Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Основы алгоритмизации и программирования (ОАиП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

Тема работы: Инженерный калькулятор

Выполнил

студент: гр. 151004 Иванов И.И.

Проверил: Фадеева Е.П.

Минск 2022

содержание

[Введение 3](#_Toc99212001)

[1 Аналитический обзор литературы 4](#_Toc99212002)

[1.1 Обзор существующих аналогов 4](#_Toc99212003)

[1.1.1 Стандартный калькулятор Windows 4](#_Toc99212004)

[1.1.2 Онлайн-калькулятор Google 5](#_Toc99212005)

[1.1.3 Онлайн-калькулятор «okcalc» 6](#_Toc99212006)

[1.1.4 Онлайн-калькулятор «calculator888» 7](#_Toc99212007)

[1.1.5 Онлайн-калькулятор «kontrolnaya-rabota» 8](#_Toc99212008)

[1.1.6 Онлайн-калькулятор «kalkulyatoronlajn» 9](#_Toc99212009)

[1.2 Анализ методов и способов разработки 10](#_Toc99212010)

[2 Структура данных 13](#_Toc99212011)

[2.1 Структура данных программы 13](#_Toc99212012)

[2.2 Структура данных алгоритма Swap 13](#_Toc99212013)

[2.3 Структура данных алгоритма Fill 13](#_Toc99212014)

[2.4 Структура данных алгоритма BubbleSort 14](#_Toc99212015)

[2.5 Структура данных алгоритма HeapSort 14](#_Toc99212016)

[2.6 Структура данных алгоритма SiftDown 15](#_Toc99212017)

[3 Схема алгоритма решения задачи по ГОСТ 19.701-90 16](#_Toc99212018)

[3.1 Схема алгоритма решения задачи 16](#_Toc99212019)

[3.2 Схема алгоритма Swap 17](#_Toc99212020)

[3.3 Схема алгоритма Fill 18](#_Toc99212021)

[3.4 Схема алгоритма BubbleSort 19](#_Toc99212022)

[3.5 Схема алгоритма HeapSort 20](#_Toc99212023)

[3.6 Схема алгоритма SiftDown 21](#_Toc99212024)

[4 Результаты расчетов 22](#_Toc99212025)

[Приложение А 23](#_Toc99212026)

[Приложение Б 27](#_Toc99212027)

# Введение

Калькулятор — электронное вычислительное устройство для выполнения операций над числами или алгебраическими формулами. Термином «калькулятор» также называют программы, работающие по принципу данного устройства.

Историю вычислительных машин, в том числе и калькуляторов, традиционно начинают с суммирующей машины Паскаля, созданной в 1643 году Блезом Паскалем, и арифмометра Лейбница, изобретённого в 1673 году немецким математиком Готфридом Вильгельмом Лейбницем.

С изобретением калькулятора темпы роста технического прогресса резко возросли. Устройство позволило проводить более сложные измерения за меньший промежуток времени. В зависимости от функционала и сферы применения калькуляторы делятся на простейшие, бухгалтерские, инженерные, финансовые и др.

Цель данной курсовой работы – разработать программное средство инженерный калькулятор с возможностью обработки выражений, определить функционал.

Данная пояснительная записка содержит следующие основные разделы:

1. Анализ прототипов, литературных источников и формирование требований к проектируемому программному средству;
2. Анализ требований к программному средству и разработка функциональных требований;
3. Проектирование программного средства;
4. Создание (конструирование) программного средства;
5. Тестирование и проверка работоспособности программного средства.

# Аналитический обзор литературы

## Обзор существующих аналогов

### Стандартный калькулятор Windows

Это программное средство целесообразно рассматривать как один из ведущих аналогов, поскольку оно поставляется вместе с операционной системой Windows.



Список выполняемых функций:

1. Ввод чисел;
2. Возможность редактирования и очистки введённого текста;
3. Выполнение различных операций над числами;
4. Работа с целыми и дробными числами;
5. Отображение истории действий.

