МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №4b

по дисциплине: Основы программирования
тема: «Обработка одномерных массивов с использованием функций»

Выполнил: ст. группы ПВ-211
Чувилко Илья Романович

Проверили:
Притчин Иван Сергеевич
Черников Сергей Викторович

Белгород 2021 г.

Лабораторная работа №4b «Обработка одномерных массивов с использованием функций»

Вариант 21 (a = 4, b = 9)

Цель работы: получение навыков написания функций при решении задач на одномерные массивы.

- Содержание отчета:
- Тема лабораторной работы.
- Номер варианта.
- Цель лабораторной работы.
- Решения задач:
  - Для первого блока необходимо решить все задачи, кроме задач с двумя звездочками. Для задач с двумя звездочками достаточно приложить код без спецификации.
  - Для второго блока необходимо решить все задачи, кроме задач с двумя звездочками. Использовать вариант оформления №1 для задач с номерами a и b, где:

$$a = n_{\text{номер варианта по журналу}}\ mod\ 9 + 1$$

$$b = (n_{\text{номер варианта по журналу}} + 5)\ mod\ 9 + 1$$

  - Необязательные для решения задачи (с двумя звёздочками) допускается оформлять произвольно.

Часть I. Реализовать следующие функции:

Задача №1. Ввод массива `a` размера `size`.

Код Программы:

```c
// Ввод массива a размера size
void inputArray(int *a, size_t const size) {
  for (size_t i = 0; i < size; i++)
    scanf("%d", &a[i]);
}
```

Задача №2. Вывод массива `a` размера `size`.

Код Программы:

```c
// Вывод массива a размера size
void outputArray(int *a, size_t const size) {
  for (size_t i = 0; i < size; i++)
    printf("%d ", a[i]);
}
```

Задача №3. Поиск позиции элемента со значением `a` с начала массива.

Код Программы:

```c
// Возвращает позицию элемента со значением x. Поиск осуществляется в массиве a
// размера size, начиная с первого элемента. После нахождения первого
// подходящего элемента — поиск прекращается.
int linearSearch(int const *a, size_t const size, int const x) {
  for (int i = 0; i < size; i++)
    if (a[i] == x)
      return i;
  return -1;
}
```

Задача №4. Поиск позиции первого отрицательного элемента.

Код Программы:

```c
//Возвращает позицию первого отрицательного элемента массива a размера size
int findNegativeIndex(const int *a, const size_t size) {
  for (int i = 0; i < size; i++)
    if (a[i] < 0)
      return i;
  return -1;
}
```

Задача №5. **Поиск позиции элемента с начала массива (по функции-предикату).
Код Программы:
```
// Возвращает позицию элемента с начала массива a размера size
// (по функции-предикату).
int findIndexIf(const int *a, size_t const size, int (*f)(int)) {
  for (int i = 0; i < size; i++)
    if (f(a[i]))
      return i;
  return -1;
}
```

Задача №6. Поиск позиции последнего четного элемента.
Код Программы:
```
// Возвращает позицию последнего четного элемента массива a размера size
int findLastEvenIndex(int const *a, int const size) {
  for (int i = size - 1; i >= 0; i--)
    if (a[i] % 2 == 0)
      return i;
  return -1;
}
```

Задача №7. **Поиск позиции с конца массива (по функции-предикату).
Код Программы:
```
// Возвращает позицию элемента с конца массива a размера size
// (по функции предикату)
int lastElementIndexIf(int const *a, int const size, int (*f)(int)) {
  for (int i = size - 1; i >= 0; i--)
    if (f(a[i]))
      return i;
  return -1;
}
```
Задача №8. Подсчёт количества отрицательных элементов.
Код Программы:
```
// Возвращает количество отрицательных элементов в массиве a рамера size.
int countNegativeElement(int const *a, size_t const size) {
  int countNegativeElement = 0;
  for (size_t i = 0; i < size; i++)
    if (a[i] < 0)
      countNegativeElement++;
  return countNegativeElement;
}
```
Задача №9. **Подсчёт элементов массива, удовлетворяющих функции-предикату.
Код Программы:
```
// Возвращает количество элементов массива a размера size,
// удовлетворяющих функции-предикату
int countElementIf(int const *a, size_t const size, int (*f)(int)) {
  int countElement = 0;
  for (size_t i = 0; i < size; i++)
    if (f(a[i]))
      countElement++;
  return countElement;
}
```

Задача №10. Изменение порядка элементов массива на обратный.
Код Программы:
```c
// Обменивает значения двух переменных a и b типа int
void swap(int *a, int *b) {
  int t = *a;
  *a = *b;
  *b = t;
}

// Изменяет порядок элементов массива a размера size на обратный
void reverse(int *a, size_t const size) {
  for (size_t firstIndex = 0, lastIndex = size - 1; firstIndex < lastIndex;
       firstIndex++, lastIndex--)
    swap(&a[firstIndex], &a[lastIndex]);
}
```

Задача №11. Проверка является ли последовательность палиндромом.
Код Программы:
```c
// Возвращает 1 (Истина), если последовательность элементов массива a размера
// size представляет собой палиндром, в ином случае 0 (Ложь)
int isPalindrome(int const *a, size_t const size) {
  for (size_t firstIndex = 0, lastIndex = size - 1; firstIndex < lastIndex;
       firstIndex++, lastIndex--)
    if (a[firstIndex] != a[lastIndex])
      return 0;
  return 1;
}
```

