**Министерство науки и высшего образования Российской Федерации**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Факультет безопасности информационных технологий**

**Дисциплина:**

«Алгоритмы и структуры данных»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1**

«Реализация алгоритма сортировки Radix»

**Выполнил:**

Арендаренко М.М., студент группы N3247

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(подпись)

**Проверил:**

Ерофеев С. А.

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(подпись)

Санкт-Петербург

2024 г.

Содержание

[Введение 3](#_Toc160885385)

[1 Описание функционала программы 4](#_Toc160885386)

[2 Блок-схема 5](#_Toc160885387)

[3 Код программы 13](#_Toc160885388)

[4 Результаты тестирования 17](#_Toc160885389)

[Заключение 18](#_Toc160885390)

Введение

Задача работы – изучить алгоритм сортировки Radix и реализовать его на одном из языков программирования с возможностью выбора типа используемого массива.

Сортировка Radix — это способ упорядочивания чисел, который не сравнивает их напрямую, как это делают многие другие алгоритмы сортировки. Вместо этого он сортирует числа, рассматривая каждую цифру числа по очереди, начиная с последней цифры (наименее значимой).

Алгоритм:

1. Найти диапазон значений: определить диапазон значений в массиве. Это поможет определить количество цифр, которые нужно учитывать при сортировке.

2. Определить количество цифр: Количество цифр в максимальном абсолютном значении в массиве определит количество проходов, необходимых для сортировки.

3. Итерация по разрядам: для каждого разряда выполнить следующие действия:

Создайте массив подсчета для хранения количества цифр в каждом разряде.

Пройдитесь по массиву и обновите массив подсчета для соответствующего разряда.

Обновить индексы подсчета, суммируя значения в массиве подсчета.

Пройтись по элементам в обратном порядке и поместить их в выходной массив в соответствии с их разрядом и индексом подсчета.

4. Объединить отсортированных элементов: после сортировки всех разрядов объединить отсортированные элементы в окончательный отсортированный массив..

Для реализации алгоритма был выбран язык программирования C++, для написания и отладки кода использовалась среда разработки XCode версии 12.4

# Описание функционала программы

Пользователю на выбор будет представлено два вида массива – статический или динамический. Длина статического будет равна 10, длину динамического будет определять пользователь. В качестве элементов массива будем использовать целые числа.

Используемые переменные:

Элементы массива – тип int (диапазон от −2 147 483 648 до 2 147 483 647)

# Блок-схема

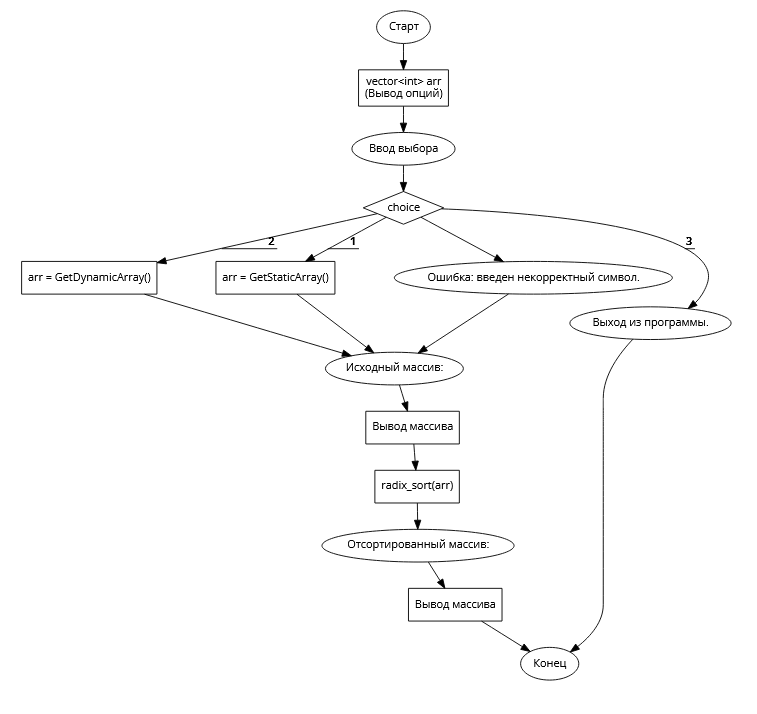


Рисунок 1 – функция «main»

