*ISSN 2079-0023 (print)*
*ISSN 2410-2857 (online)*

УДК 004.451.2

***Є. О. ЛОБОДА, Д. О. ДУБОВИЙ***

# КОНТРОЛЬ ВИКОНУВАНИХ ЕКЗЕМПЛЯРІВ ДОДАТКІВ В WINDOWS 7/8/10

Розроблено програмний модуль визначання переліку процесів, що виконуються в комп'ютері і надання інформації про їх поточний стан, ідентифікатори й пріоритети виконання активних процесів, перелік задіяних dll бібліотек. Метою науково-дослідної роботи було розробити програмний модуль сумісний з різними останніми версіями операційної системи Windows, який вперше забезпечує за допомогою діалогового вікна отримання розширеної інформації про всі діючи додатки та надає можливість позбавлення від тих з них, що вийшли з під контролю. Проведено тестування зробленої розробки додатку.

**Ключові слова:** модуль, процес, операційна система, контроль, ідентифікатор, пріоритет, тестування.

Разработан программный модуль определения перечня процессов, выполняемых в компьютере и отображения информации об их текущем состоянии, идентификаторы, приоритеты выполнения, перечень используемых dll библиотек. Целью научно-исследовательской работы было разработать программный модуль совместимый с различными последними версиями операционной системы Windows, который впервые обеспечивает с помощью диалогового окна получение расширенной информации о всех исполняемых приложениях и даёт возможность избавления от тех из них, которые вышли из под контроля. Выполнено тестирование работоспособности созданного приложения.

**Ключевые слова:** модуль, процесс, операционная система, контроль, идентификатор, приоритет тестирование.

A software module to determine the list of processes running on your computer and display information about their current status, IDs, perform the priorities list of used dll libraries. The aim of research was to develop a software module is compatible with a variety of the latest versions of the Windows operating system, which for the first time provides a dialog box to obtain extensive information on all executable applications and makes it possible to get rid of those that are out of control. Described sequentially all the steps of creating a conceptual project: carried out a detailed analysis of known attempts to accomplish the task; Described sequentially all the steps of creating a conceptual project: carried out a detailed analysis of the known attempts to perform the assigned tasks; We developed a method to obtain the optimal solution of the problem and justified project architecture with a given software functionality; logical design is made with the creation of the source code of software components and development of resources: physical design produced by running the executable. Functioning of the development has been tested on several versions of Windows. This testing fully confirmed the development operation, even when the process of getting rid of that came out under mortgaged for their control.

**Keywords:** module, a process, operating system, control, ID, priority, testing.

**Вступ.** В сучасних операційних системах дуже важливо вміти визначати які процеси виконуються в системі, спостерігати за ними, знати їх поточний стан та як позбутися процесів, наприклад, якщо вони вийшли з під контролю [1–11].

**Постановка проблеми** базується на тому, що сучасні версії Windows не дозволяють прямий доступ проникнення у системні «таємниці» інших процесів. Це можливе з великими обмеженнями лише завдяки використанню функцій API (application programming interface) операційної системи Windows. Спочатку, у перших версій Windows зовсім не було функцій, які б дозволяли виконувати ці спостереження «таємниць». Замість цього застосовувалася база даних Performance Data, що була недосконалою і постійно оновлювалася в різних версіях Windows, змінювалась архітектура та інформаційні можливості цього інструментарію. Проте, навидь цієї бази немає зараз в останніх версіях Windows. Другий можливий напрямок вирішення цієї проблеми – використовувати можливості реєстру. Використання можливостей реєстру має свої недоліки. Його лічильники продуктивності є локалізованими, тобто потрібно використовувати назви констант тільки на тій мові, на якій зараз діє надана операційна система з формуванням саме на цій мові повного шляху до лічильника продуктивності, що у сучасних умовах не можливо. Навидь загально відома утиліта Task Manager, що стандартно включена до усіх операційних систем Windows, не надає інформацію про модулі, що зараз використовуються – є активними процесами [1].

Для усунення вказаних недоліків, в описаному нижче програмному модулі був взятий напрямок на використання API бібліотеки Tool Help Library через її кращу стабільність та підтримку більшістю новітніх версій операційної системи Windows 7, 8, 10. Бібліотека Tool Help Library була створена компанією Microsoft – розробником Windows, отже програми, що використовують в ній, зненацька не втратять свою працездатність, як могло б бути з програмним забезпеченням від сторонніх виробників [2, 3, 4].