Задача №12. Сортировка массива выбором.
Код Программы:
```c
// Возвращает индекс минимального элемента в массиве a размера size
size_t minIndex(int const *a, size_t const size) {
  size_t minIndex = 0;
  for (size_t i = 0; i < size; i++)
    if (a[i] < a[minIndex])
      minIndex = i;
  return minIndex;
}

// Сортирует выбором массив a размера size по неубыванию
void selectionSort(int *a, size_t const size) {
  for (size_t i = 0; i < size - 1; i++) {
    swap(&a[i], &a[i + minIndex(&a[i], size - i)]);
  }
}
```

Задача №13. Алгоритм удаления из массива всех нечетных элементов.
Код Программы:
```c
// Удаляет все нечетные элементы из массива a размера size
void deleteUneven(int *a, size_t *size) {
  int lastEvenIndex = 0;
  for (size_t i = 0; i < *size; i++) {
    if (a[i] % 2 == 0) {
      swap(&a[lastEvenIndex++], &a[i]);
      (*size)--;
    }
  }
}
```

Задача №14. Вставка элемента в массив с сохранением относительного порядка других элементов

Код Программы:

```c
// Возвращает увеличенный массив a рамера size, с добавленным элементом x,
// который встанет на позицию pos. При том сохраняется относительная
// последовательность других элементов.
void addElement(int *a, size_t *size, int const x, int const pos) {
  int last;
  for (size_t i = 0; i <= *size; i++)
    if (i == pos) {
      last = a[i];
      a[i] = x;
    } else if (i > pos) {
      int tmp = a[i];
      a[i] = last;
      last = tmp;
    }
  (*size)++;
}
```

Задача №15. Добавление элемента в конец массива.

Код Программы:

```c
// Добавляет элемент в конец массива a размера size
void append(int *a, size_t *size, int x) {
  a[*size] = x;
  (*size)++;
}
```

Задача №16. Удаление элемента с сохранением относительного порядка других элементов.

Код Программы:

```c
// Удаляет элемент массива a размера n, находящийся на позиции pos, сохраняя
// относительную последовательность остальных элементов
void deleteElementKeeping(int *a, size_t *size, int const pos) {
  (*size)--;
  for (int i = 0; i < *size; i++) {
    if (i >= pos)
      a[i] = a[i + 1];
  }
}
```

Задача №17. Удаление элемента без сохранения относительного порядка других элементов.

Код Программы:

```c
// Удаляет элемент массива a размера size, находящийся на позиции pos, не
// сохраняя относительную последовательность остальных элементов
void deleteElementNotKeeping(int *a, size_t *size, int const pos) {
  (*size)--;
  a[pos] = a[*size];
}
```

Задача №18. **Реализуйте циклический сдвиг массива влево на `k` позиций.

Код Программы:

```c
// Совершает циклический сдвиг влево массива a размера size на k позиций
void cyclicShift(int *a, size_t *size, size_t k) {
  if (*size < k)
    k = *size;
  for (int i = 0; i < *size - k; i++) {
    a[i] = a[i + k];
  }
  *size -= k;
}
```

Задача №19. **Реализуйте функцию `forEach`, которая применяет функцию `f` к элементам массива `a` размера `size`.

Код Программы:

```c
// Применяет функцию f() к элементам массива a размера size
void forEach(int *a, size_t size, int (*f)(int)) {
  for (size_t i = 0; i < size; i++)
    a[i] = f(a[i]);
}
```

Задача №20. **Реализуйте функцию `any`, которая возвращает значение 'истина', если хотя бы один элемент массива `a` размера `size` удовлетворяют функции-предикату `f`, иначе – 'ложь'.

Код Программы:

```c
// Возвращает значение 1 (Истина), если хотя бы один элемент массива a размера
// size удовлетворяет функции-предикату f, иначе 0 (Ложь).
int any(int const *a, size_t const size, int (*f)(int)) {
  for (size_t i = 0; i < size; i++)
    if (f(a[i]))
      return 1;
  return 0;
}
```

Задача №21. **Реализуйте функцию `all`, которая возвращает значение 'истина', если все элементы массива `a` размера `size` удовлетворяют функции-предикату `f`, иначе – 'ложь'

Код Программы:

```c
// Возвращает значение 1 (Истина), если хотя бы один элемент массива a размера
// size удовлетворяет функции-предикату f, иначе 0 (Ложь).
int all(int const *a, size_t const size, int (*f)(int)) {
  for (size_t i = 0; i < size; i++)
    if (f(a[i]) == 0)
      return 0;
  return 1;
}
```

Задача №22. **Реализуйте функцию `arraySplit`, которая разделяет элементы массива `a` размера `size` на элементы, удовлетворяющие функции-предикату `f`, сохраняя в массиве `b`, иначе – в массиве `c`

Код Программы:

```c
// Разделяет массив a размера size на элементы, удовлетворяющие
// функции-предикату f, сохраняя в массиве b, иначе - в массиве c
void arraySplit(int const *a, size_t const size, int (*f)(int), int *b,
                size_t *sizeB, int *c, size_t *sizeC) {
  for (size_t i = 0; i < size; i++) {
    if (f(a[i]))
      append(b, sizeB, a[i]);
    else
      append(c, sizeC, a[i]);
  }
}
```

Часть 2. Перечень задач:

Задание №1. Определить, можно ли, переставив члены данной целочисленной последовательности длины `n` (`n` > 1), получить геометрическую прогрессию. Разрешимое допущение: знаменатель прогрессии – целое число.

