ISSN 2181-4732
ISSN 2181-4732

УДК: 004.422.11

ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ ПУЗЫРЬКОВОГО АЛГОРИТМА СОРТИРОВКИ

Узаков Заир – кандидат физико-математических наук, доцент,
ORCID: 0000-0001-8459-6072, E-mail: zair90uzakov@gmail.com

Каршинский государственный технический университет, г. Карши, Узбекистан

Аннотация. *В статье изложены содержание и особенности алгоритма метода пузырька упорядочения элементов одномерного массива по возрастанию, в котором процесс упорядочения элементов выполняется путём последовательного сравнения каждых двух соседних элементов массива, начиная с первого элемента, слева направо, и обмена мест расположения этих элементов, если они расположены не по возрастанию. Объектом исследования является практическая реализация алгоритма метода пузырька упорядочения элементов одномерного массива по возрастанию. Разработка и изложение методики практической реализации алгоритма, его программное описание на языке C++, проведение на компьютере вычислительных экспериментов по практической реализации алгоритма и анализ их результатов является целью исследования. Приведены блок-схема алгоритма метода пузырька и его программная реализация на языке программирования C++, результаты вычислительных экспериментов по практической реализации алгоритма сортировки на компьютере при различных исходных расположениях элементов одномерного массива и их анализ. Результаты вычислительных экспериментов подтверждают ранее выведенную формулу о квадратичной зависимости количества действий, которые необходимо выполнить на компьютере для упорядочения элементов одномерного массива, от количества элементов массива, что является определённой частью новизны данной работы.*

Ключевые слова: *упорядочение, элементы, массив, метод, пузырёк, вычисление, эксперимент, количество, действие.*

UDC: 004.422.11

PRACTICAL IMPLEMENTATION OF THE BUBBLE SORTING ALGORITHM

Uzakov, Zair – Candidate of Physical and Mathematical Sciences, Associate Professor

Karshi State Technical University, Karshi city, Uzbekistan

Abstract. *The article presents the content and features of the bubble method algorithm for ordering of elements of a one-dimensional array in ascending order, in which the process of elements ordering is performed by sequentially comparing every two adjacent array elements, starting with the first element, from left to right, and exchanging the locations of these elements if they are not arranged in ascending order. The object of the study is the practical implementation of the bubble method algorithm for ordering elements of a one-dimensional array in ascending order. The development and presentation of the methodology for the practical implementation of the algorithm, its software description in the C++ programming language, conducting computational experiments on a computer for the practical implementation of the algorithm, and analyzing their results are the goals of the research. A block diagram of the bubble method algorithm and its software realization in the C++ programming language, the results of computational experiments on the practical implementation of the sorting algorithm on a computer for different initial arrangements of elements of a one-dimensional array, and their analysis are given. The results of computational experiments confirm the previously derived formula about the quadratic dependence of the number of actions that must be performed on a computer to order the elements of a one-dimensional array on the number of elements of the array, which is a certain part of the novelty of this work.*

Keywords: *ordering, elements, array, method, bubble, calculation, experiment, quantity, action.*

ISSN 2181-4732
ISSN 2181-4732

UO‘K: 004.422.11

PUFAKCHALI SARALASH ALGORITMINI AMALGA OSHIRISH

Uzakov Zair – fizika-matematika fanlari nomzodi, dotsent

Qarshi davlat texnika universiteti, Qarshi sh., O‘zbekiston

Annotatsiya. *Maqolada bir o‘lchovli massiv elementlarini o‘sish tartibida tartiblashning pufakcha usuli algoritmining mazmuni va xususiyatlari bayon etilgan, unda algoritm elementlarni tartiblash jarayoni birinchi elementdan boshlab, chapdan o‘ngga qarab massivning har ikkita qo‘shni elementlarini ketma-ket taqqoslash va agar ular o‘sish tartibida joylashtirilmagan bo‘lsa, bu elementlarning joylashuvini almashish orqali amalga oshiriladi. Tadqiqot obyekti bir o‘lchovli massiv elementlarini o‘sish tartibida tartiblash uchun pufakcha usuli algoritmini amaliyotda amalga oshirishdirdan iboratdir. Algoritmni amaliyotda amalga oshirish metodikasini ishlab chiqish va taqdim etish, C++ dasturlash tilida uning dasturiy tavsifini amalga oshirish, algoritmni amaliyotda amalga oshirish uchun kompyuterda hisoblash tajribalarini o‘tkazish va ularning natijalarini tahlil qilish tadqiqot maqsadi hisoblanadi. Pufakchali usul algoritmining blok-sxemasi va uning C++ dasturlash tilida dasturiy ta’minotini amalga oshirish, bir o‘lchovli massiv elementlarining turli xil dastlabki joylashuvi uchun saralash algoritmini kompyuterda amalda qo‘llash bo‘yicha hisoblash tajribalari natijalari va ularning tahlili keltirilgan. Hisoblash tajribalari natijalari bir o‘lchovli massiv elementlarini tartiblash uchun kompyuterda bajarilishi lozim bo‘lgan amallar sonining massiv elementlari soniga kvadratik bog‘liqligi haqidagi ilgari olingan formulani tasdiqlaydi. Bu esa tadqiqotda olingan natijalar yangiligining ma’lum bir qismidir.*