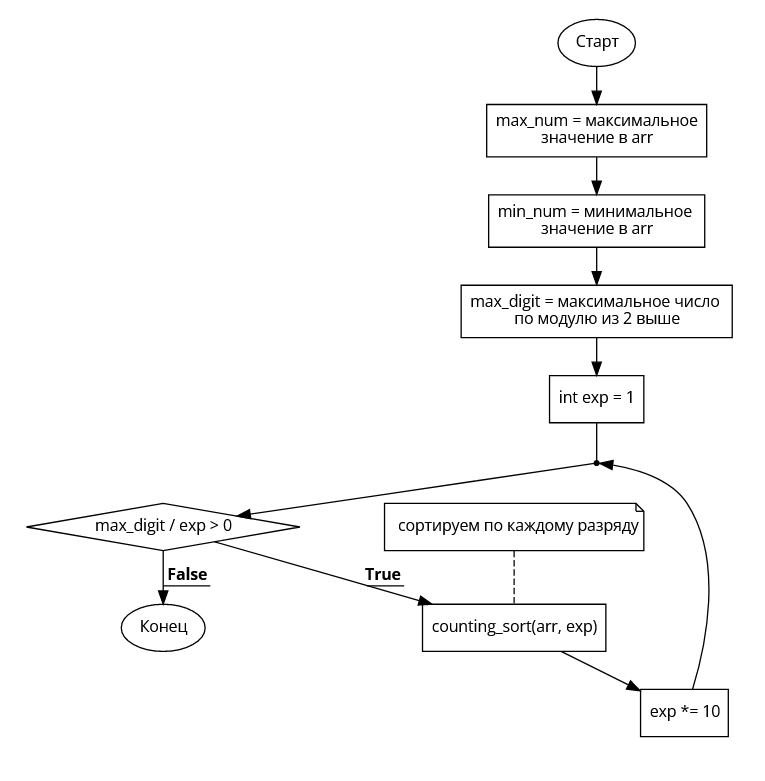


Рисунок 2 – функция «radix\_sort»