Достоинства:

1. Выполнение нескольких операций одновременно в соответствии с приоритетом;
2. Возможность работать как с градусами, так и с радианами;
3. Наличие прямых тригонометрических и гиперболических функций;
4. Наличие автоматической очистки поля при вводе нового числа.

Недостатки:

1. Отсутствие некоторых прямых тригонометрических функций, таких как синус-верзус, косинус-верзус, гаверсинус, гаверкосинус, эксеканс, экскосеканс;
2. Отсутствие обратных тригонометрических функций;
3. Отсутствие быстрых степенных функций и двойного факториала;
4. Интуитивно сложный для понимания режим ввода чисел с плавающей точкой;
5. Неверное название десятичного логарифма, вводящее пользователя в заблуждение.

### Онлайн-калькулятор Google

Это программное средство целесообразно рассматривать как один из ведущих аналогов, поскольку оно встроено в поисковую страницу Google и показывается пользователю, когда он вводит арифметический запрос в поисковую строку.



Список выполняемых функций:

1. Ввод чисел;
2. Возможность редактирования и очистки введённого текста;
3. Выполнение различных операций над числами;
4. Работа с целыми и дробными числами;
5. Отображение истории действий.

Достоинства:

1. Выполнение нескольких операций одновременно в соответствии с приоритетом;
2. Возможность работать как с градусами, так и с радианами;
3. Наличие прямых тригонометрических функций;
4. Наличие автоматической очистки поля при вводе нового числа.

Недостатки:

1. Отсутствие некоторых прямых тригонометрических функций, таких как синус-верзус, косинус-верзус, гаверсинус, гаверкосинус, эксеканс, экскосеканс;
2. Отсутствие обратных тригонометрических функций;
3. Отсутствие гиперболических функций;
4. Отсутствие быстрых степенных функций и двойного факториала;
5. Отсутствие ввода чисел с плавающей точкой;
6. Отсутствие очистки целого поля, необходимо стирать по одному символу;
7. Неверное название десятичного логарифма, вводящее пользователя в заблуждение.

### Онлайн-калькулятор «okcalc»

Это программное средство целесообразно рассматривать как один из ведущих аналогов, поскольку оно занимает лидирующее положение в поиске Google по запросу “инженерный калькулятор онлайн”.



Список выполняемых функций:

1. Ввод чисел;
2. Возможность редактирования и полной очистки введённого текста;
3. Выполнение арифметических операций;
4. Работа с целыми и дробными числами;

Достоинства:

1. Выполнение нескольких операций одновременно в соответствии с приоритетом;
2. Наличие быстрых степенных функций.

Недостатки:

1. Отсутствие прямых тригонометрических функций;
2. Отсутствие обратных тригонометрических функций;
3. Отсутствие гиперболических функций;
4. Отсутствие факториалов;
5. Отсутствие ввода чисел с плавающей точкой;
6. Отсутствие автоматической очистки поля при вводе нового числа;
7. Неверное название десятичного логарифма, вводящее пользователя в заблуждение;
8. Отсутствие истории действий.

Очевидно, что стандартный калькулятор Windows имеет наименьшее количество недостатков, поэтому он будет являться ориентиром при разработке собственного программного средства.

### Онлайн-калькулятор «calculator888»

Это программное средство целесообразно рассматривать как один из ведущих аналогов, поскольку оно появляется на первой странице поиска Google по запросу “инженерный калькулятор онлайн”.



Список выполняемых функций:

1. Ввод чисел;
2. Возможность редактирования и полной очистки введённого текста;
3. Выполнение арифметических операций;
4. Работа с целыми и дробными числами;

Достоинства:

1. Выполнение нескольких операций одновременно в соответствии с приоритетом;
2. Наличие быстрых степенных функций.
3. Наличие некоторых прямых и обратных тригонометрических функций.

