**5 ЛАБОРАТОРНАЯ РАБОТА №5**

**«Исследование возможностей управления памятью и обмена данными между процессами в ОС Windows.»**

**5.1 Цель работы**

Изучить возможности программного интерфейса приложений (API) операционных систем Windows по управления памятью и обмена данными между процессами. Приобрести практические навыки использования Win API для управления памятью и обмена данными между процессами.

**5.2 Вариант задания – 17**

**5.2.1** Написать программу MemShare, выполняющую следующие действия:

5.2.1.1 Создание(запуск) процесса MemSort, используя вызов CreateProcess;

5.2.1.2 Выделение с помощью вызова VirtualAlloc заданного по варианту задания (1стр.) количества страниц памяти (страница 4096 байт);

5.2.1.3 Заполнение выделенной памяти случайными числами в диапазоне от 0 до MAX (значение MAX выбирается исходя из типа данных, заданному по варианту задания (тип short));

5.2.1.4 Перевод режима доступа выделенной памяти в ReadOnly (с помощью вызова VirtualProtect);

5.2.1.5 Используя механизм отображения файлов на память (функции CreateFileMapping и MapViewOfFile), создание памяти общего доступа, с размером соответствующим размеру выделенной с помощью VirtualAlloc памяти (задать имя создаваемого объекта отображения "memshare");

5.2.1.6 Копирование данных из памяти, выделенной с помощью VirtualAlloc, в память общего доступа (например, с помощью вызова CopyMemory);

5.2.1.7 Ожидание процесса MemSort (используя заданный по варианту задания объект синхронизации (мютекс) необходимо ожидать пока процесс MemSort не подготовит данные);

5.2.1.8 Перевод режима доступа выделенной в п. 5.2.1.1 памяти в ReadWrite (с помощью вызова VirtualProtect);

5.2.1.9 Копирование данных из памяти общего доступа в память, выделенную с помощью VirtualAlloc (например, с помощью вызова CopyMemory);

5.2.1.10 Вывод на экран данных из памяти, выделенной с помощью VirtualAlloc;

5.2.1.11 Освобождение выделенной памяти (с помощью вызова VirtualFree);

5.2.1.12 Освободить память общего доступа (используя вызовы UnmapViewOfFile и СloseHandle);

**5.2.2** Написать программу MemSort выполняющую следующие действия:

5.2.2.1 Ожидание пока процесс MemShare не подготовит данные в памяти общего доступа (использовать заданный по варианту задания тип объекта синхронизации (мютекс));

5.2.2.2 Открытие памяти общего доступа "memshare" (используя вызовы OpenFileMapping и MapViewOfFile);

5.2.2.3 Сортировка данных в памяти общего доступа, методом указанным в варианте задания (метод Шелла).

**5.3 Ход работы**

5.3.1 В соответствии с вариантом были написаны программа на языке си и их тексты представлены ниже.

MemShare.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <windows.h>

#define MEMPAGE\_COUNT 1

#define MEMPAGE\_SIZE 4096

#define TOTAL\_MEMSIZE ((MEMPAGE\_COUNT) \* (MEMPAGE\_SIZE))

#define MAX 32767

typedef short arr\_elem\_t;

void fill\_array (arr\_elem\_t \* arrv, size\_t arrc)

{

for (size\_t i = 0; i < arrc; i++)

arrv[i] = rand() % MAX;

}

int main()

{

HANDLE mutex = CreateMutex(NULL, TRUE, "My\_mutex");

STARTUPINFO si; PROCESS\_INFORMATION pi;

memset(&si, 0, sizeof(STARTUPINFO)); memset(&pi, 0, sizeof(PROCESS\_INFORMATION));

si.cb = sizeof(STARTUPINFO); si.lpTitle = "MemSorter";

CreateProcess(NULL, "MemSort.exe", 0, NULL, FALSE, CREATE\_NEW\_CONSOLE, NULL, NULL, &si, &pi);