Тому, для вирішення вказаної проблеми було прийнято рішення розробити програмний модуль з забезпеченням наступних функціональних можливостей:

- мати діалоговий інтерфейс взаємодії з користувачем;
- виводити детальну інформацію про процеси, що виконуються в даний час у операційній системі;
- виводити список усіх модулів, що використовуються операційною системою;
- мати сумісність з різними останніми версіями Windows (5, 7, 8, 10).

Розроблений нижче програмний комплекс з одержанням користувачами всієї цієї важливої інформації, вперше надає користувачам можливості, що раніше їм не надавалися

Вказані вище вимоги потребують створення програмний модуль, який перераховує процеси, що виконуються у системі, а також надає інформацію про них: ідентифікатор, пріоритет, кількість потоків, список модулів [6].

**Аналіз останніх досліджень і публікацій** існуючих засобів отримання інформації про активні процеси показав, що програмне забезпечення, яке дозволяє отримувати таку інформацію на локальному комп'ютері та дізнаватися детальної інформації про діючи активні процеси, не є поширеним [5].

Найближчими для рішення наданої задаючи були визначені зазначені нижче методи та засоби [6–9]:

1. Бібліотека Process Status Helper (PSAPI) що надає набір функцій, які дозволяють отримати інформацію про процеси і драйвери пристроїв. Бібліотека поставляється у складі версій Windows, що з'явилася після 2000 року і доступна програмістам у якості додаткового компоненту. Для перерахування процесів бібліотека надає функцію EnumProcesses(), яка повертає лише масив ідентифікаторів запущених процесів [10].

2. Функція NtQuerySystemInformation() з не документованої бібліотеки, що експортується з відповідної системної DLL. Ця функція та деякі інші можливості цієї DLL дозволяю отримати основну інформацію про продуктивність системи, інформацією про інтенсивність використання файлу підкачки, завантаженість процесора і кількість процесів, що працюють у системі. Крім цього функція GlobalMemoryStatus() з цієї DLL бібліотеки визначає стан пам'яті в поточний момент часу, а GetIfTable() надає доступ до статистики використання мережевих інтерфейсів.

3. «Лічильники продуктивності» (performance counters) це механізм збору різної інформації, закладений в операційні системи лінійки Windows. Більша частина лічильників доступна користувачеві через оснащення (snap-in) Performance. Операційна система Windows, починаючи з версії NT, надає інтерфейс для отримання різноманітної інформації про систему у вигляді лічильників продуктивності. Цей інтерфейс дозволяє отримати набір глибоко вкладених одна в одну структур. Для отримання будь-якої інформації потрібно лише прочитати з ключа реєстру HKEY_PERFORMANCE_DATA значення із спеціально сформованим ім'ям. У результаті повертається набір вкладених структур, багато з яких мають змінний розмір. Бібліотека Performance Data Helper (PDH) надає цей інтерфейс для вимірювання даних про продуктивність. Однак ця бібліотека не входила в комплект постачання багатьох версій Windows. Останній час вона поширюється у складі Microsoft Platform SDK. Починаючи з Windows 2000 присутня за замовчуванням бібліотека PDH.dll, а система вимірювання продуктивності реалізована за допомогою поняття об'єкта, для якого здійснюється підрахунок продуктивності. Прикладами об'єктів дослідження є: процесор, жорсткий диск та ін. Кожен об'єкт може мати один або більше примірників, і для кожного об'єкта існує свій набір лічильників продуктивності. Завдання полягає в отриманні значень лічильників [11].

Основна складність у використанні лічильників продуктивності полягає в тому, що назви об'єктів і лічильників продуктивності є локалізованими. Це означає, що, наприклад, на російській версії Windows необхідно використовувати зарезервовані константи «Процес» і «Ідентифікатор процесу» замість розповсюджених «Process» і «ID Process». Для отримання локалізованих імен об'єктів і формування повного шляху до лічильника продуктивності, який нас цікавить, необхідно писати допоміжну функцію, яка б здійснювала зворотне перетворення [2]. Лічильники продуктивності – це потужний і гнучкий механізм. Але, він неочевидний і незручний.