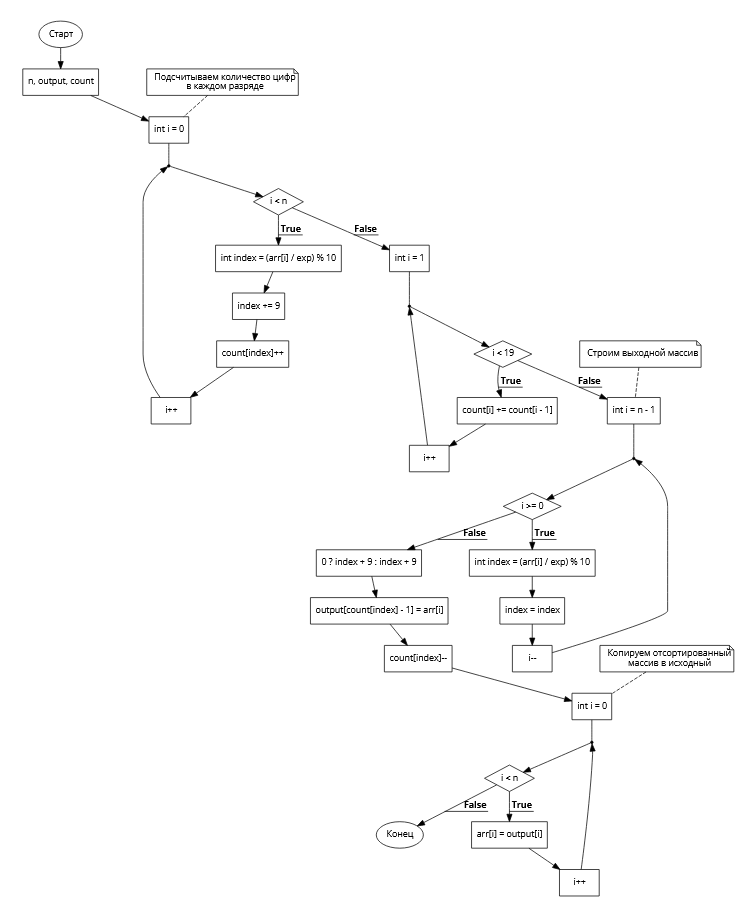


Рисунок 3 – функция «counting\_sort»

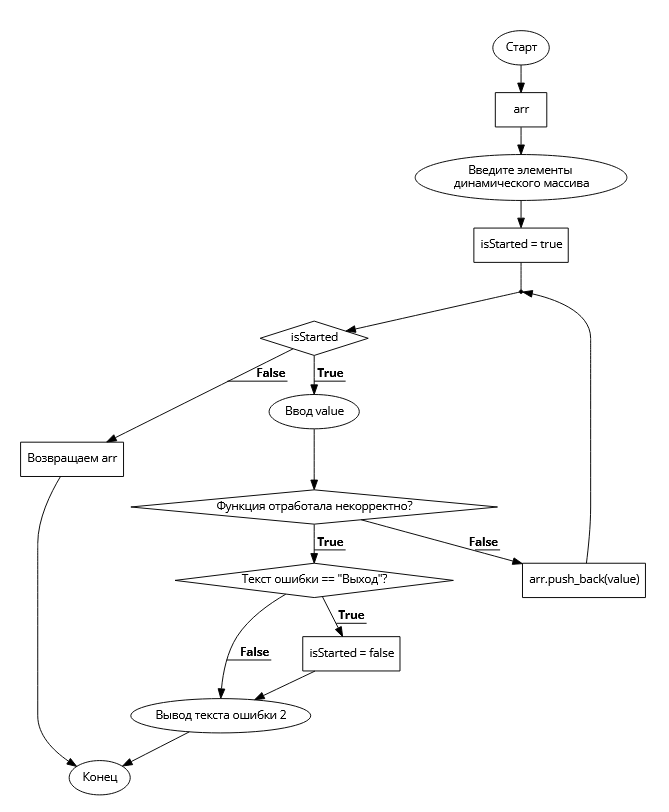
­

Рисунок 4 – функция «GetDynamicArray»

