Федеральное государственное бюджетное образовательное учреждение высшего образования «Национальный исследовательский университет «МЭИ»

Институт информационных и вычислительных технологий

Кафедра Управления и интеллектуальных технологий

**Отчёт по лабораторной работе № 3**

# «Исследование областей виртуальной памяти»

**По курсу «Системное программное обеспечение»**

# Выполнил студент группы А-01-19:

# Китайский А.С.

# Проверил:

# Козлюк Д.А.

# Москва 2022

**Задание:**

1. Получить адреса-границы анализируемой области памяти. Границы всего виртуального адресного пространства (ВАП) процесса можно определить при помощи функции GetSystemInfo(). Начальный адрес, если он вводится пользователем, нужно округлить вниз до кратного размеру страницы виртуальной памяти (определяемого GetSystemInfo()). Аналогично следует округлять и одиночный адрес, вводимый пользователем.

2. Атрибуты страниц, а также начальные адреса и размеры областей (регионов) памяти можно определить при помощи функции VirtualQuery(). Перебор всех страниц (или регионов) можно выполнить, продолжая вызывать функцию VirtualQuery() для адресов, смещаемых каждый раз на размер страницы (или региона).

**Код программы:**

#include <iostream>

#include <windows.h>

#include <iomanip>

#define PAGE(addr, size) ((uintptr\_t)addr & ~((size\_t)size - 1))

using namespace std;

uintptr\_t address\_fun(uintptr\_t least, uintptr\_t largest)

{

uintptr\_t mem\_addr;

while (true)

{

cin >> hex >> mem\_addr; //Ввод крайнего значения адреса

bool flag = (mem\_addr >= least && mem\_addr <= largest); //Переменная для проверки вхождения адреса в диапазон

if (flag)

{

break;

}

else

{

cout << "The entered value is out of range! Enter again:";

}

}

return mem\_addr;

}

void mem\_inf\_fun(MEMORY\_BASIC\_INFORMATION info)

{

cout << dec << info.RegionSize << "\t"; //Размер в байтах

//Тип выделения памяти и был ли использован регион

switch (info.State)

{

case MEM\_COMMIT:

cout << "Yes" << "\t" << "C" << "\t";

break;

case MEM\_RESERVE:

cout << "Yes" << "\t" << "R" << "\t";

break;

case MEM\_FREE:

cout << "No" << "\t" << "F" << "\t";

break;

}

//Опция защиты памяти

switch (info.AllocationProtect)

{

case PAGE\_EXECUTE:

cout << "--X-";

break;

case PAGE\_EXECUTE\_READ:

cout << "R-X-";

break;

case PAGE\_EXECUTE\_READWRITE:

cout << "RWX-";

break;

case PAGE\_EXECUTE\_WRITECOPY:

cout << "R-XC";

break;

case PAGE\_NOACCESS:

cout << "----";

break;

case PAGE\_READONLY:

cout << "R---";

break;

case PAGE\_READWRITE:

cout << "RW--";

break;

case PAGE\_WRITECOPY:

cout << "R--C";

break;

default:

cout << " ";

break;

}

cout << setw(10);

//Тип страниц в регионе

switch (info.Type)

{

case MEM\_IMAGE:

cout << "image" << "\n";

break;

case MEM\_MAPPED:

cout << "mapped" << "\n";

break;

case MEM\_PRIVATE:

cout << "private" << "\n";

break;

default:

cout << "unused" << "\n";

break;

cout << "\n";

}

}

int main()

{

SYSTEM\_INFO sys\_inf; //Указатель на структуру, которая содержит информацию о системе

MEMORY\_BASIC\_INFORMATION info{}; //Указатель на структуру, которая содержит информацию о диапазоне страниц

GetSystemInfo(&sys\_inf); //Извлекает информацию о текущей системе

auto least\_address = reinterpret\_cast<uintptr\_t>(sys\_inf.lpMinimumApplicationAddress); //Наименьший адрес памяти

auto largest\_address = reinterpret\_cast<uintptr\_t>(sys\_inf.lpMaximumApplicationAddress); //Наибольший адрес памяти

//Вывод диапазона адресов в шестнадцатеричном виде

cout << "Addresses range:";

cout << "[" << hex << showbase << least\_address << ", " << largest\_address << "]\n";

//Ввод нижней границы диапазона

cout << "Enter lower limit:";

auto least = address\_fun(least\_address, largest\_address);

//Ввод верхней границы диапазона

cout << "Enter upper limit:";

auto largest = address\_fun(least, largest\_address);

//Выравнивание по границе страницы

least = PAGE(least, sys\_inf.dwPageSize);

largest = PAGE(largest, sys\_inf.dwPageSize);

//Отображение легенды

cout << "\nLegend:\n";

cout << "\tState:\tFree, Reserved, Committed;\n";

cout << "\tAccess:\tReadable, Writable, eXecutable, Copy-on-write, Guard,\n";

cout << "\t\tNo cache, write combine (Q), no access (-------);\n";

cout << "\t\tempty space indicates no access to page protection info.\n\n";

//Вывод таблицы регионов

cout << " Address \tSize \tUsed \tState \tAccess \tType\n";

for (uintptr\_t num = least; num < largest; num += info.RegionSize)

{

if (VirtualQuery((void\*)num, &info, sizeof info) != 0) //Извлекает информацию о диапазоне страниц

{

cout << setw(10) << hex << showbase << num << "\t"; //Вывод адреса в шестнадцатеричном виде

mem\_inf\_fun(info);

}

else

{

cout << "Error:" << GetLastError();

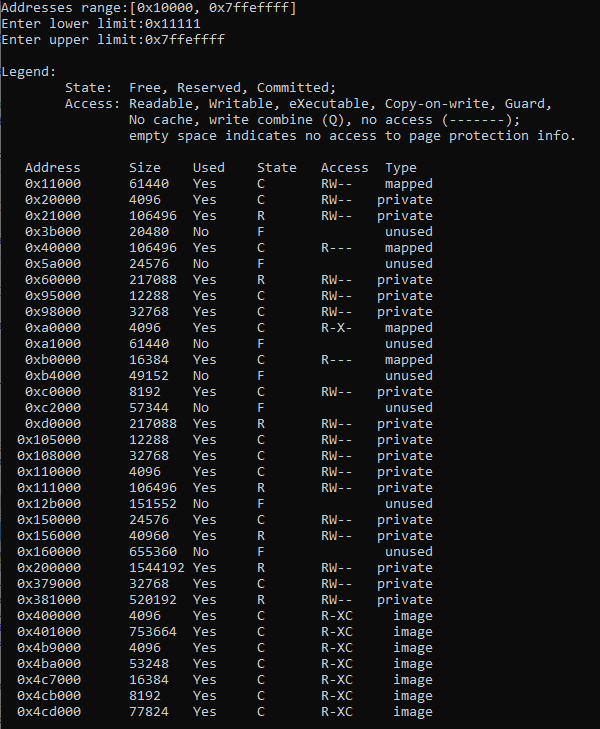
break;

}

}

}

**Результаты программы:**

****