ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ УФИМСКИЙ УНИВЕРСИТЕТ НАУКИ И ТЕХНОЛОГИЙ

Кафедра ВМиК

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1**

по предмету **«Языки программирования»**

Выполнил: студент группы МО-203б

Ярошко Е. В.

Проверил:  
ст. преподаватель

Рипатти А. В.

**Уфа 2025 г.**

**Цель работы:**

Изучить основные подходы к сортировке массивов, а также способы визуализации процесса сортировки.

**Задачи работы:**

- изучить методы сортировки массивов;

- освоить понятия устойчивых и неустойчивых сортировок;

- изучить возможность визуализации процессов, используя выделение цветом и форматированный вывод, покадровый вывод (задержку экрана и очистку консоли)

**Задание:**

Реализовать 3 метода сортировки массива с визуализацией.

Размерность массива может быть задана пользователем или константно, в диапазоне от 10 до 15 элементов. Значение элементов массива разыграть случайным образом, предусмотреть возможность корректировки пользователем предельных значений(границ диапазона).

Визуализация процесса сортировки должна отображать цветом или другим способом (кроме выделения скобками и т.п.) все сравнения и все перемещения элементов массива. При визуализации процесса сортировки можно использовать стандартные функции:

-задержка экрана;

- выделение символа и/или подложки цветом;

- очистка консоли;

-сдвиг курсора;

- и т.д.

**Теоретические основы:**

В данной лабораторной работе впервые используются понятия ANSI escape-последовательностей для форматирования текста в терминале. В рамках данной лабораторной работы используются ANSI-escape последовательности для окраски текста в различные цвета (каждая строка с данной последовательностью в листинге программы снабжена комментарием со значением данной последовательности), а также для очистки строки в терминале. Данный подход необходим для визуализации процесса сортировки непосредственно в терминале, без необходимости использования дополнительных графических библиотек.

Для сортировки массивов чисел используются приёмы сортировки: Bubble Sort, Shell Sort, Selection Sort, Bogo Sort.

Bubble Sort сравнивает каждый элемент поочерёдно, и в случае несоответствия меняет их местами. Итерации продолжаются до тех пор, пока каждый символ не будет соответствовать условиям.

