Учредитель:
КрТО МАКНС

14'04

Академический вестник

Криворожского территориального отделения Международной Академии компьютерных наук и систем (КрТО МАКНС)

Журнал зарегистрирован
Министерством информации Украины
Регистрационный номер № 3020
от 26.01.1998 г.
Издается на украинском и русском языках.
Печатается по решению Ученого Совета
Криворожского технического университета и
бюро КрТО МАКНС

Адрес редакции
50027, г.Кривой Рог,
ул.XXII партсъезда, 11

Тел. (0564) **74-14-35**
71-93-87
71-93-83
Факс **29-19-91**

Издатель:
КрТО МАКНС

Директор издательства
Назаренко М.В.

Выпускающий редактор
Марусич Ю.Ю.

Художественное оформление и компьютерная верстка
Марусич И.В.

Колонка редакции

Редакция предлагает заинтересованным лицам и организациям присылать научные и рекламные материалы для публикации в нашем журнале.

СОДЕРЖАНИЕ

АВТОМАТИЗАЦІЯ ВЕЛИКИХ СИСТЕМ, ПРОГРЕСИВНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА ПРОБЛЕМИ ШТУЧНОГО ІНТЕЛЕКТУ В ГІРНИЧІЙ СПРАВІ

НОВІ ПІДХОДИ В НАВЧАННІ ТА ВИХОВНОМУ ПРОЦЕСІ

ПРОБЛЕМИ ОХОРОНИ ІНТЕЛЕКТУАЛЬНОЇ ВЛАСНОСТІ В СФЕРІ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

навчання. Збірник матеріалів Всеукраїнської студентської науково-практичної конференції. – Кіровоград: РВГІЦ КДПУ ім. В. Винниченка. – 1999. – С.36-37.
2. Полищук А.П., Семериков С.А. Методы вычислений в классах языка С++: Учебное пособие. – Кривой Рог: Издательский отдел КГПИ, 1999. – 350с.
3. Полищук А.П., Семериков С.А., Грищенко Н.В. О выборе языка программирования для начального обучения //Комп'ютерне моделювання та інформаційні технології в природничих науках. – Кривий Ріг: Видавничий відділ КДПУ, 2000. – С. 212-228.
4. Семеріков С.О. Застосування об'єктно-орієнтованого програмування до викладання курсу чисельних методів //Математика, її застосування та викладання: Матеріали міжвузівської регіональної наукової конференції. – Кіровоград: РВГ ІЦ КДПУ ім. В. Винниченка. – 1999. – С.134-137.
5. Семеріков С.О. Чисельні методи: об'єктний підхід //Комп'ютерно-орієнтовані системи навчання. Збірник наукових праць. Випуск 2. – К.: НПУ ім. М.П. Драгоманова, 2000. – С.122-128.

УДК 378.147+518.5

Принципи застосування об'єктного підходу до розробки математичного програмного забезпечення

Семеріков С.О., канд. пед. наук, доц.
(КДПУ, м.Кривий Ріг)

В статті викладено основні принципи об'єктного підходу та його реалізація у об'єктно-орієнтованому програмуванні, показана природна спільність математичних та програмних об'єктів, наведено рекомендації щодо розробки математичного програмного забезпечення.

«Наука про те, як замість лише швидкодії машини використовувати чисельні методи та бібліотечні програми, переживає період дитинства і є однією з найважливіших областей дослідження у майбутньому» [1, с.198]. Ці слова Р.В.Хеммінга, сказані майже 40 років тому, і досі не втрачають своєї актуальності. Тенденція до створення та систематичного використання математичних бібліотек, що виникла на початку 60-х рр., наприкінці 80-х рр. стає домінуючою. Проте розвиток чисельних методів зумовлений сьогодні також розвитком засобів обчислювальної техніки та технології програмування. Так, поява паралельних ЕОМ породила новий клас чисельних методів, орієнтованих на паралельні обчислення, а розвиток об'єктно-орієнтованого програмування – новий напрямок: Object-oriented numerics (OON), або об'єктні обчислення, який зараз є провідним напрямком у чисельних методах [2].

Об'єктно-орієнтоване програмування (ООП), що отримало широке розповсюдження як потужна програмна технологія, є у наш час вагомою альтернативою традиційним процедурним методам програмування. Популярність ООП у чималій мірі визначається концептуальною цілісністю та більш сильною формою структуризації програмного забезпечення (ПЗ), що створюється на його основі. Використання ООП прискорює процес розробки програм, даючи при цьому можливість гнучкої та природної модифікації існуючого ПЗ. Найбільш рельєфно можливості ООП проявляються при створенні досить складних програмних продуктів, до яких, зокрема, відносяться проблемно-орієнтовані бібліотеки.

