Введение

Цель - Разработка программы на языке Pascal для сортировки одномерного массива целых чисел в порядке возрастания. Основной задачей является корректная реализация алгоритма сортировки, проверка его эффективности и точности на различных входных данных, а также анализ времени выполнения программы при сортировке массивов разной длины.

Задачи:

- Составить IDEF0-диаграмму проекта;
- Составить блок-схемы алгоритмов;
- Реализовать алгоритмы на языке Object Pascal;
- Протестировать реализации алгоритмов.

Аналитическая часть

Структура представлена на рисунке 1 в виде IDEF0-нотации. Задача разделяется на несколько блоков, представленных на рисунке 2:

- Ввод данных с клавиатуры;
- Выполнение вычисления;
- Создание таблицы;
- Вывод данных на экран.

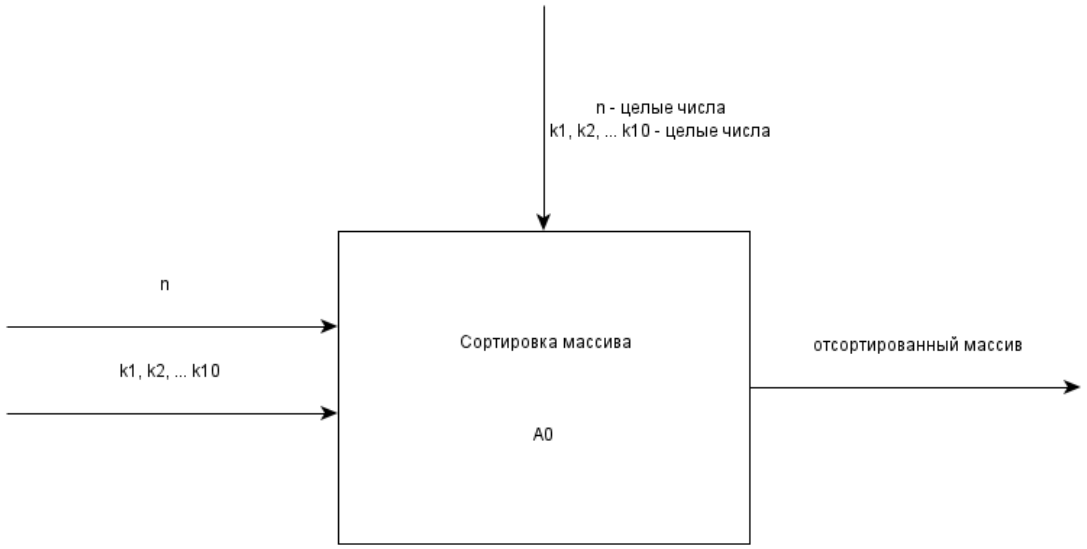

Рисунок 1 - Общая IDEF0-Нотация

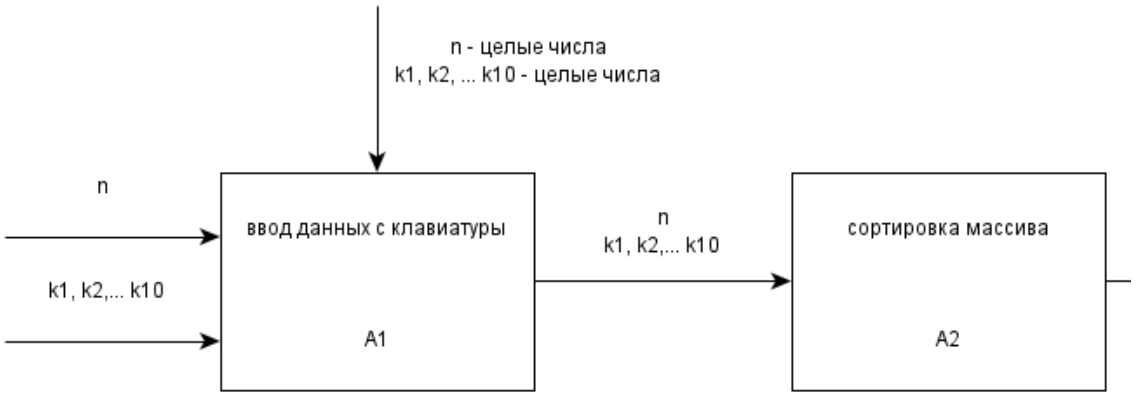

Рисунок 2 - Подробная IDEF0-Нотация

Конструкторская часть

Блоки IDEF0-диаграммы представляют собой 1-2 действия, ввиду чего рациональнее отобразить алгоритм всего проекта целиком без разбиения каждого блока на отдельные процедуры.

Блок-схема алгоритма представлена на рисунке 1. Элементы, отвечающие за интерфейс пользователя, на блок-схеме не отображены; текстовые сообщения, ввиду малозначимости их дословного приведения, представлены сокращенно.

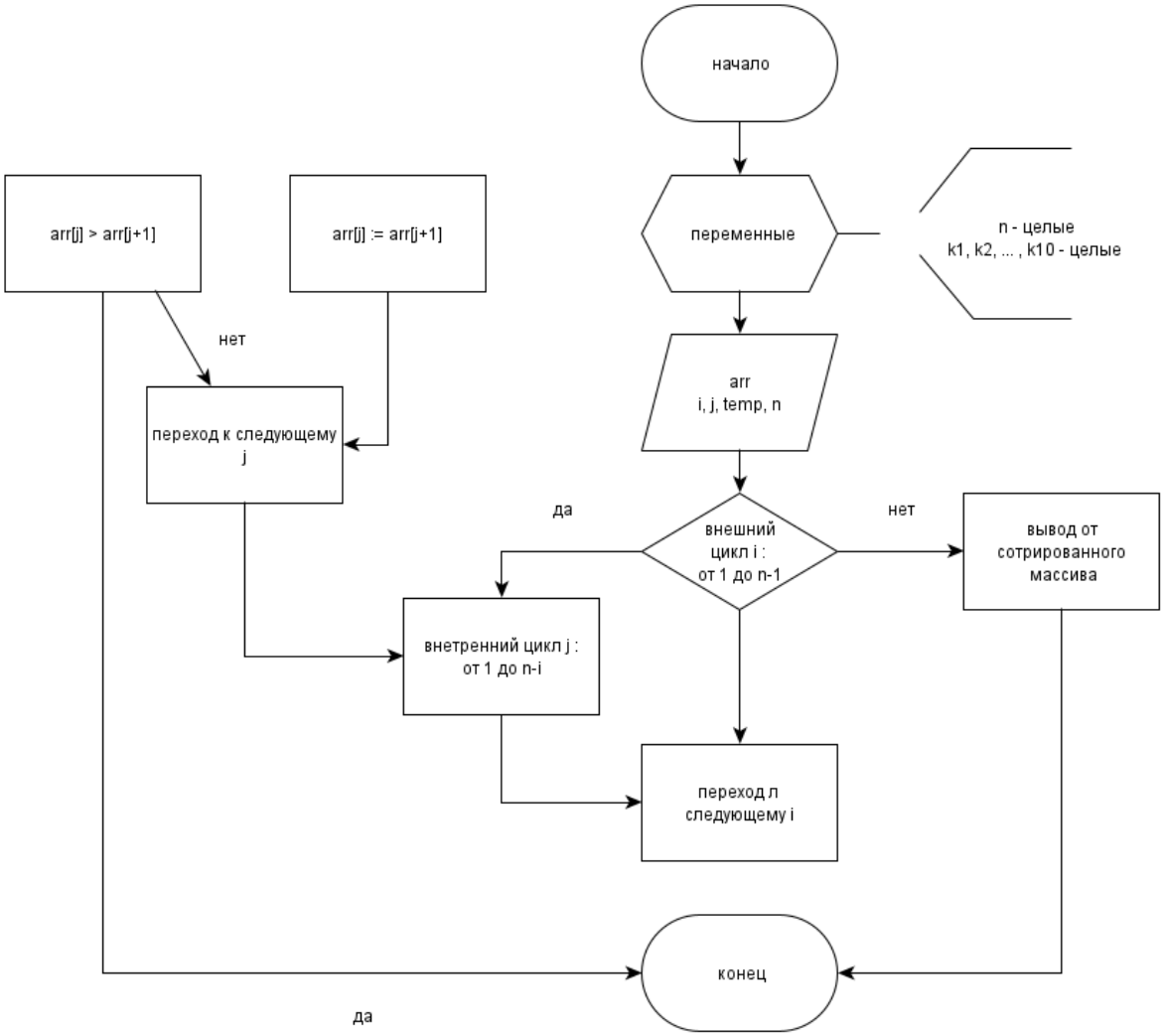

Рисунок 1 - Блок-схема алгоритма программы

- # Технологическая часть

Реализация алгоритма

В настоящем разделе представлена реализация алгоритма, чья блок-схема представлена на рисунке 1. Реализация была произведена с помощью языка программирования Pascal и представлена в листинге 1.

Листинг 1 - Программа алгоритма

```
program SortArray;
var
  arr: array[1..10] of integer;
  i, j, temp, n: integer;
begin
  { Ввод количества элементов массива }
  write('Введите количество элементов массива (до 10): ');
  readln(n);

  { Ввод элементов массива }
  writeln('Введите элементы массива:');
  for i := 1 to n do
    readln(arr[i]);

  { Сортировка массива методом пузырька }
  for i := 1 to n - 1 do
    for j := 1 to n - i do
      if arr[j] > arr[j + 1] then
      begin
        temp := arr[j];
        arr[j] := arr[j + 1];
        arr[j + 1] := temp;
      end;

  { Вывод отсортированного массива }
  writeln('Отсортированный массив:');
  for i := 1 to n do
    write(arr[i], ' ');
  writeln;
end.
```

Тестирование реализации

В данном разделе мы рассмотрим процесс тестирования программы, реализующей сортировку массива. Для тестирования использовались два метода: метод эквивалентного разбиения (черный ящик) и метод комбинаторного покрытия условий и решений (белый ящик).

Метод эквивалентного разбиения

Метод эквивалентного разбиения предполагает разбиение входных данных на несколько классов эквивалентности, где каждое тестовое значение представляет целый класс. Это позволяет провести тестирование на ограниченном наборе данных, охватывая все возможные ситуации.

Для вашей задачи сортировки массива можно выделить несколько классов эквивалентности:

1. Пустой массив.
2. Массив с одним элементом.
3. Массив, в котором все элементы одинаковы.
4. Массив уже отсортирован в порядке возрастания.
5. Массив отсортирован в порядке убывания.
6. Массив с произвольным набором положительных и отрицательных чисел.

Метод комбинаторного покрытия условий и решений

Метод комбинаторного покрытия условий и решений заключается в том, чтобы протестировать комбинации различных условий, которые могут повлиять на поведение программы. Для задачи сортировки массива можно рассмотреть такие параметры:

- **Размер массива**: пустой, один элемент, несколько элементов.
- **Порядок элементов**: возрастающий, убывающий, произвольный.
- **Тип элементов**: все положительные, все отрицательные, смешанные.

Результаты тестирования

Созданная таблица для тестирования будет включать тестовые случаи, которые покрывают все комбинации этих условий.

Тест N.	Размер массива	Порядок элементов	Тип элементов	Входной массив	Ожидаемый результат	Примечания
1	Пустой	-	-	[]	[]	Пустой массив
2	Один элемент	-	Положительные	[5]	[5]	Один положительный элемент
3	Несколько элементов	Возрастающий	Положительные	[1,2,3,4,5]	[1,2,3,4,5]	Уже отсортирован
4	Несколько элементов	Убывающий	Положительные	[5,4,3,2,1]	[1,2,3,4,5]	Обратный порядок
5	Несколько элементов	Произвольный	Положительные	[3,1,4,2,5]	[1,2,3,4,5]	Произвольный порядок
6	Несколько элементов	Произвольный	Смешанные	[0,-1,3,-2,4,-3]	[-3,-2,-1,0,3,4]	Смешанные положительные и отрицательные
7	Несколько элементов	Произвольный	Повторяющиеся элементы	[2,2,1,3,3,1]	[1,1,2,2,3,3]	Повторяющиеся элементы

Заключение

В настоящей работе была составлена программа для сортировки массива.

Задачи :

- Составлена IDEF0-диаграмма проекта;
- Составлены блок-схемы алгоритмов;
- Алгоритмы реализованы на языке Object Pascal;
- Реализации алгоритмов протестированы, были подобраны классы ошибок.