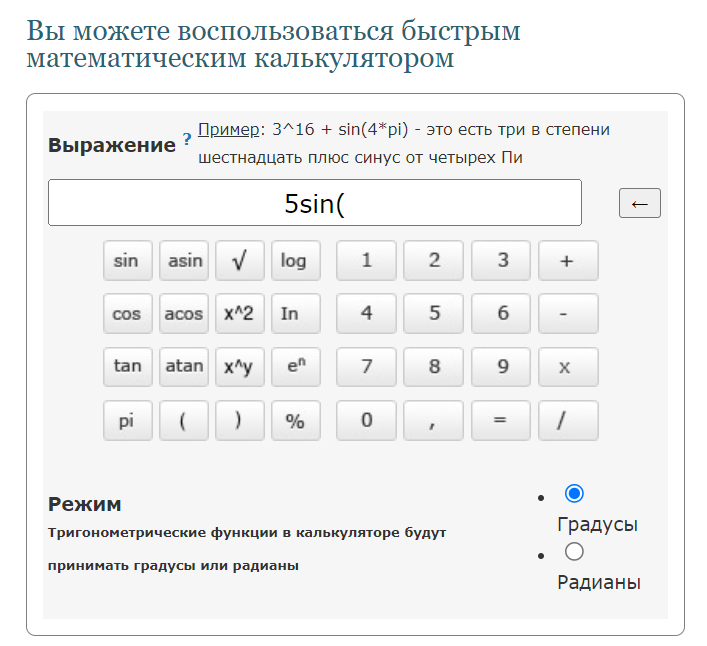
Недостатки:

1. Отсутствие некоторых прямых тригонометрических функций;
2. Отсутствие некоторых обратных тригонометрических функций;
3. Отсутствие гиперболических функций;
4. Отсутствие факториалов;
5. Отсутствие ввода чисел с плавающей точкой;
6. Отсутствие истории действий.

Очевидно, что стандартный калькулятор Windows имеет наименьшее количество недостатков, поэтому он будет являться ориентиром при разработке собственного программного средства.

### Онлайн-калькулятор «kontrolnaya-rabota»

Это программное средство целесообразно рассматривать как один из ведущих аналогов, поскольку оно появляется на первой странице поиска Google по запросу “инженерный калькулятор онлайн”.



Список выполняемых функций:

1. Ввод чисел;
2. Возможность редактирования и полной очистки введённого текста;
3. Выполнение арифметических операций;
4. Работа с целыми и дробными числами;

Достоинства:

1. Выполнение нескольких операций одновременно в соответствии с приоритетом;
2. Наличие быстрых степенных функций.
3. Наличие некоторых прямых и обратных тригонометрических функций;
4. Наличие режима градусов и радианов.

Недостатки:

1. Отсутствие некоторых прямых тригонометрических функций;
2. Отсутствие некоторых обратных тригонометрических функций;
3. Отсутствие гиперболических функций;
4. Отсутствие факториалов;
5. Отсутствие ввода чисел с плавающей точкой;
6. Отсутствие истории действий.
7. Неудобная реализация операций над одним числом, требующая обязательного закрытия скобки

### Онлайн-калькулятор «kalkulyatoronlajn»

Это программное средство целесообразно рассматривать как один из ведущих аналогов, поскольку оно появляется на первой странице поиска Google по запросу “инженерный калькулятор онлайн”.



Список выполняемых функций:

1. Ввод чисел;
2. Возможность редактирования и полной очистки введённого текста;
3. Выполнение арифметических операций;
4. Работа с целыми и дробными числами;

Достоинства:

1. Выполнение нескольких операций одновременно в соответствии с приоритетом;
2. Наличие быстрых степенных функций.
3. Наличие некоторых прямых тригонометрических функций;

Недостатки:

1. Отсутствие некоторых прямых тригонометрических функций;
2. Отсутствие обратных тригонометрических функций;
3. Отсутствие гиперболических функций;
4. Отсутствие факториалов;
5. Отсутствие ввода чисел с плавающей точкой;
6. Отсутствие истории действий.
7. Неудобная реализация операций над одним числом, требующая обязательного закрытия скобки

Очевидно, что стандартный калькулятор Windows имеет наименьшее количество недостатков, поэтому он будет являться ориентиром при разработке собственного программного средства.