Тестовые данные:

Входные данные	Выходные данные
`n = 3` `1 2 4`	“Yes”
`n = 3` `1 2 5`	“No”
`n = 2` `1 1`	“Yes”
`n = 2` `0 1`	“No”
`n = 3` `1 3 0`	“No”
`n = 5` `-1 -4 -16 2 8`	“Yes”
`n = 5` `1 1 1 -1 -1`	“Yes”
`n = 5` `1 1 1 1 -1`	“No”

Код программы:

```c
#include <stdio.h>
#include <math.h>

// Ввод массива a размера size
void inputArray(int *a, size_t const size) {
  for (size_t i = 0; i < size; i++)
    scanf("%d", &a[i]);
}

// Обменивает значения двух переменных a и b типа int
void swap(int *a, int *b) {
  int t = *a;
  *a = *b;
  *b = t;
}

// Возвращает индекс минимального элемента в массиве a размера size
size_t minIndex(int const *a, size_t const size) {
  size_t minIndex = 0;
  for (size_t i = 0; i < size; i++)
    if (a[i] < a[minIndex])
      minIndex = i;
  return minIndex;
}

// Сортирует выбором массив a размера size по неубыванию
void selectionSort(int *a, size_t const size) {
  for (size_t i = 0; i < size - 1; i++) {
    swap(&a[i], &a[i + minIndex(&a[i], size - i)]);
  }
}
```

```c
// Возвращает количество отрицательных элементов в массиве a рамера size.
int countNegativeElement(int const *a, size_t const size) {
  int countNegativeElement = 0;
  for (size_t i = 0; i < size; i++)
    if (a[i] < 0)
      countNegativeElement++;
  return countNegativeElement;
}

// Применяет функцию f() к элементам массива a размера size
void absForEach(int *a, size_t const size) {
  for (size_t i = 0; i < size; i++)
    a[i] = abs(a[i]);
}

// Возвращает знаменатель геометрической прогрессии, где a и b соседние числа
// последовательности. Если целочисленный знаменатель найти
// не получается — возвращает 0.
int denominatorGeomProgression(int const a, int const b) {
  if (b != 0 && a != 0 && a % b == 0)
    return a / b;
  return 0;
}

// Возвращает 1, если элементы отсортированного массива представляют собой
// геометрическую прогрессию.
int isGeomProgression_(int const *a, size_t const size) {
  int q = denominatorGeomProgression(a[1], a[0]);
  for (size_t i = 1; i < size - 1 && q != 0; i++)
    if (q != denominatorGeomProgression(a[i + 1], a[i]))
      return 0;
  return q != 0;
}

// Возвращает 1, если переставив элементы массива a размера size, можно
// получить геометрическую прогрессию (для целочисленных), 0 в ином случае
int isGeomProgression(int *a, size_t const size) {
  int count = countNegativeElement(a, size);
  if (size < 2 || (abs(2 * count - (int) size) > 1 && count != 0))
    return 0;
  absForEach(a, size);
  selectionSort(a, size);
  return isGeomProgression_(a, size);
}

int main() {
  size_t size;
  scanf("%llud", &size);
  int a[size];
  inputArray(a, size);

  if (isGeomProgression(a, size))
    printf("Yes");
  else
    printf("No");

  return 0;
}
```

Задание №2. Если возможно, то упорядочить данный массив размера `size` по убыванию, иначе массив оставить без изменения.

Тестовые данные:

Входные данные	Выходные данные
`n = 3` `1 2 4`	`4 2 1`
`n = 3` `4 2 4`	`4 2 4`
`n = 1` `1`	`1`

Код программы:

```c
#include <stdio.h>

// Ввод массива a размера size
void inputArray(int *a, size_t const size) {
  for (size_t i = 0; i < size; i++)
    scanf("%d", &a[i]);
}

// Вывод массива a размера size
void outputArray(int *a, size_t const size) {
  for (size_t i = 0; i < size; i++)
    printf("%d ", a[i]);
}

// Обменивает значения двух переменных a и b типа int
void swap(int *a, int *b) {
  int t = *a;
  *a = *b;
  *b = t;
}

// Возвращает индекс минимального элемента в массиве a размера size
size_t minIndex(int const *a, size_t const size) {
  size_t minIndex = 0;
  for (size_t i = 0; i < size; i++)
    if (a[i] < a[minIndex])
      minIndex = i;
  return minIndex;
}

// Сортирует выбором массив a размера size по неубыванию
void selectionSort(int *a, size_t const size) {
  for (size_t i = 0; i < size - 1; i++) {
    swap(&a[i], &a[i + minIndex(&a[i], size - i)]);
  }
}
```

```c
// Возвращает 1 (Истина), если число x не встречается в массиве a,
// в ином случае 0 (Ложь).
int isUniqueNumber(int const *const a, size_t const size, int const x) {
  for (size_t i = 0; i < size; i++)
    if (a[i] == x)
      return 0;
  return 1;
}

// Возвращает 1 (Истина), если все значения в массиве a размера size -
// уникальны, в ином случае 0 (Ложь)
int isUniqueNumbers(int const *a, size_t const size) {
  if (size == 1)
    return 1;
  for (size_t i = 0; i < size - 1; i++)
    if (isUniqueNumber(&a[i] + 1, size - i, a[i]))
      return 0;
  return 1;
}

// Упорядочивает массив a размера size по убыванию, если упорядочить по
// убыванию невозможно (есть элементы, значения которых равны), то оставляет
// массив без изменения
void sortDescending(int *a, size_t const size) {
  if (isUniqueNumbers(a, size)) {
    selectionSort(a, size);
  }
}

int main() {
  size_t size;
  scanf("%llud", &size);
  int a[size];
  inputArray(a, size);

  sortDescending(a, size);

  outputArray(a, size);

  return 0;
}
```

