Лабораторная работа № 5

Вариант 15

Новицкий

№ 1

Заполнить массив случайными числами. Удалить из массива все элементы с чётным значением

Main.c

```
#include <stdio.h>
#include <stdlib.h>
#include "task1.h"
#include "task-2.h"
#include "task-3.h"
#include <time.h>
#include <ctype.h>

int main(int argc, const char * argv[]) {
  int first_length;
   char input[100];

       do {
            printf("Введите массива в диапазоне от 1 до 50\n");

          // считывание данных с клавиатуры (куда зписать, до какого момента, что именно с клавы)
            if (fgets(input, sizeof(input), stdin) == NULL) {
               printf("Ошибка ввода, попробуйте ещё раз\n");
               continue;
            }

            // ф-ия поиска числа из строки, в которую записались данные (где искать, что искать, куда положить)
            if (sscanf(input, "%d", &first_length) != 1) {
               printf("Ошибка ввода, введите целое число\n");
               continue;
            }

            if (first_length < 1 || first_length > 50) {
               printf("Ошибка! Введите число в диапазоне от 1 до 50\n");
            }

        } while (first_length < 1 || first_length > 50);

   create_random_array_and_delete(&first_length); // создание массива случайных элементов

   return 0;
}
```

task-1.h

```
#ifndef task1_h
#define task1_h

#include <stdio.h>

void create_random_array_and_delete(int *length);
void delete_even_numbers(int **array, int *length);

#endif /* task1_h */
```

task-1.c

```
#include "task1.h"
#include <stdlib.h>
#include <stdio.h>
#include <time.h>

void create_random_array_and_delete(int *length){
    srand((unsigned int)time(NULL));

    int *array_1 = (int*)malloc(*length  * sizeof(int)); // выделение памяти для массива

    if (!array_1){ // проверка на то, что памяти хватает

        printf("Ошибка выделения памяти\n");
        return;

    }


    for(int i = 0; i < *length ; i++){ // заполнение массива

        array_1[i] = (rand() % 198) - 99; // всего 99 + 99 = 198 случайных элементов, -99 сдвиг
диапазона в (-)

    }



    printf("Исходный массив:\n");
    for(int i = 0; i < *length; i++){
        printf("%d ", array_1[i]);
    }


    delete_even_numbers(&array_1, &(*length));


    for(int i = 0; i < *length; i++){

        printf("%d ",array_1[i]);

    }
```

```
    printf("\n");

    free(array_1);

}


void delete_even_numbers(int **array, int *length){

    int new_length = 0;
    int j = 0;

    for(int i = 0; i < *length; i++){ // задаю длину нового массива, чтобы впоследствии выделить нужное кол-во памяти

        if((*array)[i] % 2 != 0){
            new_length++;
        }

    }

    int *new_array = (int*) malloc (sizeof(int) * new_length); // выделение памяти новому массиву

    if (*new_array){ // проверка на то, что памяти хватает

        printf("Ошибка выделения памяти\n");
        return;

    }


    for(int i = 0; i < *length; i++){ // заполнение нового массива

        if((*array)[i] % 2 != 0){

            new_array[j++] = (*array)[i]; // нужно получить не адрес, а значения массива, поэтому сначала нужно его разыменовать

        }

    }

    free(*array);
    *array = new_array; // Обновляем указатель на новый массив
    *length = new_length;

    printf("\n");
    printf("Массив без четных элементов: ");
    printf("\n");


}
```

№ 2

В двумерном массиве натуральных чисел (количество чисел в строке может быть различным, последнее число - 0) удалить строку с минимальной суммой элементов (порядок остальных строк не менять).

Main.c

```
#include <stdio.h>
#include <stdlib.h>
#include "task1.h"
#include "task-2.h"
#include "task-3.h"
#include <time.h>
#include <ctype.h>

int main(int argc, const char * argv[]) {

        int rows_count;

       char input[100];

          do {
               printf("Введите массив в диапазоне от 1 до 50\n");

             // считывание данных с клавиатуры (куда записать, до какого момента, что именно с клавы)
               if (fgets(input, sizeof(input), stdin) == NULL) {
                   printf("Ошибка ввода, попробуйте ещё раз\n");
                   continue;
               }

               // ф-ия поиска числа из строки, в которую записались данные (где искать, что искать, куда положить)
               if (sscanf(input, "%d", &rows_count) != 1) {
                   printf("Ошибка ввода, введите целое число\n");
                   continue;
               }

               if (rows_count < 1 || rows_count > 50) {
                   printf("Ошибка! Введите число в диапазоне от 1 до 50\n");
               }

           } while (rows_count < 1 || rows_count > 50);

   // реализация мта аватвао вотамто ваоа создания массива и удаления строки с наименьшеецц суммой

   create_array(&rows_count);
```

```
    return 0;
}
```

task-2.h

```
#ifndef task_2_h
#define task_2_h

#include <stdio.h>

void create_array(int *length);
int create_row( int *elements, int row_len );

#endif /* task_2_h */
```