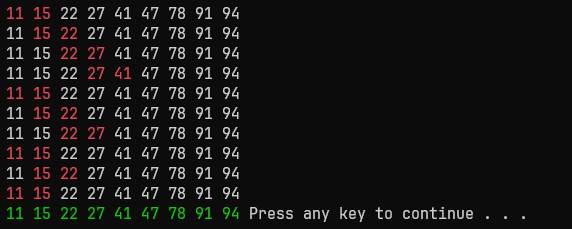
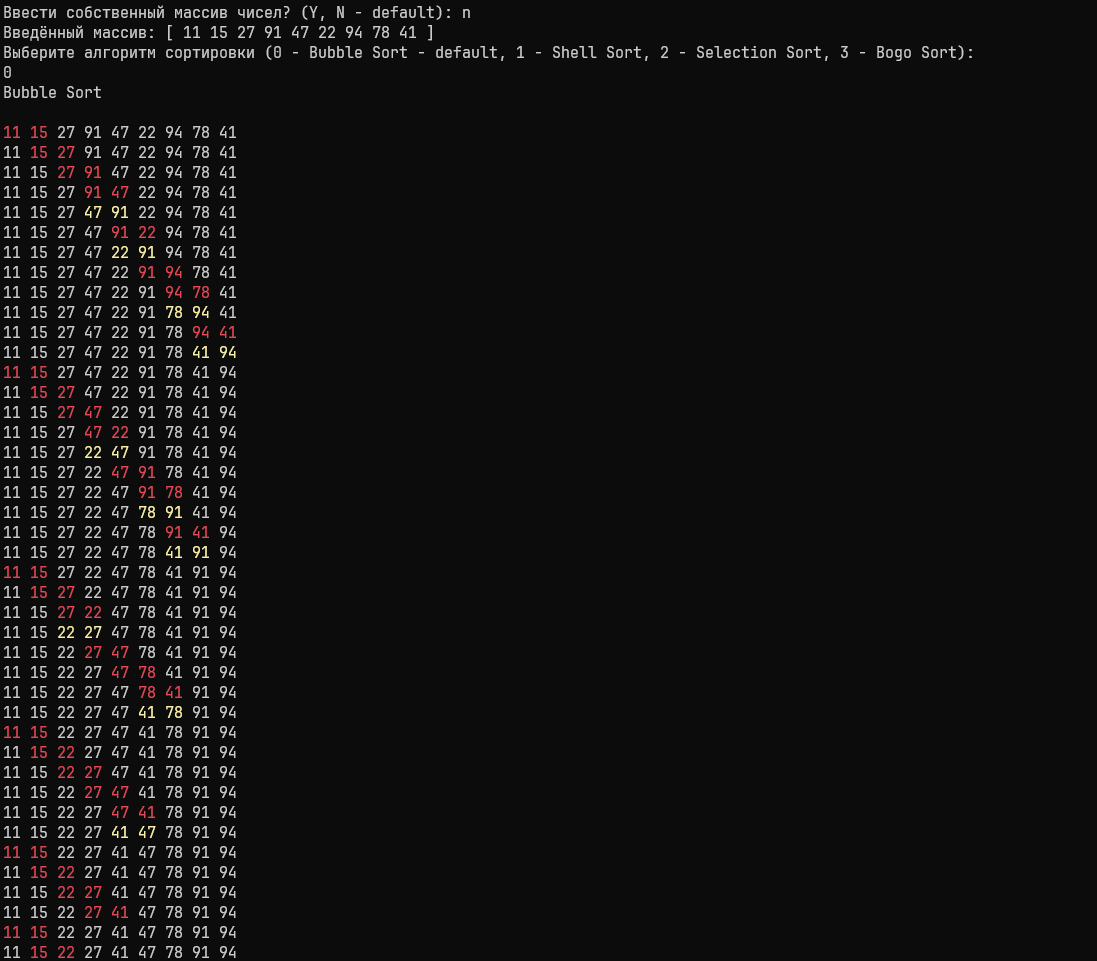
Shell Sort сравнивает каждый элемент с определённым шагом, начинающимся с половины размера массива (array\_size/2). С каждой итерацией шаг уменьшается, тем самым сортируя элементы, не прибегая к поочерёдному сравниваю. Сложность данной сортировки будет меньше по сравнению с Bubble Sort.

Selection Sort выбирает минимальный элемент в массиве и ставит его в начало массива. Процесс продолжается до тех пор, пока последний минимальный элемент не встанет в конец массива. Временная сложность приёма одинакова с Bubble Sort.

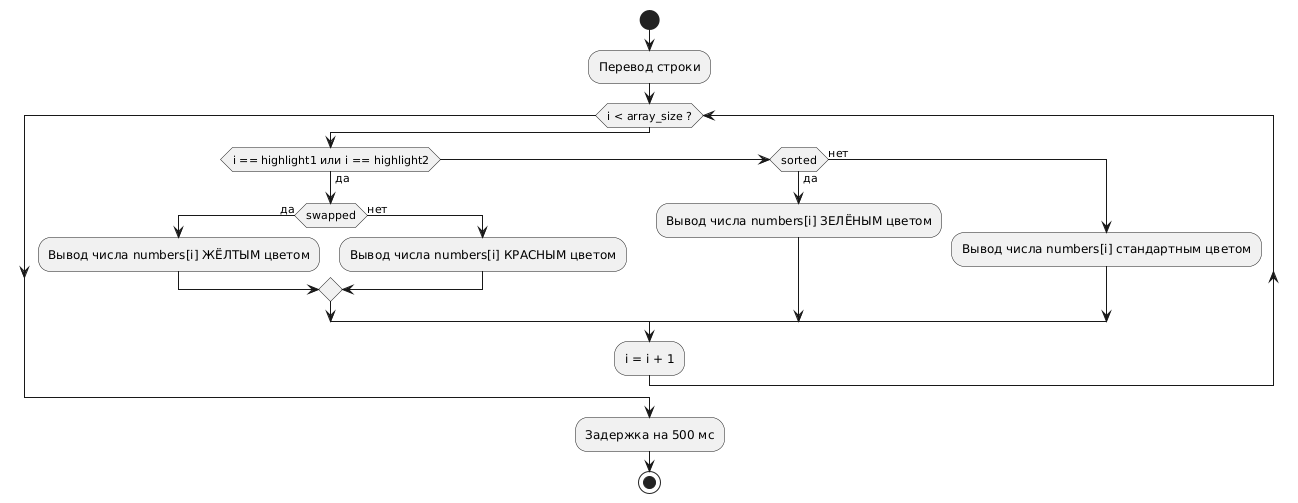
Bogo Sort прибегает к сортировке массива посредством случайной перестановки всех элементов массива. Сортировка продолжается до тех пор, пока каждый элемент массива не встанет на соответствующее место. Временная сложность данного подхода не может быть определена, так как сортировка происходит за счёт случайной перестановки элементов.