DWORD old\_sec\_attrs;

LPVOID base\_address = VirtualAlloc(NULL, TOTAL\_MEMSIZE, MEM\_COMMIT, PAGE\_READWRITE);

arr\_elem\_t \*arr = (arr\_elem\_t \*) base\_address;

fill\_array (arr, TOTAL\_MEMSIZE / sizeof(arr\_elem\_t));

VirtualProtect(base\_address, TOTAL\_MEMSIZE, PAGE\_READONLY, &old\_sec\_attrs);

HANDLE file\_map = CreateFileMapping(INVALID\_HANDLE\_VALUE, NULL, PAGE\_READWRITE, 0, TOTAL\_MEMSIZE, "memshare");

LPVOID view = MapViewOfFile(file\_map, FILE\_MAP\_ALL\_ACCESS, 0, 0, TOTAL\_MEMSIZE);

CopyMemory(view, base\_address, TOTAL\_MEMSIZE);

ReleaseMutex(mutex);

WaitForSingleObject(pi.hProcess, INFINITE);

VirtualProtect(base\_address, TOTAL\_MEMSIZE, PAGE\_READWRITE, &old\_sec\_attrs);

CopyMemory(base\_address, view, TOTAL\_MEMSIZE);

for (size\_t i = 0; i < TOTAL\_MEMSIZE / sizeof(arr\_elem\_t); i++)

printf("%d ", (int) arr[i]);

printf("\r\n");

VirtualFree(base\_address, TOTAL\_MEMSIZE, MEM\_RELEASE);

UnmapViewOfFile(view);

CloseHandle(file\_map);

system("pause");

return 0;

}

MemSort.c

#include <stdio.h>

#include <stdlib.h>

#include <windows.h>

#define MEMPAGE\_COUNT 1

#define MEMPAGE\_SIZE 4096

#define TOTAL\_MEMSIZE ((MEMPAGE\_COUNT) \* (MEMPAGE\_SIZE))

#define MAX 32767

typedef short arr\_elem\_t;

void sort(arr\_elem\_t \*arrv, size\_t arrc)

{

int i, j, step;

arr\_elem\_t tmp;

for (step = arrc/2; step > 0; step /= 2)

for (i = step; i < arrc; i++)

{

tmp = arrv[i];

for (j = i; j >= step; j -= step)

{

if (tmp < arrv[j-step])

arrv[j] = arrv[j-step];

else

break;

}

arrv[j] = tmp;

}

}

int main()

{

HANDLE mutex = OpenMutex(MUTEX\_ALL\_ACCESS, FALSE, "My\_mutex");

WaitForSingleObject(mutex, INFINITE);

HANDLE shared\_mem = OpenFileMapping(FILE\_MAP\_ALL\_ACCESS, FALSE, "memshare");

arr\_elem\_t \*arr = (arr\_elem\_t \*) MapViewOfFile(shared\_mem, FILE\_MAP\_ALL\_ACCESS, 0, 0, TOTAL\_MEMSIZE);

sort(arr, TOTAL\_MEMSIZE / sizeof(arr\_elem\_t));

ReleaseMutex(mutex);

UnmapViewOfFile((LPVOID) arr);

CloseHandle(shared\_mem);

return 0;

}

5.3.2 Тестовый пример продемонстрирован на рисунке 5.1.

Были написаны 2 программы на языке си. На рисунке 5.1 можно увидеть демонстрацию вывода основной программы – отсортированные по возрастанию данные.