Задание №3. Дана целочисленная последовательность. Упорядочить по неубыванию часть последовательности, заключенную между первым вхождением максимального значения и последним вхождением минимального.

Тестовые данные:

Входные данные	Выходные данные
n = 5 10 **3 2 1** 0	10 **1 2 3** 0
n = 5 0 **3 2 1** 10	0 **1 2 3** 10
n = 8 10 **5 4** 4 7 8 10 10	10 **4 5** 4 7 8 10 10

Код программы:

```c
#include <stdio.h>

// Ввод массива a размера size
void inputArray(int *a, const size_t size) {
  for (size_t i = 0; i < size; i++)
    scanf("%d", &a[i]);
}

// Вывод массива a размера size
void outputArray(int *a, const size_t size) {
  for (size_t i = 0; i < size; i++)
    printf("%d ", a[i]);
}

// Обменивает значения двух переменных a и b типа int
void swap(int *a, int *b) {
  int t = *a;
  *a = *b;
  *b = t;
}

// Обменивает значения двух переменных a и b типа size_t
void swapIndex(size_t *a, size_t *b) {
  size_t t = *a;
  *a = *b;
  *b = t;
}

// Возвращает индекс минимального элемента в массиве a размера size
size_t minIndex(int const *a, size_t const size) {
  size_t minIndex = 0;
  for (size_t i = 0; i < size; i++)
    if (a[i] < a[minIndex])
      minIndex = i;
  return minIndex;
}

// Сортирует выбором массив a размера size по неубыванию
void selectionSort(int *a, size_t const size) {
  for (size_t i = 0; i < size - 1; i++) {
    swap(&a[i], &a[i + minIndex(&a[i], size - i)]);
  }
}

// Осуществляет поиск минимального и максимального значения в массиве a размера
// size. В случае, если такие значения удалось найти - изменяет значение
// minIndex на порядковый номер последнего минимального значения, а maxIndex
// на порядковый номер максимального значения, иначе оставляет без изменения.
void findMinMaxIndex(int const *const a, size_t size, size_t *minIndex,
                     size_t *maxIndex) {
  for (size_t i = 1; i < size; i++) {
    if (a[i] <= a[*minIndex])
      *minIndex = i;
    else if (a[i] > a[*maxIndex])
      *maxIndex = i;
  }
}
```

```c
// Сортирует числа массива a размера size, находящиеся между первым вхождением
// максимального значения и последним вхождением минимального.
void sortDescendingBetweenMinMax(int *a, size_t size) {
  size_t minIndex = 0, maxIndex = 0;
  findMinMaxIndex(a, size, &minIndex, &maxIndex);
  if (minIndex < maxIndex)
    swapIndex(&minIndex, &maxIndex);
  selectionSort(&a[maxIndex + 1], minIndex - 1);
}


int main() {
  size_t n;
  scanf("%llud", &n);
  int a[n];
  inputArray(a, n);

  sortDescendingBetweenMinMax(a, n);

  outputArray(a, n);

  return 0;
}
```

Задание №4. Если данная последовательность не упорядочена ни по неубыванию, ни по невозрастанию, найти среднее геометрическое положительных членов.

