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 №19.8

по дисциплине: «Реализация функций
для работы с одномерными массивами в стиле С»

Выполнил/а: ст. группы ВТ-231
Кисиль Николай Владимирович

Проверили:
Черников Сергей Викторович
Новожен Никита Викторович

Белгород, 2023 г.

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

Содержание работы

Задача 1: Ввод массива a размера n

Спецификация функции inputArray:

1. Заголовок `void inputArray(int *a, const int n)`
2. Назначение: ввод массива a размера n

Код:

```
void inputArray(int *a, const int n) {
    for (size_t i = 0; i < n; i++)
        scanf("%d", &a[i]);
}
```

Задача 2: Вывод массива a размера n

Спецификация функции outputArray:

1. Заголовок `void outputArray(int *a, const int n)`
2. Назначение: вывод массива a размера n

Код:

```
void outputArray(int *a, const int n) {
    for (size_t i = 0; i < n; i++)
        printf("%d ", a[i]);
    printf("\n");
}
```

Задача 3: Поиск позиции элемента со значением x с начала массива

Пример тестовых данных:

№	Входные данных	Выходные данные
1	6 4 5 4	1
2	1 2 3 4 1	0
3	4 2 4 0	-1

Спецификация функции getIndex:

1. Заголовок `unsigned getIndex(const int *a, const size_t n, const int x)`
2. Назначение: поиск позиции элемента со значением x с начала массива

Код:

```
unsigned getIndex(const int *a, const size_t n, const int x) {
    for (size_t i = 0; i < n; i++) {
        if (a[i] == x) {
            return i;
        }
    }
    return -1;
}
```

Задача 4: Поиск позиции первого отрицательного элемента

Пример тестовых данных:

№	Входные данных	Выходные данные
1	6 -4 -5	1
2	6 4 5	-1

Спецификация функции getFirstNegativeIndex:

1. Заголовок `size_t getFirstNegativeIndex(const int *a, const size_t n)`
2. Назначение: поиск позиции первого отрицательного элемента

Код:

```
size_t getFirstNegativeIndex(const int *a, const size_t n) {
    for (size_t i = 0; i < n; i++)
        if (a[i] < 0)
            return i;
    return -1;
}
```

Задача 5: Поиск позиции элемента с начала массива (по функции-предикату).

Пример тестовых данных:

№	Входные данных	Выходные данные
1	6 4 5 четное	0
2	1 2 3 4 отрицательное	-1
3	1 8 6 2 простое	3

Спецификация функции getFirstIndexPredicate:

1. Заголовок `size_t getFirstIndexPredicate(const int *a, const size_t n, int (*predicate) (int))`
2. Назначение: поиск позиции элемента с начала массива (по функции-предикату).

Код:

```
int isEven(int x) {
    return x % 2 == 0;
}
int isNegative(int x) {
    return x < 0;
}

int isPrime(const int n) {
    int max_d = sqrt(n);
    int d = 3;
    int is_prime = !(n == 1 || n % 2 == 0 && n != 2);

    while (d <= max_d && is_prime) {
        is_prime = n % d;
        d += 2;
    }
    return is_prime;
}

size_t getFirstIndexPredicate(const int *a, const size_t n, int (*predicate)
(int)) {
    for(size_t i = 0; i < n; i++) {
        if(predicate(a[i])) {
            return i;
        }
    }
    return -1;
}
```

Задача 6: ** Поиск позиции последнего чётного элемента

Пример тестовых данных:

№	Входные данных	Выходные данные
1	1 2 3 4	3
2	1 2 2 1	2
3	1 3 5 7	-1

Спецификация функции getLastEvenIndex:

1. Заголовок `size_t getLastEvenIndex(const int *a, const size_t n)`
2. Назначение: поиск позиции последнего чётного элемента

Код:

```
size_t getLastEvenIndex(const int *a, const size_t n) {
    for (size_t i = n - 1; i > 0; i--)
        if (a[i] % 2 == 0)
            return i;
    return -1;
}
```

Задача 7: **Поиск позиции с конца массива (по функции-предикату)

Пример тестовых данных:

№	Входные данных	Выходные данные
1	2 5 6 четное	2
2	1 2 3 4 отрицательное	-1
3	1 5 6 2 простое	3

Спецификация функции getLastIndexPredicate:

1. Заголовок `size_t getLastIndexPredicate(const int *a, const size_t n, int (*predicate) (int))`
2. Назначение: поиск позиции с конца массива (по функции-предикату)

Код:

```
size_t getLastIndexPredicate(const int *a, const size_t n, int (*predicate)
(int)) {
    for(size_t i = n - 1; i > 0; i--) {
        if(predicate(a[i])) {
            return i;
        }
    }
    return -1;
}
```

Задача 8: Вычисление количества отрицательных элементов.

Пример тестовых данных:

№	Входные данных	Выходные данные
1	6 -4 -5	2
2	6 4 5	0
3	1 -2 3 -4	2

Спецификация функции `getCountNegative`:

1. Заголовок `size_t getCountNegative(const int *a, const size_t n)`
2. Назначение: вычисление количества отрицательных элементов

Код:

```
size_t getCountNegative(const int *a, const size_t n) {
    int result = 0;
    for(size_t i = 0; i < n; i++) {
        if(a[i] < 0) {
            result++;
        }
    }
    return result;
}
```

Задача 9: ** Вычисление количества элементов массива, удовлетворяющих функции-предикату.

