Министерство образования Республики Беларусь

г. Минск

Государственное учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Лабораторная работа №6

Выполнил: Новицкий Е.Ю

студент группы 458302

Проверила: Желтко Ю.Ю.

Минск 2024

// 1.В одномерном массиве выполнить сортировку элементов методом Хоара

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <ctype.h>
#include <string.h>

void execute_verification(int *variable){

   do {
      char input[100];
                  printf("Введите длину 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", variable) != 1) {
                     printf("Ошибка ввода, введите целое число\n");
                     continue;
                  }

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

               } while (*variable < 1 || *variable > 50);

}
```

```
void binary_verification(int *var01){
   do {

     //  printf("Введите 0 либо 1!");

       if (scanf("%d", var01) != 1){
          printf("Ошибка ввода! Введите 0 либо 1\n");
           while(getchar() != '\n');
          continue;
       }

       if( *var01 < 0 || *var01 > 1){
          printf("Ошибка ввода! Введите 0 либо 1\n");
          continue;
       }



                } while (*var01 < 0 || *var01 > 1);

}




// быстрая сортировка

void hoar_sort(int *array, int left, int right){


if(left >= right) return; // проверка что массив не пустой
```

```
int control_point = array[(right + left) / 2]; // находим контрольную точку с которой будем сравнивать
int i = left; // определяем левую границу
int j = right; // определяем правую границу

while (i <= j){ // выполняем пока начало и конец не перешагнули через друг друга

while(array[i] < control_point) i++; // указатель движется вправо пока не найдет больше опорного
while(array[j] > control_point) j--; // указатель движется влево пока не найдет меньше опорного

    if( i <= j){ // если начало не перешагнуло конец (относительно опорного)
int temp = array[i]; // то
array[i] = array[j]; // меняем их
array[j] = temp; // местами

i++; // двигаемся дальше
j--; // для продолжения поиска (гарантия пересечения, чтобы разбить массив)

    }

}

hoar_sort(array, left, j); // вызываю мою функцию для левой части массива
                                                    //(рекурсивные вызовы)
hoar_sort(array, i, right); // вызываю мою функцию для правой части массива

}
```

```
int main(void){

// 1.В одномерном массиве выполнить сортировку элементов методом Хоара

        int length;
        int flag;
        srand((unsigned int)time(NULL));
```

```
    execute_verification(&length);

    int *user_array = (int*) calloc (length, sizeof(int));




    printf("Выберите тип заполенения массива: (1 - случайные значения, 0 - ручной ввод)\n");

    binary_verification(&flag); // обработка ввода 0 и 1

    if (flag == 1){
        for(int i = 0; i < length; i++){
            user_array[i] = (rand() % 20) - 10; // от - 10 до 10
        }
    }

    else if (flag == 0){
        printf("Введите массив: ");
        for(int i = 0; i < length; i++){
            scanf("%d", &user_array[i]);
        }
    }




    printf("Массив пользователя: ");
    for (int i = 0; i < length; i++){
        printf("%d ", user_array[i]);
    }

    printf("\n");



// вызов функции соритровки
```

```
 hoar_sort(user_array, 0, length - 1);



printf("Отсоритрованный массив: ");
for(int i = 0; i < length; i++){
    printf("%d ", user_array[i]);
 }

 printf("\n");


free(user_array);


    return 0;
}
```

//2. В матрице размером NxM выполнить сортировку строк, номер которых кратен k, по убыванию суммы положительных элементов.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <ctype.h>
#include <string.h>


void verification_for_k(int *k_rate, int count_rows){
   do {

       printf("Введите целое число k ( не превышающее количества строк матрицы ):
\n");

       if (scanf("%d", k_rate) != 1){
          printf("Ошибка ввода! Введите целое число\n");
           while(getchar() != '\n');
          continue;
       }

       if( *k_rate > count_rows ){
          printf("Ошибка ввода! k больше, чем количество строк\n");
          continue;
       }



                } while (*k_rate > count_rows);

}
```

```
void binary_verification(int *var01){
   do {

     //  printf("Введите 0 либо 1!");

       if (scanf("%d", var01) != 1){
          printf("Ошибка ввода! Введите 0 либо 1\n");
           while(getchar() != '\n');
          continue;
       }

       if( *var01 < 0 || *var01 > 1){
          printf("Ошибка ввода! Введите 0 либо 1\n");
          continue;
       }



                } while (*var01 < 0 || *var01 > 1);

}









void execute_verification(int *variable){

   do {
      char input[100];
                  //  printf("Введите количество строк 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", variable) != 1) {
                printf("Ошибка ввода, введите целое число\n");
                continue;
            }

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

        } while (*variable < 1 || *variable > 50);

}
```

```
void sort_rows(int **array, int rows, int cols, int k ){

for (int i = k - 1; i < rows; i += k) {  // Обрабатываем строки, кратные k     типо первый кратный

   for (int j = i + k; j < rows; j += k) { // цикл сравнения сум строк        типо второй кратный

        int sum_i = 0;
        int sum_j = 0;

        for (int c = 0; c < cols; c++) {


           if (array[i][c] > 0){ // для первого
               sum_i += array[i][c];
           }
```

```
            if (array[j][c] > 0) { // для второго
                sum_j += array[j][c];
            }

        }



        if (sum_j > sum_i) { // замена
            int *temp = array[i];
            array[i] = array[j];
            array[j] = temp;
        }
    }
  }

}
```

```
int main(void){
    int rows;
    int cols;
    int flag;
    int k;
    srand((unsigned int) time(NULL) );

    // проверка переменных на ввод

    printf("Введите количество строк от 1 до 50\n");
```

```
    execute_verification(&rows);

    printf("Введите количество столбцов от 1 до 50\n");
    execute_verification(&cols);

    // выделение памяти для двумерного массива
    int **array = (int**) calloc (rows, sizeof(int*)); // память для массива указателей (для строк)

    for(int i = 0; i < rows; i++){
        array[i] = (int*) calloc (cols, sizeof(int)); // выделение памяти для каждого столбца
    }

    // создание матрицы
    printf("Выберите тип заполенения массива: (1 - случайные значения, 0 - ручной ввод)\n");

    binary_verification(&flag); // обработка ввода 0 и 1

    if(flag == 1){ // обработка действий
        for(int i = 0; i < rows; i++){
            for (int j = 0; j < cols; j++){
                array[i][j] = (rand() % 20 ) - 10;
              // array[i][j] = (rand() % 10 );
            }
        }
    }

    else if(flag == 0){ // обработка действий
        for(int i = 0; i < rows; i++){
            for(int j = 0; j < cols; j++){
                scanf("%d", &array[i][j]);
            }
        }
    }
```

```
printf("Введите k (сортироваться будут строки, кратные k)\n");
```

```
verification_for_k(&k, rows); // проверка ввода k
printf("\n");




// вывод исходной матрицы

    printf("Исходная матрица:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%d ", array[i][j]);
        }
        printf("\n");
    }
printf("\n");




sort_rows(array, rows, cols, k); // сортировка строк матрицы




printf("Отсортированная матрица:\n");
for(int i = 0; i < rows; i++){
    for(int j = 0; j < cols; j++){
        printf("%d ", array[i][j]);
    }
    printf("\n");
}




// очистка памяти

for(int i = 0; i < rows; i++){
    free(array[i]);
```

```
}
free(array);




return 0;

}
```