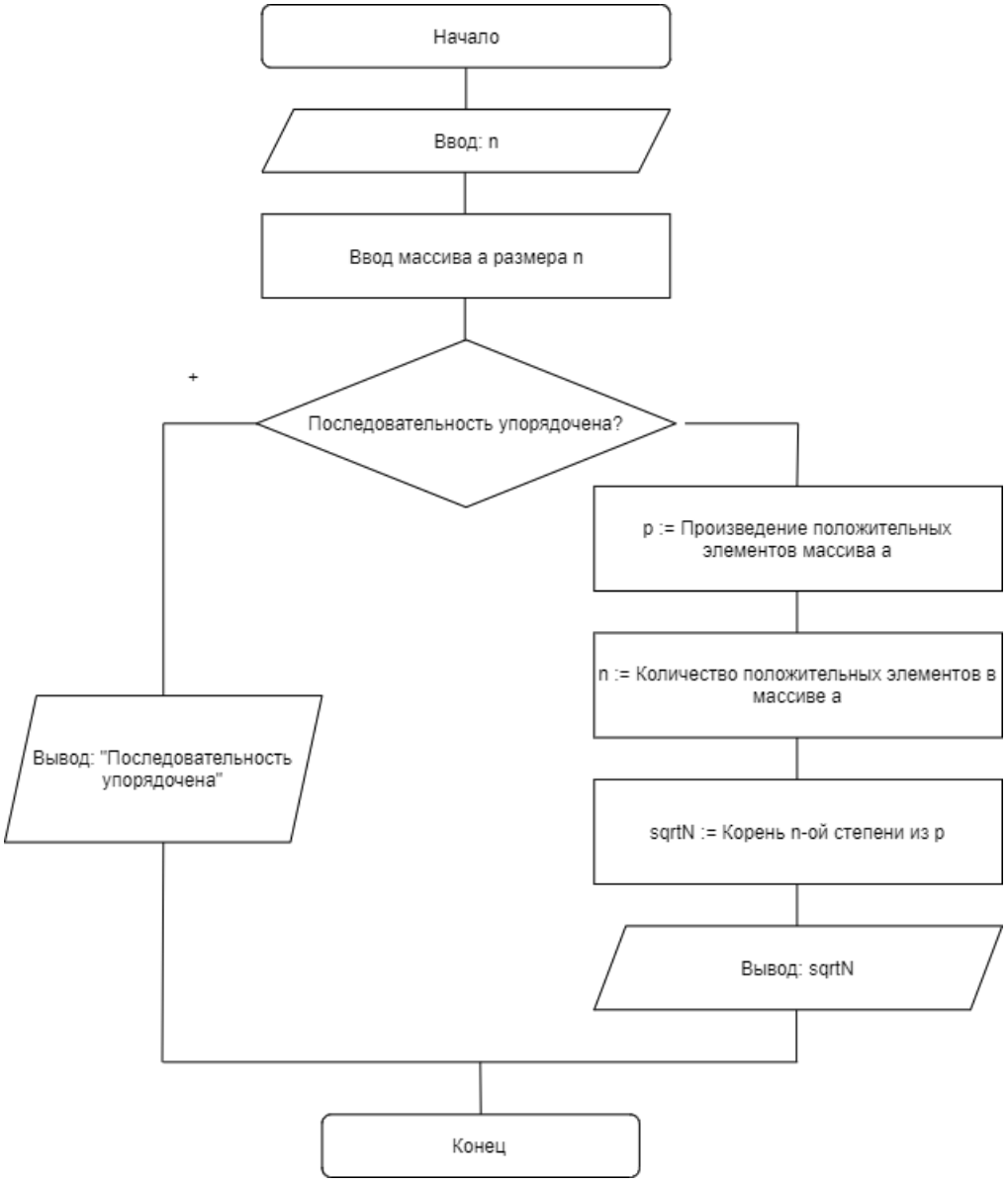
Тестовые данные:

Входные данные	Выходные данные
n = 3 4 1 2	2
n = 5 9 1 3 3 0	3
n = 4 2 -1 -1 0	2
n = 3 -1 -2 -1	0
n = 3 -1 -1 -1	"Последовательность упорядочена"
n = 3 1 2 4	"Последовательность упорядочена"
n = 3 4 2 2	"Последовательность упорядочена"

Выделение подзадач:

1. Ввод массива
2. Определить, является ли последовательность упорядоченной.
3. Нахождение среднего геометрического.
    - **(a)** Нахождение произведения положительных членов.
    - **(b)** Нахождение количество положительных членов.
    - **(c)** Нахождение корня от произведения положительных членов.

Алгоритм в укрупненных блоках



Код программы:

```c
#include <stdio.h>
#include <math.h>
#include <windows.h>

// Ввод массива a размера size
void inputArray(int *a, const size_t size) {
  for (size_t i = 0; i < size; i++)
    scanf("%d", &a[i]);
}

// Возвращает 1 ('Истина'), если члены массива a размера size упорядочены по
// невозрастанию или по неубыванию, в ином случае возвращает 0 ('Ложь').
int isRegularSequence(int const *const a, size_t const size) {
  int status;
  if (a[0] < a[1])
    status = 1; // Неубывание
  else if (a[0] > a[1])
    status = 2; // Невозрастание
  else
    status = 3;
  for (int i = 1; i < size - 1; i++) {
    if ((status == 1 && a[i] > a[i + 1]) || (status == 2 && a[i] < a[i + 1]))
      return 0;
    else if (status == 3 && a[i] < a[i + 1])
      status = 1;
    else if (status == 3 && a[i] > a[i + 1])
      status = 2;
  }
  return 1;
}

// Возвращает среднее геометрическое положительных элементов массива a размера
// size, если массив не упорядочен по невозрастанию или по неубыванию
double geometricAverage(int const *const a, size_t const size) {
  int p = 1, n = 0;
  for (size_t i = 0; i < size; i++)
    if (a[i] > 0) {
      p *= a[i];
      n++;
    }
  if (n == 0)
    return 0;
  double sqrtN = 1.0 / n;
  return pow(p, sqrtN);
}

int main() {
  SetConsoleOutputCP(CP_UTF8);
  size_t n;
  scanf("%llud", &n);
  int a[n];
  inputArray(a, n);

  if (isRegularSequence(a, n))
    printf("Последовательность упорядочена");
  else
    printf("%lf", geometricAverage(a, n));

  return 0;
}
```

Задание №5. Если число x встречается в данной целочисленной последовательности, то упорядочить по неубыванию часть последовательности после первого вхождения x.

