**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра Вычислительной техники**

**ОТЧЕТ**

**по лабораторной работе №2**

**по дисциплине «Операционные системы»**

**Тема: «Управление памятью»**

Студентки гр. 3312 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Шахов К.С.

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Тимофеев А.В.

Санкт-Петербург

2025

**Цель работы.**

исследовать механизмы управления виртуальной памятью Win32.

**Задание 2.1.**

Исследовать виртуальное адресное пространство

процесса.

Указания к выполнению.

1. Создайте консольное приложение с меню (каждая выполняемая

функция и/или операция должна быть доступна по отдельному пункту

меню), которое выполняет:

− получение информации о вычислительной системе (функция Win32 API – GetSystemInfo);

− определение статуса виртуальной памяти (функция Win32 API – GlobalMemoryStatus);

− определение состояния конкретного участка памяти по заданному

с клавиатуры адресу (функция Win32 API – VirtualQuery); − раздельное резервирование региона и передачу ему физической

памяти в автоматическом режиме и в режиме ввода адреса начала

региона (функция Win32 API – VirtualAlloc, VirtualFree); − одновременное резервирование региона и передача ему физической

памяти в автоматическом режиме и в режиме ввода адреса начала региона (функция Win32 API – VirtualAlloc, VirtualFree);

− запись данных в ячейки памяти по заданным с клавиатуры адресам;

− установку защиты доступа для заданного (с клавиатуры) региона

памяти и ее проверку (функция Win32 API – VirtualProtect).

2. Запустите

приложение и проверьте его работоспособность на нескольких наборах

вводимых данных. Запротоколируйте результаты в отчет. Дайте свои

комментарии в отчете относительно выполнения функций Win32 API.

2. Подготовьте итоговый отчет с развернутыми выводами по заданию.

**Описание программы:**

Программа представляет собой консольное приложение на C++, которое взаимодействует с системой для получения информации о памяти, выделения памяти, изменения защиты памяти и отображения состояния памяти. Она использует API Windows, такие как SYSTEM\_INFO, GlobalMemoryStatus, VirtualAlloc, VirtualQuery, VirtualProtect и другие, чтобы выполнить операции с памятью и вывести соответствующую информацию.

**Основные функции программы:**

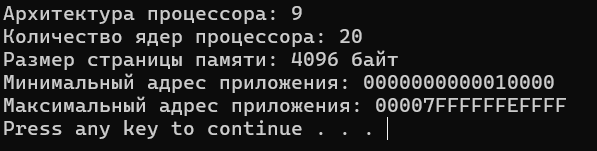
1. **show\_system\_info()**:
   * Использует структуру SYSTEM\_INFO для получения информации о системе.
   * Выводит:

Архитектура процессора.

Количество ядер процессора.

Размер страницы памяти.

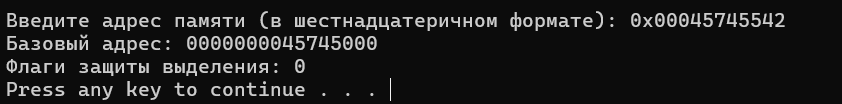
Минимальный и максимальный адреса, доступные для приложений.



1. **set\_memory\_protection()**:
   * Запрашивает у пользователя адрес памяти (в шестнадцатеричном формате).
   * Использует VirtualProtect для изменения защиты памяти на указанный адрес на PAGE\_READONLY (т.е. защита от записи).
   * Выводит сообщение об успешности или неудаче операции.



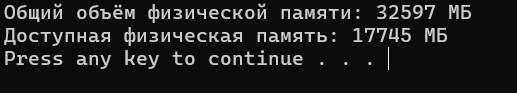
1. **get\_memory\_info()**:
   * Запрашивает у пользователя адрес памяти.
   * Использует VirtualQuery для получения информации о памяти по этому адресу.
   * Выводит базовый адрес и флаги защиты выделения памяти.



1. **display\_memory\_status()**:
   * Использует GlobalMemoryStatus для получения общего состояния физической памяти в системе.
   * Выводит:

Общий объём физической памяти.

Доступную физическую память.



1. **allocate\_and\_commit\_memory()**:
   * Выделяет и резервирует память с помощью VirtualAlloc на 1 МБ.
   * Если память успешно выделена, выводит адрес, где она была выделена, и затем освобождает её с помощью VirtualFree.

**Главная функция main()**:

В главной функции программы реализовано меню с возможностью выбора различных операций:

Показать информацию о системе.

Показать состояние памяти.

Запросить информацию о памяти.

Выделить и зарезервировать память.

Изменить защиту памяти.

Выход из программы.

Меню работает в цикле и позволяет пользователю выбирать действия, а после выполнения каждого действия программа делает паузу и очищает экран.

**Детали реализации:**

1. **Меню**: В цикле выводится меню с различными действиями. Пользователь вводит свой выбор, после чего выполняется соответствующее действие. После каждого действия экран очищается с помощью команды system("cls"), а затем программа делает паузу, чтобы пользователь мог ознакомиться с результатами.
2. **Использование системных функций Windows API**:
   * Для работы с памятью используются функции:

GetSystemInfo для получения информации о системе.

