**Лабораторная работа №2**

**Задание для выполнения**

Разработать программу которая:

1. выдает информацию, получаемую при использовании API GlobalMemoryStatus (при выводе информации использовать диаграммы);

2. составляет карту виртуальной памяти для любого процесса.

Программа должна выводить полную информацию о занятой и свободной физической и виртуальной памяти, составлять карту процессов, создает виртуальную память для каждого процесса.

**Решение первого задания:**

import ctypes

import matplotlib.pyplot as plt

class MEMORYSTATUS(ctypes.Structure):

    \_fields\_ = [

        ("dwLength", ctypes.c\_ulong),

        ("dwMemoryLoad", ctypes.c\_ulong),

        ("dwTotalPhys", ctypes.c\_ulong),

        ("dwAvailPhys", ctypes.c\_ulong),

        ("dwTotalPageFile", ctypes.c\_ulong),

        ("dwAvailPageFile", ctypes.c\_ulong),

        ("dwTotalVirtual", ctypes.c\_ulong),

        ("dwAvailVirtual", ctypes.c\_ulong)

    ]

def get\_memory\_status():

    memory\_status = MEMORYSTATUS()

    memory\_status.dwLength = ctypes.sizeof(MEMORYSTATUS)

    ctypes.windll.kernel32.GlobalMemoryStatus(ctypes.byref(memory\_status))

    return memory\_status

def draw\_memory\_chart(mem):

    labels = ['Total Phys', 'Avail Phys', 'Total Virtual', 'Avail Virtual']

    values = [

        mem.dwTotalPhys / (1024\*\*2),

        mem.dwAvailPhys / (1024\*\*2),

        mem.dwTotalVirtual / (1024\*\*2),

        mem.dwAvailVirtual / (1024\*\*2)

    ]

    plt.figure(figsize=(8, 5))

    plt.bar(labels, values, color=['blue', 'green', 'orange', 'red'])

    plt.ylabel('MB')

    plt.title('GlobalMemoryStatus - Память в системе')

    for i, v in enumerate(values):

        plt.text(i, v + 50, f'{v:.0f} MB', ha='center')

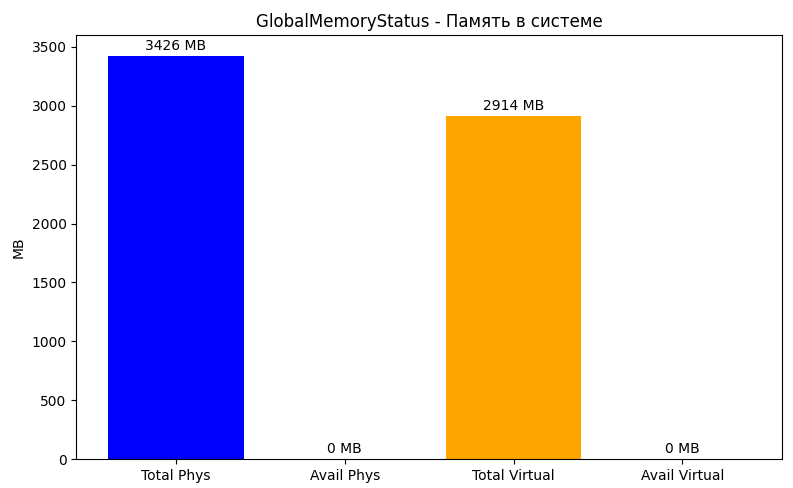
    plt.tight\_layout()

    plt.show()

mem = get\_memory\_status()

draw\_memory\_chart(mem)

**Результат первого задания:**



**Решение второго задания:**

import ctypes

import ctypes.wintypes as wintypes

import psutil

# Константы WinAPI

PROCESS\_QUERY\_INFORMATION = 0x0400

PROCESS\_VM\_READ = 0x0010

MEM\_COMMIT = 0x1000

# Структура MEMORY\_BASIC\_INFORMATION

class MEMORY\_BASIC\_INFORMATION(ctypes.Structure):

    \_fields\_ = [

        ("BaseAddress", ctypes.c\_void\_p),

        ("AllocationBase", ctypes.c\_void\_p),

        ("AllocationProtect", wintypes.DWORD),

        ("RegionSize", ctypes.c\_size\_t),

        ("State", wintypes.DWORD),

        ("Protect", wintypes.DWORD),

        ("Type", wintypes.DWORD)

    ]

# Получение карты виртуальной памяти одного процесса

def get\_virtual\_memory\_map(pid):

    entries = []

    try:

        h\_process = ctypes.windll.kernel32.OpenProcess(

            PROCESS\_QUERY\_INFORMATION | PROCESS\_VM\_READ,

            False,

            pid

        )

        if not h\_process:

            return ["  [!] Не удалось открыть процесс."]

        mbi = MEMORY\_BASIC\_INFORMATION()

        addr = 0

        while ctypes.windll.kernel32.VirtualQueryEx(

            h\_process, ctypes.c\_void\_p(addr),

            ctypes.byref(mbi), ctypes.sizeof(mbi)

        ):

            if mbi.State == MEM\_COMMIT:

                line = (

                    f"  Адрес: {hex(mbi.BaseAddress)} | "

                    f"Размер: {mbi.RegionSize / 1024:.0f} KB | "

                    f"Защита: {mbi.Protect}"

                )

                entries.append(line)

            addr += mbi.RegionSize

        ctypes.windll.kernel32.CloseHandle(h\_process)

        return entries

    except Exception as e:

        return [f"  Ошибка: {e}"]

# Главная функция: записывает всё в файл

def generate\_memory\_map\_report(filename="memory\_map.txt"):

    with open(filename, "w", encoding="utf-8") as f:

        for proc in psutil.process\_iter(['pid', 'name']):

            pid = proc.info['pid']

            name = proc.info.get('name', 'Неизвестно')

            f.write(f"\n=== PID: {pid} | Процесс: {name} ===\n")

            entries = get\_virtual\_memory\_map(pid)

            for entry in entries:

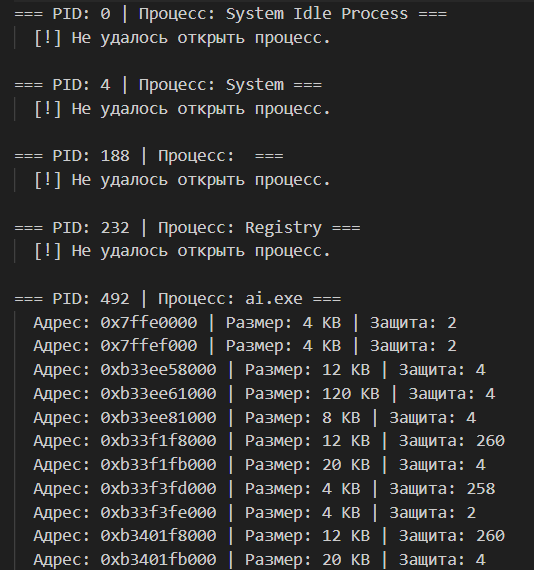
                f.write(entry + "\n")

    print(f"Карта виртуальной памяти сохранена в файл: {filename}")

# Запуск

generate\_memory\_map\_report()

**Результат второго задания:**

****