Семеріков С.О., Мінтій І.С., Словак К.І.,
Теплицький І.О., Теплицький О.І.
Криворізький державний педагогічний університет

Мобільне програмне забезпечення навчання інформатичних дисциплін у вищій школі

Постановка проблеми. Як було показано у [1], фундаменталізація навчання виступає насамперед інструментом стабілізації змісту навчання засобами, адекватними предметній галузі навчання в умовах швидких темпів її розвитку. Так, у [1, 73] зазначено, що стабілізація курсів інформатики досягається поширенням на методичну систему навчання інформатики властивостей *відкритих систем*: розширюваності, масштабованості, мобільності; інтероперабельності та «люб'язності».

Аналіз останніх досліджень. В останні десятиліття «центр уваги змістився ... на мобільність програмного продукту як міру уніфікації та подовження терміну його життя, можливість використання з наступними поколіннями ЕОМ» [2, 395]. У першому розділі [1] було показано, що стабілізація програмного забезпечення разом з усталенням змісту навчання веде до фундаменталізації навчання інформатичних дисциплін у вищій школі, саме тому **метою статті** є розгляд стабільного мобільного програмного забезпечення, що виступає технічним засобом фундаменталізованого навчання.

Основна частина. *Мобільність програмного забезпечення* визначається як властивість, що полягає у можливості його перенесення з ЕОМ одного типу на ЕОМ іншого з низькими працевитратами [3, 328]. Властивість мобільності важлива при створенні програмного забезпечення для електронного навчання. Під *мобільністю пакета прикладних програм* будемо розуміти можливість перенесення його з одного операційного середовища в інше. Мобільність пакета прикладних програм є «найважливішою властивістю пакетів прикладних програм ... [в комп'ютерній технології навчання], оскільки сприяє спрощенню їх тиражування, супроводу, а також полегшує навчання роботи з ними (не виникає необхідності повторного навчання при зміні технічної бази комп'ютерної технології навчання)» [3, 329].

1. Мобільні операційні системи

1.1. Історія створення й поточний статус стандарту POSIX. Один із загальноприйнятих способів підвищення мобільності (кросплатформенності, портабельності) програмного забезпечення – стандартизація програмного оточення: програмних інтерфейсів, утиліт тощо [4]. На рівні системних сервісів подібне оточення описується в стандарті POSIX (Portable Operating System Interface – мобільний інтерфейс операційної системи); назва запропонована відомим фахівцем, засновником Фонду вільно поширюваного програмного забезпечення Ричардом Столменом.

Розглянемо сучасну версію стандарту POSIX у редакції 2003 р., яку можна назвати «потрійним стандартом», а саме: стандартом IEEE Std 1003.1, Технічним стандартом Open Group та міжнародним стандартом ISO/IEC 9945.

Історія створення цієї версії така. На початку 1998 р. представники трьох організацій – Комітету зі стандартів мобільних додатків IEEE, Open Group та робочої групи 15 підкомітету 22 спільного технічного комітету 1 (JTC1/SC22/WG15) Міжнародної організації зі стандартизації (ISO) – почали консультації з питання злиття й розвитку курованих ними стандартів інтерфейсів до системних сервісів: IEEE Std 1003.1, IEEE Std 1003.2, Базових специфікацій від Open Group, ISO/IEC 9945-1, ISO/IEC 9945-2. У вересні того ж року в місті Остін, штат Техас, в офісі корпорації IBM відбулося організаційне засідання групи, сформованої для досягнення поставленої мети.

Основним документом для переглянутого стандарту, перший проект якого був представлений у липні 1999 року, стали Базові специфікації від Open Group, оскільки вони включали положення стандартів IEEE і ISO/IEC. В 2001 році, після завершення підготовчої роботи, стандарт містив наступні чотири частини:

– основні означення (терміни, концепції та інтерфейси, спільні для всіх частин);
– опис прикладного програмного С-інтерфейсу до системних сервісів;
– опис інтерфейсу до системних сервісів на рівні командної мови й службових програм;
– детальне роз'яснення положень стандарту, обґрунтування ухвалених рішень.

Далі в ISO, IEEE і Open Group в 2001-2002 рр. пройшло формальне затвердження нового стандарту POSIX. Тим часом накопичувалися відносно дрібні виправлення, враховані в редакції 2003-го року.

З розвитком стандарту розширювалося й трактування терміну «POSIX». Спочатку він відносився до документа IEEE Std 1003.1-1988, де описувався прикладний програмний інтерфейс ОС класу Unix. Після стандартизації інтерфейсу на рівні командної мови й службових програм більш правильно розуміти під словом «POSIX» стандарт у цілому, позначаючи перераховані вище частини 2 і 3 через POSIX.1 і POSIX.2 відповідно до нумерації документів IEEE і ISO/IEC.

1.2. Основні ідеї стандарту POSIX. В стандарті POSIX описується множина базових системних сервісів, необхідних для функціонування прикладних програм. Доступ до них надається за допомогою інтерфейсу, специфікованого для мови С, командної мови й загальновикористовуваних службових програм.

У кожного інтерфейсу є дві сторони: та, що викликає, і та, що викликається. Стандарт POSIX орієнтований у першу чергу на ту, що викликає. Його призначення – зробити програми мобільними на рівні вихідної мови. Це значить, зокрема, що при перенесенні С-програм на іншу операційну платформу буде потрібна перекомпіляція. Про мобільність виконуваних програм чи об'єктних файлів у стандарті мова не йде – це забезпечується додатковими засобами (середовища .NET та Mono для C#, інтерпретатори байт-коду для Java, Python тощо).

Стандарт POSIX аж ніяк не обмежений рамками Unix-середовища: практично для всіх сучасних

операційних системи існують розширення (наприклад, для Windows – Cygwin, MinGW, SFU і т.п.), через які надаються необхідні сервіси і тим самим підтримується виконання POSIX-сумісних програм. Можна стверджувати, що підтримка стандарту POSIX полегшує перенесення прикладних програм практично на будь-яку скільки-небудь поширену операційну платформу. Додаткові зусилля стосовно підвищення мобільності, прикладені на етапі розробки, безумовно, окупаються незалежністю POSIX-програм від обраної операційної системи.

Разом з визначенням інтерфейсу системних викликів в POSIX залишається за рамками розгляду їх реалізація. Зокрема, не розрізняються системні виклики й бібліотечні функції. Не є об'єктом стандартизації засоби адміністрування, апаратні обмеження та функції, необхідні тільки адміністратору, що ще раз підкреслює спрямованість стандарту POSIX на прикладні програми, а не на операційні системи.

POSIX нейтральний стосовно системної архітектури й розрядності процесора. Це дуже важливий аспект мобільності програм.

Орієнтація на міжнародний стандарт мови С визначила не тільки стиль опису функцій, але й певною мірою напрям розвитку специфікацій POSIX у плані синхронізації обох стандартів. Як відомо, у затвердженій в 1999 р. редакції специфікацій мови С узаконений комплексний тип даних, що викликало відповідне поповнення POSIX-функцій.

У стандарті POSIX виконаний поділ на обов'язкові та додаткові функції, причому обов'язкове ядро зроблене, за можливості, компактним. Особлива увага приділяється способам реалізації стандартизованих функцій як в «класичному» Unix-середовищі, так і на інших операційних платформах, у мережних і розподілених конфігураціях.

Розробники нової версії стандарту POSIX дбайливо віднеслися і до його передісторії, і до передісторії Unix-систем, і, головне, до прикладних програм, що задовольняли більше раннім версіям стандарту. Існуючі інтерфейси намагалися зберігати; у процесі розвитку дотримувався принцип зворотної сумісності; нові інтерфейси додавалися так, щоб вони не конфліктували з попередніми. Повністю уникнути внесення змін у додатки не вдалося із цілком зрозумілих причин: треба було усунути протиріччя між різними вихідними специфікаціями, а також відмовитися від підтримки традиційного варіанту мови С і перейти на його міжнародний стандарт.