VirtualAlloc и VirtualFree для выделения и освобождения памяти.

VirtualProtect для изменения защиты памяти.

VirtualQuery для получения информации о состоянии памяти.

GlobalMemoryStatus для получения общего состояния физической памяти.

1. **Обработка ошибок**: Программа проверяет успешность выполнения системных функций и выводит сообщения об ошибках в случае неудачи, например, если не удалось выделить память или изменить защиту.
2. **Простота интерфейса**: Программа использует текстовый интерфейс и работает в командной строке. Для удобства взаимодействия с пользователем предоставляется меню с четкими и понятными вариантами.

**Заключение:**

Программа является полезным инструментом для работы с памятью в Windows, предоставляя пользователю информацию о системе, выделении памяти, изменении защиты и отображении состояния памяти. Основные операции выполняются корректно, но могут быть улучшены в плане пользовательского ввода и дополнительных проверок на ошибки.

**Текст программы**

*Main.cpp*

#include <iostream>

#include <windows.h>

void show\_system\_info()

{

SYSTEM\_INFO sys\_info;

GetSystemInfo(&sys\_info);

std::cout << "Архитектура процессора: " << sys\_info.wProcessorArchitecture << "\n";

std::cout << "Количество ядер процессора: " << sys\_info.dwNumberOfProcessors << "\n";

std::cout << "Размер страницы памяти: " << sys\_info.dwPageSize << " байт\n";

std::cout << "Минимальный адрес приложения: " << sys\_info.lpMinimumApplicationAddress << "\n";

std::cout << "Максимальный адрес приложения: " << sys\_info.lpMaximumApplicationAddress << "\n";

}

void set\_memory\_protection()

{

void\* addr;

DWORD old\_protection;

std::cout << "Введите адрес памяти для изменения защиты (в шестнадцатеричном формате): ";

std::cin >> addr;

if (VirtualProtect(addr, 4096, PAGE\_READONLY, &old\_protection))

std::cout << "Защита памяти успешно обновлена!\n";

else

std::cerr << "Ошибка: Не удалось обновить защиту памяти!\n";

}

void get\_memory\_info()

{

void\* addr;

MEMORY\_BASIC\_INFORMATION mem\_info;

std::cout << "Введите адрес памяти (в шестнадцатеричном формате): ";

std::cin >> addr;

if (VirtualQuery(addr, &mem\_info, sizeof(mem\_info)))

{

std::cout << "Базовый адрес: " << mem\_info.BaseAddress << "\n";

std::cout << "Флаги защиты выделения: " << mem\_info.AllocationProtect << "\n";

}

else

std::cerr << "Ошибка: Запрос памяти не удался!\n";

}

void display\_memory\_status()

{

MEMORYSTATUS memory\_status;

GlobalMemoryStatus(&memory\_status);

std::cout << "Общий объём физической памяти: " << memory\_status.dwTotalPhys / (1024 \* 1024) << " МБ\n";

std::cout << "Доступная физическая память: " << memory\_status.dwAvailPhys / (1024 \* 1024) << " МБ\n";

}

void allocate\_and\_commit\_memory()

{

SIZE\_T memory\_size = 1024 \* 1024;

LPVOID allocated\_memory = VirtualAlloc(NULL, memory\_size, MEM\_RESERVE | MEM\_COMMIT, PAGE\_READWRITE);

if (allocated\_memory)

{

std::cout << "Память выделена по адресу: " << allocated\_memory << "\n";

VirtualFree(allocated\_memory, 0, MEM\_RELEASE);

}

else

std::cerr << "Ошибка: Не удалось выделить память!\n";

}

int main()

{

bool menu\_flag = true;

int menu\_choice;

while (menu\_flag)

{

std::cout << "Меню\n";

std::cout << "1 Показать информацию о системе\n";

std::cout << "2 Показать состояние памяти\n";

std::cout << "3 Запросить информацию о памяти\n";

std::cout << "4 Выделить и зарезервировать память\n";

std::cout << "5 Изменить защиту памяти\n";

std::cout << "0 Выход\n";

std::cout << "Введите ваш выбор: ";

std::cin >> menu\_choice;

switch (menu\_choice)

{

case 1:

system("cls");

show\_system\_info();

system("pause");

system("cls");

break;

case 2:

system("cls");

display\_memory\_status();

system("pause");

system("cls");

break;

case 3:

system("cls");

get\_memory\_info();

system("pause");

system("cls");

break;

case 4:

system("cls");

allocate\_and\_commit\_memory();

system("pause");

system("cls");

break;

case 5:

system("cls");

set\_memory\_protection();

system("pause");

system("cls");

break;

case 0:

system("cls");

menu\_flag = false;

std::cout << "Выход из программы\n";

break;

default:

std::cout << "Неверный вариант! Пожалуйста, попробуйте снова.\n";

}

}

return 0;

}