Тестовые данные:

Входные данные	Выходные данные
x = 4, n = 5 16 8 **4 2 1**	16 8 **4 1 2**
x = 16, n = 5 **16 8 4 2 1**	**16 1 2 4 8**
x = 1, n = 5 16 8 4 2 **1**	16 8 4 2 **1**
x = 3, n = 5 16 8 4 2 1	16 8 4 2 1

Код программы:

```c
#include <stdio.h>

// Ввод массива a размера size
void inputArray(int *a, size_t const size) {
  for (size_t i = 0; i < size; i++)
    scanf("%d", &a[i]);
}

// Вывод массива a размера size
void outputArray(int *a, size_t const size) {
  for (size_t i = 0; i < size; i++)
    printf("%d ", a[i]);
}

// Возвращает позицию элемента со значением x. Поиск осуществляется в массиве a
// размера size, начиная с первого элемента. После нахождения первого
// подходящего элемента — поиск прекращается.
int linearSearch(int const *const a, size_t const size, int x) {
  for (int i = 0; i < size; i++)
    if (a[i] == x)
      return i;
  return -1;
}

// Обменивает значения двух переменных a и b типа int
void swap(int *a, int *b) {
  int t = *a;
  *a = *b;
  *b = t;
}

// Возвращает индекс минимального элемента в массиве a размера size
size_t minIndex(int const *const a, size_t const size) {
  size_t minIndex = 0;
  for (size_t i = 0; i < size; i++)
    if (a[i] < a[minIndex])
      minIndex = i;
  return minIndex;
}
```

```c
// Сортирует выбором массив a размера size по неубыванию
void selectionSort(int *a, size_t const size) {
  for (size_t i = 0; i < size - 1; i++) {
    swap(&a[i], &a[i + minIndex(&a[i], size - i)]);
  }
}

// Сортирует, в порядке неубывания, члены последовательности массива a размера
// size, идущие после первого вхождения числа x
void sortAfterX(int *a, size_t const size, int const x) {
  int indexX = linearSearch(a, size, x);
  if (indexX < size - 1 && indexX != -1)
    selectionSort(&a[indexX + 1], size - indexX - 1);
}


int main() {
  int x;
  size_t n;
  scanf("%d %llud", &x, &n);
  int a[n];
  inputArray(a, n);

  sortAfterX(a, n, x);

  outputArray(a, n);

  return 0;
}
```

Задание №6. Даны две последовательности. Получить упорядоченную по невозрастанию последовательность, состоящую из тех членов первой последовательности, которых нет во второй

Тестовые данные:

Входные данные	Выходные данные
sizeA = 3, sizeB = 1 a = {1, 2, 4} b = {4}	c = {1, 2}
sizeA = 4, sizeB = 2 a = {1, 2, 2, 4} b = {1, 4}	c = {2, 2}
sizeA = 4, sizeB = 1 a = {1, 2, 2, 4} b = {3}	c = {4, 2, 2, 1}

Код программы:

```c
#include <stdio.h>

// Ввод массива a размера size
void inputArray(int *a, size_t const size) {
  for (size_t i = 0; i < size; i++)
    scanf("%d", &a[i]);
}

// Вывод массива a размера size
void outputArray(int *a, size_t const size) {
  for (size_t i = 0; i < size; i++)
    printf("%d ", a[i]);
}

// Возвращает наибольшее число из двух введенных целочисленных
// переменных a и b типа size_t
size_t maxSize(size_t const a, size_t const b) {
  return a > b ? a : b;
}

// Обменивает значения двух переменных a и b типа int
void swap(int *a, int *b) {
  int t = *a;
  *a = *b;
  *b = t;
}

// Возвращает индекс минимального элемента в массиве a размера size
size_t minIndex(int const *a, size_t const size) {
  size_t minIndex = 0;
  for (size_t i = 0; i < size; i++)
    if (a[i] < a[minIndex])
      minIndex = i;
  return minIndex;
}

// Сортирует выбором массив a размера size по неубыванию
void selectionSort(int *a, size_t const size) {
  for (size_t i = 0; i < size - 1; i++) {
    swap(&a[i], &a[i + minIndex(&a[i], size - i)]);
  }
}

// Добавляет элемент в конец массива a размера size
void append(int *a, size_t *size, int x) {
  a[*size] = x;
  (*size)++;
}

// Изменяет порядок элементов массива a размера size на обратный
void reverse(int *a, const size_t size) {
  for (size_t firstIndex = 0, lastIndex = size - 1; firstIndex < lastIndex;
       firstIndex++, lastIndex--)
    swap(&a[firstIndex], &a[lastIndex]);
}

// Возвращает 1 (Истина), если число x не встречается в массиве a размера size,
// в ином случае 0 (Ложь).
int isUniqueNumber(int const *const a, const size_t size, const int x) {
  for (size_t i = 0; i < size; i++)
    if (a[i] == x)
      return 0;
  return 1;
}
```

```c
// Отбирает из массива a размера sizeA и массива b размера sizeB уникальные
// элементы, которые встречаются в массиве a, но не встречается в массиве b,
// и сохраняет их в массив c размера sizeC в порядке невозрастания
void sortUniqueNumbers(int *const a, size_t const sizeA,
                       int *const b, size_t const sizeB,
                       int *const c, size_t *sizeC) {
  selectionSort(a, sizeA);
  selectionSort(b, sizeB);
  for (size_t i = 0; i < sizeA; i++)
    if (isUniqueNumber(b, sizeB, a[i]))
      append(c, sizeC, a[i]);
  reverse(c, *sizeC);
}


int main() {
  // Ввод массива a
  size_t sizeA;
  scanf("%llud", &sizeA);
  int a[sizeA];
  inputArray(a, sizeA);
  // Ввод массива b
  size_t sizeB;
  scanf("%llud", &sizeB);
  int b[sizeB];
  inputArray(b, sizeB);
  // Выделение места под массив c
  size_t sizeC = 0;
  int c[sizeA];

  sortUniqueNumbers(a, sizeA, b, sizeB, c, &sizeC);

  outputArray(c, sizeC);

  return 0;
}
```