1.3. Основні поняття операційних систем, що відповідають стандарту POSIX: користувач; файл; процес; термінал; хост; вузол мережі; час; мовно-культурне середовище.

Це первинні поняття. Їх не можна строго означити, але можна пояснити за допомогою конкретних прикладів. Для кожного з виділених понять будуть описані їх характеристики – властиві їм атрибути й застосовні до них операції.

У тексті стандарту POSIX подані наступні характеристики основних понять разом з посиланнями на атрибути й операції.

У *користувача* є ім'я й числовий ідентифікатор.

Файл – об'єкт, стосовно якого допускається читання, запис, та який має такі атрибути, як права доступу до нього та тип (звичайний файл, символьні й блокові спеціальні файли, канал, символьне посилання, сокет і каталог). В реалізації можуть підтримуватися й інші типи файлів.

Процес – адресний простір разом з виконуваними в ньому потоками управління, а також системними ресурсами, необхідними для їх виконання.

Термінал (або термінальний пристрій) – символьний спеціальний файл, що задовольняє специфікації загального термінального інтерфейсу.

Мережа – сукупність взаємозалежних вузлів.

Мовно-культурне середовище – частина оточення користувача, що залежить від мовних та культурних домовленостей.

Для роботи з великою кількістю об'єктів завжди надаються механізми групування й побудови ієрархій. Існує ієрархія файлів та каталогів, групи користувачів і процесів, підмережі й т.п.

1.4. Середовище компіляції POSIX-сумісних програм. Часто (хоча це не завжди усвідомлюється) розробка програм ведеться в крос-режимі, тобто платформа розробки (еквівалентний термін – інструментальна платформа) не збігається із платформою виконання (що зветься також цільовою платформою). На інструментальній платформі створюється середовище компіляції програм, а результат компіляції може бути перенесений для наступного виконання на цільову платформу.

Найважливіша частина середовища компіляції – інтерфейсні файли, що містять прототипи функцій, визначення констант, макросів, типів даних, структур і т.п. Для кожної описаної в стандарті POSIX функції визначено, які інтерфейсні файли повинні бути включені для компіляції програми (найчастіше потрібен один файл).

В стандарті POSIX передбачено симетричний механізм перевірки функціональності, що надає можливість з програм в разі потреби надсилати запити на одержання доступу до певних прототипів і імен.

1.5. Мобільність POSIX-сумісних програм принципово досяжна завдяки двом основним факторам. По-перше, це наявність величезного числа стандартизованих системних сервісів, а по-друге, можливість динамічного з'ясування характеристик цільової платформи й налаштовування програми під них. (Природно, мається на увазі мобільність у рамках, регламентованих стандартом.)

Програми, що відповідають стандарту POSIX, можуть бути одно- і багатопроцесними, з динамічною адаптацією конфігурації до властивостей цільової платформи. Стандартизовано засоби породження й завершення процесів, зміни відповідних програм, опитування та зміни різноманітних характеристик. Процеси можна призупиняти й активізувати в заданий час. За допомогою механізму

сигналів можна сповіщати про події й завершувати процеси ззовні. Для їх групування передбачені засоби управління завданнями. Програми оснащені регуляторами для управління плануванням і пріоритетами процесів. Наявний широкий спектр засобів управління пам'яттю та обміну даними між процесами: черги повідомлень, поділювана (спільно використовувана) пам'ять, семафори. Нарешті, у межах процесу можна організувати кілька потоків управління.

Необхідний ступінь детермінізму виконання досягається завдяки засобам підтримки реального часу (до них відносяться управління дисципліною виділення процесорів, сигнали реального часу, утримання сторінок в оперативній пам'яті, точні таймери і т.п.).

Функції для роботи з файлами призначені для виконання запитів від програм на читання й запис даних тривалого зберігання, захист таких даних від несанкціонованого доступу. Механізм блокування фрагментів файлів дозволяє забезпечити атомарність транзакцій. Наявність асинхронного введення/виведення дозволяє об'єднувати операції обміну, оптимізуючи виконання програми. У стандарті POSIX ретельно пророблені питання доступу до зовнішніх пристроїв, під'єднаних через послідовні лінії, особливо до терміналів.

Для багатокористувацьких систем необхідна організація одночасної роботи великої кількості людей. В POSIX ця проблема вирішується через регламентацію засобів безпосереднього й поштового обміну повідомленнями.

Для роботи з програмами надаються стандартизовані засоби для з'ясування як «крупноблочних» характеристик цільової системи (наприклад, спектру підтримуваних необов'язкових засобів), так і більш дрібних характеристик (наприклад, поточний розмір вільного дискового простору).

Проблеми мобільності програм надзвичайно складні, і було б перебільшенням стверджувати, що в стандарті POSIX-2001 вони вирішуються повністю. За його рамками залишаються такі найважливіші питання, як графіка, багатовіконний інтерфейс і цілий ряд інших.

1.6. Підтримка стандарту POSIX у сучасних операційних системах. Повністю POSIX-сумісними (такими, що мають відповідний сертифікат сумісності) є наступні операційні системи: A/UX, BSD/OS, HP-UX, IBM AIX, INTEGRITY, IRIX, LynxOS, Mac OS X, Minix, MPE/iX, OpenSolaris, OpenVMS, QNX, RTEMS, Solaris, UnixWare, velOSity, VxWorks.

POSIX-сумісними (такими, що офіційно не сертифіковані, але відповідні стандарту) є BeOS, FreeBSD, GNU/Linux, NetBSD, Nucleus RTOS, OpenBSD, RTEMS, Sanos, SkyOS, Syllable, VSTa.

Серед перелічених систем найбільш поширеними є:

Mac OS X – POSIX-сумісна операційна система фірми Apple Inc, що базується на мікроядрі Mach та підсистемі BSD-UNIX Каліфорнійського університету в Берклі, випускається для комп'ютерів Macintosh на базі процесорів PowerPC та Intel.

Minix – вільно поширювана POSIX-сумісна мікроядерна операційна система, що поширюється за ліцензією BSD. Ендрю Таненбаум створив першу версію Minix в 1987 в якості ілюстрації до підручника [5].

QNX – комерційна POSIX-сумісна операційна система реального часу. QNX призначена в першу чергу для вбудованих систем. Вважається однією з найкращих реалізацій концепції мікроядерних операційних систем.

Linux – монолітне ядро, що використовується для створення POSIX-сумісних операційних систем. Це один із найвидатніших прикладів розробки з відкритими джерельними кодами та вільно поширюваного програмного забезпечення. Спершу розроблювалася та використовувалася індивідуальними ентузіастами на персональних комп'ютерах. З тих пір Linux завдяки підтримці таких компаній, як IBM, Sun Microsystems, Hewlett-Packard, Novell та інших набув неабиякої популярності як серверна операційна система. Linux портовано на велику кількість апаратних платформ. Тепер вона досить успішно використовується як на суперкомп'ютерах, так і на мобільних телефонах. Значна кількість спеціалізованих дистрибутивів Linux, що розробляються та підтримуються різними спільнотами, забезпечує широкий вибір програмного забезпечення.

В операційні системи сімейства *Windows* включена підсистема Microsoft POSIX, в якій реалізовано одну з перших версій стандарту POSIX. POSIX-сумісні програми функціонують як процеси-нащадки posix.exe. Обмеженість реалізації не дає можливості створювати в них потоки управління, вікна, використовувати засоби обміну даними між процесами тощо.