Kalit so‘zlar: *tartiblash, elementlar, massiv, usul, pufakcha, hisoblash, tajriba, miqdor, amallar.*

Введение

Алгоритм решения задачи представляет собой совокупность четко сформулированных инструкций, выполнение которых приводит к решению поставленной задачи. Процесс решения задачи является на самом деле процессом обработки входных данных, их преобразования в решение задачи. Алгоритмы сортировки элементов исследуемого объекта, к которым относится и алгоритм метода пузырька, являются наиболее часто применяемыми на практике алгоритмами. Алгоритмы сортировки различаются между собой в основном стратегией сортировки [1-6]. В качестве простого примера задачи сортировки можно привести, например, задачу упорядочения последовательности чисел в порядке возрастания. В общем случае, задачу сортировки можно определить в виде совокупности входных и выходных данных: входными данными является любая последовательность чисел, удовлетворяющих определённым условиям, а выходными данными является последовательность, например, переставленных в порядке возрастания чисел, являющихся элементами входных данных. В такой последовательности чисел меньшие элементы расположены перед большими.

В работе [7] приведены примеры применения алгоритма пузырькового метода сортировки элементов одномерного массива по возрастанию в трёх случаях, когда начальное расположение элементов массива является: 1) наилучшим по отношению к поставленной задаче упорядочения элементов массива по возрастанию; 2) средним по отношению к поставленной задаче; 3) наихудшим по отношению к поставленной задаче. Сложность алгоритма решения конкретной задачи оценивается по времени, которое требуется для выполнения алгоритма, и по объёму оперативной памяти компьютера, которую занимает алгоритм во время выполнения. В общем случае, сложность алгоритмов измеряется анализом времени, которое потребуется для обработки очень большого набора входных данных. Такой анализ называется асимптотическим анализом сложности алгоритма. Степень сложности

ISSN 2181-4732
ISSN 2181-4732

алгоритма принято оценивать по наихудшему случаю входных данных относительно поставленной задачи, так как ресурсы, то есть время на выполнение алгоритма и объем памяти компьютера, выделенные для решения задачи в наихудшем случае, будут достаточны для решения задачи в любом другом случае.

В статье [8] рассмотрен вопрос о степени сложности алгоритма метода пузырька упорядочения элементов одномерного массива по возрастанию применительно к массиву int A[n] ={a_1, a_2, a_3, …, a_i, …, a_{n-1}, a_n}; , в котором $a_1>a_2>a_3>….>a_i>…>a_{n-1}>a_n$, то есть исходное расположение элементов массива является наихудшим относительно поставленной задачи упорядочения элементов массива по возрастанию. Получена формула асимптотической оценки $N=5/2*n*(n-1)=5/2*n^2-5/2*n$ (1) степени сложности пузырькового алгоритма сортировки, где n- количество элементов массива, N – общее количество операций сравнения и присваивания, которые необходимо выполнить для упорядочения элементов массива по возрастанию при наихудшем по отношению к поставленной задаче исходном расположении элементов массива, то есть элементы расположены в порядке убывания. Формула (1) показывает, что асимптотическая сложность алгоритма пузырькового метода имеет вид $O(n^2)$. Представляет интерес практическая реализация пузырькового алгоритма сортировки, определение характеристик сложности алгоритма в вычислительных экспериментах. Разработка методики выполнение этих работ и их изложение является целью данной статьи.