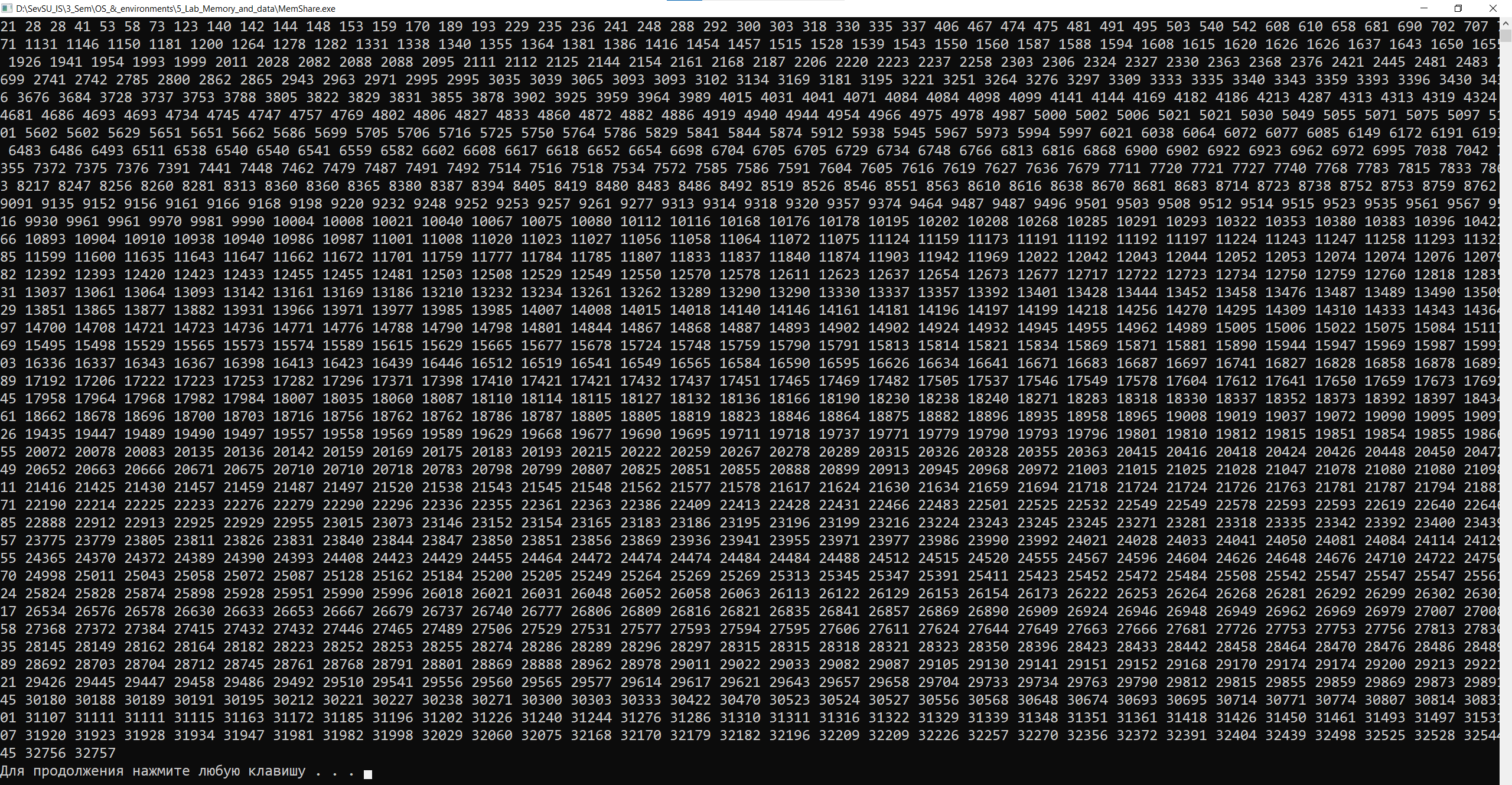


Рисунок 5.1 ­– Результат выполнения программы первый пример

Результаты тестирования полностью соответствуют ожиданиям.

**Выводы**

При выполнении данной лабораторной работы были получены навыки создания и работы с общей памятью, выделения памяти с помощью средств WinAPI. Также были повторно закреплены навыки работы с мьютексами. Полученные знания позволят создавать многопоточные приложения, которые на многоядерных системах будут работать эффективнее.