Для забезпечення повної POSIX-сумісності застосовуються наступні програми:

1. *Cygwin* – набір вільно поширюваних програмних інструментів, розроблених фірмою Cygnus Solutions (входить до складу Red Hat), застосування яких надає можливість перетворення Windows на POSIX-сумісну систему. Cygwin розроблявся як середовище для перенесення програм з POSIX-сумісних операційних систем у Windows. Cygwin містить бібліотеку, за допомогою якої реалізується інтерфейс прикладного програмування POSIX на основі системних викликів Win32. Крім того, Cygwin містить у собі інструменти розробки GNU для виконання основних завдань програмування, а також і деякі прикладні програми, еквівалентні базовим програмам UNIX. В 2001 році до складу Cygwin був включений пакет X Window System.

2. *Microsoft Windows Services for UNIX* (Сервіси Microsoft Windows для UNIX, SFU) – програмний пакет Interix, розроблений компанією Microsoft, що забезпечує POSIX-сумісність Windows. Windows Vista Enterprise і Ultimate Editions також містять елементи SFU, перейменовану в підсистему для прикладних додатків UNIX (Subsystem for UNIX-based applications, SUA).

3. *MinGW* або Mingw32 (Minimalist GNU for Windows) – колекція вільно розповсюджуваних заголовних файлів і бібліотек у сполученні з набором інструментів GNU (компілятор GCC і ін.), що надає можливість створення програм Windows, в яких не використовуються сторонні динамічні

бібліотеки. Спочатку MinGW створювався як відгалуження Cygwin для роботи з бібліотекою Microsoft MSVCRT (Windows API); бібліотека MinGW менш вимоглива до обсягу оперативної й дискової пам'яті, поширюється під більше вільною ліцензією й може використовуватися з будь-яким програмним забезпеченням, але функціональність специфікації POSIX реалізовані в ній не так повно, як в Cygwin.

4. *GnuWin32* – проект із портування програм за вільними ліцензіями на платформу Windows. Програми компілюються для безпосереднього виконання в Windows, не вимагаючи запуску в емуляторах Unix-оточення, таких як Cygwin або MSYS.

5. *CoLinux* (Cooperative Linux) – технологія, за допомогою якої можна запускати Linux у Windows. В CoLinux використовується модифікований Linux і спеціальний драйвер Windows для відображення системних викликів Linux у виклики Windows. Пам'ять програми використовується як системна пам'ять операційної системи. Використовуючи цю технологію, можна запускати один або кілька екземплярів Linux у середовищі Windows без втрати швидкості (для користувача екземпляри Linux виглядають як запущені на іншому комп'ютері та доступні через мережу).

Таким чином, можна зробити висновок, що **POSIX-сумісність є засобом уніфікації операційних систем, а дотримання стандартів POSIX при розробці програмного забезпечення – засобом уникнення залежності від використовуваної операційної системи.**

2. Мобільні компілятори

2.1. GCC. Колекція компіляторів GNU (GNU Compiler Collection, GCC) – набір компіляторів для різних мов програмування. GCC – вільно поширюване програмне забезпечення, що розробляється Фондом вільного програмного забезпечення (FSF) під ліцензією GNU GPL та GNU LGPL, і є ключовою складовою набору інструментів розробки GNU (GNU development toolchain). Це стандартний набір компіляторів для POSIX-сумісних операційних систем [6].

GCC започаткований Ричардом Столменом у 1985 році як компілятор мови C (GNU C Compiler) для проекту GNU. Перша версія випущена навесні 1987 року, у 1988 році з'явилася підтримка C++. GCC був першим незалежно створеним (не базувався на препроцесорі CFront Б'ярна Страуструпа) та першим власне компілятором (а не препроцесором у C) мови C++.

У 1997 група розробників, незадоволена повільним темпом і закритістю офіційної розробки GCC, створила проект EGCS (Experimental/Enhanced GNU Compiler System – Експериментальна/Покращена збірка компіляторів GNU), в якому об'єднано кілька експериментальних відгалужень GCC. Розробка EGCS з часом виявилась більш життєздатною, ніж GCC, і у квітні 1999 року EGCS був оголошений офіційною версією GCC.

GCC сьогодні розробляється широкою групою розробників зі всього світу. Він перенесений на більшу кількість типів процесорів та операційних систем, ніж будь-який інший компілятор. Версія GCC для Windows забезпечується проектами MinGW та Cygwin, під DOS – проектом DJGPP (лише C/C++).

У версії 4.3.3 (випущеній у січні 2009 року), у типовій збірці підтримуються наступні мови: Ada, C, C++, Fortran 95, Java, Objective-C, Objective-C++. В додаткових проектах підтримуються мови програмування Pascal, Modula-2, Modula-3, Mercury, VHDL, PL/I та D.

За допомогою GCC версії 4.3 створюється код для таких процесорних архітектур: Alpha, ARM, Atmel AVR, Blackfin, HC12, H8/300, IA-32 (x86), x86-64, IA-64, Motorola 68000, MIPS, PA-RISC, PDP-11, PowerPC, R8C/M16C/M32C, SPU, System/390/zSeries, SuperH, SPARC і VAX.

Менш відомі серед підтримуваних процесорів включають A29K, ARC, ETRAX CRIS, D30V, DSP16xx, FR-30, FR-V, Intel i960, IP2000, M32R, 68HC11, MCORE, MMIX, MN10200, MN10300, Motorola 88000, NS32K, ROMP, Stormy16, V850, Xtensa та AVR32.

Окремими проектами підтримуються D10V, LatticeMico32, MeP, Motorola 6809, MicroBlaze, MSP430, Nios II та Nios, PDP-10, TIGCC (варіант для m68k), Z8000 та PIC24/dsPIC.

Слід зазначити, що компілятори GCC включають десятки опцій і користуватися ними напряму не зовсім зручно, тому для спрощення роботи рекомендується використовувати оболонки або інтегровані середовища розробки – Code::Blocks, Dev-C++, KDevelop, NetBeans, Eclipse.

2.2. Free Pascal (повна назва Free Pascal Compiler, FPC) – це вільно поширюваний компілятор програм, описаних мовою програмування Паскаль.

FPC – кросплатформенний інструмент, призначений для різних апаратних платформ, зокрема i386, x86-64 (тільки Linux і Windows), PPC, PPC64 (тільки Linux), SPARC (тільки Linux), ARM. Важливою особливістю даного компілятора, на відміну, наприклад, від GNU Pascal, є орієнтація на поширені комерційні діалекти мов Object Pascal і Delphi, які широко застосовуються у вітчизняній системі освіти. Поряд з цим FPC включає ряд додаткових засобів, наприклад, підтримується перевизначення операторів.

В останні роки у рамках проекту Free Pascal також розробляється Lazarus – вільно поширюваний аналог середовища розробки Delphi і Lazarus Components Library (LCL) – вільно поширювана бібліотека візуальних компонентів, аналогічна VCL у Delphi.

У Free Pascal підтримується компіляція в кількох режимах, якими забезпечується сумісність з різними діалектами й реалізаціями мови:

TP	режим сумісності з Turbo Pascal: сумісність практично повна, за винятком кількох моментів, пов'язаних з тим, що за допомогою FPC компілюються програми для захищеного режиму процесора, в якому неможливо пряме звертання до пам'яті, портів і т.д.
FPC	власний діалект: відповідає попередньому, розширеному додатковими засобами, такими як, наприклад, перевизначення операторів
DELPHI	режим сумісності з Borland Delphi: включає підтримку класів і інтерфейсів
OBJFPC	поєднано об'єктно-орієнтовані засоби Delphi і власні розширення мови

MACPAS	режим сумісності з Mac Pascal
GNU	режим часткової сумісності з GNU Pascal

Застосування Free Pascal надало можливість, з одного боку, підтримати існуючі методики навчання на основі Borland Pascal 7 та Kylix/Delphi, а з іншого – розширити сферу застосування компіляторів Pascal в навчанні POSIX-сумісних операційних систем [7; 8].