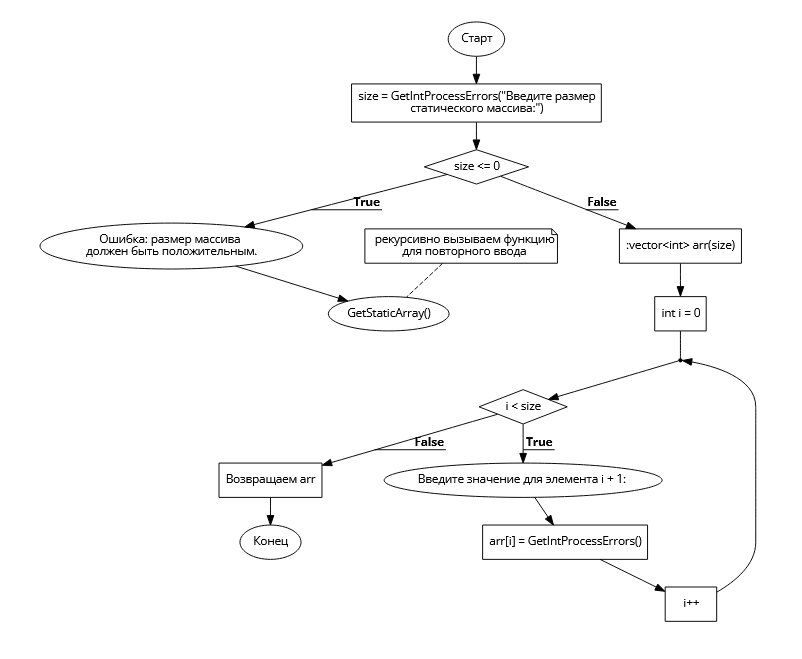


Рисунок 5 – функция «GetStaticArray»

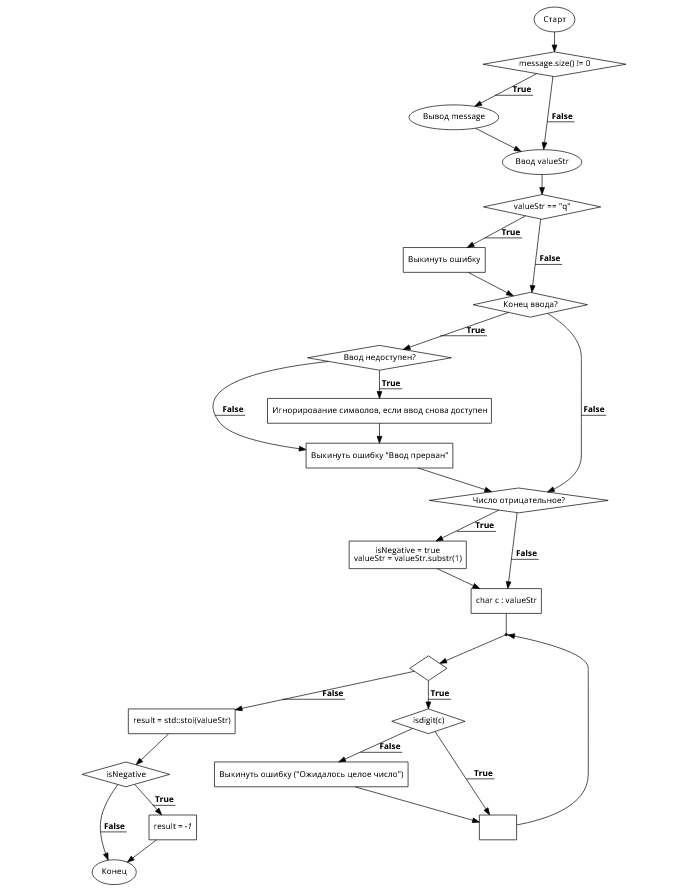


Рисунок 6 – функция «GetInt»