Методика исследования и материалы

Графическое описание алгоритма метода пузырька приведено ниже. В данной блок-схеме алгоритма и ниже в программе n – количество элементов массива; j – параметр цикла ввода элементов массива A[n] в память компьютера, их вывода на печать, прохода по элементам массива; i- порядковый номер прохода по элементам массива при их сортировке; CNOAO - количество операций присваивания, которые выполняются при данном i-ом проходе по элементам массива; NOPTAE - количество проходов по элементам массива; AV – вспомогательная переменная, используемая при перестановке местами двух соседних элементов массива, не расположенных в возрастающем порядке (блоки 13-15, три операции присваивания); блок 9 – начало проходов по элементам массива; блоки 10-16 – выполнение текущего i-го прохода по элементам массива; блок 17 – проверка условия, будут ли прекращены (да) или продолжены (нет) проходы по элементам массива; блоки 18-19 - вывод на экран монитора элементов отсортированного массива; блок 20 –завершение алгоритма сортировки элементов массива.

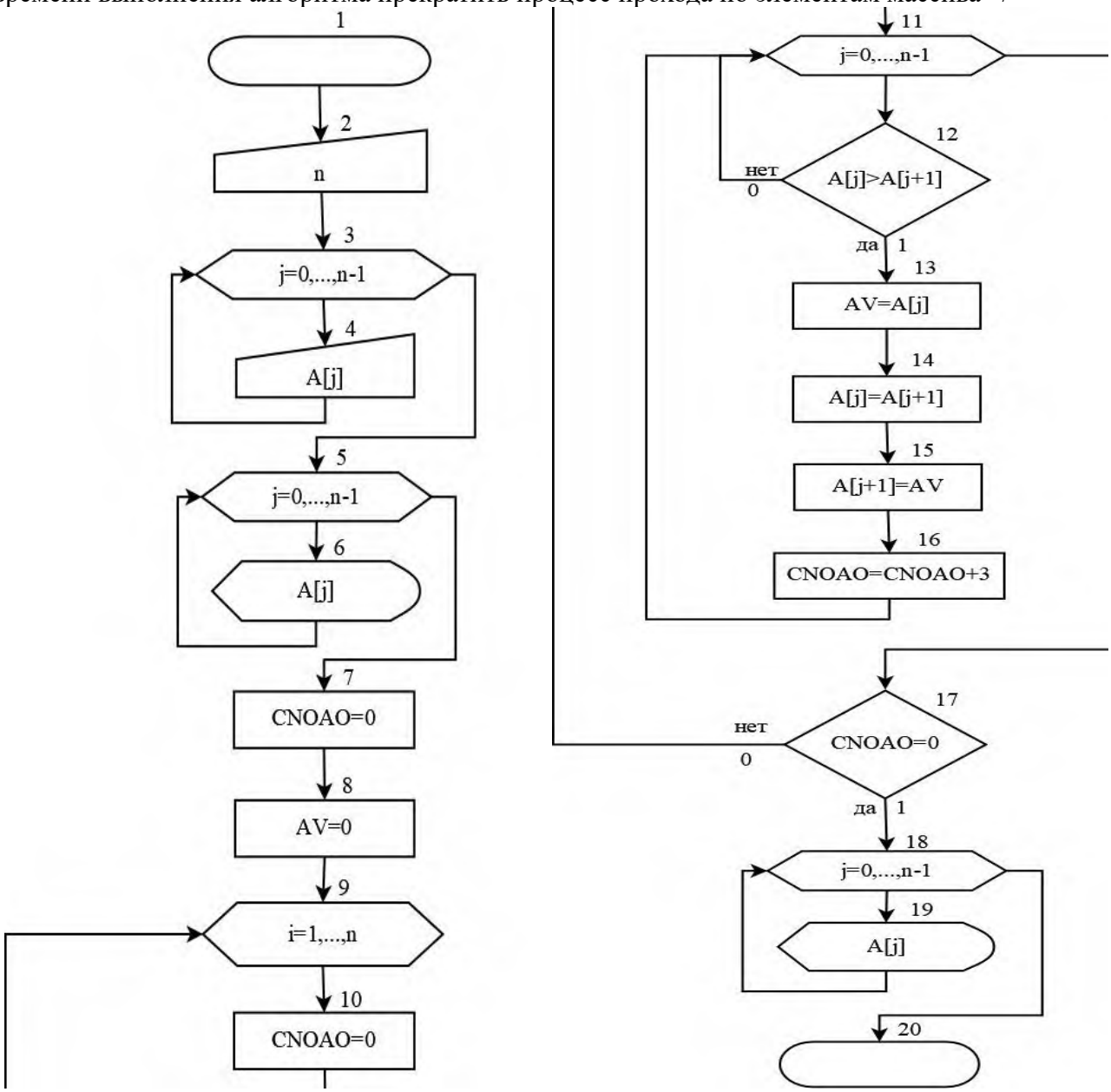
Программа реализации пузырькового алгоритма [9-12].

```
#include <iostream>
#include <Windows.h>
using namespace std;
/* NOCO - количество операций сравнения; NOAO - количество операций присваивания;
TNOO - общее количество операций сравнения и присваивания; NOPTAE- количество
проходов по элементам массива; A[100]- максимально допустимый в данной программе
массив, элементы которого упорядочиваются по возрастанию; AV - вспомогательная
переменная */
int main()
{
setlocale(LC_ALL,"Russian");  int NOCO=0;  int NOAO=0;  int CNOAO;
int TNOO=0;  int AV;  int n;
/* ввод количества элементов массива*/
cout<<"n=";  cin>>n;  cout<<endl;
int A[100];
/* инициализация массива */
```