4 Використання ToolHelp32 API:

Набір функцій під назвою ToolHelp API є в останніх версіях ОС Windows 7/8/10. Ця бібліотека дозволяє стороннім розробникам отримати доступ до системної інформації. Спочатку створюється моментальний знімок (snapshot) списку процесів за допомогою функції CreateToolhelp32Snapshot(), а потім здійснюється зчитування інформації зі списку за допомогою функцій Process32First() і Process32Next(). Структура PROCESSENTRY32, що заповнюється цими функціями, містить всю інформацію про поточний стан системи незалежно від версії та локалізації Windows [2, 10].

Виконаний огляд існуючих засобів отримання інформації про активні процеси показав, що програмне забезпечення, яке дозволяє отримувати потрібну інформацію на локальному комп'ютері та дізнаватися поглиблених даних про діючи активні процеси, не є поширеним [3, 4, 5].

Тому остаточне **формулюванням цілі статті** для вирішення поставленого завдання зводиться до: потрібно створити модуль, який перераховує процеси, що виконуються у системі, а також надає інформацію про них, таку як: ідентифікатор, пріоритет, кількість потоків, список модулів та dll-бібліотек, що вони використовують.

**Виклад основного матеріалу**

Остаточний вибір методу рішення поставленого цілі було виконано завдяки даних наведеного аналізу існуючих засобів отримання інформації про активні процеси. В нашому проекті використовується бібліотека Tool Help Library через кращу стабільність та підтримку її методу більшістю останніх версій операційної системи Windows (7/8/10). Вона створена компанією Microsoft, отже програми, що будуть її використовувати, зненацька не втратять свою працездатність, як могло б бути з програмним забезпеченням від сторонніх виробників, при використанні ними не документованих можливостей операційної системи. Дана бібліотека, що розроблювалась з метою спрощення процесу написання спеціалізованих утиліт, системних відгадчиків, дозволяє також отримувати інформацію про процеси, що виконуються в даний момент на комп'ютері.

**Обґрунтування архітектури** створюваного проекту базувалося для розробленого додатку для забезпечення керування користувачем з допомогою мишки або клавіатури. Одержана інформація відображається в елементах діалогового вікна. Для реалізації такої програми не потрібні спеціалізовані сервери, багатопроцесорні системи та мережі. Тому,

*ISSN 2079-0023 (print)*
*ISSN 2410-2857 (online)*

поставлене завдання цілком може виконати програма на базі звичайного персонального комп'ютера

Забезпечення експлуатаційних характеристик програмного продукту проекту базувалося на необхідності забезпечення максимальної автоматизації процесу функціонування розробки. Автоматизація полягає в тому, що безпосередньо процес аналізу повинен виконуватися без участі людини та/або додаткових спеціалізованих пристроїв.

Використання програмного продукту, який створюється, зменшує трудомісткість процесу аналізу. програмного засобу створеного проекту. Практично кожний ПК має необхідні засоби емуляції відсутніх в комп'ютері пристроїв. Необхідні як мінімум: клавіатура, мишка й конфігурація ПК повинна забезпечувати роботу системи Windows останніх версій 7/8/10 [4, 6–10].

Для розширення області застосування програми передбачається можливість оперативної зміни користувачем параметрів аналізу. Простий, інтуїтивно зрозумілий інтерфейс та можливість оперативного конфігурування програми забезпечують «гнучкість» програми та можливість роботи із програмою користувачів з низьким рівнем знань в області комп'ютерів.

Вигляд діалогового вікна розробки наведено нижче на рис. 1.