task-2.c

```
#include "task-2.h"
#include <stdio.h>
#include <stdlib.h>

void create_array(int *length){

    int **array = (int**) calloc (*length, sizeof(int*)); // массив указателей на строки

    for(int i = 0; i < *length; i++ ){
        int row_len;

        printf("Введите длину %d-й строки: ", i+1);

        scanf("%d", &row_len);

        array[i] = (int*) calloc (row_len + 1, sizeof(int)); // выделение памяти для строки длинной
row_len

        printf("Введите элементы %d-й строки: ", i+1);

        for(int j = 0; j < row_len; j++){ // цикл заполнения массива
            scanf("%d", &array[i][j]);
        }


    }



    // поиск строки с наименьшей суммой элементов
    int min_sum = 0;
    int min_row = 0;
    int sum = 0;

    // поиск суммы элементов первой строки

    for(int i = 0; i < 1; i++){
```

```
        int k = 0;
        while(array[i][k] != 0){
            min_sum += array[i][k];
            k++;
        }
    }

    // поиск всех остальных сумм и сравнение их между собой

    for(int i = 0; i < *length; i++ ){
        sum = 0; // сброс суммы
        int k = 0;
        while (array[i][k] != 0){
            sum += array[i][k];
            k++;
        }
        if (sum < min_sum){
            min_sum = sum;
            min_row = i;
        }

        }

    // удаление строки с наименьшей суммой

        free(array[min_row]);


        for(int i = min_row; i < *length-1; i++){
            array[i] = array[i+1]; // сдвиг влево строк, т.к двумерный массив это массив массивов
        }

        (*length)--;
        *array = (int**) realloc(*array, (*length) * sizeof(int*)); // обновляю размер выделенной
памяти


    // вывод массива
    printf("\n");
    printf("Новый массив:\n");
    for(int i = 0; i < *length; i++){

        int j = 0;
        while (array[i][j] != 0) {
            printf("%d ", array[i][j]);
            j++;
        }

        printf("0\n"); // для обозначения конца строки
    }

    printf("Строка с минимальной суммой = %d, сумма равна = %d \n", min_row + 1, min_sum);

    for(int i = 0; i < *length; i++){ // освобождение памяти (тобишь каждой строки по порядку)
        free(array[i]);
    }

            free(array); // освобождаю массив указателей }
```

// удалить строку с минимальной суммой элементов

№ 3

Дан двумерный массив ненулевых целых чисел. Определить максимально длинную последовательность положительных чисел. Массив просматривается построчно сверху вниз, а в каждой строке - слева направо. Сохранение знака при переходе на новую строку также учитывать.

Main.c

```
int main(int argc, const char * argv[]) {

    int row_count;
    char input[100];

        do {
            printf("Введите массива в диапазоне от 1 до 50\n");

          // считывание данных с клавиатуры (куда зписать, до какого момента, что именно с клавы)
            if (fgets(input, sizeof(input), stdin) == NULL) {
                printf("Ошибка ввода, попробуйте ещё раз\n");
                continue;
            }

            // ф-ия поиска числа из строки, в которую записались данные (где искать, что искать, куда положить)
            if (sscanf(input, "%d", &row_count) != 1) {
                printf("Ошибка ввода, введите целое число\n");
                continue;
            }

            if (row_count < 1 || row_count > 50) {
                printf("Ошибка! Введите число в диапазоне от 1 до 50\n");
            }

        } while (row_count < 1 || row_count > 50);


    // создание массива строк + выделение памяти для него

    int **array = (int**) calloc (row_count, sizeof(int*));

    // функция создания масссива

    create_new_array(array, &row_count);

    // функция вывода массива

    print_array(array, &row_count);


    printf("\n"); // убрать

    // поиск длины последовательности положительных элементов
```

```
  printf("Макимальная длина последовательности = %d\n", find_sequence(array, &row_count));

    return 0; }
```

task-3.h

```
#ifndef task_3_h
#define task_3_h

#include <stdio.h>

void create_new_array(int **array, int *length);
void print_array(int **array, int *length);
int find_sequence(int **array, int *length);


#endif /* task_3_h */
```

task-3.c

```
#include "task-3.h"
#include <stdlib.h>
#include <stdio.h>

void create_new_array(int **array, int *length){
    int row_len;

    for(int i = 0; i < *length; i++){

        printf("Введите длину %d-й строки: ",i+1);
        scanf("%d", &row_len);

        array[i] = (int*) calloc (row_len + 1, sizeof(int)); // чтобы добавить ноль в конце строки,
понять когда конец

        printf("Введите элементы %d-й строки: ", i+1);
        for(int j = 0; j < row_len; j++){
            scanf("%d", &array[i][j]);
        }



    }


}

void print_array(int **array, int *length){

    printf("Исходный массив:\n");

    for(int i = 0; i < *length; i++){


        for(int j = 0; array[i][j] != 0; j++){
```

```
            printf("%d ", array[i][j]);

        }

        printf("\n");

    }


}

int find_sequence(int **array, int *length){

    int max_length = 0;
    int current_length = 0;
    int flag = 0;



    for(int i = 0; i < *length; i++){

        int j = 0;

        while(array[i][j] != 0){

            if (array[i][j] > 0) {
                current_length++;
                flag = 1; // надежда, что последний элемент положительный (условное
сохранение)
            }
            else {
                if(current_length > max_length){
                    max_length = current_length;
                }

                current_length = 0;
                flag = 0; // надежда, что последний элемент отрицательный, чтобы проверить в
будущем след строку
            }

            j++;

        }

        if(!flag){ // флаг нужен ради того, чтобы проверить начало следующей строки
            current_length = 0;
        }

    }

    if(current_length > max_length ){

        max_length = current_length;

    }



    return max_length;}
```