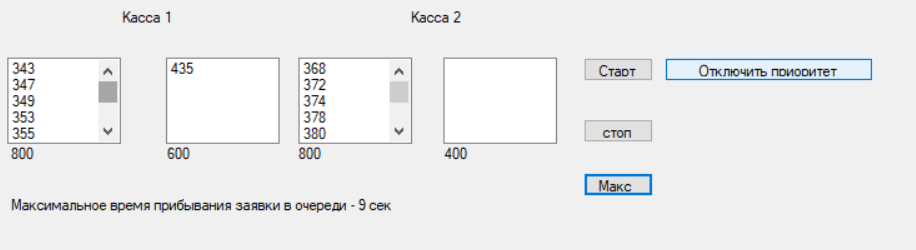
}

}

textBox3.Text = res.ToString();

# **Задание 3. Структура данных очередь.**

2. Система состоит из двух обслуживающих аппаратов. Очереди на каждый аппарат две: для обычных и приоритетных заявок. Определить максимальное время пребывания заявки в очереди.





Подача людей на обычные заявки:

int klientC = 1;

private void klients\_Tick(object sender, EventArgs e)

{

listBox1.Items.Add(klientC);

klientC++;

listBox3.Items.Add(klientC);

klientC++;

}

Подача людей на приоритетные заявки:

private void VIPCLIENTS\_Tick(object sender, EventArgs e)

{

listBox2.Items.Add(klientC);

klientC++;

listBox4.Items.Add(klientC);

klientC++;

}

Кнопка для подачи или отключения приоритетных заявок:

private void button4\_Click(object sender, EventArgs e)

{

if (VIPCLIENTS.Enabled)

{

VIPCLIENTS.Enabled = false;

button4.Text = "Включить вип клиентов ";

}

else

{

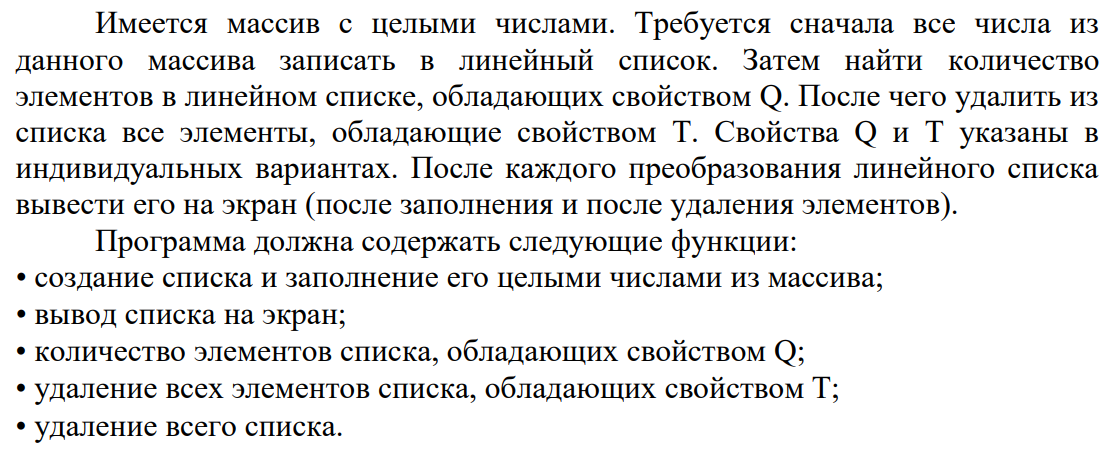
VIPCLIENTS.Enabled = true;

button4.Text = "Отключить вип клиентов ";

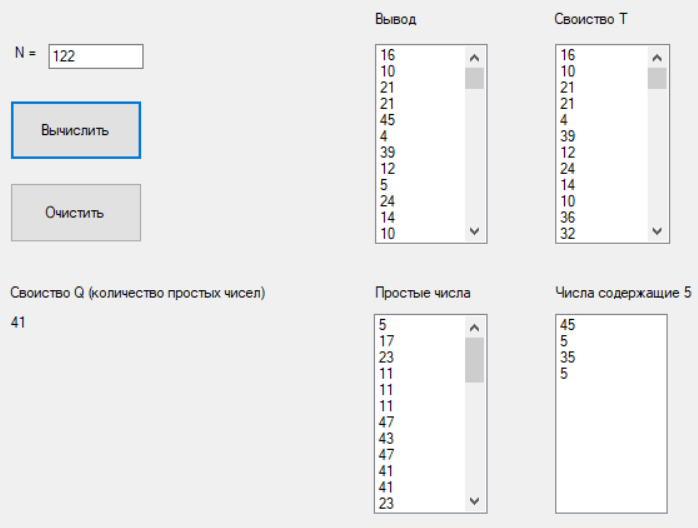
}

}

# **Задание 4. Структура данных связные списки.**



6. Q: число является простым. T: число содержит в своем составе цифру 5.



Число является простым:

int count = 0;

bool Prim(int n)

{

for (int d = 2; d <= n / 2; d++)

if (n % d == 0)

return false;

return true;

}

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

{

if (arr[i] > 1)

{

if (Prim(arr[i]))

{

listBox3.Items.Add(arr[i]);

count++;

}

}

}

Число содержит в своем составе цифру 5:

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

{

if (arr[i] % 10==5)

{

listBox4.Items.Add(arr[i]);

continue;

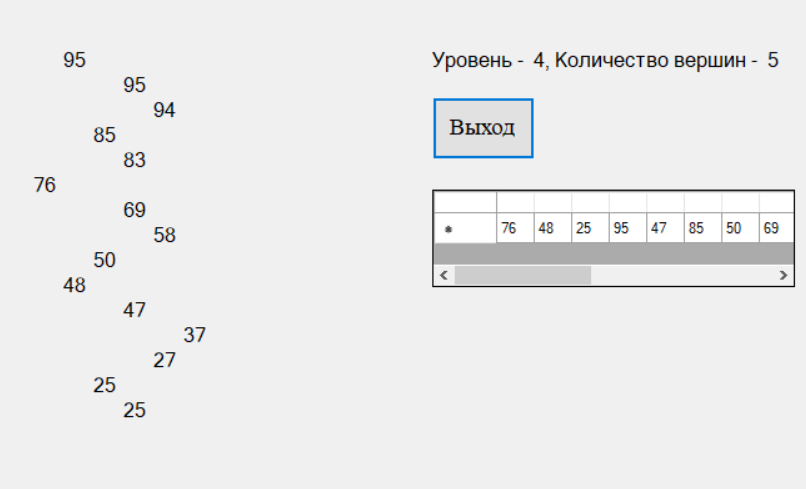
}

listBox2.Items.Add(arr[i]);

}

# **Задание 5. Структура данных дерево.**

7. Дан массив целых чисел. Построить дерево поиска. Определить уровень, на котором находится наибольшее количество вершин – листьев.



Класс дерева:

internal class Tree

{

private IntPtr data;

private Tree left;

private Tree right;

Добавление элементов в дерево:

public void Add(int key)

{

if (dannye == IntPtr.Zero)

{

dannye = Marshal.AllocHGlobal(sizeof(int));

Marshal.WriteInt32(dannye, key);

return;

}

else

{

if (key <= Marshal.ReadInt32(dannye))

{

if (left == null)

{

left = new Tree();

left.Add(key);

}

else

{

left.Add(key);

}

}

else

{

if (right == null)

{

right = new Tree();

right.Add(key);

}

else

{

right.Add(key);

}

}

}

Определение количества вершин листьев в определенном уровне:

public int GetSizeOfLevel(Tree tree, int height, int depth = 1)

{

int razmer = 0;

if (tree.left != null)

{

razmer += GetSizeOfLevel(tree.left, height, depth + 1);

}

if (tree.dannye != IntPtr.Zero & height == depth)

{

razmer++;

}

if (tree.right != null)

{

razmer += GetSizeOfLevel(tree.right, height, depth + 1);

}

return razmer;

}

Максимально возможный уровень дерева:

private static int GetHeight(Tree root)

{

int height = 0;

if (root != null)

{

int left = GetHeight(root.left);

int right = GetHeight(root.right);

int maximum = left > right ? left : right;

height = maximum + 1;

}

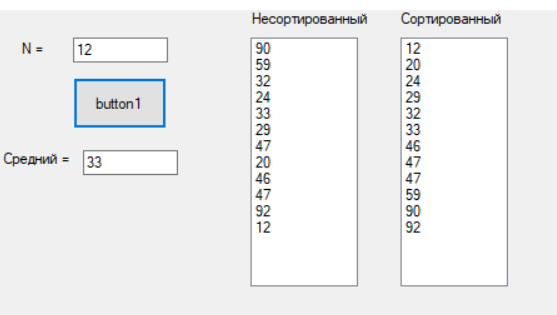
return height;

}

# **Задание 6. Сортировка массивов.**

11. Известны марки и стоимость n моделей автомобилей. Определить марку автомобиля, стоимость которого является «средней» (т.е. величина которой оказалась в середине массива в случае его сортировки).

a. прямого включения



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

{

int value = arr[i];

int index = i;

while (index>0&&arr[index-1]>value)

{

arr[index] = arr[index - 1];

index--;

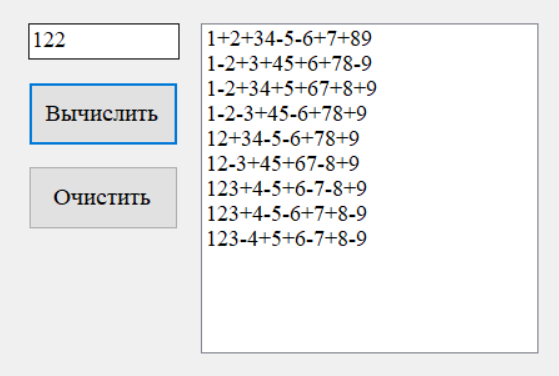
}

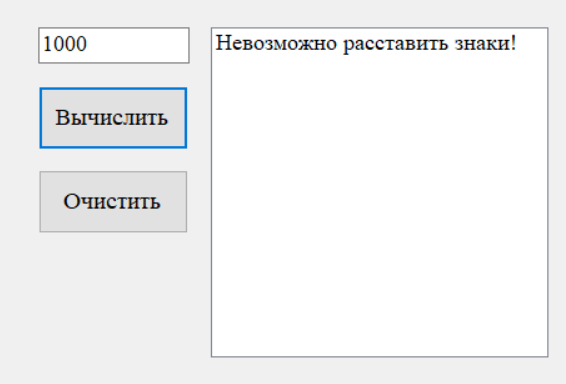
arr[index] = value;

}

# **Задание 7. Рекурсия с возвратом.**

3. Расстановка знаков. Дано целое число m. Вставить между некоторыми цифрами 1,2,3,4,5,6,7,8,9, записанными именно в таком порядке, знаки «+» и «-» так, чтобы значением получившегося выражения было число m. Например если m=122, то подойдет выражение: 12+34-5-6+78+9. Если расставить знаки требуемым образом невозможно, сообщить об этом.





public void Check(int begin, int result = 0, char oper = '+', int index = 2){

int m\_2;

if (oper == '+'){

m\_2 = result + begin;

}

else{

m\_2 = result - begin;

}

if (index <= 9){

signs[index] = '+';

Check(index, m\_2, '+', index + 1);

signs[index] = '-';

Check(index, m\_2, '-', index + 1);

signs[index] = ' ';

Check(begin \* 10 + index, result, oper, index + 1);

}

else{

if (m\_2 == m){

string Sum = "";

for (int i = 2; i <= 10; i++)

{

if (signs[i] == ' '){

Sum += (i - 1).ToString();

}

else{

Sum += (i - 1).ToString() + signs[i].ToString();

}

}

listBox1.Items.Add(Sum);

}

}

}

# **Заключение.**

В ходе выполнения курсовой работы были выполнены все поставленные задачи и реализованы необходимые структуры данных: массив, стек, очередь, связной список, дерево. Четкая структуризация кода и грамотно подобранные имена переменных, структур данных, функций и процедур способствуют удобочитаемости программы.

В заключение хочу написать, что я узнал, что такое структура данных, и какие структуры данных бывают. В частности, такие как: стек, очередь, связной список, дерево и т.д. Узнал для чего нужны структуры данных, как строится математический анализ рекурсивного алгоритма, узнал, что такое числа Фибоначчи. Узнал, какие бывают виды связных списков, графов. Понял, как реализовывать сортировку с включением. Научился решать много интересных задач на разные структуры данных.

# **Список литературы.**

*1) 8 известных структур данных, о которых спросят на собеседовании*. (2018). Библиотека программиста. https://proglib.io/p/8-data-structures

2) Icu, I. (2021, December 11). *Дерево как структуры данных , Двоичные (бинарные) деревья*. Intellect. https://intellect.icu/derevo-kak-struktury-dannykh-dvoichnye-binarnye-derevya-9856

3) S. (2017, November 2). *О стеке простыми словами — для студентов и просто начинающих*. Хабр. https://habr.com/ru/post/341586/

*4) Вы точно человек?* (2008). КиберЛенинка. https://cyberleninka.ru/article/n/matematicheskie-metody-analiza-rekursivnyh-algoritmov

5) Д. (2020, November 28). *Очередь (queue) в C++: реализация и что это вообще такое*. CODELessons.ru | Пошаговые уроки по программированию. https://codelessons.ru/cplusplus/ochered-queue-v-c-realizaciya-i-chto-eto-voobshhe-takoe.html

6) Е. (2018, April 16). *Сортировка прямыми включениями*. Prog-Cpp.Ru. https://prog-cpp.ru/sort-include/

*7) Числа Фибоначчи: циклом и рекурсией. Язык Python*. (2021). linux. https://younglinux.info/algorithm/fibonacci

8) Штукатуров, С. (2021, November 15). *Введение в связанные списки*. Tproger. https://tproger.ru/translations/introduction-to-linked-lists/