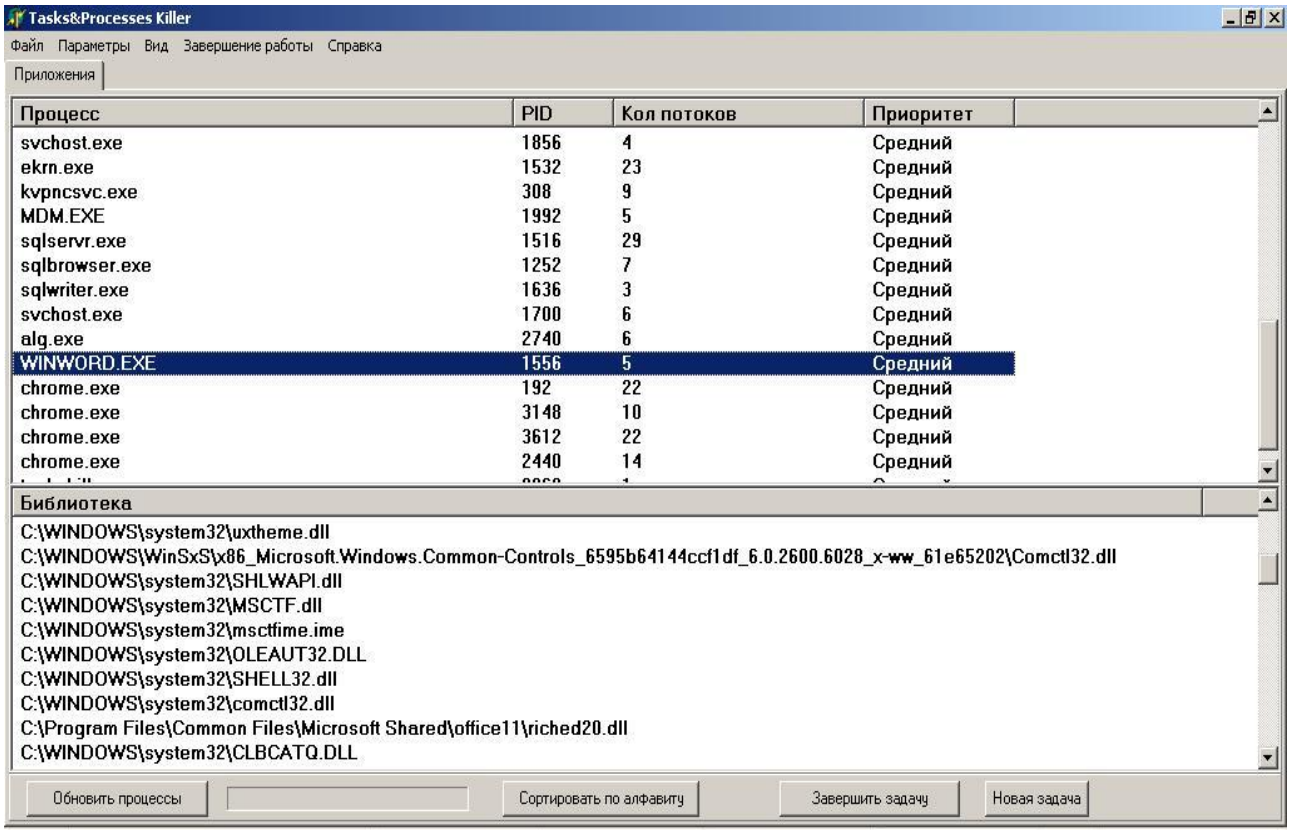


Рис. 1 – Діалогове вікно розробки з отриманням інформації про процес WINWORD.EXE

**Логічне проектування** дизайну інтерфейсу користувача розроблено таким, що для спрощення керування створено діалогове вікно з елементами зрозумілого керування до кожної з наданих в завданні інформації. Доступ користувача до виконання кожної функції можливий як з клавіатури, так і за допомогою мишки.

Особливості виконання роботи засобами Tool Help Library (THL) на її основі забезпечуються поняттям «snapshot» миттєвий знімок стану запущених додатків. Перед викликом всіх функцій бібліотеки необхідно створити snapshot. Це здійснюється викликом функції:

HANDLE WINAPI CreateToolhelp32Snapshot (DWORD dwFlags, DWORD th32ProcessID );

Параметр dwFlags вказує, яка саме інформація цікавить користувача. Параметр th32ProcessID ідентифікує процес, стан якого досліджується. Він потрібен для використання dwFlags зі значеннями TH32CS_SNAPHEAPLIST, TH32CS_SNAPMODULE і TH32CS_SNAPTHREAD. В інших випадках цей параметр ігнорується.

Знищується об'єкт snapshot стандартним викликом CloseHandle (hSnapshot);

Всі помилки, що виникають при виконанні функцій THL, повертаються стандартним викликом GetLastError і FormatMessage.

Щоб отримати список процесів, які працюють в даний момент, необхідно написати код, що використовує дві наступні функції:

BOOL WINAPI Process32First (HANDLE hSnapshot(LPPROCESSENTRY32 lppe);

BOOL WINAPI Process32Next (HANDLE hSnapshot(LPPROCESSENTRY32 lppe);

В обох цих функціях э два параметри. Перший – дескриптор, що повертається попереднім викликом CreateToolhelp32Snapshot, а другий вказує на структуру PROCESSENTRY32, в якій повертається резуль-

тат виконання функції. Перед викликом Process32First необхідно встановити в поле dwSize структури PROCES-SENTRY32 значення, що дорівнює розміру самої структури – sizeof (PROCESSENTRY32), інакше функція поверне помилку.