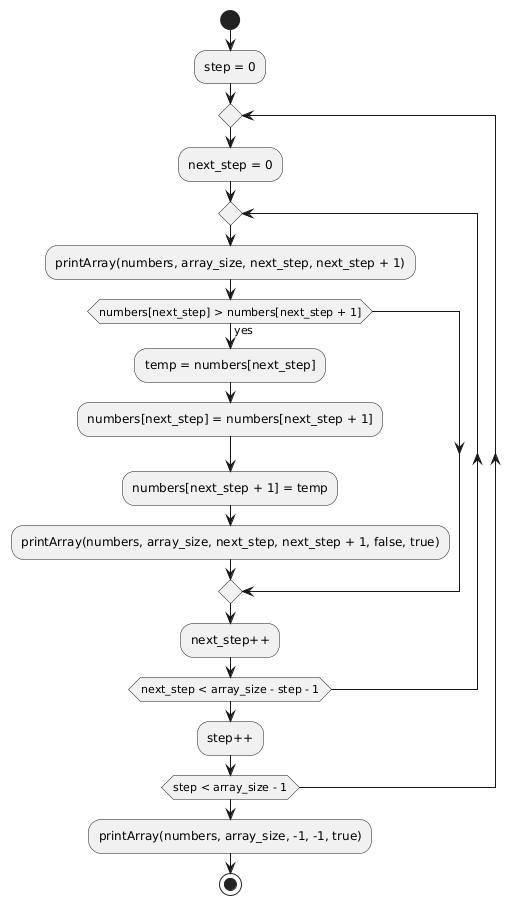
**Листинг программы:**

*#include* <iostream>  
*#include* <thread>  
*#include* <chrono>  
*#include* <locale>  
*#include* <windows.h>  
*#include* <random>  
  
using namespace std;  
  
*// Вывод в терминал*void *printArray*(int *numbers*[], int *array\_size*, int *highlight1* = -1, int *highlight2* = -1, bool *sorted* = false, bool *swapped* = false){  
 cout *<< endl*;  
 for (int i = 0; i < *array\_size*; ++i) {  
 if (i == *highlight1* || i == *highlight2*){  
 if (*swapped*){  
 cout *<<* "\033[1;33m" *<< numbers*[i] *<<* "\033[0m "; *// Жёлтый цвет* }else{  
 cout *<<* "\033[1;31m" *<< numbers*[i] *<<* "\033[0m "; *// Красный цвет* }  
 }else if (*sorted*){  
 cout *<<* "\033[1;32m" *<< numbers*[i] *<<* "\033[0m "; *// Зелёный цвет* }else{  
 cout *<< numbers*[i] *<<* " "; *// Обычный вывод* }  
 }  
 this\_thread::*sleep\_for*(chrono::*milliseconds(*500*)*); *// Задержка 500 мс*}  
  