ISSN 2181-4732
ISSN 2181-4732

```
for(int j=0; j<n; j++)
{
cout<<"A["<<j<<"]=";  cin>>A[j];  cout<<endl;  }  cout<<endl;
/* Вывод элементов созданного массива на экран монитора */
cout<<"Исходный массив";  cout<<endl;  cout<<endl;
for(int j=0; j<n; j++) { cout<<"A["<<j<<"]="<<A[j];  cout<<endl;  }  cout<<endl;/*
начало сортировки элементов массива, начало прохода по элементам массива */
for(int i=1; i<=n; i++) {  int CNOAO=0;     NOPTAE=i;
/* начало цикла по элементам массива, выполнение i-го прохода по элементам массива */
for(int j=0; j<(n-1); j++)  {  NOCO=NOCO+1;  if(A[j]>A[j+1])  {  AV=A[j];
A[j]=A[j+1];  A[j+1]=AV;  NOAO=NOAO+3;  CNOAO=CNOAO+3; } }
```

/* если при данном проходе по элементам массива CNOAO=0, то есть при данном проходе не выполнилась ни одна операция присваивания, то это означает, что ни в одной паре соседних элементов массива не выполнилась перестановка элементов местами, элементы массива пришли в упорядоченное по возрастанию состояние, можно и необходимо в целях экономии времени выполнения алгоритма прекратить процесс прохода по элементам массива */



ISSN 2181-4732
ISSN 2181-4732

```
if(CNOAO==0) break;     }
cout<<endl;   cout<<endl;   TNOO=NOCO+NOAO;
/* вывод на экран монитора элементов отсортированного массива */
cout<<"Отсортированный массив";    cout<<endl;   cout<<endl;
for(int j=0; j<n; j++) {cout<<"A["<<j<<"]="<<A[j];   cout<<endl;   }  cout<<endl;
cout<<"Количество выполненных операций сравнения";   cout<<endl;
cout<<"NOCO= "<<NOCO;    cout<<endl;
 cout<<"Количество выполненных операций присваивания";   cout<<endl;
cout<<"NOAO= "<<NOAO;   cout<<endl;
cout<<"Общее количество выполненных операций ";   cout<<endl;
cout<<"TNOO= "<<TNOO;   cout<<endl;   cout<<endl;
system("pause");    return 0;     }
```

Примечания к тексту программы. Значение переменной CNOAO регулирует продолжение проходов по элементам массива в целях их упорядочения по возрастанию или прекращение проходов (оператор if(CNOAO==0) break; в тексте программы). Текущее значение переменной CNOAO равно количеству операций присваивания, выполненных над элементами массива при только что завершившемся очередном проходе по элементам массива. Если значение переменной CNOAO отлично от нуля, то это означает, что при данном проходе над элементами массива выполнены по меньшей мере один раз действия по перестановке местами двух соседних элементов массива, и связанные с этим действием операции присваивания, а может быть и больше действий, по причине, что эти элементы были расположены не по возрастанию. Необходимо продолжить действия прохода по элементам массива. Если же значение переменной CNOAO равно нулю, то это означает, что при данном, очередном проходе по элементам массива не были выполнены операции присваивания, то есть не было выполнено ни одно действие по перестановке местами элементов массива, в массиве нет неупорядоченных элементов. В этом случае, то есть в случае CNOAO=0, можно и нужно, в целях сокращения времени выполнения алгоритма, прекратить действия прохода по элементам массива, что выполняется в программе с помощью оператора break (break - прекращение, прекращение проходов по элементам массива). Обмен местами двух соседних элементов массива в данной программе осуществляется с помощью операторов присваивания AV=A[j]; A[j]=A[j+1]; A[j+1]=AV. Обменять местами два соседних элемента массива можно также с использованием стандартной функции swap(A[j], A[j+1]).