## Анализ методов и способов разработки

Предполагается, что данное средство, помимо выполнения арифметических и тригонометрических операций над числами, будет выполнять ряд дополнительных функций, таких как сохранение истории действий в текстовый файл, чтение истории действий из текстового файла, чтение и массовая операция над всеми числами из текстового файла.

В данном программном средстве будут использоваться две основные структуры даных:

* записи;
* массивы;
* множества;
* файлы.

Запись - это структура данных, состоящая из фиксированного числа элементов, которые являются полями. В пределах записи имя поля должно быть уникальным. Описание записи происходит с помощью зарезервированного слова record и дает возможность объединять данные разных типов вместе.

Массивом называется упорядоченная совокупность однотипных элементов, у которых общее имя. Массивы в языке Delphi во многом схожи с аналогичными  типами данных в других языках программирования. В состав элементов массива могут входить данные любых типов, в том числе и структурированные. Любой элемент массива определяется именем массива и индексом (индекс может быть не один, тогда данный массив будет многомерным). В качестве индексных типов в языке Delphi используются перенумерованные типы. Обычно используется тип Integer.

Множество - это набор элементов (объектов) одного и того же типа. Оно может состоять не более чем из 256 элементов, но может и не иметь ни одного элемента (пустое множество). Все элементы множества должны быть различны (если вы и попытаетесь до­бавить элемент, который уже имеется в множестве, то просто ничего не произойдёт). Объявляются множества с помощью зарезервированных слов set и of, по­сле которых указывается тип элементов (базовый тип). Базовым типом могут быть: логический тип, символьный тип char, пере­числяемый и интервальный типы, содержащие не более 256 элементов. Типы целых и вещественных чисел не могут быть базовым типом, по­скольку они заведомо содержат более 256 элементов.

Файловый тип - это произвольная последовательность элементов, длина которой заранее не определена, а конкретизируется в процессе выполнения программы. Это определение логического файла, т.е. того, который использу-ется в программе (файл с точки зрения программиста). Физический файл (набор данных) - это именованная область памяти на внешнем носителе, в ко-торой хранится некоторая информация (файл с точки зрения пользователя).

B Delphi существует три типа файлов:

* типизированные файлы;
* текстовые файлы;
* нетипизированные файлы.

Типизированные файлы связываются с файловыми переменными, объ-явленными как file of <Тип>. Файл считается состоящим из элементов, каж-дый из которых имеет тип «Тип>.

Нетипизированные файлы могут быть связаны только с теми файловы-ми переменными, которые были объявлены как file. Файл считается состоя-щим из элементов, размер которых определяется при открытии файла.

Текстовый файл представляет собой последовательность символов, од-нако он не эквивалентен файлу типа file of Char. Файл типа file of Char - это единая последовательность символов. Текстовые файлы связываются с фай-ловыми переменными, принадлежащими стандартному типу TextFile. Особен-ность текстовых файлов состоит в том, что содержащиеся в них символы раз-биваются на строки. Строки могут быть различной длины (пустые в том чис-ле). В конце каждой строки помещается специальный управляющий символ: возврат каретки (#13 или ^М - международное обозначение CR) и перехода новую строку (#10 или ^J - международное обозначение LF). С наличием это-го маркера связана логическая функция Eoln (End of line).

Для доступа к отдельным элементам файлов в Delphi существуют спе-циальные стандартные процедуры и функции. Их называют процедурами и функциями ввода-вывода. Обращение к ним осуществляется обычным образом.