Перш ніж розглядати ООП, доцільніше спочатку розглянути його підґрунтя – об'єктний підхід, який є більш загальною технологією дослідження та пізнання, як це пропонує А.П.Єршов. У своїй роботі «Про об'єктно-орієнтовану взаємодію з ЕОМ» він надає об'єктно-орієнтованому програмуванню більш широкий зміст, ніж програмуванню лише з використанням об'єктно-орієнтованих мов. У якості одного з прикладів об'єктно-орієнтованої взаємодії програмуючого користувача з ЕОМ А.П.Єршов посилається на Е-практикум як реальну систему автоматизованого конструювання програм, особливо підкреслюючи при цьому тезу про перспективність та універсальність об'єктно-орієнтованої взаємодії. Отже, розглянемо основні передумови впровадження об'єктного підходу у практику викладання чисельних методів, і почнемо з психологічних передумов.

Основне утруднення, що постає перед розробником складного програмного продукту, полягає в неможливості одночасно утримувати в пам'яті усі необхідні деталі. Г.Р.Міллер та його послідовники стверджують, що максимальна кількість об'єктів, з якою здатен одночасно оперувати людський мозок, не перевищує 7±2 (так званий «гаманець Міллера»). Це «магічне число», скоріше за все, пов'язане з обсягом короткострокової пам'яті у людини. Ще одним обмежуючим фактором тут виступає швидкість опрацювання мозком нової інформації: йому потрібно приблизно 5 секунд на сприймання кожного нового об'єкту. Як бачимо, природна здатність людського мозку до роботи із складними системами є низькою.

Проте, як услід за Е.Дейкстрою зазначає Б.Страуструп, ще з давніх давен людству відомий простий та ефективний спосіб управління складними системами: «Розділяй та володарюй». Тому при проектуванні складної системи (програми) необхідно складати її з окремих невеликих підсистем (підпрограм) – у цьому випадку ми не виходимо за межі можливостей людини: при розробці будь-якого рівня системи необхідно одночасно утримувати в пам'яті інформацію лише про деякі її частини.

Такий підхід називається алгоритмічною декомпозицією і забезпечує психологічне підґрунтя для процедурного програмування, визначаючи головну вимогу до написання підпрограми: «усі дії, що виконуються в підпрограмі, повинні усвідомлюватися одночасно», і якщо ця вимога не виконується, підпрограму слід поділити на дрібніші блоки. До того ж у шкільному віці «гаманець Міллера» має менші розміри, тому при вивченні основ алгоритмізації поняття допоміжного алгоритму (процедури) та алгоритмічної декомпозиції, на наш погляд, повинні вводитися одними з перших, а самі процедури не повинні містити більше 4–5 усвідомлюваних дій.

Проте число подій, що одночасно може опрацювати людина, не залежить від обсягу інформації, що міститься у кожній події, і це дає людині надзвичайно ефективний механізм опрацювання складних повідомлень – абстрагування. Не маючи можливості відтворити у всіх деталях складний об'єкт, ми ігноруємо несуттєві для нас деталі і, таким чином, маємо справу з узагальненою, ідеалізованою моделлю об'єкта. І хоча при цьому ми, як і раніше, змушені охоплювати одночасно значну кількість властивостей об'єкту, та завдяки абстракції ми використовуємо узагальнені властивості суттєво більшого семантичного обсягу. Це особливо вірно, коли ми розглядаємо світ з позицій об'єктно-орієнтованої взаємодії, оскільки об'єкти як абстракції реального світу являють собою насичені зв'язні інформаційні одиниці. При цьому ми також обмежені кількістю об'єктів, яку можемо сприйняти у кожний окремий момент, все одно, використовуючи абстрактні поняття, ми отримуємо можливість працювати із складними системами, а, отже, і створювати складні програмні продукти.

Складні системи можна досліджувати, концентруючи основну увагу або на об'єктах, що фігурують у системі, або на процесах, що протікають в ній. Проте доцільніше розглядати систему як впорядковану сукупність об'єктів, які в процесі взаємодії один з одним забезпечують функціонування системи як єдиного цілого. Об'єкти, що складають систему, можуть утворювати ієрархії. При такому підході основним способом дослідження складної системи є об'єктна декомпозиція.

Таким чином, з'являється можливість розширити межі когнітивних можливостей людини, використовуючи методи декомпозиції, виділення абстракцій та створення ієрархій. Саме ці методи покладено в основу об'єктного підходу, який утворює концептуальний базис об'єктно-орієнтованої методології. (Тут під методологією ми розуміємо сукупність методів, що застосовуються при розробці програм і об'єднаних одним загальним філософським підходом.)

Реалізацією об'єктно-орієнтованої методології дослідження складних систем є об'єктно-орієнтоване програмування – методологія програмування, заснована на представленні програми у вигляді сукупності об'єктів, кожен з яких є реалізацією деякого класу, а класи утворюють ієрархію за принципами наслідуваності.

Таким чином, об'єктно-орієнтоване програмування є найбільш природною методологією програмування, яка, враховуючи особливості психічних процесів, дає можливість створювати чітко структуровані та досяжні складні програмні продукти.

Об'єктний підхід відомий ще з давніх часів. Так, давнім грекам належить ідея про те, що світ можна розглядати як у термінах об'єктів, так і подій. А у XVII ст. Декарт підкреслював, що люди зазвичай мають об'єктно-орієнтований погляд на світ. У 60–70-х р.р. XX ст. ця думка була розвинена в одній з течій когнітивної філософії – об'єктивістській епістемології (Е.Ранд), а на початку 80-х р.р. М.Мінскі запропонував модель людського мислення, у якій розум лю-

дини розглядається як спільнота агентів, що по-різному мислять. На його думку, лише спільні дії таких агентів приводять людину до осмисленої поведінки.

Основні ідеї об'єктного підходу (абстрагування, типізація, ієрархія тощо) у тому чи іншому вигляді були присутні у практиці програмування, починаючи з перших мов високого рівня. Одну з можливих класифікацій мов високого рівня наведено у табл. 1.

Табл.1. Класифікація мов програмування високого рівня (за П.Вегнером, [3, с.44])

Генерація	*Основні мови*	*Особливості, нові конструкції*
Перша (1954–1958)	Fortran I	Математичні формули
	Algol–58	Математичні формули
	Flowmatic	Математичні формули
	IPL V	Математичні формули
Друга (1959–1961)	Fortran II	Підпрограми, роздільна компіляція
	Algol–60	Блочна структура, типи даних
	Cobol	Опис даних, робота з файлами
	Lisp	Опрацювання списків, вказівників, збирання сміття
Третя (1962–1970)	PL/1	Fortran+Algol+Cobol
	Algol–68	Більш строгий спадкоємець Algol–60
	Pascal	Більш простий спадкоємець Algol–60
	Simula	Класи, абстрактні дані

У кожній наступній генерації змінювалися підтримувані мовами механізми абстракції. Мови першої генерації орієнтувалися виключно на науково-інженерне застосування, і словник цієї предметної галузі був майже виключно математичним. Такі мови, як Fortran I, були створені для спрощення програмування математичних формул, щоб звільнити програміста від труднощів асемблера та машинного коду. Перша генерація мов високого рівня була кроком, що наближує програмування до предметної галузі та віддаляє його від конкретної машини.

Характерною тенденцією мов першої та початку другої генерації став розвиток алгоритмічних абстракцій. У цей час потужність комп'ютерів швидко росла, що дозволяло розширити межі їх застосування. Можна відмітити, що для таких мов, як Fortan та Cobol, основним будівельним блоком є підпрограма. Програми, написані такими мовами, мають відносно просту структуру, що складається лише з глобальних даних та підпрограм. У процесі розробки можна логічно упорядкувати різнотипні дані, проте механізми цих мов практично не підтримують такого упорядкування. Помилка у будь-якій частині програми може мати глобальні наслідки, тому що область даних відкрито усім підпрограмам. В процесі розробки програм цими мовами вже через короткий час виникає плутанина, викликана великою кількістю перехресних зв'язків між підпрограмами, заплутаними схемами управління, що знижує надійність та ясність програми.

Починаючи з середини 60-х рр. поступово усвідомлюється роль підпрограм як програмної (процедурної) абстракції. Погляд на підпрограми як механізм абстрагування дозволив розробити мови, що підтримували різні механізми передавання параметрів. Були закладені основи структурного програмування, що відобразилося у мовній підтримці механізмів вкладеності підпрограм та науковому дослідженні структур управління та областей видимості, виникли методи структурного проектування, які стимулювали розробляти програми, використовуючи підпрограми як готові будівельні блоки.

Наприкінці 60-х рр. з появою транзисторів, а потім інтегральних схем, вартість комп'ютерів різко знизилася, а їх потужність виросла майже експонентно. З'явилася можливість розв'язувати все більш складні задачі, проте це вимагало вміння обробляти найрізноманітніші типи даних. Такі мови, яка Algol–68, а далі Pascal стали підтримувати абстракцію даних у явному вигляді. Виразом потреби у незалежній розробці окремих частин задачі став окремо компільований модуль, який спочатку був просто більш-менш випадковим набором даних та підпрограм. У такі модулі збиралися підпрограми, які, як уявлялося, скоріше за все будуть змінюватися сумісно, і мало хто розглядав їх як нову техніку абстракції. У більшості мов третьої генерації, хоча і підтримувалося модульне програмування, проте не вводилося жодних правил, що регламентували б узгодження інтерфейсів модулів. Програміст, наприклад, написавши підпрограму в одному з модулів,

міг очікувати, що її будуть викликати із трьома параметрами: дійсним числом, десятиелементним масивом та логічною змінною. Але у якомусь іншому модулі ця підпрограма могла помилково викликатися з фактичними параметрами у вигляді: ціле число, п'ятиелементний масив, дійсне число. На жаль, оскільки більшість мов надавали у кращому випадку рудиментарну підтримку абстрактних даних та типів, такі помилки виявлялися лише при виконанні програм.

Абстрагування, що досягається за допомогою процедур, добре підходить для опису абстрактних дій, проте не годиться для опису абстрактних об'єктів. Усвідомлення цього дало два важливих напрямки, що розвивалися у 70-х рр. По-перше, виникають методи проектування на основі потоків даних, які вносять упорядкування в абстракцію даних у мовах, орієнтованих на алгоритми. По-друге, з'являється теорія типів, яка втілюється у таких мовах, яка Pascal. У 70-ті рр. було створено більше двох тисяч різних мов та їх діалектів, та лише деякі з них збереглися до нашого часу. Проте багато принципів, розвинених у попередніх мовах програмування, знайшли своє адекватне відображення у нових. Так, наприклад, мова Smalltalk є спадкоємцем мови Simula, Ada – спадкоємець Algol–68 та Pascal з елементами Simula, Alphard та Clu, Clos об'єднав Lisp, Loops та Flavors, C++ виник від поєднання C та Simula, Eiffel – від Simula та Ada. Ці мови отримали назву об'єктних та об'єктно-орієнтованих, і саме цей клас мов являтиме для нас подальший інтерес.

Таким чином, класифікацію генерацій мов програмування, подану у табл.1, можна переформулювати з позицій підтримуваних ними абстракцій: математичні, алгоритмічні, орієнтовані на дані та об'єктно-орієнтовані.

У свою чергу, об'єктно-орієнтовані мови також можна класифікувати за різними ознаками. Дж.Саундерс поділяє їх на 7 таких категорій:

1. Actor-подібні – мови, що підтримують механізм делегування.
2. Паралельні – мови, що реалізують паралелізм.
3. Розподілені – мови, націлені на опрацювання розподілених об'єктів.
4. Фреймові – мови, що реалізують теорію фреймів.
5. Гібридні – об'єктно-орієнтовані надбудови над звичайними мовами.
6. Smalltalk-подібні – мова Smalltalk та її діалекти.
7. Ідеологічні – мови, орієнтовані на сферу застосування.
8. Інші – мови, які не відносяться до жодної з категорій.

У відповідності до введеного нами визначення не всі мови програмування є об'єктно-орієнтованими. Автор об'єктно-орієнтованої мови програмування C++ Б.Страуструп зазначає, що «… якщо термін об'єктно-орієнтована мова взагалі щось означає, то він повинен означати мову, що має засоби гарної підтримки об'єктно-орієнтованого стилю програмування… Забезпечення такого стилю у першу чергу означає, що у мові зручно використовувати такий стиль. Якщо написання програм у стилі ООП вимагає спеціальних зусиль чи воно неможливо зовсім, то ця мова не відповідає вимогам ООП». Теоретично можлива імітація об'єктно-орієнтованого програмування навіть мовою асемблера, проте це надзвичайно важко. За Вегнером, мова програмування є об'єктно-орієнтованою тоді і лише тоді, коли виконуються такі умови:

- підтримуються об'єкти, тобто абстракції даних, що мають інтерфейс у вигляді іменованих операцій та власні дані, з обмеженням доступу до них;
- об'єкти відносяться до відповідних типів (класів);
- типи (класи) можуть наслідувати атрибути супертипів (суперкласів).

Таким чином, можна дати два такі означення:

Об'єктними називають мови, які підтримують абстракцію даних та класи.

Об'єктно-орієнтованими називають ті об'єктні мови, які підтримують наслідування та поліморфізм.

Побудову спеціалізованих математичних бібліотек, зокрема – бібліотеки класів для підтримки курсу чисельних методів, починають з етапу проектування.

Об'єктно-орієнтоване проектування – це методологія проектування, що поєднує у собі процес об'єктної декомпозиції та прийоми подання логічної та фізичної, а також статичної та динамічної моделей проектованої системи.

Саме об'єктно-орієнтована декомпозиція відрізняє об'єктно-орієнтоване проектування від структурного; у першому випадку логічна структура системи відображається абстракціями у вигляді класів і об'єктів, у другому – алгоритмами. Об'єктно-орієнтований аналіз спрямований на створення моделей реальної дійсності на основі об'єктно-орієнтованого світогляду.

Об'єктно-орієнтований аналіз – це методологія, при якій вимоги до системи

сприймаються з точки зору класів та об'єктів, виявлених у предметній галузі.

Між всіма цими визначеннями існує тісний взаємозв'язок: на результатах об'єктно-орієнтованого аналізу формуються моделі, на яких ґрунтується об'єктно-орієнтоване проектування; у свою чергу, об'єктно-орієнтоване проектування створює фундамент для фінальної реалізації системи з використанням методології ООП.

Проведений нами аналіз дозволяє зробити висновок про те, що найбільш придатною мовою для навчання програмуванню у об'єктній методології є мова C++. Ця мова є гібридною, її ядро – широко відома мова програмування C, що дозволяє організувати неперервне навчання студентів процедурній та об'єктній методології у межах однієї мови. Спочатку вивчається та підмножина C++, що містить основні конструкції мови C, а далі на вивченому базисі надбудовуються компоненти, що надають цій мові об'єктно-орієнтованих властивостей. Такий підхід, зокрема, застосовано у навчальному посібнику [4].

В.А.Семенов виділяє такі головні можливості, що їх надає об'єктний підхід при розробці математичного програмного забезпечення:

1. Інкапсуляція чисельних методів у класах математичних об'єктів є більш сильною формою структуризації обчислювальних модулів, що підвищує концептуальну наочність застосовуваних чисельних підходів та регламентує коректну дисципліну роботи з даними об'єктів без порушення їх цілісності.
2. Розробник прикладного ПЗ виступає у якості користувача бібліотечних класів і замість підтримки внутрішніх даних може сконцентрувати зусилля на предметному аспекті розроблюваних програм (приймаючи до уваги складність динамічно розміщуваних даних математичних об'єктів, зокрема, розріджених матричних об'єктів, зазначена перевага є більш ніж помітною).
3. Механізм захисту даних, що забезпечує об'єктна технологія, сприяє підвищенню надійності розроблюваних програм.
4. Наслідування математичних об'єктів та поліморфізм чисельних методів, інкапсульованих ними, надає можливість гнучкої модернізації і розвитку математичного забезпечення як з урахуванням проблемної орієнтації, так і у відповідності до детальної класифікації об'єктів за їх математичними та обчислювальними ознаками і властивостями.
5. Реалізація чисельних методів у вигляді наслідуваних методів споріднених математичних класів дозволяє природним чином виразити міру спільності математичних понять та ступінь універсальності застосування тих чи інших чисельних підходів у кожному конкретному випадку.

Таким чином, математична об'єктно-орієнтована бібліотека може розглядатися у якості базового інструментального середовища для програмування чисельних методів.

При цьому, по-перше, відпадає необхідність починати нову програму з нуля: базові типи векторів, поліномів, матриць тощо, оформлені у вигляді відповідних класів, можуть бути використані поряд із вбудованими числовими типами. Інструментальний характер базових класів тим більше усвідомлюється за необхідності перевизначення методу чи групи методів або породження нового, більш спеціалізованого типу даних.

По-друге, нові типи даних, що виражають потужні концептуальні поняття, утворюють своєрідну мову, максимально наближену до природної математичної, використовуючи яку можна програмувати чисельні методи у загальноприйнятих термінах та позначеннях.

По-третє, побудована система класів може бути доповнена та розширена у відповідності до предметної галузі, в якій її планується застосовувати. Специфічна для проблемної області система понять, властивостей, відношень знаходить своє відображення в системі класів, побудованій за принципом ієрархічної деталізації. Так, розроблена нами бібліотека математичних класів знайшла застосування не лише в курсі чисельних методів, а й в курсах автоматики, фізики твердого тіла, електротехніки та інших.

Застосування об'єктного підходу до розробки математичного програмного забезпечення передбачає перш за все проведення попереднього об'єктно-орієнтованого аналізу розглядуваної предметної галузі. Результатом такого аналізу повинна стати система узагальнень, що виражає основні предметні поняття, їх властивості та встановлює між ними необхідні класифікаційні відношення.

Складовою частиною об'єктно-орієнтованого аналізу є виявлення усіх можливих співвідношень наслідування між об'єктами, за яких множина класів може бути подана ієрархією споріднених об'єктів. В основі побудови такої ієрархії лежить та чи інша

система ознак, що ідентифікує кожен об'єкт у відповідності до обраної класифікації. Оскільки способи виділення таких ознак можуть варіюватися у широких межах, при побудові ієрархій математичних класів будемо виходити з конструктивних властивостей об'єктів – в такому випадку будь-який обчислювальний алгоритм, який можна застосувати до об'єктів деякого математичного класу, також може бути застосований і до об'єктів будь-якого його нащадка, причому з високою ефективністю на множинах еквівалентних задач, що різняться лише типами математичних об'єктів.

Разом з тим, кожного разу, коли наявність спеціальних математичних властивостей об'єкта – класифікаційних критеріїв – допускає застосування більш ефективного обчислювального методу, відповідний метод загального класу може бути перевизначений. Це стає тим більш виправданим, якщо математичні особливості частинного об'єкта дозволяють замінити обчислювальну процедуру відповідним аналітичним перетворенням.

Таким чином, наслідування і поліморфізм чисельних методів, інкапсульованих у відповідних математичних класах, забезпечує компроміс між необхідністю мати надійне, функціонально повне і уніфіковане алгоритмічне ядро та можливість заміщення універсальних методів на частинні реалізації.

Список літератури.

1. Хемминг Р.В. Численные методы для научных работников и инженеров. – М.: Наука, 1968. – 400с.
2. Sullivan S.J., Zorn B.G. Numerical Analysis Using Nonprocedural Paradigms //ACM Transactions on Mathematical Software. – 1995. – Vol. 21, No. 3. – P.267-298.
3. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++. – 2-е изд. / Пер. с англ. – М.: Бином, СПб.: Невский диалект, 1999. – 560с.
4. Полищук А.П., Семериков С.А. Методы вычислений в классах языка C++: Учебное пособие. – Кривой Рог: Издательский отдел КГПИ, 1999. – 350 с.

УДК 371

Информационная культура и информационная безопасность как факторы выживания в информационном обществе

Теплицкий И.А., канд. пед. наук, доц.
(КГПУ, г.Кривой Рог)

В статье рассматривается понятие информационного общества, раскрываются его сущностные признаки, выделяются различные аспекты информационного воздействия на человека и социум. Особое внимание уделено формированию понятий информационной культуры и информационной безопасности у студентов компьютерных специальностей.

В наступившую постиндустриальную эпоху индустрия информационных технологий органически входит во все сферы человеческой деятельности. Информационные технологии приходится не только использовать, но и жить, сотрудничать и конкурировать с ними. «При этом под информационной технологией понимается совокупность метолов и технических средств сбора, организации, хранения, обработки, передачи и представления информации, расширяющая знания людей и развивающая их возможности по управлению техническими и социальными процессами» [2, с.4].

Однако развитие информационных сетей, процесс превращения граждан в пользователей всемирных сетей далеко не всегда способствуют изменению личностных форм бытия людей, их самореализации. «Существенным компонентом компьютеризации и информатизации является дополнительная форма отчуждения человеческого знания, его активизация и использование как непосредственной производительной силы в виде программного обеспечения и машинных банков данных и знаний» [1, с.9].

Влияния глобальных сетей в информационном обществе становится настолько интенсивным, что естественным образом возникают вопросы:

1. Возможна ли свобода личности в информационном обществе?