*// Алгоритмы  
  
  
// Bubble Sort*void *bubbleSort*(int *numbers*[], int *array\_size*){  
 for (int step = 0; step < *array\_size* - 1; step++){  
 for(int next\_step = 0; next\_step < *array\_size* - step - 1; next\_step++){  
 *printArray*(*numbers*, *array\_size*, next\_step, next\_step + 1); *// Выводит промежуточный результат* if(*numbers*[next\_step] > *numbers*[next\_step + 1]){  
 int temp = *numbers*[next\_step];  
 *numbers*[next\_step] = *numbers*[next\_step + 1];  
 *numbers*[next\_step + 1] = temp;  
 *printArray*(*numbers*, *array\_size*, next\_step, next\_step + 1, false, true); *// Выводит промежуточный результат* }  
 }  
 }  
 *printArray*(*numbers*, *array\_size*, -1, -1, true);  
}  
  
*// Shell Sort*void *shellSort*(int *numbers*[], int *array\_size*){  
 for (int gap = *array\_size* / 2; gap > 0; gap /= 2){  
 for (int i = gap; i < *array\_size*; i++){  
 bool swapped = false;  
 int temp = *numbers*[i];  
 int j = i;  
 *printArray*(*numbers*, *array\_size*, i, i - gap);  
 for (j; j >= gap; j -= gap){  
 if(gap && *numbers*[j - gap] > temp){  
 swapped = true;  
 *numbers*[j] = *numbers*[j - gap];  
 }else{  
 break;  
 }  
 }  
 *numbers*[j] = temp;  
 if (swapped){  
 *printArray*(*numbers*, *array\_size*, i, i - gap, false, true);  
 }  
 }  
 }  
 *printArray*(*numbers*, *array\_size*, -1, -1, true);  
}  
  
*// Selection Sort*void *selectionSort*(int *numbers*[], int *array\_size*){  
 for (int step = 0; step < *array\_size* - 1; step++){  
 int min\_idx = step;  
 for (int next\_step = step + 1; next\_step < *array\_size*; next\_step++){  
 *printArray*(*numbers*, *array\_size*, min\_idx, next\_step);  
 if (*numbers*[next\_step] < *numbers*[min\_idx]){  
 min\_idx = next\_step;  
 }  
 }  
 if(min\_idx != step){  
 int temp = *numbers*[min\_idx];  
 *numbers*[min\_idx] = *numbers*[step];  
 *numbers*[step] = temp;  
 *printArray*(*numbers*, *array\_size*, min\_idx, step, false, true);  
 }  
 }  
 *printArray*(*numbers*, *array\_size*, -1, -1, true);  
}  
  
*// Bogo Sort*void *bogoSort*(int *numbers*[], int *array\_size*){  
 while (true){  
 bool sorted = true;  
 for (int i = 0; i < *array\_size* - 1; i++){  
 if (*numbers*[i] > *numbers*[i + 1]){  
 sorted = false;  
 break;  
 }  
 }  
 if (sorted){  
 *printArray*(*numbers*, *array\_size*, -1, -1, true);  
 break;  
 }  
  
 int random\_index = *rand*() % *array\_size*;  
 int another\_random\_index = *rand*() % *array\_size*;  
 *printArray*(*numbers*, *array\_size*, random\_index, another\_random\_index);  
 if ((*numbers*[random\_index] > *numbers*[another\_random\_index]) && (random\_index < another\_random\_index)) {  
 int temp = *numbers*[random\_index];  
 *numbers*[random\_index] = *numbers*[another\_random\_index];  
 *numbers*[another\_random\_index] = temp;  
 *printArray*(*numbers*, *array\_size*, random\_index, another\_random\_index, false, true);  
 }  
 }  
}  
  
int *main*() {  
 *SetConsoleCP*(CP\_UTF8);  
 *SetConsoleOutputCP*(CP\_UTF8);  
 *setlocale*(LC\_ALL, ".UTF8");  
  
 int numbers[9];  
 *random\_device* rd;  
 *mt19937* gen*(*rd()*)*;  
 *uniform\_int\_distribution*<int> dist*(*0, 100*)*;  
 for (int & number : numbers) {  
 number = dist(gen);  
 }  
  