Процедура AssignFile(F, Name) связывает файловую переменную с внешним файлом на диске. Здесь F - имя файловой переменной любого типа; Name - выражение строкового типа. Назначение процедуры следующее: организация связи между конкретным физическим файлом на внешнем устройстве (конкретным набором данных) и файловой переменной (логическим файлом) F.

Процедура Reset(F) открывает существующий файл F в режиме чтения. Фактически при этом открывается внешний файл с именем, присвоенным переменной F процедурой AssignFile. Если файл с данным именем не существует, возникает исключение. Указатель файла устанавливается на первую позицию файла. Если файл был предварительно открыт, то он закрывается и повторно открывается. Процедура Reset(F) может быть применена к файлулюбое количество раз. При выполнении данной процедуры содержимое файла не изменяется. Для текстовых файлов - в режиме только для чтения (read-only).

Процедура Rewrite(F) создает новый файл и открывает его. Если файл уже открыт, то закрывает и снова открывает его. Указатель файла устанавливается на начало файла (файл создается пустым). Для текстовых файлов - в режиме только для записи (write-only).

Процедура Append(F) предопределена только для текстовых файлов. Она открывает существующий файл для добавления. Если файл уже открыт, то закрывает его, а затем вновь открывает. Если файл с данным именем не существует, возникает исключение. В данном случае указатель файла устанавливается на конец файла. Файл открывается в режиме только для записи (write-only).

Процедура CloseFile(F) закрывает файл. В данном случае F - файловая переменная, открытая с помощью Reset(F), Rewrite(F) или Append(F).

Функция Eof(F) проверяет, является ли текущей позицией конец файла, и возвращает результат логического типа. В данном случае F - файловая переменная, открытая с помощью Reset(F) или Rewrite(F).

Функция Eoln(F) проверяет, является ли текущей позицией конец строки, и возвращает результат логического типа. В данном случае F - файловая переменная, открытая с помощью Reset(F), Rewrite(F) или Append(F).

Для файлов с типом предопределены процедуры считывания компонентов файла в переменные и записи в файл компонентов из переменных Read (F, V1, V2, …, Vn) и Write (F, V1, V2, …, Vn).

Для файлов без типа есть аналоги процедур Read и Write. Это процедуры BlockRead (F, Buf, Count [, Done]) и BlockWrite (F, Buf, Count [, Done]).

Также есть функции и процедуры, перемещающие указатель файла в указанную позицию, возвращающие текущее положение указателя файла в байтах, возвращающие размер файла в компонентах и др.

# Структура данных

## Структура данных программы

Таблица 2 – Структура данных

|  |  |  |
| --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение |
| I, J | Integer | Параметры цикла |
| Comp | Integer | Число сравнений |
| Arr1, Arr2 | Array[1..3000] Of Integer | Массивы для сортировки |
| N | Array[1..6] Of Integer | Массив из размерностей массивов |
| S | Array[1..3] Of String | Массив из имён массивов |

## Структура данных алгоритма Swap

Таблица 3 – Структура данных алгоритма Swap(A, B)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| A | Integer | Обмениваемое число | Формальный |
| B | Integer | Обмениваемое число | Формальный |
| T | Integer | Сохранённое число | Локальный |

## Структура данных алгоритма Fill

Таблица 4 – Структура данных алгоритма Fill(Arr, N, Opt)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Arr | Array[1..3000] | Массив | Формальный |
| N | Integer | Количество элементов массива | Формальный |
| Opt | Integer | Выбор условия, по которому заполняется массив | Формальный |
| I | Integer | Параметр цикла | Локальный |

## Структура данных алгоритма BubbleSort

Таблица 5 – Структура данных алгоритма BubbleSort(Arr, N, Comp)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Arr | Array[1..3000] | Массив | Формальный |
| N | Integer | Количество элементов массива | Формальный |
| Comp | Integer | Число сравнений | Формальный |
| I, J | Integer | Параметры цикла | Локальный |
| Sorted | Boolean | «Флажок» сортировки | Локальный |