Завершуючи огляд, можна зробити висновок, що **застосування мобільних компіляторів є засобом уникнення залежності від використовуваного середовища програмування.**

3. Мобільні інтерпретовані мови програмування. Як було показано вище, застосування мобільних компіляторів виступає засобом стабілізації таких компільованих мов програмування, як C, C++, Java, Pascal та ін. У той же час слід зазначити, що у вищій школі в навчанні інформатичних дисциплін все більшого застосування набувають *мобільні інтерпретовані мови загального призначення*. Яскравим прикладом такої мови є мова **Scheme**, застосування якої надало можливість створити незалежний від використовуваної операційної системи вступний курс з програмування.

Scheme – невеликий і елегантний діалект мови Lisp – інтерпретована функціональна мова програмування високого рівня. Перший варіант мови Scheme був створений Дж. Дж. Сассманом та Г. Стілом в Массачусетському технологічному інституті у 1975 р. як «простий та конкретний експериментальний інструмент для досліджень в галузі семантики та стилю програмування» [9]. Найбільш інтенсивна розробка ядра мови відбувалась у період з 1975 по 1980 рр.

Серед переваг Scheme можна назвати як такі, що характеризують її як функціональну мову програмування, так і її власні:

– відкладені обчислення (альтернативою виклику за значенням є виклик за необхідності);

– стислість і простота (програми, описані функціональними мовами зазвичай набагато коротші і простіші за ті ж самі програми, описані імперативними мовами);

– модульність (механізм модульності дозволяє розділяти програми на кілька порівняно незалежних частин (модулів) з чітко визначеними зв'язками між ними, що полегшує процес проектування і наступної підтримки великих програмних систем);

– використання функцій як значень (функції можуть бути передані іншим функціям як аргумент або повернуті як результат);

– чистота (відсутність побічних ефектів);

– невеликий розмір ядра мови (менше 20 ключових слів);

– мобільність програм (внаслідок інтерпретованої природи мови);

– можливість використання в напівавтоматичному режимі (корисно для експериментування та розв'язування простих задач);

– зручність для розв'язування математичних задач (можливість опрацьовувати числові типи даних, подані як у вигляді десяткового та десяткового періодичного дробу, так і звичайного дробу; також Scheme містить засоби роботи з комплексними числами, цілими числами довільної довжини);

– простота синтаксису (завдяки цьому програми на Scheme зазвичай є коротшими, їх легше писати, читати та розуміти; студенти вже на перших заняттях можуть зрозуміти такі концепції, як рекурсія; можливість зосередитись на засвоєнні загальних принципів програмування, а не на вивченні синтаксису):

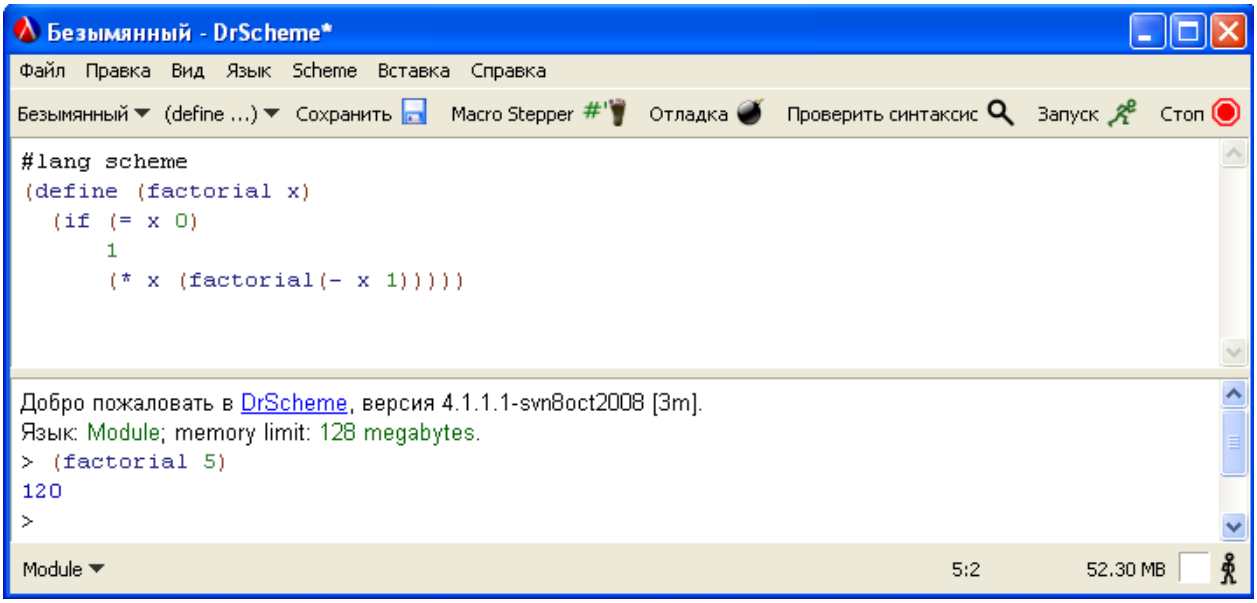
Програма мовою C	*Програма мовою Scheme*

int factorial(int x) {
  if (x == 0) {
  return 1;
  } else {
  return x * factorial(x-1);
  }
}
	(define (factorial x)
	  (if (= x 0)
	  1
	  (* x (factorial(- x 1)))))

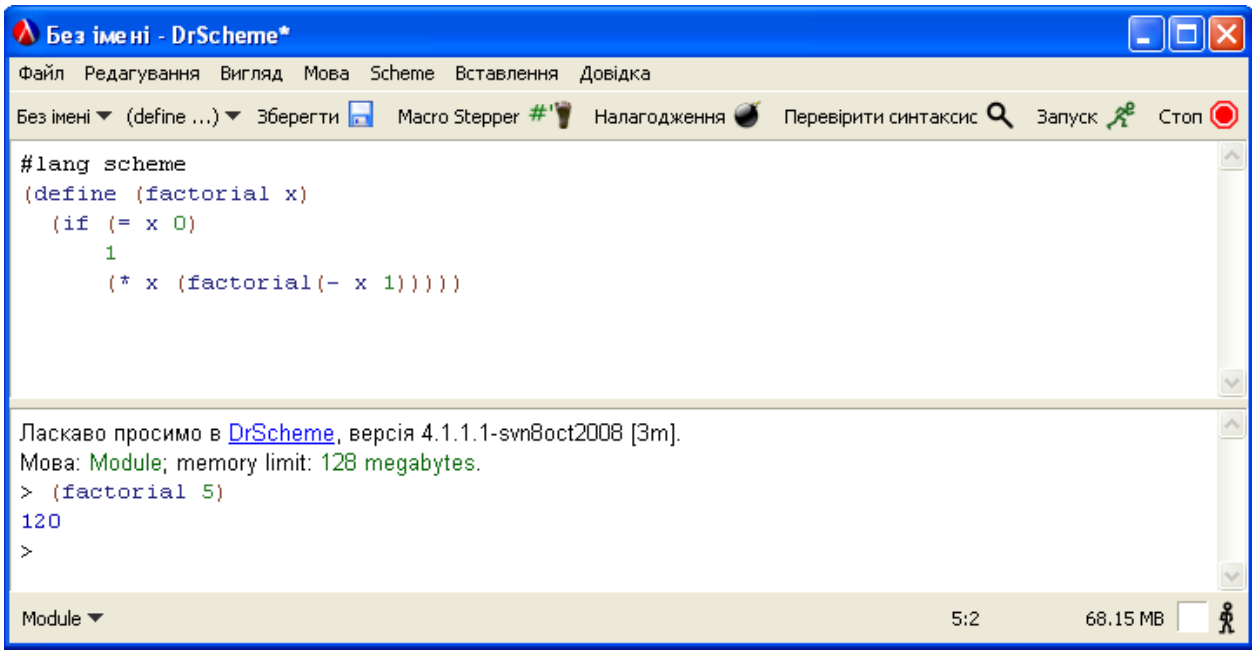
– підтримка багатьох парадигм (функціональної, об'єктно-орієнтованої та імперативної).

Інтерпретатор Scheme та стандартні бібліотеки доступні на всіх основних платформах, тобто сам інтерпретатор є мобільним.

Вказані переваги стали причиною застосування мови Scheme як у вступних курсах інформатики [10], так і при розгляді загальних принципів програмування у провідних ВНЗ світу (за матеріалами журналу Times [11]): Гарвардський університет, Сполучені Штати Америки (США); Йельський університет, США; Чиказький університет, США; Массачусетський технологічний університет, США; Каліфорнійський технологічний університет, США; Токійський технологічний інститут, Японія; Гонконгзький університет науки та технологій, Гонконг; університет Торонто, Канада; Національний університет Сінгапуру, Сінгапур та ін. На рівні середніх навчальних закладів Scheme також активно впроваджується, витісняючи, таким чином, традиційні мови програмування Basic, Pascal, або C [12].



а) Scheme російською



б) Scheme українською

Рис. 1. Локалізована версія DrScheme

Саме тому при розробці курсу «Вступ до програмування» для студентів молодших курсів вищих педагогічних навчальних закладів нами було використано середовище програмування DrScheme. На даний момент нами виконано локалізацію цього середовища російською (рис. 1а) та українською мовами (рис. 1б).

3. Мобільні Web-середовища. Всесвітні програми Intel World Ahead Education та One Laptop per Child, спрямовані на насищення освітнього ринку недорогими комп'ютерами для забезпечення рівного доступу до засобів ІКТ, викликали до життя концепцію *нетбука* (з англ. netbook) – невеликого ноутбуку, призначеного для доступу до Інтернет та роботи з офісними додатками. Нетбуки відрізняються компактними розмірами (діагональ екрану 7–10 дюймів), невеликою вагою, низькими енерговитратами і відносно невисокою вартістю.

Дані пристрої займають проміжне положення між мобільними Інтернет-пристроями (MID) та КПК «знизу» і субноутбуками «зверху». Через високу вартість швидкісної флеш-пам'яті вони оснащені твердотільними жорсткими дисками (SSD) відносно малого розміру (порядку 4 Гб), що поряд із невисокою швидкодією та малим обсягом оперативної пам'яті суттєво ускладнює застосування таких ресурсоємних додатків, як розвинені середовища програмування, системи комп'ютерної математики і т.п.

Для розв'язання цієї проблеми доцільно перейти до *мережецентричної моделі* навчання, за якої ресурсоємні прикладні програми виконуються на Інтернет-серверах, а головною клієнтською програмою виступає Web-браузер.

3.1. Онлайн-IDE. Software as a service (SaaS) – *програмне забезпечення як послуга* – це модель пропозиції програмного забезпечення користувачеві, при якій постачальник розробляє Web-додаток, розміщує його і управляє ним (самостійно або через третіх осіб) з метою і можливістю використання замовниками через Інтернет. Замовники платять не за володіння програмним забезпеченням як таким, а

за його використання (через API, що доступний через Web і в якому часто використовуються Web-служби). Близьким до терміну SaaS є термін «додаток за запитом» (On-Demand).

Принциповою відмінністю моделі SaaS від інших (Hosted Applications і Application Service Provider (ASP)) є те, що отримується саме послуга і інтерфейс (призначений для користувача або програмний), тобто деяка функціональність без жорсткої прив'язки до способу її реалізації.

У моделі SaaS:

– прикладнийй програмний засіб пристосований для віддаленого використання;

– одним прикладним засобом користуються кілька клієнтів;

– модернізація прикладного засобу відбувається плавно і прозоро для клієнтів;

– для платних послуг: оплата здійснюється як щомісячна абонентська плата або на основі об'єму транзакцій, а підтримка програмного засобу входить до складу оплати.

Тенденція розташування настільних прикладних програмних засобів у мережі та надання їх в якості послуг не є новою: так, з 2006 р. на ринку онлайнових текстових процесорів працюють Google Docs та Zoho Writer. У 2007 р. в Інтернет з'явились і перші онлайнові інтегровані середовища програмування.

Більшість Інтернет-IDE має досить специфічний характер (на відміну від IDE загального призначення, таких як Eclipse): IDE не є послугою, що надається, а скоріше інструментом для користувачів, які використовують інші послуги. Прикладами таких послуг є Coghead, Zoho Creator, Bungee Builder, Microsoft PopFly і Yahoo Pipes. Всі ці сервіси є пропрієтарними, в деяких з них використовуються свої власні мови, і вимагається розміщення всіх послуг виключно на їх серверах.

Проте існує кілька служб, які мають більш загальний характер, та базуються на стандартних мовах. Наприклад, в Heroku застосовується мова Ruby і він може бути використаний для розробки та розгортання Ruby-додатків. Найближчим часом на основі Google AppEngine планується розгортання Python-орієнтованих сервісів.

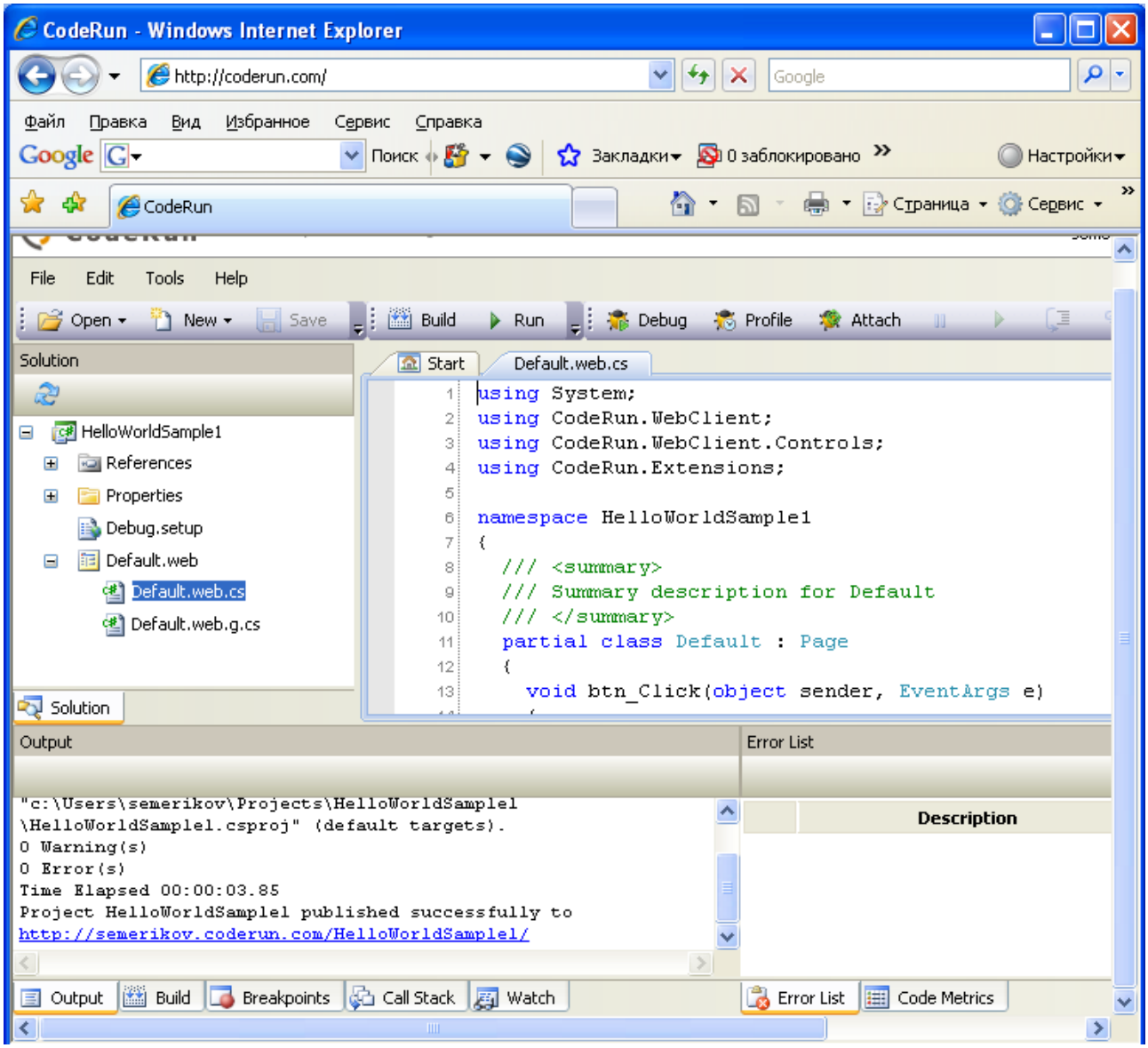


Рис. 2. CodeRun в режимі виконання програми мовою C#

CodeIDE є повноцінною онлайн-IDE, де підтримуються мови BASIC, Pascal, C, C++, Perl, Java, JavaScript, HTML, MySQL та LISP. Це середовище є гарним інструментом для залучення до програмування: його використання надає користувачеві можливість швидко почати кодування і миттєво

отримати результати. Хоча CodeIDE не є повноцінним інтегрованим середовищем, робота з ним дає уявлення про те, як в цілому може виглядати онлайн-IDE.

При роботі з онлайн-IDE від їх користувача не вимагається нічого, крім Web-браузера, хоча деякі з них (особливо ті, що оснащені розвиненим інтерфейсом користувача) є орієнтованими на конкретну платформу. На рис. 2 наведено зовнішній вигляді онлайн-IDE CodeRun у середовищі Web-браузера Internet Explorer.

Поки що в жодній з онлайн-IDE не надається такого багатого набору функцій, як в «справжніх» (настільних) IDE, такі як Eclipse або Visual Studio. Цей розрив не є унікальним для IDE – достатньо порівняти інструментальні панелі Google Docs та Microsoft Word: Google Docs містить всі основні функції, тим не менше він ще далекий від повномасштабного настільного текстового процесора.

Можна очікувати, що ця ситуація зміниться. Як стверджують аналітики Gartner, перехід до серверних додатків поки що, як і раніше, знаходиться в зародковому стані, але незабаром ця тенденція стане провідною. Так, якщо сьогодні для запуску 70–80% корпоративних додатків потрібна Windows, то до 2011 року більшість цих програм будуть незалежними від операційної системи та існувати у формі, наприклад, Web-додатків [13].

Безпосередніми перевагами онлайн-IDE є відсутність необхідності у їх інсталяції та миттєве розгортання проекту. Крім того їх застосування відкриває нові можливості для обміну навчальними матеріалами та співробітництва (досить згадати про можливість віддаленого парного програмування та спільної роботи над проектами).

Перспективним напрямом роботи є розробка своїх власних спеціалізованих онлайн-IDE та онлайнових плагінів до існуючих середовищ (зокрема, Eclipse). Так, нами пропонується аплет, де реалізується інтерпретатор Scheme з мінімалістичним інтерфейсом користувача, придатним для роботи в середовищі мобільного браузера (рис. 3).

3.2. Мобільне математичне середовище Sage. Однією з проблем, що постають в процесі навчання математичної інформатики [14], є вибір середовища для роботи. Як комерційні, так і вільно поширювані системи суттєво різняться за функціональністю (загального призначення, спеціалізовані), інтерфейсом (командного рядка, графічним), розміром (від кількасот кілобайт до кількох гігабайт), вбудованою мовою програмування тощо. Безальтернативне ознайомлення лише з однією системою комп'ютерної математики (навіть такої розвиненої, як Maxima) неминуче впливатиме на подальшу професійну діяльність, обмежуючи клас розв'язуваних задач особливостями конкретного програмного продукту.

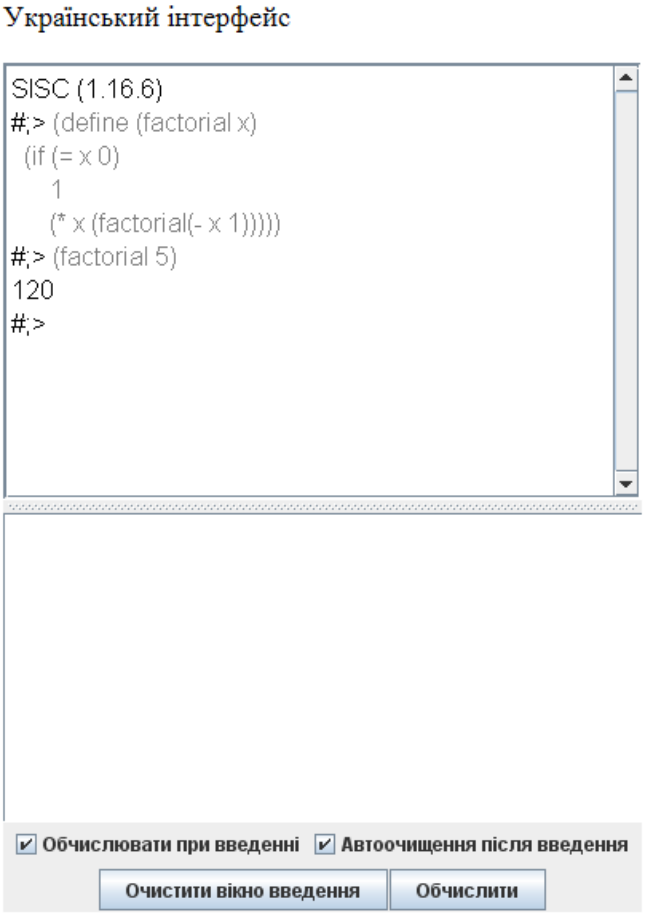


Рис. 3. Online-версія Scheme для мобільних пристроїв

Разом з тим існують причини необхідності знання основ роботи з кількома математичними системами: «необхідність раціонального вибору математичної системи з урахуванням особливостей задачі, що розв'язується; необхідність розв'язування складних задач за допомогою різних систем, щоб перевірити правильність результатів, не покладаючись на одну систему (збільшити вірогідність одержаного результату); необхідність підготовки математичних документів (статей, звітів, книг,

навчальних занять і т.д.) підвищеної якості. Останнє говорить на користь інтеграції математичних систем між собою і з іншими програмами, що може розглядатися як один з перспективних напрямів розвитку систем комп'ютерної математики» [15, 364–365].

Фактором, що ускладнює вивчення та застосування різних СКМ, є синтаксичні відмінності у застосуванні одних і тих самих команд, що можуть змінюватися навіть в межах однієї СКМ (наприклад, при виході нової версії). Інша проблема – те, що досить часто в системах загального призначення не вистачає функціональності, що є в спеціалізованих системах, та навпаки. Зауважимо, що частково ця проблема може бути розв'язана через застосування сучасних об'єктних технологій інтеграції програмного забезпечення, таких як COM та CORBA. В результаті для роботи буває необхідним цілий набір СКМ, встановлених на одному комп'ютері, що в умовах загальноосвітньої школи та ВНЗ реалізувати практично неможливо через ліцензійні чи адміністративні перешкоди.

Проблема вибору СКМ та підтримки великої інсталяційної бази може бути розв'язана через застосування мережних технологій, коли користувач за допомогою спеціалізованого клієнтського програмного забезпечення звертається до серверної частини СКМ, де виконуються команди користувача та повертається результат до клієнтського ПЗ. Такі послуги надаються, зокрема, Matlab Web Server, webMathematica та wxMaxima. І хоча далеко не у всі СКМ включені вбудовані мережні засоби, для тих з них, в яких поряд з візуальним підтримується командний інтерфейс, можливе створення мережної надбудови.

Логічним наступним кроком буде створення оболонки, що інтегрує в собі послуги різних СКМ за допомогою клієнт-серверних технологій, надаючи користувачеві рівний доступ до різних СКМ за допомогою єдиної командної мови (а за необхідності – легко переходити до мови будь-якої СКМ). Тоді на клієнтському комп'ютері не буде потреби у встановленні та налагодженні спільного функціонування різних СКМ – достатньо звернутися до математичного сервера засобами Web-браузера.

Новим перспективним напрямом розвитку СКМ є мобільні математичні середовища; перші представники таких систем з'явились лише на початку XXI століття. Інтеграція СКМ у єдине мережне середовище – найкраща ілюстрація сучасної концепції «комп'ютер – це мережа», що знаходить своє відображення у перенесенні прикладного ПЗ (навіть «робочих столів») у Web-середовища.

Для користувача за рахунок цього надається можливість мобільного доступу до програм та даних, для адміністратора комп'ютерного класу знімаються проблеми підтримки великої інсталяційної бази та ліцензування ПЗ, для викладачів – суттєво розширюється спектр використовуваного ПЗ, а для учнів та студентів створюються умови для реалізації мобільного навчання.

Мобільне математичне середовище (ММС) – мережне програмне забезпечення, за допомогою якого надається можливість доступу до математичних об'єктів в будь-який зручний час та у будь-який спосіб.

Важливим є те, що застосування мобільних Web-середовищ у навчальному процесі надає можливість інтегрувати аудиторну й позааудиторну роботу у безперервний навчальний процес. До визначальних характеристик ММС відносяться:

– виконуваність на широкому спектрі комп'ютерних пристроїв, що дозволяє залучити до навчального процесу мультимедійні панелі, нетбуки та смартфони;

– виконання та зберігання математичних об'єктів на Інтернет-серверах, що дозволяє надати уніфікований мережний доступ до навчальних матеріалів як в аудиторії, так і поза її межами;

– можливість природного застосування ефективних педагогічних технологій організації спільної роботи (соціального конструктивізму, коннективізму та конструкціонізму) над навчальними проектами у навчальних спільнотах (як аудиторних, так і розподілених);

– можливість організації в межах одного середовища повного циклу навчання: а) зберігання та подання навчальних матеріалів; б) математичних досліджень; в) індивідуальної та колективної роботи; г) оцінювання навчальних досягнень.

Яскравим представником ММС є Web-СКМ Sage, що виступає в якості інтегратора різних математичних пакетів із забезпеченням спільного Web-інтерфейсу. За допомогою Sage можна:

1) виконувати будь-які обчислення, як аналітичні (дії з алгебраїчними виразами, розв'язування рівнянь, диференціювання, інтегрування тощо), так і чисельні (точні – з будь-якою розрядністю, наближені – з будь-якою, наперед заданою точністю);

2) подавати результати обчислень у зручній для сприйняття формі, будувати дво- та тривимірні графіки кривих та поверхонь, гістограми та будь-які інші зображення (в тому числі анімаційні);

3) поєднувати обчислення, текст та графіку на робочих листах з можливістю їх друкування, оприлюднення в мережі та спільної роботи над ними;

4) створювати за допомогою вбудованої у Sage мови Python моделі для виконання навчальних досліджень;

5) описувати нові функції та класи мовою Python [16].

Розвинена функціональність Sage забезпечується широким застосуванням технології AJAX, що є основою більшості продуктів Web 2.0.

Починаючи з 2008 р. Sage використовується у ВНЗ м. Кривого Рогу в процесі навчання теорії алгоритмів, моделювання, методів оптимізації, чисельних методів, теорії кодування, паралельних та розподілених обчислень, розпізнавання образів та інших навчальних дисциплін.

Також Sage може бути використаний в процесі навчання елементарної та вищої математики, у тому числі лінійної та вищої алгебри, геометрії, математичного аналізу, методів математичної фізики, теорії чисел, комбінаторики, теорії графів та багатьох розділів математичної інформатики.

Так, до основних напрямків застосування Sage в процесі навчання вищої математики відноситься:

– графічна інтерпретація математичних моделей та теоретичних понять;
– автоматизація рутинних обчислень;
– підтримка самостійної роботи;
– організація математичних досліджень.

Для їх реалізації було створено мобільний курс вищої математики, визначальною особливістю якого є динамічна природа навчальних матеріалів: будь-який опублікований у мережі об'єкт може автоматично змінюватися у відповідності до зміни вмісту пов'язаного з ним робочого аркуша; зміни програмного забезпечення, що входить до складу ММС; зміни пристрою доступу до навчальних матеріалів; зміни початкових умов для моделей тощо.

Для первинної апробації мобільного курсу вищої математики було створено дистрибутив, який, крім лекцій, практикуму та моделей, містить локалізовану версію ММС Sage, посібник «Основи роботи з Sage» та відеоуроки (рис. 4).

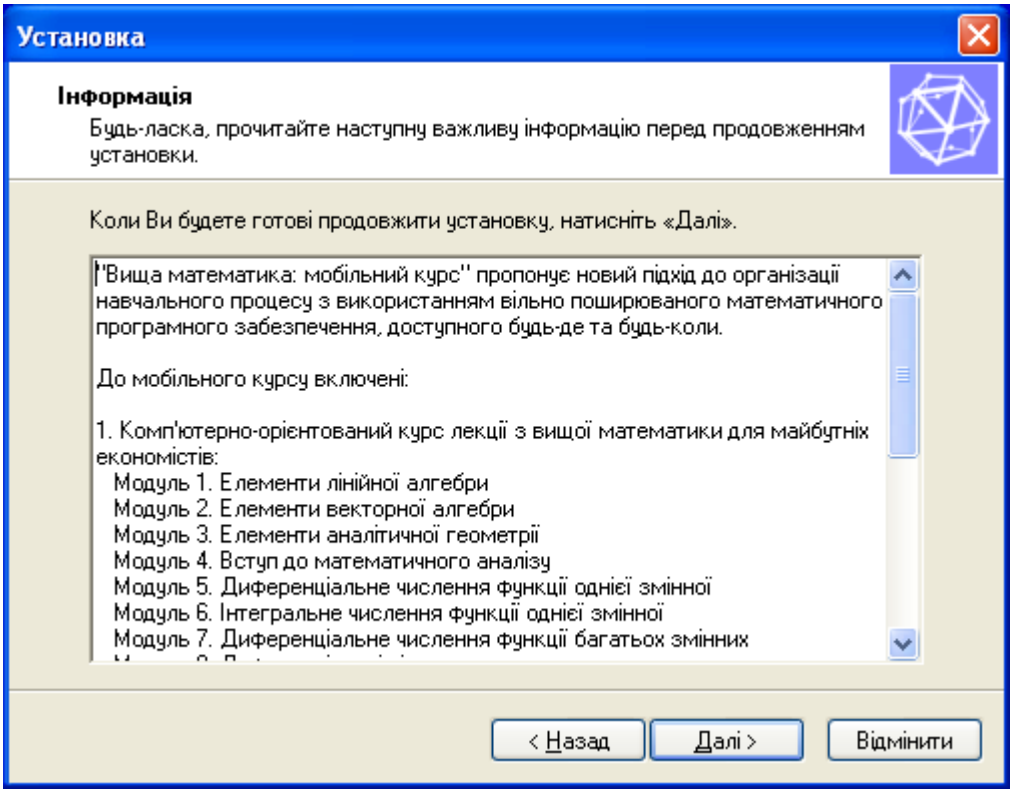


Рис. 4. Інсталяція ППЗ «Вища математика: мобільний курс»

З метою популяризації ММС Sage у вітчизняній системі освіти на базі Криворізького металургійного факультету Національної металургійної академії України організовано відкритий доступ до локалізованої версії Sage за адресою http://korpus21.dyndns.org:60000.