Задание №7. Дана целочисленная последовательность, содержащая как положительные, так и отрицательные числа. Упорядочить последовательность следующим образом: сначала идут отрицательные числа, упорядоченные по невозрастанию, потом положительные, упорядоченные по неубыванию.

Тестовые данные:

Входные данные	Выходные данные
n = 7 3 2 1 1 -4 -5 -6	-4 -5 -6 1 1 2 3
n = 8 -3 -2 -1 0 1 2 3 4	-1 -2 -3 0 1 2 3 4

Код программы:

```c
#include <stdio.h>

// Ввод массива a размера size
void inputArray(int *a, const size_t size) {
  for (size_t i = 0; i < size; i++)
    scanf("%d", &a[i]);
}

// Вывод массива a размера size
void outputArray(int *a, const size_t size) {
  for (size_t i = 0; i < size; i++)
    printf("%d ", a[i]);
}
```

```c
// Возвращает количество отрицательных элементов в массиве a рамера size.
int countNegativeElement(const int *a, const size_t size) {
  int countNegativeElement = 0;
  for (size_t i = 0; i < size; i++)
    if (a[i] < 0)
      countNegativeElement++;
  return countNegativeElement;
}

// Обменивает значения двух переменных a и b типа int
void swap(int *a, int *b) {
  int t = *a;
  *a = *b;
  *b = t;
}

// Изменяет порядок элементов массива a размера size на обратный
void reverse(int *a, const size_t size) {
  for (size_t firstIndex = 0, lastIndex = size - 1; firstIndex < lastIndex;
       firstIndex++, lastIndex--)
    swap(&a[firstIndex], &a[lastIndex]);
}

// Возвращает индекс минимального элемента в массиве a размера size
size_t minIndex(int const *a, size_t const size) {
  size_t minIndex = 0;
  for (size_t i = 0; i < size; i++)
    if (a[i] < a[minIndex])
      minIndex = i;
  return minIndex;
}

// Сортирует выбором массив a размера size по неубыванию
void selectionSort(int *a, size_t const size) {
  for (size_t i = 0; i < size - 1; i++) {
    swap(&a[i], &a[i + minIndex(&a[i], size - i)]);
  }
}

// Сортирует массив a размера size таким образом, что сначала идут
// отрицательные элементы в порядке невозрастания, а потом положительные
// элементы в порядке неубывания
void sortNegativeDecreasePositiveIncrease(int *a, size_t const size) {
  selectionSort(a, size);
  size_t lastNegativeIndex = countNegativeElement(a, size);
  reverse(a, lastNegativeIndex);
}

int main() {
  size_t n;
  scanf("%llud", &n);
  int a[n];
  inputArray(a, n);

  sortNegativeDecreasePositiveIncrease(a, n);

  outputArray(a, n);

  return 0;
}
```

Задание №8. Дана целочисленная последовательность и целое число `x`. Определить, есть ли `x` среди членов последовательности, и если нет, то найти члены последовательности, ближайшие к `x` снизу и сверху.

Тестовые данные:

Входные данные	Выходные данные
`x = 6, n = 5` `1 3 6 2 5`	“x – элемент последовательности“
`x = 4, n = 5` `1 3 6 2 5`	3 5
`x = 0, n = 5` `1 3 6 2 5`	“-inf” 1
`x = 7, n = 5` `1 3 6 2 5`	6 “+inf”