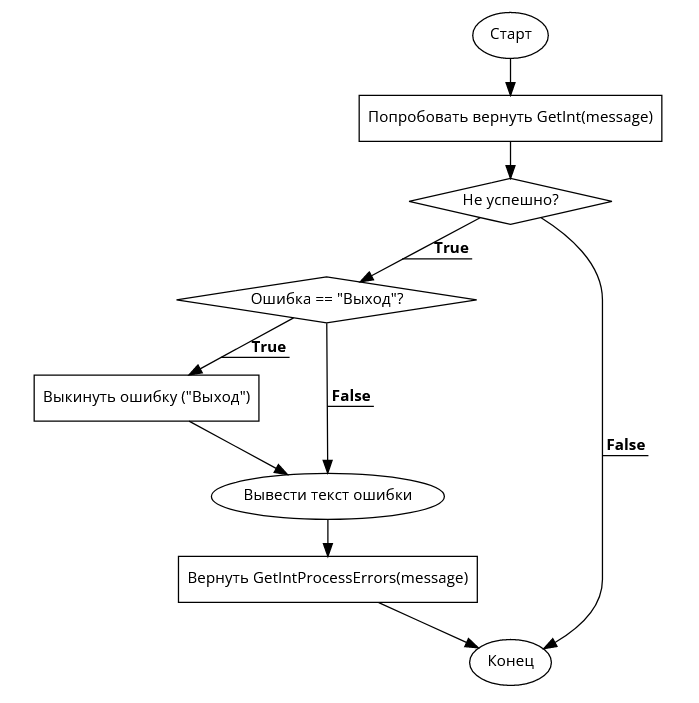


Рисунок 7 – функция «getIntProcessErrors»

# Описание алгоритма сортировки на примере

Рассмотрим массив чисел [170, 45, 75, 90, 802, 24, 2, 66].

Выбор опорного элемента – 802.

Разделим массив. Необходимо распределить остальные элементы массива по разрядам (один младший, один старший), чтобы числа, меньшие 802, например, 45 и 24, находились слева от него, а числа, большие 802, находились справа.

[170, 90, 802, 2, 24, 45, 75, 66]

Рекурсивное применение алгоритма. Теперь мы рекурсивно применяем алгоритм к левому и правому подмассивам относительно опорного элемента 802, сортируя их по старшему и младшему разрядам. Продолжаем делить и сортировать подмассивы по разрядам, пока массив не будет полностью отсортирован.

Финальный результат. После завершения всех рекурсивных вызовов и правильной сортировки по разрядам, массив будет полностью отсортирован. В данном случае массив будет отсортирован в порядке: [2, 24, 45, 66, 75, 90, 170, 802].

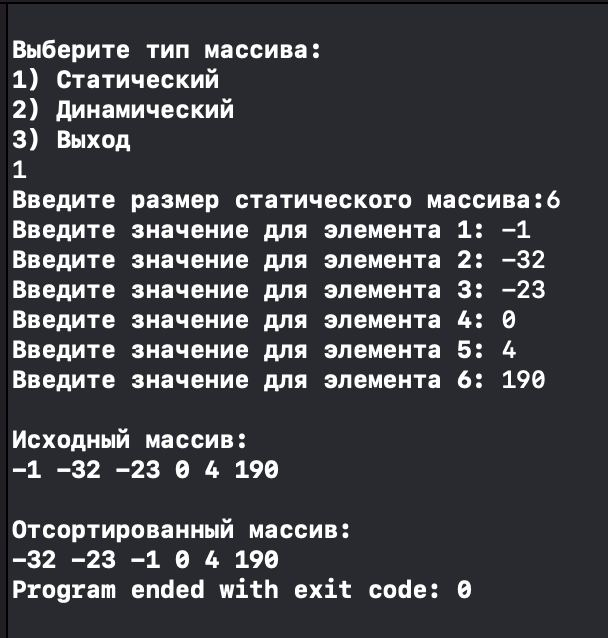
# Код программы

|  |
| --- |
| #include <cstdlib>  #include <iostream>  #include <vector>  #include <algorithm> // for max\_element  const char **\***options **=** "\nВыберите тип массива:\n"  "1) Статический\n"  "2) Динамический\n"  "3) Выход\n"**;**  // Проверка Ctrl+D Ctrl+Z  int GetInt**(**std**::**string message **=** ""**)**  **{**  **if** **(**message**.**size**()** **!=** 0**)**  **{**  std**::**cout **<<** message**;**  **}**  std**::**string valueStr**;**  std**::**cin**.**clear**();**  std**::**getline**(**std**::**cin**,** valueStr**);**  **if** **(**valueStr **==** "q"**)**  **{**  **throw** std**::**runtime\_error**(**"Выход"**);**  **}**  **if** **(**std**::**cin**.**eof**())**  **{**  // std::cin.clear(); // Очистка флага EOF  **if** **(!**std**::**cin**.**good**())**  **{**  std**::**cin**.**ignore**(**std**::**numeric\_limits**<**std**::**streamsize**>::**max**(),** '\n'**);** // Игнорирование символов, если ввод снова доступен  **}**  **throw** std**::**runtime\_error**(**"Ввод прерван"**);**  **}**  // Проверка на отрицательное число  bool isNegative **=** **false;**  **if** **(!**valueStr**.**empty**()** **&&** valueStr**[**0**]** **==** '-'**)**  **{**  isNegative **=** **true;**  valueStr **=** valueStr**.**substr**(**1**);** // Удаление знака "-"  **}**  // Проверка на то, что введенная строка состоит только из цифр  **for** **(**char c **:** valueStr**)**  **{**  **if** **(!**isdigit**(**c**))**  **{**  **throw** std**::**runtime\_error**(**"Ожидалось целое число"**);**  **}**  **}**  // Преобразование строки в целое число  int result **=** std**::**stoi**(**valueStr**);**  // Умножаем результат на -1, если число отрицательное  **if** **(**isNegative**)**  **{**  result **\*=** **-**1**;**  **}**  **return** result**;**  **}**  int GetIntProcessErrors**(**std**::**string message **=** ""**)**  **{**  **try**  **{**  **return** GetInt**(**message**);**  **}**  **catch** **(**const std**::**runtime\_error **&**e**)**  **{**  **if** **(**strcmp**(**e**.**what**(),** "Выход"**)** **==** 0**)**  **{**  **throw** std**::**runtime\_error**(**"Выход"**);**  **}**  std**::**cout **<<** "Ошибка 1: " **<<** e**.**what**()** **<<** std**::**endl**;**  **return** GetIntProcessErrors**(**message**);**  **}**  **}**  std**::**vector**<**int**>** GetStaticArray**()**  **{**  int size **=** GetIntProcessErrors**(**"Введите размер статического массива:"**);**  **if** **(**size **<=** 0**)**  **{**  std**::**cout **<<** "Ошибка: размер массива должен быть положительным." **<<** std**::**endl**;**  **return** GetStaticArray**();** // рекурсивно вызываем функцию для повторного ввода  **}**  std**::**vector**<**int**>** arr**(**size**);**  **for** **(**int i **=** 0**;** i **<** size**;** i**++)**  **{**  std**::**cout **<<** "Введите значение для элемента " **<<** i **+** 1 **<<** ": "**;**  arr**[**i**]** **=** GetIntProcessErrors**();**  **}**  **return** arr**;**  **}**  std**::**vector**<**int**>** GetDynamicArray**()**  **{**  std**::**vector**<**int**>** arr**;**  std**::**cout **<<** "Введите элементы динамического массива" **<<** std**::**endl**;**  bool isStarted **=** **true;**  **while** **(**isStarted**)**  **{**  **try**  **{**  int value **=** GetIntProcessErrors**(**"Введите следующее значение (для завершения введите 'q'): "**);**  arr**.