Результаты исследования и их обсуждение

Ниже приводятся результаты вычислительных экспериментов по упорядочению элементов массивов массив (1) - массив (3) по возрастанию, для которых n=7 и элементы массивов имеют различное исходное расположение по отношению к поставленной задаче их упорядочения по возрастанию (соответственно. наилучшее, среднее и наихудшее), и массива (4), для которого n=21 и элементы массива имеют наихудшее исходное расположение по отношению к поставленной задаче. В каждом массиве заключительным проходом по его элементам является проход, удостоверяющий, что элементы массива упорядочены по возрастанию, то есть проход, в котором количество выполненных операций присваивания NOAO= 0.

Массив (1). n=7. Исходный массив int A[7]={-15, -9, 4, 7, 13, 28, 30};
Отсортированный массив int A[7]={-15, -9, 4, 7, 13, 28, 30};
Количество выполненных операций сравнения NOCO= 6.
Количество выполненных операций присваивания NOAO= 0.
Общее количество выполненных операций TNOO=6.
Количество проходов по элементам массива NOPTAE=1.
После первого прохода по элементам массива (1) количество выполненных операций

ISSN 2181-4732
ISSN 2181-4732

присваивания равно нулю NOAO= 0, и это означает, что элементы массива упорядочены по возрастанию и процесс прохода по элементам массива прекращается.

Массив (2). n=7. Исходный массив int A[7]={-15, -9, 13, 4, 28, 30, 7};
Отсортированный массив int A[7]={-15, -9, 4, 7, 13, 28, 30};
Количество выполненных операций сравнения NOCO= 24.
Количество выполненных операций присваивания NOAO= 12.
Общее количество выполненных операций TNOO=36.
Количество проходов по элементам массива NOPTAE=4.

Массив (3). n=7. Исходный массив int A[7]={ 30, 28, 13, 7, 4, -9, -15};
Отсортированный массив int A[7]={-15, -9, 4, 7, 13, 28, 30};
Количество выполненных операций сравнения NOCO= 42.
Количество выполненных операций присваивания NOAO= 63.
Общее количество выполненных операций TNOO=105.
Количество проходов по элементам массива NOPTAE=7.
Массив (4). n=21. Исходный массив int A[21]={30, 28, 13, 7, 4, -9, -15,
-16, -17, -18, -19, -20, -21, -22, -23, -24, -25, -26, -27, -28, -29 };
Отсортированный массив int A[21]={ -29, -28, -27, -26, -25, -24, -23, -22, -21,
-20, -19, -18, -17, -16, -15, -9, 4, 7, 13, 28, 30};
Количество выполненных операций сравнения NOCO= 420.
Количество выполненных операций присваивания NOAO= 630.
Общее количество выполненных операций TNOO=1050.
Количество проходов по элементам массива NOPTAE=21.

Выше было сказано, что в статье [8] получена асимптотическая оценка

$$N=5/2*n*(n-1) \quad (1)$$

степени сложности пузырькового алгоритма сортировки. В массиве (3) n=7 и, согласно формуле (1), N=5/2*7*(7-1) = 5/2*7*6=105. Результаты вычислительных экспериментов по практическому применению пузырькового алгоритма сортировки также показывают, что выполнено такое количество, 105 операций сравнения и присваивания. В массиве (4) n=21 и, согласно формуле (1), N=5/2*21*(21-1) = 5/2*21*20=1050. Это значение N=1050 также совпадает с результатами вычислительных экспериментов.

Отметим, что в случае массива (2) количество операций (36), выполняемых для упорядочения элементов массива по возрастанию, меньше, чем в случае массива (3) (105), потому что в исходном состоянии часть элементов расположена по возрастанию. В случае же массива (1) выполняются только 6 операций сравнения соседних элементов, не выполняется ни одна операция по перестановке соседних элементов местами, так как в исходном состоянии все элементы расположены по возрастанию.