## Структура данных алгоритма HeapSort

Таблица 6 – Структура данных алгоритма HeapSort (Arr, N, Comp)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Arr | Array[1..3000] | Массив | Формальный |
| N | Integer | Количество элементов массива | Формальный |
| Comp | Integer | Число сравнений | Формальный |
| NodeCurr | Integer | Текущий индекс | Локальный |
| NodeLast | Integer | Сохранённый индекс | Локальный |

## Структура данных алгоритма SiftDown

Таблица 7 – Структура данных алгоритма SiftDown(Arr, Node, N, Comp)

|  |  |  |  |
| --- | --- | --- | --- |
| Элементы данных | Рекомендуемый тип | Назначение | Тип параметра |
| Arr | Array[1..3000] | Массив | Формальный |
| Node | Integer | Текущий индекс | Формальный |
| N | Integer | Количество элементов массива | Формальный |
| Comp | Integer | Число сравнений | Формальный |
| Root | Integer | Текущий индекс | Локальный |
| Child | Integer | Дочерний индекс | Локальный |
| Sifted | Boolean | Закончена ли сборка дерева | Локальный |

# Схема алгоритма решения задачи по ГОСТ 19.701-90

## Схема алгоритма решения задачи

|  |
| --- |
|  |
| Рисунок 1 – Схема алгоритма решения задачи |

## Схема алгоритма Swap

|  |
| --- |
|  |
| Рисунок 2 – Схема алгоритма Swap |

## Схема алгоритма Fill

|  |
| --- |
|  |
| Рисунок 3 – Схема алгоритма Fill |

## Схема алгоритма BubbleSort

|  |
| --- |
|  |
| Рисунок 4 – Схема алгоритма BubbleSort |

## Схема алгоритма HeapSort

|  |
| --- |
|  |
| Рисунок 5 – Схема алгоритма HeapSort |

## Схема алгоритма SiftDown

|  |
| --- |
|  |
| Рисунок 6 – Схема алгоритма SiftDown |

# Результаты расчетов

Вследствие результатов программы мы получаем следующие результаты:

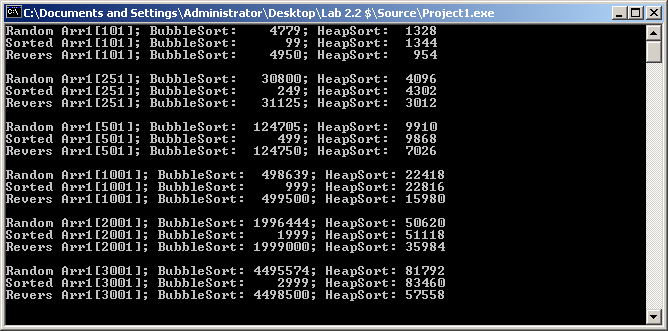


Рисунок 7 – Результаты расчетов

Приложение А

(обязательное)

Исходный код программы (постановка 1)

Program Lab2;

{Comp sorts}

//Use app

{$APPTYPE CONSOLE}

//Declare modules

Uses

SysUtils;

//Declare types

Type

TArr = Array[1..3000] Of Integer;

//TArr - our array

//Declare Vars

Var

I, J: Integer;

Comp: Integer;

Arr1, Arr2: TArr;

N: Array[1..6] Of Integer = (100, 250, 500, 1000, 2000, 3000);

S: Array[1..3] Of String = ('Random', 'Sorted', 'Revers');

//I,J - loop params

//N - array sizes

//Comp - quantity of comparisons

//Arr1, Arr2 - our arrays

//N - num of elements

//S - name of array

//Swaps 2 elements

//A, B - elements

Procedure Swap(Var A, B: Integer);

Var

T: Integer;

//T - temp

Begin

T:= A;

A:= B;

B:= T;

End;

//Fills array with elements

//Arr - array, N - array size, Opt - filler type

Procedure Fill(Var Arr: TArr; Const N, Opt: Integer);

Var

I: Integer;