Для одержання списку модулів, завантажених процесом протягом роботи практично будь-якого додатка Win32 завантажуються ті чи інші бібліотеки, компоненти та інші програмні модулі. Природно, інколи виникає необхідність отримати список завантажених модулів. Для цього в THL в першу чергу, викликається CreateToolhelp32Snapshot з параметрами TH32CS_SNAPMODULE і ідентифікатором процесу, модулі якого нас цікавлять. Далі, як і в попередньому розділі, у полі dwSize, але тепер вже до структури MODULEENTRY32, встановлюється значення, що дорівнює розміру цієї структури. До речі, це необхідно робити для всіх функцій перерахування THL. Цикл перебору використовує структуру MODULEENTRY32, в яку повертається інформація про модуль.

Механізм роботи з потоками процесу ідентичний механізму, описаному для модулів, тільки викликаються функції Thread32First і Thread32Next, а замість структури MODULEENTRY32 використовується THREADENTRY32.

Доступ до пам’яті процесу має великі труднощі. Відомо, що в Win32 отримати доступ до пам’яті одного процесу з іншого офіційно неможливо. Проте в THL є функція ProcessMemory(), що дозволяє отримати копію блоку пам’яті іншого процесу.

Для дослідження динамічної пам’яті процесу виділяються області у віртуальній пам’яті використанням функції HeapCreate(), яка створює відповідний об’єкт (heap) в адресному просторі процесу. Для виділення пам’яті у середині цього блоку використовується функція HeapAlloc(). Всі інші функції роботи з динамічною пам’яттю (GlobalAlloc(), malloc(), calloc() і т. д.) зводяться до використання цього механізму.

Очевидно, що потрібно вміти знаходити список виділених областей віртуальної пам’яті і для кожного з них – список виділених блоків. Подивимося, як це робиться за допомогою функцій THL. У термінах THL область пам’яті називається heap. Є багато областей пам’яті – heap list. Виділеному всередині неї блоку відповідає термін «block of a heap». У циклі за допомогою функцій Heap32ListFirst() і Heap32ListNext() повертається список блоків пам’яті, які були розподілені. Знаючи ідентифікатор процесу та ідентифікатор області віртуальної пам’яті, який повертається в поле th32HeapID структури HEAPLIST32, можна побачити, що розподіляється пам’ять усередині цього блоку. Виділені області пам’яті можна отримати за допомогою функцій Heap32First() і Heap32Next(). Вони повертають структуру HEAPENTRY32.

Знаючи значення полів dwAddress і dwBlockSize, за допомогою функції Toolhelp32ReadProcessMemory() неважко отримати дані, що знаходяться у динамічній пам’яті процесу [3, 10, 11].

Фізичне проектування інтерфейсу середи розробки при створенні нового проекту зображено на рис. 2. На ньому наведено зображення оболонки у якій створювався модуль проекту.

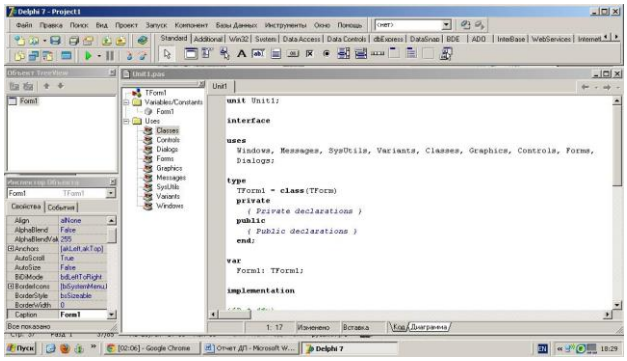

Рис. 2 – Оболонка створення проекту

Було проведено тестування зробленої розробки. Доказом тестування створеного проекту э демонстрація на рис. 1 роботи програми при отриманні інформації про добре відомий професіоналам процес WINWORD.EXE ().

Обравши процес у верхній частині програмі ми можемо отримати інформацію про бібліотеки та модулі, які використовує обраний нами процес. На рис. 1 ми бачимо які бібліотеки використовує процес WINWORD.EXE.

Тестування розробки «Отримання інформації про активні процеси» було успішно проведено на кількох версіях ОС Windows, тобто можна говорити про досягнення сумісності, яка була одним з критеріїв вибору даного методу отримання інформації про активні процеси. Ніяких збоїв у роботі розробленого додатку не виникало.

**Висновки.** В роботі були розглянуті механізми створення процесів та структур, які забезпечують роботу зробленого додатку. Безпосередньо використовувати ці механізми програміст не може – вони виконуються операційною системою (можна «побачити» структури, за допомогою різних відгадчиків, наприклад WinDbg, SoftIce), проте їх розуміння важливе для ефективного використання розглянутих методів отримання списку активних процесів.

Було визначено, що найбільш доцільним є використання засобів бібліотеки Tool Help Library через кращу стабільність та підтримку цього методу усіх наданих версій операційної системи Windows 7/8/10.

Використання цієї бібліотеки полягає в виготовленні «знімків» стану процесів у системі, ця дія виконується функцією CreateToolhelp32Snapshot. Функцією Process32First «отримуємо» перший процес із зробленого «знімку». Перехід на наступні процеси виконується функцією Process32Next. Обидві останні функції заповнюють структуру PROCESSENTRY32, яка містить доволі детальну інформацію щодо процесу.

За допомогою API функцій було створено програмний продукт, що отримує детальну інформацію про активні процеси і вирішує поставлене завдання для якого раніше не було розроблених додатків.

**Список літератури**

1. *Руссинович М.* Внутреннее устройство Microsoft Windows / *М. Руссинович, Д. Соломон*. – М. : Русская редакция; СПб. : Питер; 2008. – 992 с.
2. *Рихтер Д.* WinRT: программирование на C# для профессионалов = Windows Runtime via C# / *Джеффри Рихтер, Мартен ван де Боспурт*. – М. : Вильямс, 2014. – 368 с.
3. *Джонсон М. Харт.* Системное программирование в среде Windows. / *М. Харт Джонсон.* – М. : Вильямс, 2005. – 592 с.
4. *Deitel P.* C++11 for Programmers (2nd Edition) / *Paul Deitel, Harvey Deitel*. – Pearson Education Inc., 2013. – 848 p.
5. *Machado M.* Customizing the Microsoft NET Framework Common Language Runtime / *Manel Machado*. – 2008. – 896 p.
6. *Deshpande M.* Item-based top-N recommendation algorithms / *M. Deshpande, G. Karypis* // ACM Transactions on Information Systems. – 2011. – Vol. 29, no. 1. – P. 143–177.
7. *Nikovski D.* Induction of compact decision trees for personalized recommendation / *D. Nikovski, V. Kulev* // Proceedings of the 2006 ACM Symposium on Applied Computing (SAC 2006). Dijon, France, April 2006. – Vol. 1. – Dijon, 2006. – P. 575–581.
8. *Su X.* A survey of collaborative filtering techniques. – Advances in Artificial Intelligenc e/ *Xiaoyuan Su, Taghi M. Khoshgoftaar*. – New York : Hindawi Publishing Corporation, 2009. – P. 1–19. – doi: 10.1155/2009/421425
9. *Linden G.* Item-to-Item Collaborative Filtering / *G. Linden, B. Smith, J. York* // IEEE Internet Computing. – Los Alamitos, CA USA. – 2003. – P. 76–80.
10. *Рихтер Д.* Создание эффективных WIN32-приложений с учётом специфики 64-разрядной версии Windows. – *Джеффри Рихтер*. – СПб. : Питер «Русская редакция», 2011. – 752 с.
11. *Krizhevsky A.* ImageNet classification with deep convolutional neural networks / *Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton* // Advances in Neural Information Processing Systems 25 (NIPS 2012). Vol. 25. – Lake Tahoe, Nevada, 2012. – P. 1097–1105.

**References (transliterated)**

1. Russinovich M., Solomon D. *Vnutrennee ustroystvo Microsoft Windows* [Windows Internals]. Moscow, Russkaja Redakcija Publ. and Saint Petersburg, Piter Publ., 2008. – 992 p.
2. Jeffrey Richter, Marten van de Bospoort. *WinRT: prohrammyrovanye na C# dlya professyonalov* [Windows Runtime via C#]. Moscow, Vil'jams Publ., 2014. 368 p.
3. Johnson M. Hart. *Systemnoe prohrammyrovanye v srede Windows* [Windows System Programming]. Moscow, Vil'jams Publ., 2005. 592 p.
4. Deitel Paul, Deitel Harvey. *C++11 for Programmers (2nd Edition)*. Pearson Education Inc. Publ., 2013. 848 p.
5. Manel Machado Customizing the Microsoft .NET Framework Common Language Runtime 2008. 896 p.
6. Deshpande M., Karypis G. Item-based top-N recommendation algorithms. *ACM Transactions on Information Systems*. 2011, vol. 29, no. 1, pp. 143–177.
7. Nikovski D., Kulev V. Induction of compact decision trees for personalized recommendation. *Proceedings of the 2006 ACM Symposium on Applied Computing (SAC 2006). Vol. 1*. Dijon, France, April 2006, pp. 575–581.
8. Su Xiaoyuan, Khoshgoftaar Taghi M. A survey of collaborative filtering techniques. Advances in Artificial Intelligence. New York, Hindawi Publishing Corporation, 2009, pp. 1–19. doi: 10.1155/2009/421425
9. Linden G., Smith B., York J. Item-to-Item Collaborative Filtering. *IEEE Internet Computing*. Los Alamitos, CA USA, 2003, p. 76–80
10. Richter J. Sozdanie jeffektivnyh WIN32-prilozhenij s uchjotom specifiki 64-razrjadnoj versii Windows [Programming Applications for Microsoft WINDOWS]. – Saint Petersburg, Piter Publ. and Moscow, Russkaja Redakcija Publ., 2011. – 752 p.
11. Krizhevsky A., Sutskever I., Hinton G. ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems 25 (NIPS 2012). Vol. 25.* Lake Tahoe, Nevada, 2012, pp. 1097–1105.

*Надійшла (received) 05.02.2017*

*Бібліографічні описи / Библиографические описания / Bibliographic descriptions*

**Контроль виконуваних екземплярів додатків в Windows 7/8/10 / Є. О. Лобода, Д. О. Дубовий** // Вісник НТУ «ХПІ». Серія: Системний аналіз, управління та інформаційні технології. – Харків : НТУ «ХПІ», 2017. – № 28 (1250). – С. 60–64. – Бібліогр.: 11 назв. – ISSN 2079-0023.

**Контроль исполняемых экземпляров приложений в Windows 7/8/10 / Е. А. Лобода, Д. А. Дубовой** // Вісник НТУ «ХПІ». Серія: Системний аналіз, управління та інформаційні технології. – Харків : НТУ «ХПІ», 2017. – № 28 (1250). – С. 60–64. – Библиогр.: 11 назв. – ISSN 2079-0023.

**Control of the executable applications in Windows 7/8/10 / E. A. Loboda, D. A. Dubovoy** // Bulletin of NTU "KhPI". Series: System analysis, control and information technology. – Kharkov : NTU "KhPI", 2017. – No. 28 (1250). – P. 60–64. – Bibliogr.: 11. – ISSN 2079-0023.

*Відомості про авторів / Сведения об авторах / About the Authors*

***Лобода Євген Олександрович*** – кандидат технічних наук, доцент, Національний технічний університет «Харківський політехнічний інститут», професор кафедри обчислювальної техніки та програмування; м. Харків, Україна; тел.: +38(057) 336-41-68; e-mail: loboda.eugene@gmail.com.

***Лобода Евгений Александрович*** – кандидат технических наук, доцент, Национальный технический университет «Харьковский политехнический институт», профессор кафедры вычислительной техники и программирования; г. Харьков, Украина; тел.: +38(057) 336 41 68; e-mail: loboda.eugene@gmail.com.

***Loboda Eugene Alexandrovich*** – Candidate of Technical Sciences (Ph. D.), Docent, National Technical University "Kharkiv Polytechnic Institute", Professor at the Department of Computer Engineering and Programing; tel.: +38(057) 336-41-68; e-mail: loboda.eugene@gmail.com.

***Дубовий Дмитро Олександрович*** – Національний технічний університет «Харківський політехнічний інститут»; студент факультет комп'ютерних та інформаційних технологій; тел.: +38(095) 871-48-66.

***Дубовой Дмитрий Александрович*** – Национальный технический университет «Харьковский политехнический институт»; студент факультета компьютерных и информационных технологий; тел.: +38(095) 871-48-66.

***Duboviy Dmitro Aleksandrovych*** – National Technical University "Kharkiv Polytechnic Institute", student, Faculty of Computer and Information Technology; tel.: +38(095) 871-48-66.