Код программы:

```c
#include <stdio.h>
#include <windows.h>

// Ввод массива a размера size
void inputArray(int *a, size_t const size) {
  for (size_t i = 0; i < size; i++)
    scanf("%d", &a[i]);
}

// Возвращает позицию элемента со значением x. Поиск осуществляется в массиве a
// размера size, начиная с первого элемента. После нахождения первого
// подходящего элемента — поиск прекращается.
int linearSearch(const int *const a, const size_t size, int x) {
  for (int i = 0; i < size; i++)
    if (a[i] == x)
      return i;
  return -1;
}

// Определяет индексы элементов массива a размера size, значения которых
// наиболее близки к значению x. Где minIndex - индекс элемента, значение
// которого меньше x. maxIndex - индекс элемента, значение которого больше x.
void closestToX(int const *const a, size_t const size, int const x,
                int *minIndex, int *maxIndex) {
  for (int i = 0; i < size; i++) {
    if (*minIndex == -1 && a[i] < x ||
        *minIndex != -1 && a[i] < x && a[i] > a[*minIndex])
      *minIndex = i;
    else if (*maxIndex == -2 && a[i] > x ||
             *maxIndex != -2 && a[i] > x && a[i] < a[*maxIndex])
      *maxIndex = i;
  }
}

// Выводит: "-inf", если x равен -1, "+inf, если x равен -2 и x-овый элемента
// массива a в остальных случаях.
void printIf(int const *const a, int const x) {
  if (x == -1)
    printf("-inf ");
  else if (x == -2)
    printf("+inf ");
  else
    printf("%d ", a[x]);
}
```

```c
int main() {
  SetConsoleOutputCP(CP_UTF8);
  int x;
  size_t n;
  scanf("%d %llud", &x, &n);
  int a[n];
  inputArray(a, n);

  if (linearSearch(a, n, x) != -1)
    printf("x - Элемент последовательности");
  else {
    int minIndex = -1, maxIndex = -2;
    closestToX(a, n, x, &minIndex, &maxIndex);
    printIf(a, minIndex);
    printIf(a, maxIndex);
  }

  return 0;
}
```

Задание №9. Дана целочисленная последовательность. Получить массив из уникальных элементов последовательности.

Тестовые данные:

Входные данные	Выходные данные
n = 5 1 2 4 1 2	4
n = 5 1 2 3 4 5	1 2 3 4 5
n = 5 1 1 1 1 1	“Последовательность пуста”

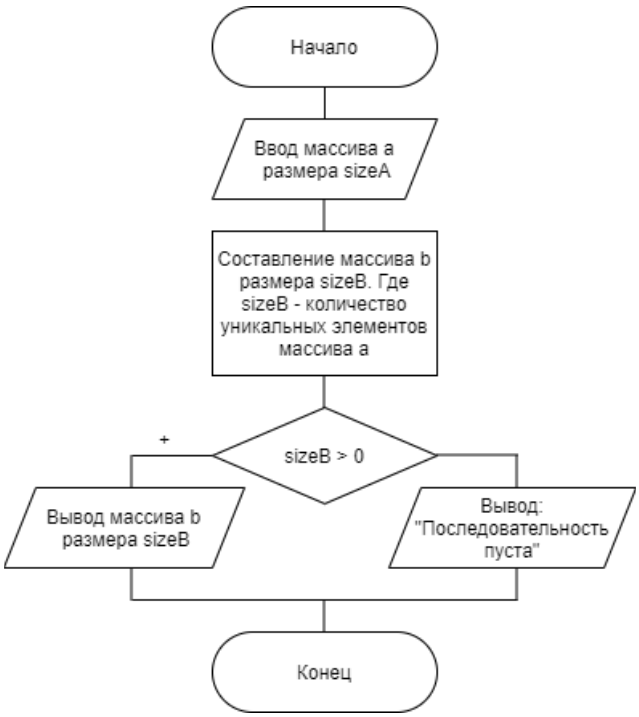
Выделение подзадач:

1. Ввод массива

2. Создание массива из уникальных элементов
   (a) Подсчет количества вхождений значения x в массив
   (b) Добавление элемента в массив

3. Вывод массива

Алгоритм в укрупненных блоках.



Код программы:

```c
#include <stdio.h>
#include <windows.h>

// Ввод массива a размера size
void inputArray(int *a, size_t const size) {
  for (size_t i = 0; i < size; i++)
    scanf("%d", &a[i]);
}

// Вывод массива a размера size
void outputArray(int *a, size_t const size) {
  for (size_t i = 0; i < size; i++)
    printf("%d ", a[i]);
}

// Добавляет элемент в конец массива a размера size
void append(int *a, size_t *size, int x) {
  a[*size] = x;
  (*size)++;
}

// Возвращает количество элементов массива a размера size, равных x
int countElementX(int const *a, size_t const size, int const x) {
  int countElement = 0;
  for (size_t i = 0; i < size; i++)
    if (a[i] == x)
      countElement++;
  return countElement;
}

// Составляет массив b размера sizeB из уникальных элементов
// массива a размера sizeA
void sequenceUniqueNumbers(int const *const a, size_t const sizeA,
                           int *b, size_t *sizeB) {
  for (int i = 0; i < sizeA; i++) {
    if (countElementX(a, sizeA, a[i]) == 1)
      append(b, sizeB, a[i]);
  }
}

int main() {
  SetConsoleOutputCP(CP_UTF8);
  size_t sizeA, sizeB = 0;
  scanf("%llud", &sizeA);
  int a[sizeA], b[sizeA];
  inputArray(a, sizeA);

  sequenceUniqueNumbers(a, sizeA, b, &sizeB);

  if (sizeB > 0)
    outputArray(b, sizeB);
  else
    printf("Последовательность пуста");

  return 0;
}
```