//I - select type of filling

Begin

Randomize;

Case Opt Of

1:

For I:= Low(Arr) To N Do

Arr[I]:= Random(N);

2:

For I:= Low(Arr) To N Do

Arr[I]:= I;

3:

For I:= Low(Arr) To N Do

Arr[I]:= N - I;

End;

End;

//Sorts an array and calculates the number of comparisons

//Arr - array, N - array size, Comp - comparisons

Procedure BubbleSort(Var Arr: TArr; Const N: Integer; Var Comp: Integer);

Var

I, J: Integer;

Sorted: Boolean;

//I, J - loop params

//Sorted - condition to exit

Begin

Sorted:= False;

I:= 1;

While (I <= N - 1) And Not Sorted Do

Begin

Sorted:= True;

For J:= 1 To N - I Do

Begin

If Arr[J] > Arr[J + 1] Then

Begin

Sorted:= False;

Swap(Arr[J], Arr[J + 1]);

End;

Inc(Comp);

End;

Inc(I);

End;

End;

//Sorts an array and calculates the number of comparisons

//Arr - array, Node - current index, N - array size, Comp - comparisons

Procedure SiftDown(Var Arr: TArr; Node: Integer; Const N: Integer; Var Comp: Integer);

Var

Root, Child: Integer;

Sifted: Boolean;

//Root, Child - indexes

//Sifted - condition to exit

Begin

Root:= Node;

Sifted:= False;

While (Not Sifted) And (Root \* 2 - Node + 1 <= N) Do

Begin

Comp:= Comp + 2;

Child:= Root \* 2 - Node + 1;

If (Child + 1 <= N) And (Arr[Child] < Arr[Child + 1]) Then

Inc(Child);

If Arr[Root] < Arr[Child] Then

Begin

Swap(Arr[Root], Arr[Child]);

Root:= Child;

End

Else

Sifted:= True;

End;

End;

//Sorts an array and calculates the number of comparisons

//Arr - array, N - array size, Comp - comparisons

Procedure HeapSort(Var Arr: TArr; Const N: Integer; Var Comp: Integer);

Var

NodeLast, NodeCurr: Integer;

//NodeLast, NodeCurr - indexes

//Comps - counter

Begin

NodeCurr:= N Div 2 - 1;

While NodeCurr >= Low(Arr) Do

Begin

SiftDown(Arr, NodeCurr, N, Comp);

Dec(NodeCurr);

End;

NodeLast:= N;

While NodeLast > Low(Arr) Do

Begin

Swap(Arr[Low(Arr)], Arr[NodeLast]);

Dec(NodeLast);

SiftDown(Arr, Low(Arr), NodeLast, Comp);

End;

End;

Begin

For I:= 1 To 6 Do

Begin

For J:= 1 To 3 Do

Begin

Fill(Arr1, N[I], J);

//Show arr size

Write(S[J], ' Arr[', N[I], ']; ');

//Copy array

Arr2:= Arr1;

//Enzero to show true value

Comp:= 0;

BubbleSort(Arr1, N[I], Comp);

Write('BubbleSort: ', Comp: 7, '; ');

//Enzero to show true value

Comp:= 0;

HeapSort(Arr2, N[I], Comp);

Write('HeapSort: ', Comp: 5);

WriteLn;

End;

WriteLn;

End;

ReadLn;

End.

Приложение Б

(обязательное)

Тестовые наборы

Формула числа сравнений для BubbleSort: C = (N2-N)/2

Поскольку в постановке задачи применялся не простой BubbleSort, а BubbleSort с флажком, то отсортированный массив будет проверен всего 1 раз, а количество сравнений будет высчитываться не по обычной формуле, а по формуле C = N – 1.

Формула среднего числа сравнений для HeapSort: C = 2 \* N \* log2N

Результаты на практике подтверждают теоретические результаты, рассчитанные в Microsoft Excel.

|  |  |
| --- | --- |
|  |  |