Пример тестовых данных:

№	Входные данных	Выходные данные
1	1 5 6 4 7 четное	2
2	3 5 8 10 простое	2
3	-1 4 5 6 отрицательное	1
4	1 2 3 4 отрицательное	0

Спецификация функции `getCountPredicate`:

1. Заголовок `size_t getCountPredicate(const int *a, const size_t n, int (*predicate) (int))`
2. Назначение: вычисление количества элементов массива, удовлетворяющих функции-предикату

Код:

```
size_t getCountPredicate(const int *a, const size_t n, int (*predicate)
(int)) {
    int result = 0;
    for(size_t i = 0; i < n; i++) {
        if(predicate(a[i])) {
            result++;
        }
    }
    return result;
}
```

Задача 10: Изменение порядка элементов массива на обратный.

Пример тестовых данных:

№	Входные данных	Выходные данные
1	1 2 3 4	4 3 2 1
2	6 -4 -5	-5 -4 6

Спецификация функции `reverseArray`:

1. Заголовок `void reverseArray(int * a, const size_t n)`
2. Назначение: изменение порядка элементов массива на обратный

Код:

```
void swap(int * const a, int * const b) {
    const int temp = *a;
    *a = *b;
    *b = temp;
}

void reverseArray(int * a, const size_t n) {
    for(size_t i = 0, j = n - 1; i < j; i++, j--) {
        swap(&a[i], &a[j]);
    }
}
```

Задача 11: Проверка, является ли последовательность палиндромом.

Пример тестовых данных:

№	Входные данных	Выходные данные
1	1	1
2	1 2 1	1
3	1 2 2 1	1
4	1 2 3 4	0

Спецификация функции `isPalindrome`:

1. Заголовок `bool isPalindrome(const int * a, const size_t n)`
2. Назначение: проверка, является ли последовательность палиндромом

Код:

```
bool isPalindrome(const int * a, const size_t n) {
    int is_palindrome = 1;
    for(size_t i = 0, j = n - 1; i < j; i++, j--) {
        if(a[i] != a[j]) {
            is_palindrome = 0;
        }
    }
    return is_palindrome;
}
```

Задача 12: Сортировка массива выбором

Пример тестовых данных:

№	Входные данных	Выходные данные
1	1 2 3 4	1 2 3 4
2	4 2 1 3	1 2 3 4
3	-1 2 -3 4	-3 -1 2 4

Спецификация функции `selectionSort`:

1. Заголовок `void selectionSort(int *a, const size_t n)`
2. Назначение: сортировка выбором по неубыванию

Код:

```
void selectionSort(int *a, const size_t n) {
    for (int i = 0; i < n - 1; i++) {
        int min_index = i;
        for (int j = i + 1; j < n; j++)
            if (a[j] < a[min_index])
                min_index = j;
        swap(&a[i], &a[min_index]);
    }
}
```

Задача 13: Удаление из массива всех нечетных элементов

Пример тестовых данных:

№	Входные данных	Выходные данные
1	1 2 3 4	2 4
2	1 3 5 6	6
3	1 3 5 7	

Спецификация функции deleteOdd:

1. Заголовок `void deleteOdd(int * a, const size_t n)`
2. Назначение: удаление из массива всех нечетных элементов

Код:

```
void deleteOdd(int * a, const size_t n) {
    int b[n];
    int j = 0;
    for(size_t i = 0; i < n; i++) {
        if(isEven(a[i])) {
            b[j] = a[i];
            j++;
        }
    }
    outputArray(b, j);
}
```

Задача 14: Вставка элемента в массив с сохранением относительного порядка других элементов

Пример тестовых данных:

№	Входные данных	Выходные данные
1	1 2 3 4 Pos = 3 Value = 5	1 2 3 5 4
2	2 4 5 6 7 Pos = 3 Value = 5	2 4 5 5 6 7

Спецификация функции insert:

1. Заголовок void insert(int *a, int * n, const size_t pos, const int value)
2. Назначение: вставка элемента в массив с сохранением относительного порядка других элементов

Код:

```
void insert(int *a, int * n, const size_t pos, const int value) {
    for (size_t i = *n - 1; i >= pos; i--)
        a[i + 1] = a[i];
    a[pos] = value;
    (*n)++;
}
```

Задача 15: Добавление элемента в конец массива

Пример тестовых данных:

№	Входные данных	Выходные данные
1	1 2 3 4 Value = 5	1 2 3 4 5

Спецификация функции insertEnd:

1. Заголовок void insertEnd(int *a, int * n, const int value)
2. Назначение: вставка элемента в конец массива

Код:

```
void insertEnd(int *a, int * n, const int value) {
    a[*n] = value;
    (*n)++;
}
```

Задача 16: Удаление элемента с сохранением относительного порядка других элементов

Пример тестовых данных:

№	Входные данных	Выходные данные
1	1 2 3 4 Pos = 2	1 2 4

Спецификация функции deleteByPosSaveOrder:

1. Заголовок void deleteByPosSaveOrder(int *a, int * n, const size_t pos)
2. Назначение: удаление элемента с сохранением относительного порядка других элементов

Код:

```
void deleteByPosSaveOrder(int *a, int * n, const size_t pos) {
    for (size_t i = pos; i < *n - 1; i++)
        a[i] = a[i + 1];
    (*n)--;
}
```

Задача 17: Удаление элемента без сохранения относительного порядка других элементов

Пример тестовых данных:

№	Входные данных	Выходные данные
1	1 2 3 4 Pos = 1	1 4 3

Спецификация функции deleteByPosUnsaveOrder:

1. Заголовок void deleteByPosUnsaveOrder(int *a, int * n, size_t pos)
2. Назначение: удаление элемента без сохранения относительного порядка других элементов

Код:

```
void deleteByPosUnsaveOrder(int *a, int * n, size_t pos) {
    a[pos] = a[*n - 1];
    (*n)--;
}
```

Задача 18: ** Реализуйте циклический сдвиг массива влево на k позиций

Пример тестовых данных:

№	Входные данных	Выходные данные
1	1 2 3 4 5 1	2 3 4 5 1
2	1 2 3 4 5 2	3 4 5 1 2
3	1 2 3 4 5 5	1 2 3 4 5
4	1 2 3 4 5 212	3 4 5 1 2

Спецификация функции shift:

1. Заголовок void shift(int *a, size_t n, int k)
2. Назначение: циклический сдвиг массива влево на k позиций.

Код:

```
void shift(int *a, size_t n, int k) {
    int new_k = k % n;
    for(size_t i = 0; i < new_k; i++) {
        int temp = a[0];
        for(size_t j = 0; j < n - 1; j++) {
            a[j] = a[j + 1];
        }
        a[n - 1] = temp;
    }
}
```

Задача 19: ** Реализуйте функцию $forEach$, которая применяет функцию f к элементам массива a размера $size$.

Пример тестовых данных:

№	Входные данных	Выходные данные
1	-1 -2 -3 4 -5 f = модуль	1 2 3 4 5
2	1 -2 3 -4 5 $f = x^2$	1 4 9 16 25

Спецификация функции forEach:

1. Заголовок void forEach (int *a, size_t n, int (*predicate) (int))
2. Назначение: применяет функцию f к элементам массива a размера $size$.

Код:

```
int abs(int a) {
    return a < 0 ? -a : a;
}

int squaring(int a) {
    return a * a;
}

void forEach (int *a, size_t n, int (*predicate) (int)) {
    for(size_t i = 0; i < n; i++) {
        a[i] = predicate(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'.

Пример тестовых данных:

№	Входные данных	Выходные данные
1	-1 -2 -3 4 -5 x > 0	1
2	-1 -2 -3 -4 -5 x > 0	0

Спецификация функции anyPredicate:

1. Заголовок bool anyPredicate(int *a, size_t n, int (*predicate) (int))
2. Назначение: которая возвращает значение 'истина', если хотя бы один элемент массива a размера $size$ удовлетворяют функции-предикату f, иначе – 'ложь'

Код:

```
int isPositive(int a) {
    return a > 0;
}

bool anyPredicate(int *a, size_t n, int (*predicate) (int)) {
    for(size_t i = 0; i < n; i++) {
        if(predicate(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’.

Пример тестовых данных:

№	Входные данных	Выходные данные
1	1 2 3 4 5 x > 0	1
2	1 2 3 -4 5 x > 0	0

Спецификация функции allPredicate:

1. Заголовок bool allPredicate(int *a, size_t n, int (*predicate) (int))
2. Назначение: возвращает значение ’истина’, если все элементы массива a размера $size$ удовлетворяют функции-предикату f, иначе – ’ложь’

Код:

```
bool allPredicate(int *a, size_t n, int (*predicate) (int)) {
    int is_predicate = true;
    for(size_t i = 0; i < n; i++) {
        if(!predicate(a[i])) {
            is_predicate = false;
        }
    }
    return is_predicate;
}
```

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.

Пример тестовых данных:

№	Входные данных	Выходные данные
1	0 2 -3 -4 5 x > 0	2 5 0 -3 -4
2	1 2 3 4 5 x - простое	2 3 5 1 4

Спецификация функции splitArray:

1. Заголовок void splitArray(int *a, size_t n, int (*predicate) (int))
2. Назначение: разделяет элементы массива a размера $size$ на элементы, удовлетворяющие функции-предикату f, сохраняя в массиве b, иначе – в массиве c.

Код:

```
void splitArray(int *a, size_t n, int (*predicate) (int)) {
    int b[n], c[n];
    int bj = 0, cj = 0;
    for(size_t i = 0; i < n; i++) {
        if(predicate(a[i])) {
            b[bj] = a[i];
            bj++;
        } else {
            c[cj] = a[i];
            cj++;
        }
    }
    outputArray(b, bj);
    outputArray(c, cj);
}
```

Вывод: получили навыки в написании функций для работы с одномерными массивами в стиле С.