 int array\_size = sizeof(numbers) / sizeof(numbers[0]);  
 char choice = 'N';  
  
 cout *<<* "Ввести собственный массив чисел? (Y, N - default): ";  
 cin *>>* choice;  
  
 if(choice == 'Y'){  
 cout *<<* "Введите количество чисел: ";  
 cin *>>* array\_size;  
 cout *<<* "Введите числа через пробел: ";  
 for (int i = 0; i < array\_size; i++){  
 cin *>>* numbers[i];  
 }  
 }  
 cout *<<* "Введённый массив: [ ";  
 for (int i = 0; i < array\_size; i++){  
 cout *<<* numbers[i] *<<* " ";  
 }  
 cout *<<* "]" *<< endl*;  
  
 choice = '0';  
  
 cout *<<* "Выберите алгоритм сортировки (0 - Bubble Sort - default, 1 - Shell Sort, 2 - Selection Sort, 3 - Bogo Sort): " *<< endl*;  
 cin *>>* choice;  
 switch (choice) {  
 case '0':  
 cout *<<* "Bubble Sort" *<< endl*;  
 *bubbleSort*(numbers, array\_size);  
 break;  
 case '1':  
 cout *<<* "Shell Sort" *<< endl*;  
 *shellSort*(numbers, array\_size);  
 break;  
 case '2':  
 cout *<<* "Selection Sort" *<< endl*;  
 *selectionSort*(numbers, array\_size);  
 break;  
 case '3':  
 cout *<<* "Bogo Sort" *<< endl*;  
 *bogoSort*(numbers, array\_size);  
 break;  
 default:  
 cout *<<* "Неверный выбор. Используется Bubble Sort по умолчанию." *<< endl*;  
 *bubbleSort*(numbers, array\_size);  
 }  
 *system*("pause");  
 return 0;  
}