Асимптотическая сложность алгоритма пузырькового метода имеет вид $O(n^2)$, который означает, что когда количество элементов массива возрастает в m раз, количество операций, выполняемых для упорядочения элементов массива возрастает в m^2 раз: $(m*n)^2 = m^2 * n^2$. Если, например, m=3, то количество операций возрастает в 9 раз. В массиве (4) число элементов больше, чем в массиве (3), в 3 раза, а количество операций, выполняемых для упорядочения элементов массива (4) возросло в 10 раз (1050 операций), чем в массиве (3) (105 операций). Это говорит о том, что оценка $O(n^2)$ асимптотической сложности алгоритма пузырькового метода является приближённым описанием точной формулы (1).

Заключение

Таким образом, в статье показана практическая реализация пузырькового алгоритма сортировки элементов одномерного массива. Результаты вычислительных экспериментов подтверждают формулу N=5/2*n*(n-1) зависимости количества действий N, выполняемых при

ISSN 2181-4732
ISSN 2181-4732

упорядочении элементов одномерного массива от количества элементов массива n, а также оценку $O(n^2)$ асимптотической сложности алгоритма пузырькового метода.

Литература

[1] Балаева М.О., Кальгин Ю.А., Погорелов Д.А. Исследование сложности и сравнение скорости алгоритмов сортировки методами пузырька, простого выбора, простых вставок. Саратов, Журнал «Научные междисциплинарные исследования». 2021 г., №1, с. 8-12.
[2] Сборник статей IX Международной научно-практической конференции «Научные междисциплинарные исследования», Саратов, Научная общественная организация (НОО) «Цифровая наука», 2021.
[3] Томас Х. Кормен. Алгоритмы. Вводный курс. 2014 г.
[4] Томас Кормен, Чарльз Лейзерсон, Рональд Ривест, Клиффорд Штайн. Алгоритмы: построение и анализ, 3-е изд.: Пер. с англ. - М.: ООО “И. Д. Вильямс”, 2013. – 1328 с.: ил. Парал. тит. англ. ISBN 978-5-8459-1794-2 (рус).
[5] Вирт Н. Алгоритмы и структуры программы // М., Оберон, 2010 г.
[6] Левитин, Ананий В. Алгоритмы: введение в разработку и анализ: Пер. с англ. – М.: здательский дом “Вильямс”, 2006. – 576 с.: ил. - Парал. тит. англ. ISBN 5-8459-0987-2 (рус). (Москва. Санкт-Петербург. Киев.)
[7] Лойко В.И. Структура и алгоритмы обработки данных. Учебное пособие для вузов. – Краснодар: Куб. ГАУ. 2004.- 261 с., ил.
[8] Заир Узаков. Методика асимптотического анализа сложности алгоритма пузырькового метода. O‘zbekiston Respublikasi Mudofaa vazirligi. O‘zbekiston Respublikasi Harbiy aviatsiya instituti. Axborot texnologiyalari kafedrasi. “Ta’limda zamonaviy axborot-kommunikatsiya exnologiyalarini qo’llash afzalliklari, muammolar va yechimlari mavzsidagi Respublika ilmiy-nazariy anjumanining materiallari to'plami. 26-132 betlar. 2025 yil, 26-mart. Qarshi, 2025
[9] З.Узаков. Оценка степени сложности пузырькового алгоритма сортировки. Материалы 40-й международной конференции «Ляпуновские чтения – 2024» (г. Иркутск, 2 – 6 декабря 2024 г.). Федеральное государственное бюджетное учреждение науки Институт динамики систем и теории управления имени В.М. Матросова Сибирского отделения Российской Академии Наук. Стр. 222-224.
[10] Adam Drozdek. Data structures and algorithms in C++. Fourth edition. Cengage Learning. 013 y.
[11] Кнут Д.Э. Искусство программирования. Том 3. Сортировка и поиск. - М.: Вильямс, 2012. – 824 с.
[12] Мадраҳимов Ш.Ф., Гайназаров С.М. С++ тилида программалаш асослари. – Тошкент, 2009. - 160 б.
[13] Подбельский В.В. Язык С++: учеб. пособие. – 5-е изд. – М.: Финансы и статистика, 2007. – 560 с.: ил.
[14] https://codelessons.dev/ru/puzyrkovaya-sortirovka-v-c-glavnye-momenty/
[15] https://uchet-jkh.ru/i/kak-otsortirovat-massiv-po-vozrastaniyu-s/