**push\_back**(**value**);**  **}**  **catch** **(**const std**::**runtime\_error **&**e**)**  **{**  **if** **(**strcmp**(**e**.**what**(),** "Выход"**)** **==** 0**)**  isStarted **=** **false;**  std**::**cout **<<** "Ошибка 2: " **<<** e**.**what**()** **<<** std**::**endl**;**  **}**  **}**  **return** arr**;**  **}**  void counting\_sort**(**std**::**vector**<**int**>** **&**arr**,** int exp**)**  **{**  int n **=** arr**.**size**();**  std**::**vector**<**int**>** output**(**n**);** // выходной массив  std**::**vector**<**int**>** count**(**19**,** 0**);** // массив подсчета, 19 для цифр от -9 до 9  // Подсчитываем количество цифр в каждом разряде  **for** **(**int i **=** 0**;** i **<** n**;** i**++)** **{**  int index **=** **(**arr**[**i**]** **/** exp**)** **%** 10**;**  index **=** index **<** 0 **?** index **+** 9 **:** index **+** 9**;** // сдвигаем индекс для поддержки отрицательных чисел  count**[**index**]++;**  **}**  // Модифицируем массив count так, чтобы count[i] содержал индекс  // начала элементов с цифрой i в массиве output  **for** **(**int i **=** 1**;** i **<** 19**;** i**++)** **{**  count**[**i**]** **+=** count**[**i **-** 1**];**  **}**  // Строим выходной массив  **for** **(**int i **=** n **-** 1**;** i **>=** 0**;** i**--)** **{**  int index **=** **(**arr**[**i**]** **/** exp**)** **%** 10**;**  index **=** index **<** 0 **?** index **+** 9 **:** index **+** 9**;** // сдвигаем индекс для поддержки отрицательных чисел  output**[**count**[**index**]** **-** 1**]** **=** arr**[**i**];**  count**[**index**]--;**  **}**  // Копируем отсортированный массив в исходный  **for** **(**int i **=** 0**;** i **<** n**;** i**++)**  **{**  arr**[**i**]** **=** output**[**i**];**  **}**  **}**  void radix\_sort**(**std**::**vector**<**int**>** **&**arr**)** **{**  int max\_num **=** **\***std**::**max\_element**(**arr**.**begin**(),** arr**.**end**());**  int min\_num **=** **\***std**::**min\_element**(**arr**.**begin**(),** arr**.**end**());**  int max\_digit **=** std**::**max**(**std**::**abs**(**max\_num**),** std**::**abs**(**min\_num**));for** **(**int exp **=** 1**;** max\_digit **/** exp **>** 0**;** exp **\*=** 10**)** **{**  counting\_sort**(**arr**,** exp**);** // сортируем по каждому разряду  **}**  **}**  int main**()**  **{**  std**::**vector**<**int**>** arr**;**  std**::**cout **<<** options**;**  int choice **=** GetIntProcessErrors**();**  **switch** **(**choice**)**  **{**  **case** 1**:**  arr **=** GetStaticArray**();**  **break;**  **case** 2**:**  arr **=** GetDynamicArray**();**  **break;**  **case** 3**:**  std**::**cout **<<** "Выход из программы." **<<** std**::**endl**;**  exit**(**0**);**  **default:**  std**::**cout **<<** "Ошибка: введен некорректный символ." **<<** std**::**endl**;**  **break;**  **}**  std**::**cout **<<** std**::**endl  **<<** "Исходный массив:" **<<** std**::**endl**;**  **for** **(**int num **:** arr**)**  **{**  std**::**cout **<<** num **<<** " "**;**  **}**  std**::**cout **<<** std**::**endl**;**  radix\_sort**(**arr**);**  std**::**cout **<<** std**::**endl  **<<** "Отсортированный массив:" **<<** std**::**endl**;**  **for** **(**int num **:** arr**)**  **{**  std**::**cout **<<** num **<<** " "**;**  **}**  std**::**cout **<<** std**::**endl**;**  **return** 0**;**  **}** |

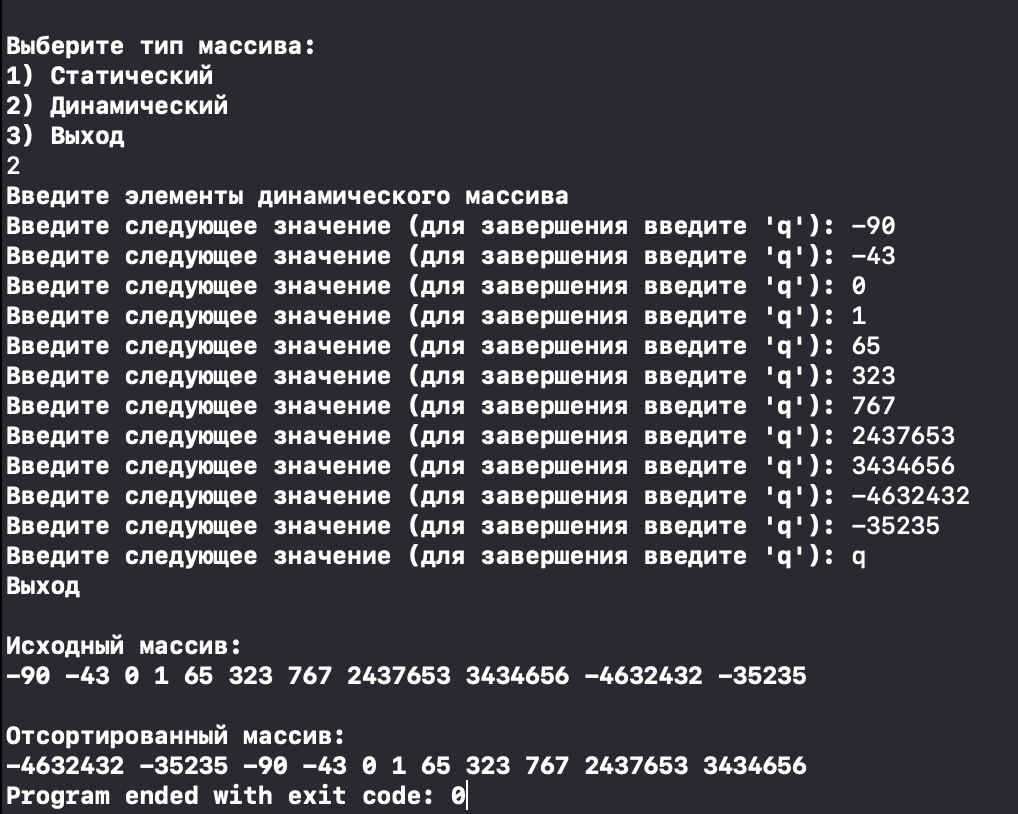
# Результаты тестирования

Были проведены тесты программы с различными входными данными. Все реализованные случаи вывода корректно обрабатываются и отображаются.

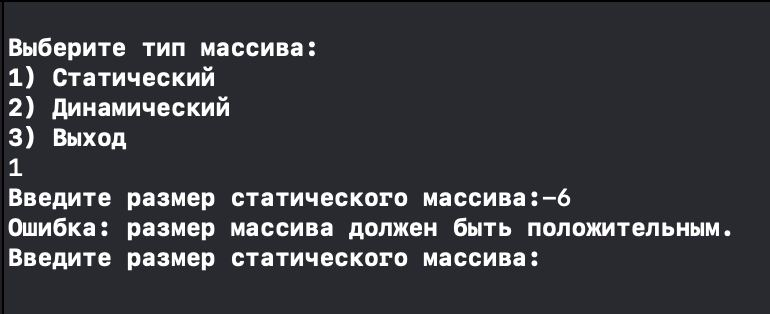
Статический массив:



Динамический массив:



Неверно введенные данные:



# Заключение

В ходе выполнения данной лабораторной работы была достигнута поставленная задача. А именно: мною был изучен и реализован алгоритм сортировки Radix на языке программирования С++ с возможностью выбора статического или динамического массива.

В процессе решения поставленной задачи были реализованы следующие функции «radix\_sort» и «counting\_sort».

counting\_sort: Эта функция выполняет сортировку подсчетом для данного разряда. Она подсчитывает количество цифр в каждом разряде и использует эти подсчеты для правильного размещения элементов в выходном массиве.

radix\_sort: Эта функция вызывает функцию counting\_sort для каждого разряда, эффективно сортируя массив по одному разряду за раз.

С помощью среды разработки Xcode версии 12.4 и языка программирования С++ было произведено тестирование программы при различных входных данных. Программа отработала корректно, сортировка была произведена правильно.