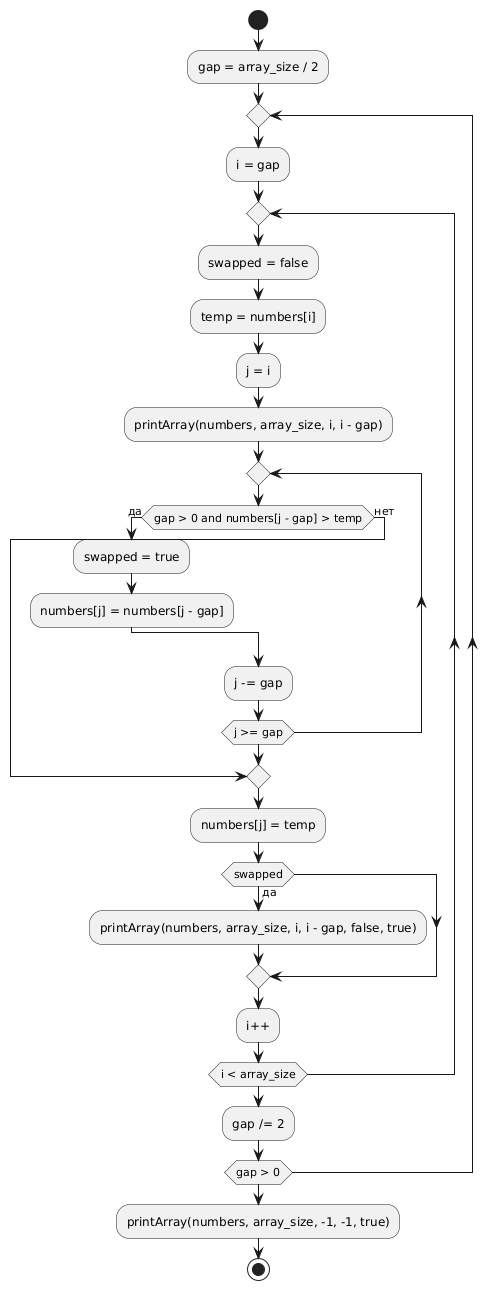
**Запуск программы:**

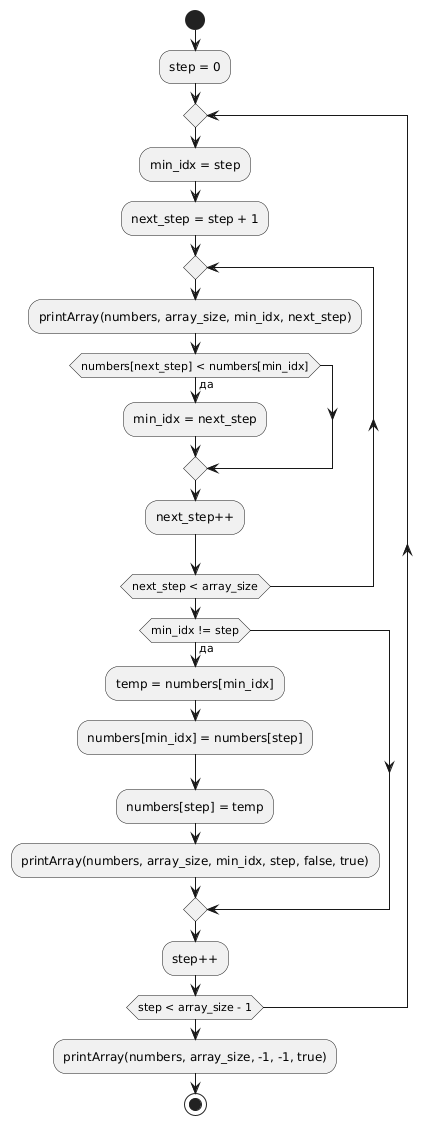
**Блок-схемы:**

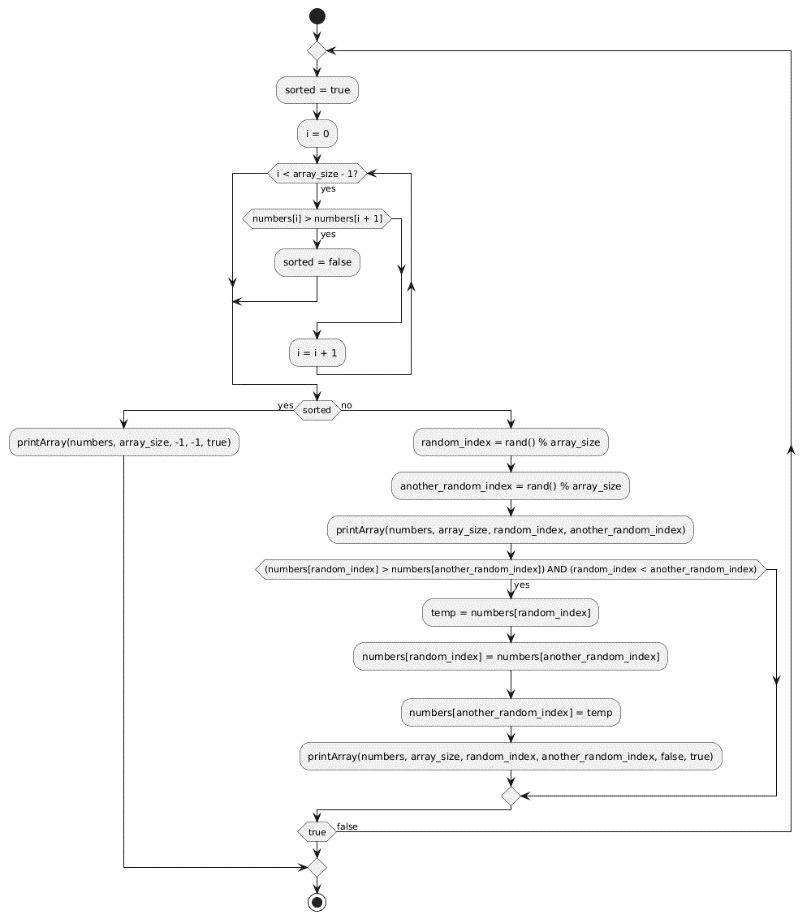
printArray:

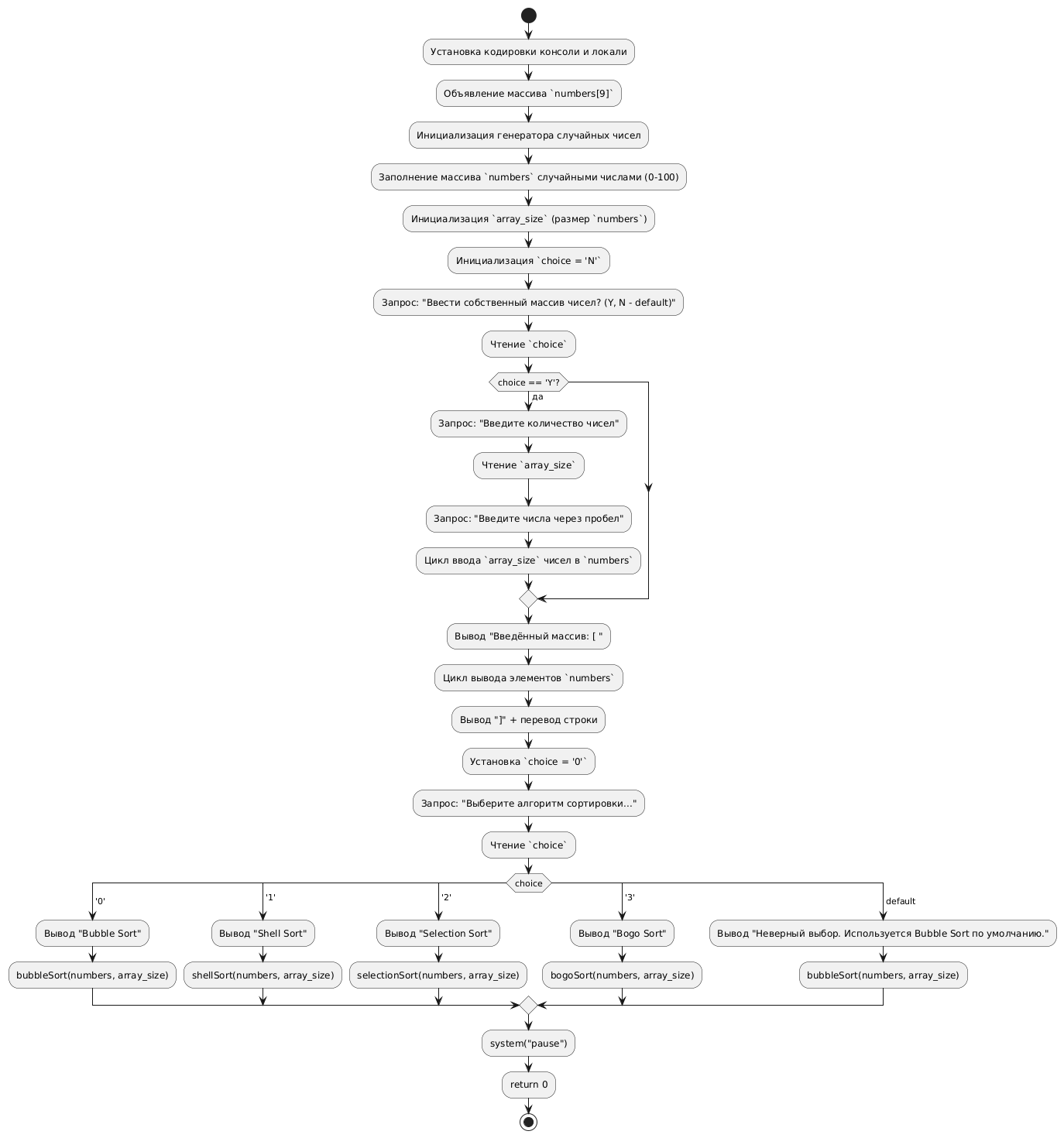
bubbleSort:

shellSort:



selectionSort:

bogoSort:

main: