Программа сортировки (задания 1 и 2)

Входные и выходные параметры

Программа имеет три входных параметра `n`, `t` и `c` типа `int`.

Параметр `n` представляет собой положительное целое число. Функция сортировки тестируется на массивах длины n и $2n$.

Параметр `t` задает тип массивов, на которых тестируется функция сортировки. Возможные значения: `'r'` – случайные массивы, `'i'` – возрастающие массивы, `'d'` – убывающие массивы.

Параметр `c` принимает два значения `'y'` – проверять корректность проведенной сортировки или `'n'` – не проверять.

Программа имеет три выходных параметра: `t1` – время сортировки массива длины n, `t2` – время сортировки массива длины $2n$ и `t2 / t1` типа `double`.

Пример запуска программы.

```
$ ./sort<Enter>
Input n: 50000<Enter>
Input t (r/i/d): r<Enter>
Input c (y/n): n<Enter>
t1: 1.817137000000e+00
t2: 7.159167000000e+00
3.939805859437
```

Массивы

Программа должна выделять память под хранение четырех массивов `a`, `t` длины n и `a2`, `t2` длины $2n$. Сортировке подвергаются массивы `a` и `a2`.

Массив `t` является копией массива `a` до его сортировки. Массив `t2` является копией массива `a2` до его сортировки.

Рекомендуется использовать один вызов функции `malloc` для выделения памяти сразу под четыре массива.

```
a  = malloc(6 * n * sizeof(int));
...
t  = a  + n;
a2 = t  + n;
t2 = a2 + 2 * n;
```

Элементы возрастающего массива задаются с помощью выражения `a[i] = i`. Элементы убывающего массива задаются с помощью выражения `a[i] = n - i`. Элементы случайного массива задаются с помощью выражения `a[i] = rand()`.

Функция `rand` возвращает очередное псевдослучайное число. Перед формированием случайного массива рекомендуется сделать вызов функции `srand(0)`. Это обеспечит воспроизводимость результата при повторном запуске программы.

Проверка

Вспомогательная функция

```
int check(int *a, int *t, int n);
```

используется для проверки корректности сортировки. На вход эта функция получает указатель `a` на отсортированный массив, указатель `t` на копию массива до его сортировки, а также длину массива `n`. Функция возвращает `0`, если результат сортировки корректен, иначе возвращает `-1`.

Результат сортировки корректен, если одновременно выполняются два условия:

- верно неравенство `a[i] <= a[i+1]` для всех допустимых значений переменной `i`;
- каждое значение встречается в массивах `a` и `t` одинаковое число раз.

Время сортировки

Для замера времени сортировки используется стандартная функция `clock`, прототип которой определен в заголовочном файле `time.h`. Эта функция возвращает число тактов процессора с момента запуска программы. Для представления подобных значений используется специальный тип беззнаковых целых чисел `clock_t`.

Для замера времени сортировки необходимо запомнить число `s` произведенных тактов процессора непосредственно перед вызовом функции сортировки, а также запомнить число `e` произведенных тактов процессора сразу после завершения вызова функции сортировки. Тогда число `e - s` будет равно времени выполнения сортировки, вычисленному в произведенных тактах процессора. Чтобы перевести эту величину в секунды, необходимо воспользоваться выражением

```
(double)(e - s) / CLOCKS_PER_SEC
```

Ниже приведен фрагмент программы иллюстрирующий работу с функцией `clock`.

```
clock_t s, e;
double  t1;
...
s = clock();
sort(a, n);
e = clock();

t1 = (double)(e - s) / CLOCKS_PER_SEC;
```