Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Операционные среды и системное программирование

ОТЧЁТ

к лабораторной работе №3

на тему

УПРАВЛЕНИЕ ПАМЯТЬЮ И ВВОДОМ-ВЫВОДОМ, РАСШИРЕННЫЕ ВОЗМОЖНОСТИ ВВОДА-ВЫВОДА WINDOWS. ФУНКЦИИ API ПОДСИСТЕМЫ ПАМЯТИ WIN 32. ОРГАНИЗАЦИЯ И КОНТРОЛЬ АСИНХРОННЫХ ОПЕРАЦИЙ ВВОДА-ВЫВОДА. ОТОБРАЖЕНИЕ ФАЙЛОВ В ПАМЯТЬ.

Выполнил студент гр.153502 Богомолов М.А.

Проверил ассистент кафедры информатики Гриценко Н.Ю.

Минск 2023

**СОДЕРЖАНИЕ**

[1 Формулировка задачи 3](#_Toc148046360)

[2 Теоретические сведения 4](#_Toc148046361)

[3 Описание функций программы 5](#_Toc148046362)

[Список использованных источников 8](#_Toc148046363)

[Приложение А (обязательное) Листинг кода 9](#_Toc148046364)

# **1 ФОРМУЛИРОВКА ЗАДАЧИ**

Целью выполнения лабораторной работы является создание приложения для мониторинга и управления системной памятью, отображающее текущее потребление памяти различными процессами.

В качестве задачи необходимо реализовать возможность просмотра списка всех запущенных процессов и количество потребляемой ими памяти, а также просмотр информации об общем количестве занятой в системе памяти.

# **2 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Для получения списка всех запущенных процессов в операционной системе *Windows* используется функция *CreateToolhelp32Snapshot*. Эта функция делает снимок текущих процессов в системе и возвращает его в виде переменной типа *HANDLE*.

Для получения имени процесса используется структура *PROCESSENTRY32*. Структура *PROCESSENTRY32* описывает запись из списка процессов, находящихся в системном адресном пространстве при получении *snapshot* [1].

Функция *GetProcessMemoryInfo* (*psapi.h*) извлекает сведения об использовании памяти указанным процессом [2].

Непосредственно для отображения данных используется компонент *ListView*. *ListView* в *WinAPI* – это элемент управления, который используется для отображения и редактирования списков элементов в виде таблицы. Он предоставляет множество опций для отображения данных в виде колонок и строк. *ListView* может использоваться для создания файловых менеджеров, таблиц данных, списков файлов, и многих других приложений, где нужно отображать данные в табличной форме.

# **3 ОПИСАНИЕ ФУНКЦИЙ ПРОГРАММЫ**

При запуске приложения в элементе *ListView* отсутствует какое-либо содержимое (Рисунок 1).

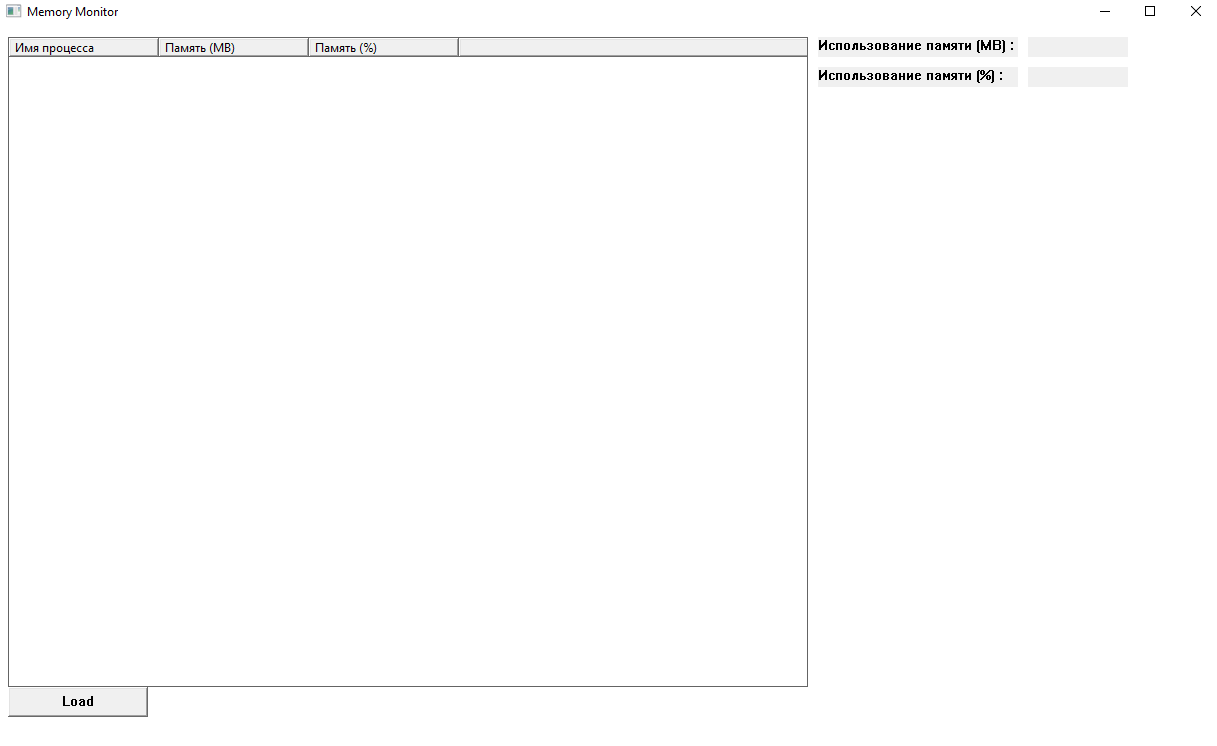


Рисунок 1 – Главное окно приложения при запуске

Чтобы информация о текущих процессах в системе появилась в элементе *ListView*, необходимо нажать на кнопку *Load*. Результат выполнения операции *Load* (Рисунок 2).

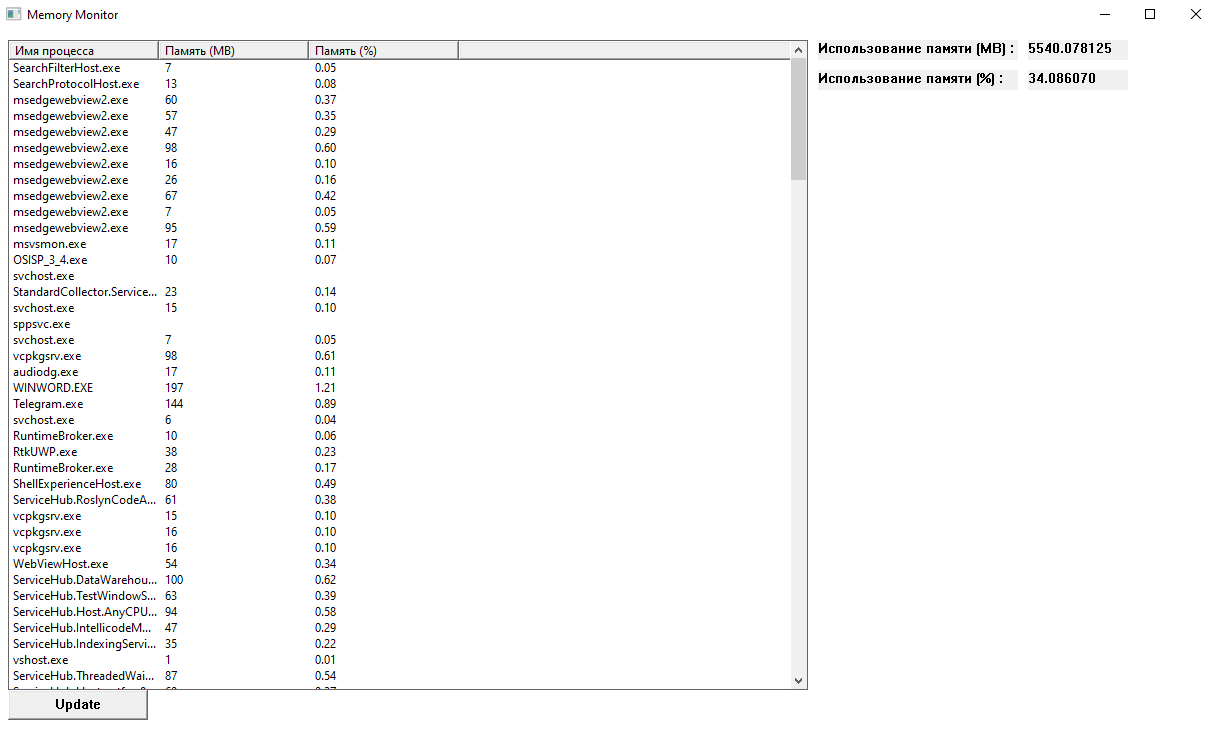


Рисунок 2 – Результат выполнения операции *Load*

После нажатия на кнопку *Load* содержимое *ListView* заполняется данными о текущих процессах в системе и данными о потреблении памяти каждым из процессов в мегабайтах и в процентном соотношение от количества всей физической памяти устройства. Также справа от компонента *ListView* можно наблюдать общую загруженность памяти в мегабайтах и в процентном соотношении от количества всей физической памяти устройства. Следует отметить, что после нажатия кнопка *Load* меняет своё название на *Update,* при этом оставляя за собой такой же функционал. К примеру, откроем приложение *PyCharm* и приложение *Steam* и увидим, как изменится потребление ресурсов и перечень текущих процессов. Результат открытия вышеупомянутых приложений (Рисунок 3).

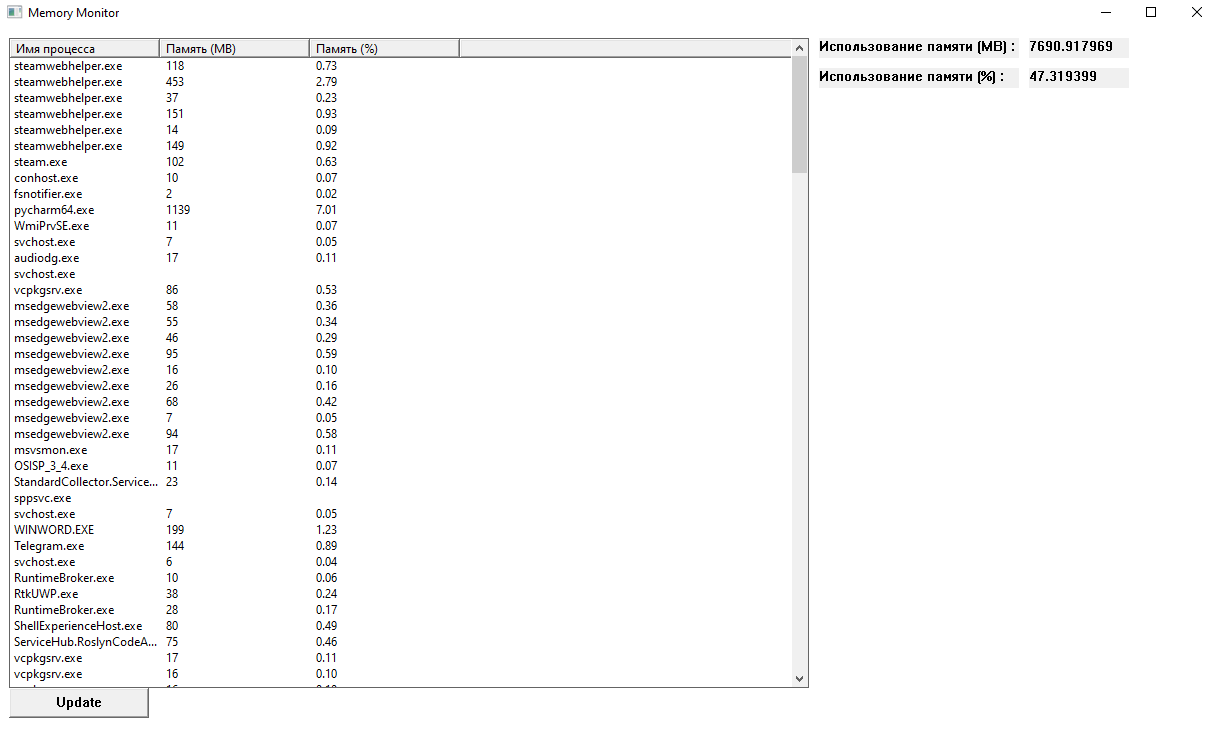


Рисунок 3 – Результат открытия приложения *PyCharm* и приложения *Steam*

Для доказательства корректности работы программы сравним значения загруженности памяти со значениями из Диспетчера Задач *Windows* (Рисунок 4).

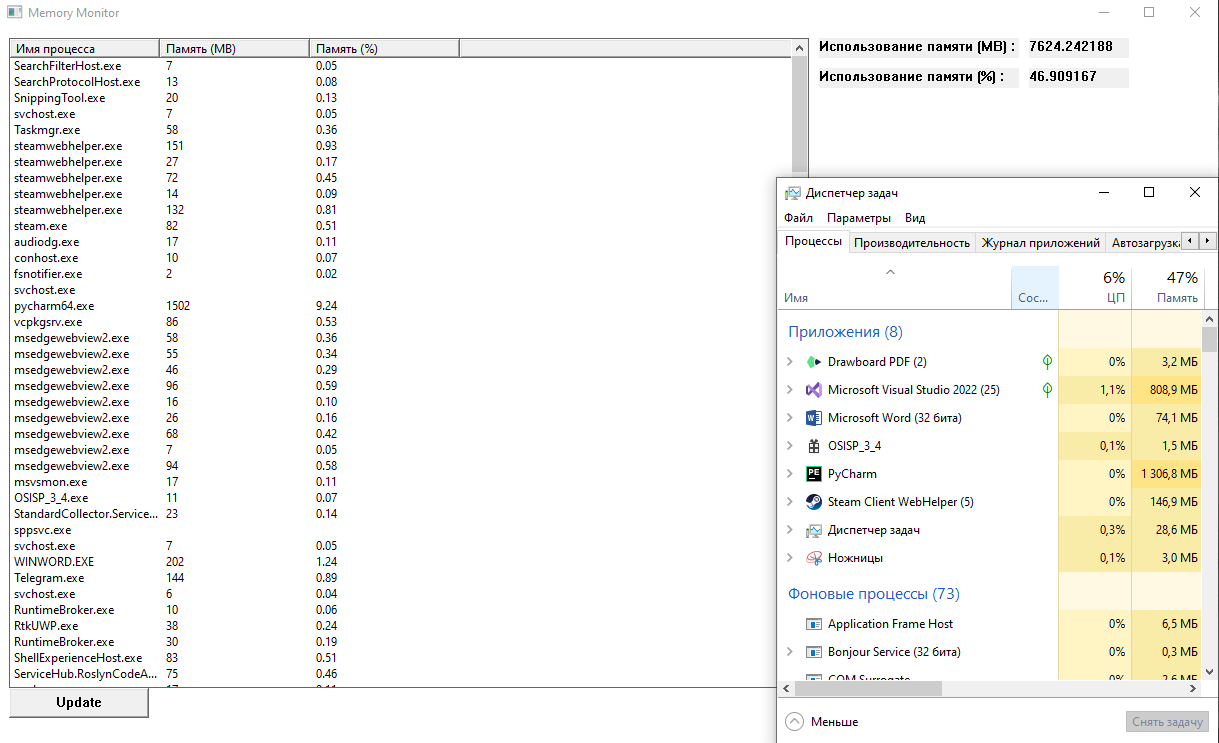


Рисунок 4 – Сравнение показателей загруженности памяти

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Структура PROCESSENTRY32 (tlhelp32.h) [Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/ru-ru/windows/win32/api/tlhelp32/ns-tlhelp32-processentry32> – Дата доступа: 01.11.2023.

[2] Функция GetProcessMemoryInfo (psapi.h) [Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/ru-ru/windows/win32/api/psapi/nf-psapi-getprocessmemoryinfo> – Дата доступа: 01.11.2023.

# **ПРИЛОЖЕНИЕ А (обязательное) Листинг кода**

Листинг 1 – Файл OSISP\_3.cpp

#include <windows.h>

#include <TlHelp32.h>

#include <CommCtrl.h>

#include <Psapi.h>

#include <stdio.h>

#include <thread>

#include <string>

HINSTANCE hInst;

HWND hWnd;

HWND hListView;

HWND hUpdateButton;

HWND hMemoryUsageMBLabel;

HWND hMemoryUsagePercentageLabel;

HWND hMemoryUsageMB;

HWND hMemoryUsagePercentage;

#define IDC\_UPDATE\_BUTTON 1001

void GetProcessList(HWND hListView);

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

int APIENTRY WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)

{

hInst = hInstance;

WNDCLASSEX wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_HREDRAW | CS\_VREDRAW;

wcex.lpfnWndProc = WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(hInstance, IDI\_APPLICATION);

wcex.hCursor = LoadCursor(NULL, IDC\_ARROW);

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW + 1);

wcex.lpszMenuName = NULL;

wcex.lpszClassName = L"MyMemoryMonitorApp";

wcex.hIconSm = LoadIcon(wcex.hInstance, IDI\_APPLICATION);

RegisterClassEx(&wcex);

INITCOMMONCONTROLSEX icex;

icex.dwICC = ICC\_LISTVIEW\_CLASSES;

InitCommonControlsEx(&icex);

hWnd = CreateWindow(L"MyMemoryMonitorApp", L"Memory Monitor", WS\_OVERLAPPEDWINDOW, CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, NULL, NULL, hInstance, NULL);

hListView = CreateWindowEx(0, WC\_LISTVIEW, L"", WS\_VISIBLE | WS\_CHILD | WS\_BORDER | LVS\_REPORT | LVS\_SINGLESEL,

10, 10, 800, 650, hWnd, NULL, hInst, NULL);

LV\_COLUMN lvc;

lvc.mask = LVCF\_FMT | LVCF\_WIDTH | LVCF\_TEXT;

lvc.fmt = LVCFMT\_LEFT;

lvc.cx = 150;

lvc.pszText = const\_cast<LPWSTR>(L"Имя процесса");;

ListView\_InsertColumn(hListView, 0, &lvc);

lvc.cx = 150;

lvc.pszText = const\_cast<LPWSTR>(L"Память (MB)");

ListView\_InsertColumn(hListView, 1, &lvc);

lvc.cx = 150;

lvc.pszText = const\_cast<LPWSTR>(L"Память (%)");

ListView\_InsertColumn(hListView, 2, &lvc);

hMemoryUsageMBLabel = CreateWindow(L"STATIC", L"Использование памяти (MB) :",

WS\_VISIBLE | WS\_CHILD, 820, 10, 200, 20, hWnd, NULL, hInst, NULL);

hMemoryUsagePercentageLabel = CreateWindow(L"STATIC", L"Использование памяти (%) :",

WS\_VISIBLE | WS\_CHILD, 820, 40, 200, 20, hWnd, NULL, hInst, NULL);

hMemoryUsageMB = CreateWindow(L"STATIC", L"",

WS\_VISIBLE | WS\_CHILD, 1030, 10, 100, 20, hWnd, NULL, hInst, NULL);

hMemoryUsagePercentage = CreateWindow(L"STATIC", L"",

WS\_VISIBLE | WS\_CHILD, 1030, 40, 100, 20, hWnd, NULL, hInst, NULL);

hUpdateButton = CreateWindow(L"BUTTON", L"Load", WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

10, 660, 140, 30, hWnd, (HMENU)IDC\_UPDATE\_BUTTON, hInst, NULL);

if (!hWnd)

{

return FALSE;

}

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

MSG msg;

while (GetMessage(&msg, NULL, 0, 0))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return (int)msg.wParam;

}

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

switch (message)

{

case WM\_COMMAND:

{

int wmId = LOWORD(wParam);

int wmEvent = HIWORD(wParam);

if (wmId == IDC\_UPDATE\_BUTTON && wmEvent == BN\_CLICKED)

{

SetWindowText(hUpdateButton, L"Update");

std::thread(GetProcessList, hListView).detach();

}

}

break;

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

EndPaint(hWnd, &ps);

return 0;

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

return 0;

}

void GetProcessList(HWND hListView)

{

double AllMemoryUsageMB = 0;

double AllMemoryUsagePercentage = 0;

ListView\_DeleteAllItems(hListView);

HANDLE hProcessSnap;

PROCESSENTRY32 pe32;

hProcessSnap = CreateToolhelp32Snapshot(TH32CS\_SNAPPROCESS, 0);

if (hProcessSnap == INVALID\_HANDLE\_VALUE)

{

return;

}

pe32.dwSize = sizeof(PROCESSENTRY32);

if (Process32First(hProcessSnap, &pe32))

{

do

{

LVITEM lvi;

lvi.mask = LVIF\_TEXT;

lvi.iItem = 0;

lvi.iSubItem = 0;

lvi.pszText = pe32.szExeFile;

ListView\_InsertItem(hListView, &lvi);

PROCESS\_MEMORY\_COUNTERS pmc;

HANDLE hProcess = OpenProcess(PROCESS\_QUERY\_INFORMATION | PROCESS\_VM\_READ, FALSE, pe32.th32ProcessID);

if (hProcess)

{

if (GetProcessMemoryInfo(hProcess, &pmc, sizeof(pmc)))

{

lvi.iSubItem = 1;

WCHAR szMemoryUsage[64];

swprintf\_s(szMemoryUsage, L"%lu", pmc.WorkingSetSize / 1024 / 1024);

AllMemoryUsageMB += (double)pmc.WorkingSetSize / 1024 / 1024;

lvi.pszText = szMemoryUsage;

ListView\_SetItem(hListView, &lvi);

lvi.iSubItem = 2;

MEMORYSTATUSEX memInfo;

memInfo.dwLength = sizeof(MEMORYSTATUSEX);

GlobalMemoryStatusEx(&memInfo);

double memoryUsagePercentage = (double)(pmc.WorkingSetSize) / memInfo.ullTotalPhys \* 100;

WCHAR szMemoryUsagePercentage[64];

swprintf\_s(szMemoryUsagePercentage, L"%.2lf", memoryUsagePercentage);

AllMemoryUsagePercentage += (double)(pmc.WorkingSetSize) / memInfo.ullTotalPhys \* 100;

lvi.pszText = szMemoryUsagePercentage;

ListView\_SetItem(hListView, &lvi);

}

CloseHandle(hProcess);

}

} while (Process32Next(hProcessSnap, &pe32));

std::wstring AllMemoryUsageMBAsString = std::to\_wstring(AllMemoryUsageMB);

SetWindowText(hMemoryUsageMB, AllMemoryUsageMBAsString.c\_str());

std::wstring AllMemoryUsagePercentageAsString = std::to\_wstring(AllMemoryUsagePercentage);

SetWindowText(hMemoryUsagePercentage, AllMemoryUsagePercentageAsString.c\_str());

}

CloseHandle(hProcessSnap);

}