Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Факультет Компьютерных Технологий и Управления

Кафедра Информатики и прикладной математики

**Системное программное обеспечение**

Лабораторная работа №5

Выполнил:

Хлопков Д. С.

Группа: Р3217

Преподаватель:

Зыков Анатолий Геннадьевич

Санкт-Петербург

2018 г.

**Листинг программы:**

1. #include "stdafx.h"
2. using namespace std;
4. #define ARRAY\_LENGTH 10000
5. #define BUFFER\_SIZE  8
7. typedef enum { ASC, DESC } order\_t;
8. typedef struct { int\* arr; order\_t order; int x; } args\_t;
10. int cmp(int a, int b)
11. {
12. if (a > b) return 1;
13. if (a < b) return -1;
14. return 0;
15. }
17. void sort(int\* arr, size\_t n, order\_t order)
18. {
19. int gap, i, j, temp, ord = (order == ASC ? 1 : -1);
21. for (int gap = n / 2; gap > 0; gap /= 2)
22. {
23. for (int i = gap; i < n; i += 1)
24. {
25. temp = arr[i];
27. j = i;
28. while (j >= gap && cmp(arr[j - gap], temp) == ord)
29. {
30. arr[j] = arr[j - gap];
31. j -= gap;
32. }
34. arr[j] = temp;
35. }
36. }
37. }
39. int bisect(int\* arr, size\_t n, order\_t order, int x)
40. {
41. int leftCmp = (order == ASC ? -1 : 1);
43. int first = 0, last = n;
44. while (first < last)
45. {
46. int i = (first + last) / 2;
48. if (x == arr[i]) return i;
49. else if (cmp(x, arr[i]) == leftCmp) last = i;
50. else first = i + 1;
51. }
53. return -1;
54. }
56. time\_t threadTime(HANDLE hThread)
57. {
58. FILETIME creationTime, exitTime, kernelTime, userTime;
59. ZeroMemory(&creationTime, sizeof(creationTime));
60. ZeroMemory(&exitTime, sizeof(exitTime));
62. GetThreadTimes(hThread, &creationTime, &exitTime, &kernelTime, &userTime);
64. time\_t startTime = ((time\_t)creationTime.dwHighDateTime << 32) + creationTime.dwLowDateTime;
65. time\_t endTime = ((time\_t)exitTime.dwHighDateTime << 32) + exitTime.dwLowDateTime;
66. return (endTime - startTime) / 10;
67. }
69. DWORD WINAPI ThreadSort(LPVOID lpParam)
70. {
71. args\_t args = \*(args\_t\*)lpParam;
73. sort(args.arr, ARRAY\_LENGTH, args.order);
74. bisect(args.arr, ARRAY\_LENGTH, args.order, args.x);
76. return EXIT\_SUCCESS;
77. }
79. int\* readArray()
80. {
81. char buffer[BUFFER\_SIZE];
82. int arr[ARRAY\_LENGTH];
83. FILE\* file;
84. fopen\_s(&file, "..**\\**array.txt", "r");
85. for (int i = 0; i < ARRAY\_LENGTH; i++)
86. {
87. if (fgets(buffer, BUFFER\_SIZE, file) == nullptr) break;
88. arr[i] = atoi(buffer);
89. }
90. fclose(file);
92. return arr;
93. }
95. int main(int argc, char\* argv[])
96. {
97. if (argc != 3 || (string)argv[1] != "asc" && (string)argv[1] != "desc")
98. {
99. cout << "Usage: " << endl << "spo4 {asc|desc} x" << endl;
100. system("pause");
101. return EXIT\_FAILURE;
102. }
104. order\_t order = (string)argv[1] == "asc" ? ASC : DESC;
105. int x = atoi(argv[2]);
107. int\* arr = readArray();
108. args\_t\* args[3];
109. for (int i = 0; i < 3; i++)
110. {
111. args[i] = (args\_t\*)malloc(sizeof(args\_t));
112. args[i]->arr = (int\*)malloc(ARRAY\_LENGTH \* sizeof(int));
113. memcpy(args[i]->arr, arr, ARRAY\_LENGTH \* sizeof(int));
114. args[i]->order = order;
115. args[i]->x = x;
116. }
118. HANDLE hThread[3];
119. DWORD dwThreadId[3];
120. for (int i = 0; i < 3; i++)
121. {
122. hThread[i] = CreateThread(NULL, 0, ThreadSort, args[i], 0, &dwThreadId[i]);
124. if (hThread[i] == NULL)
125. {
126. cout << "CreateThread failed: " << GetLastError() << endl;
127. system("pause");
128. return EXIT\_FAILURE;
129. }
130. }
132. WaitForMultipleObjects(3, hThread, TRUE, INFINITE);
134. DWORD handleCount;
135. GetProcessHandleCount(GetCurrentProcess(), &handleCount);
137. cout << "+------+----------------+" << endl <<
138. "|  ID  |      TIME      |" << endl <<
139. "+------+----------------+" << endl;
141. for (int i = 0; i < 3; i++)
142. {
143. time\_t time = threadTime(hThread[i]);
144. printf("|%6u|%12lli mcs|**\n**", dwThreadId[i], time);
146. CloseHandle(hThread[i]);
147. free(args[i]->arr);
148. free(args[i]);
149. }
151. cout << "+-------+----------------+" << endl << endl;
152. cout << "Handles count: " << handleCount << endl << endl;
154. system("pause");
155. }

**Пример работы программы:**