Висновки:

1. Фундаменталізація навчання інформатичних дисциплін вимагає стабілізації засобів навчання через надання їм властивостей відкритих систем. Стабілізація програмних засобів надає широкі можливості для варіювання програмного забезпечення навчання інформатичних дисциплін замість штучної прив'язки до окремих програмних продуктів. До стабільного програмного забезпечення навчання інформатичних дисциплін у вищій школі відносяться мобільні операційні системи, мобільні компілятори, мобільні інтерпретовані мови програмування та мобільні Web-середовища.

2. Один із загальноприйнятих способів підвищення мобільності програмного забезпечення – стандартизація програмного оточення: програмних інтерфейсів, утиліт тощо. На рівні системних сервісів подібне оточення описується в стандарті POSIX, підтримка якого полегшує перенесення прикладних програм практично на будь-яку скільки-небудь поширену операційну платформу. Мобільність програм, що відповідають стандарту POSIX, принципово досяжна завдяки наявності великої кількості стандартизованих системних сервісів та можливості динамічного з'ясування характеристик цільової платформи й налаштовування програми під них. POSIX-сумісність є засобом уніфікації операційних систем, а дотримання стандартів POSIX при розробці програмного забезпечення – засобом уникнення залежності від використовуваної операційної системи.

3. Застосування мобільних компіляторів є засобом уникнення залежності від використовуваних середовища програмування (через потужний інтерфейс командного рядка та легкість інтеграції у IDE), операційної системи (через забезпечення POSIX-сумісності) та мови програмування (через надання спільних бібліотек).

4. Застосування мобільних інтерпретованих мов загального призначення є засобом забезпечення мобільності програм, створених на POSIX-несумісних платформах.

5. Використання мобільних пристроїв з невисокою швидкодією та малим обсягом оперативної пам'яті суттєво ускладнює застосування таких ресурсоємних програм, як середовища програмування, системи комп'ютерної математики і т.п. Для розв'язання цієї проблеми доцільно перейти до

мережецентричної моделі, за якої ресурсоємні програми працюють на Інтернет-серверах, а головною клієнтською програмою виступає Web-браузер. Перенесення прикладного програмного забезпечення у Web-середовище (онлайн-IDE, Web-СКМ ті ін.) створює нові можливості для обміну навчальними матеріалами та організації співробітництва між усіма учасниками навчального процесу:

– для будь-якого користувача за рахунок цього надається можливість мобільного доступу до програм та даних;

– для адміністратора комп'ютерного класу знімаються проблеми підтримки великої інсталяційної бази та ліцензування програмного забезпечення;

– для викладачів суттєво розширюється спектр використовуваного програмного забезпечення, а для студентів – використовуваних засобів мобільного навчання.

Література

1. Семеріков С.О. Фундаменталізація навчання інформатичних дисциплін у вищій школі: монографія / Семеріков С.О.; науковий редактор академік АПН України, д.пед.н., проф. М. І. Жалдак. – Кривий Ріг: Мінерал; К.: НПУ ім. М.П. Драгоманова, 2009. – 340 с.: іл.
2. Хоменко Л.Г. История отечественной кибернетики и информатики : монография / Хоменко Л.Г. – К.: Ин-т кибернетики им. В.М. Глушкова НАН Украины, 1998. – 455 с.
3. Компьютерная технология обучения : словарь-справочник / Под редакцией Гриценко В.И., Довгялло А.М., Савельева А.Я. – К. : Наукова думка, 1992. – 650 с.
4. Diehl, S. Distributed virtual worlds: Found. a. implementation techn. using VRML, Java, a. CORBA / Stephan Diehl. – Berlin [etc.]: Springer, Cop. 2001. – XII+166 p.
5. Таненбаум Э. Операционные системы: разработка и реализация [+CD] / Таненбаум Э., Вудхалл А. – 3-е изд. – СПб.: Питер, 2007. – 704 с.: ил. – (Классика CS)
6. Гриффитс А. GCC. Настольная книга пользователей, программистов и системных администраторов / Гриффитс А. – К.: Диасофт, 2004. – 624 с.
7. Полищук А.П. Программирование в X Window средствами Free Pascal [Электронный ресурс]: учебное пособие / Полищук А. П., Семериков С. А. – Кривой Рог: Издательский отдел КГПУ, 2005. – 128 с. – Режим доступа: http://rus-linux.net/MyLDP/BOOKS/ProgrX/xwin-contents.shtml
8. Полищук А. П. Системное программирование в UNIX средствами Free Pascal [Электронный ресурс] / Полищук А. П., Семериков С. А. – Кривой Рог : Издательский отдел КГПУ, 2005. – 418 с. – Режим доступа : http://www.interface.ru/home.asp?artId=1617
9. Sussman G.J., Steele G.L.Jr. Scheme: An Interpreter for Extended Lambda Calculus // MIT Artificial Intelligence Memo 349. – December 1975. – Cambridge: MIT, 1975.
10. Мінтій І.С. Математичні основи функціонального підходу /І. С. Мінтій // Розвиток інтелектуальних умінь і творчих здібностей учнів та студентів у процесі навчання математики: матеріали Всеукр. наук.-метод. конф. (3-4 грудня 2009 р., м. Суми). – Суми: Вид-во СумДПУ імені А.С. Макаренка, 2009. – С. 219-220.
11. Times Higher Education-QS World University Rankings 2009. [Electronic resource] – Mode of access: http://www.timeshighereducation.co.uk/Rankings2009-Top200.html
12. Schools Using Scheme. [Electronic resource] – Mode of access: http://www.schemers.com/schools.html
13. Schindler, E. Gartner Explains Why Windows Is Broken [Electronic resource] / Esther Schindler. – 2008. – April, 09 – Mode of access: http://www.cio.com/article/331963/Gartner_Explains_Why_Windows_Is_Broken
14. Кобильник Т. П. Методична система навчання математичної інформатики у педагогічному університеті: автореф. дис... канд. пед. наук : 13.00.02 – теорія та методика навчання (інформатика) / Кобильник Тарас Петрович; Національний педагогічний ун-т ім. М.П. Драгоманова. – К., 2009. – 20с.
15. Триус Ю. В. Комп'ютерно-орієнтовані методичні системи навчання математичних дисциплін у вищих навчальних закладах: дис. ... доктора пед. наук: 13.00.02 / Триус Ю.В.; Черкаський нац. ун-т ім. Б. Хмельницького. – Черкаси, 2005. – 649 с.
16. Інноваційні інформаційно-комунікаційні технології навчання математики: навчальний посібник / В. В. Корольський, Т. Г. Крамаренко, С. О. Семеріков, С. В. Шокалюк; науковий редактор академік АПН України, д.пед.н., проф. М. І. Жалдак. – Кривий Ріг: Книжкове видавництво Кирєєвського, 2009. – 316 с.