УДК 37.016.09

КРАСА ПРОСТИХ ЗАДАЧ АБО ДО ПИТАННЯ ПРО ВИКОРИСТАННЯ МОВ ПРОГРАМУВАННЯ У НАВЧАННІ ШКОЛЯРІВ ІНФОРМАТИКИ

Білоусова Людмила Іванівна,

зав. кафедри інформатики Харківського державного педагогічного університету імені Г.С. Сковороди, кандидат фізико-математичних наук, професор, lib215@list.ru.

Анотація. У статті розкривається значимість навчання школярів основ алгоритмізації і програмування; на конкретних прикладах висвітлено ті можливості реалізації провідних цілей навчання інформатики, які надає належна організація роботи над задачею на розробку алгоритму.

Ключові слова: інформатика, основи алгоритмізації та програмування, навчання школярів, задачі на розробку алгоритму, робота над задачею.

Навчання школярів основ алгоритмізації і програмування є одним з найважливіших завдань курсу інформатики, оскільки це дає змогу ознайомити учнів із високозатребуваною на ринку праці професією і зорієнтувати на подальший вибір спеціалізації в ІТ-галузі.

Алгоритмізація і програмування є суто інформатичними видами діяльності, які вирізняють інформатику серед інших наук. На думку автора бестселера «Мистецтво програмування» Дональда Кнута, який, за даними Вікіпедії, на січень 2013 року посідає 37 позицію у рейтингу авторів у галузі комп'ютерних наук за кількістю цитувань, «кращий ... спосіб визначити інформатику — це сказати, що вона займається вивченням алгоритмів» [1].

Розв'язання задач на алгоритмізацію та програмування є найцікавішим з того, що може відбуватися на уроці інформатики. Такі задачі дають простір для розвитку аналітичного, логічного, критичного, операційного, комбінаторного, творчого мислення учня. «Програмування — дзеркало розуму», як влучно зазначив Джеральд Вейнберг. Навчати учнів розмірковувати над розв'язком задачі, вибудовувати логіку дій, здійснювати аналіз знайденого рішення, добиватися його прозорості, лаконічності, нарешті, естетичності — все це можна і слід робити на уроках інформатики, і це є актуальним як з точки зору реалізації одного з провідних завдань загальноосвітньої інформатичної підготовки школярів, так і в плані підвищення конкурентоздатності молодої людини в умовах сучасного динамічного інформатизованого суспільства.

Проблемам використання задач у предметному навчанні і навчання школярів розв'язанню задач присвячено чимало психолого-педагогічних досліджень, зокрема фундаментальні праці Г. Балла [2], Д. Пойя [3]. З появою в школі нового шкільного предмета «Інформатика» великий загал науковців і практиків сконцентрував свої зусилля на розробці педагогічних підходів, методичних прийомів, дидактичних засобів, зорієнтованих на сприяння успішності навчання учнів розв'язанню задач на побудову алгоритмів і складання програм (Я.М. Глинський, І.О. Завадський, Т.П. Караванова, М.Б. Культін, С.М. Окулов, К.Ю. Поляков, А.В. Присяжнюк, С.А. Присяжнюк та інші). У рамках окресленого напряму були виконані спеціальні наукові дослідження, зокрема спрямовані на з'ясування особливостей пізнавальних задач з інформатики, визначення педагогічних умов їх ефективного використання у навчанні школярів інформатики (Н.О. Пономарьова [4]), оптимізацію форм організації навчальної діяльності школярів у процесі розв'язування задач з інформатики (Н.В. Олефіренко [5]), підготовлено збірники систематизованих за тематикою і складністю задач з програмування та інформаційних технологій [6, 7]). Разом з тим можна зазначити, що серед великої кількості розробок, зокрема й представлених в Інтернеті, переважна більшість присвячена аналізу задач високого або підвищеного рівня складності, які застосовуються у практиці проведення олімпіад з програмування і підготовці до них, у той час як розглядом так званих «простих» задач, призначених для використання на звичайному уроці інформатики, у повсякденній практиці роботи вчителя, опікуються не так багато дослідників, і ще менше напрацювань, де розкриваються деталі роботи над задачею.

Метою даної роботи є висвітлення тих можливостей реалізації провідних цілей навчання інформатики, які надає належна організація роботи над простою задачею на алгоритмізацію і програмування.

Ключовою відмінністю задач з інформатики є варіативність їх розв'язку. Жоден навчальний предмет не надає таких можливостей, як інформатика для вироблення різних рішень практично будь-якої задачі. Інші предмети природничо-математичного циклу (за А.П. Єршовим, інформатику слід відносити саме до таких предметів) здебільшого концентрують увагу учнів на опануванні алгоритмів розв'язування задач, який дає змогу шляхом певних дій дістатися шуканого результату. Задачі, що розглядаються у навчанні інформатики, й особливо основ алгоритмізації та програмування, попри наявність певних методів їх розв'язання, завжди залишають місце для альтернативних рішень, розгляд

і аналіз яких є не менш цінним, ніж досягнення цілі, сформульованої в постановці задачі.

Проілюструємо це на конкретних прикладах.

Задача 1. Вирощування курчат — складна справа. Птахофабрика роздала курчат школярам для вирощування в домашніх умовах, запропонувавши за це таку винагороду: якщо кількість вирощених курчат не перевищить 40, то за кожне буде виплачено 1 грн.; за більшу кількість вирощених курчат винагорода зростає: за кожне курча понад 40 виплачують 1 грн. 20 коп., за кожне понад 50 — 1 грн. 50 коп., за кожне понад 60 — 5 грн. Розробіть алгоритм обчислення винагороди V, яку отримає школяр за вирощування N курчат.

Проаналізуємо варіанти розв'язку задачі (табл. 1). Для скорочення будемо розглядати тільки ядро алгоритму. В усіх розв'язках залишатимемо числові значення, задані в умові задачі, з тим, щоб було зрозумі-

Таблиця 1

Варіант 1. якщо N <= 40 то V:= N*1; якщо N > 40 and N <= 50 то V:= V + 1.2*(N−40); якщо N > 50 and N <= 60 то V:= V + 1.5*(N−50); якщо N > 60 то V:= V + 5*(N−60);	**Коментар**. Такий варіант часто пропонують ті, хто намагається швидко впоратися з рішенням, але цілком зрозуміло, що алгоритм забезпечує правильне обчислення значення V тільки для випадку *N*<=40
Варіант 2. якщо N <= 40 то V := N*1 інакше якщо 40 < N <= 50 то V:=(N−40)*1.2+40*1 інакше якщо 50 < N <= 60 то V :=(N−50)*1.5+10*1.2+40*1 інакше якщо N > 60 то V:=(N−60)*5+10*1.5+10*1.2+40*1;	**Коментар.** Алгоритм дає правильну відповідь, але використання вкладених умов, тим більш із таким ступенем вкладеності, виписування кожного разу повної формули для обчислення значення V призводить до зайвої громіздкості алгоритму
Варіант 3. V: = N*1; якщо N > 40 то V := V + (N−40)*0.2; якщо N > 50 то V: = V + (N−50)*0.3; якщо N > 60 то V := V + (N−60)*3.5;	**Коментар.** Рішення правильне і прозоре, застосовується послідовне проходження крізь «сито» простих умов. Сума винагороди формується поетапно: спочатку вона визначається за найменшим тарифом, а далі коригується: якщо значення *N* перевищує черговий рубіж (40, 50, 60), винагорода відповідним чином збільшується. Наприклад, для *N*=52 винагорода обчислюється в три етапи: спочатку вона однакова за всіх вирощених курчат (52 грн.), далі за 12 курчат (понад 40) додається доплата по 0,2 грн. (2.40грн.), потім за 2-х курчат (понад 50) доплачується ще по 0,3грн. (0,6грн). Разом — 55 грн.
Варіант 4. V: = 0; якщо N > 60 то поч V:= V + (N−60)*5; N:=60 кін; якщо N > 50 то поч V: = V + (N−50)*1.5; N:=50 кін; якщо N > 40 то поч V: = V + (N−40)*1.2; N:=40 кін; V: = V + N*1;	**Коментар**. Якщо є варіант 3 з простими умовами «знизу догори» (за зростанням граничних значень), то має бути і зворотний варіант — «згори донизу», і спортивний інтерес змушує знайти його. Як бачимо, зворотний варіант алгоритму виявився менш раціональним
Варіант 5. V = N*C1; якщо N > M1 то V = V + (N−M1)*(C2−C1); якщо N > M2 то V = V + (N−M2))*(C3−C2); якщо N > M3 то V = V + (N−M3))*(C4−C3);	**Коментар**. Узявши варіант 4 за базовий, узагальнимо його на випадок довільних значень цін (C1,C2,C3,C4) і граничних значень (M1, M2, M3)

ло, як цей розв'язок узагальнити на випадок довільних значень вхідних даних.

В останньому варіанті запису алгоритму проглядається певна система дій і виникає бажання записати весь алгоритм в один рядок. Для цього потрібна функція $F(N,M)$, яка буде залежати від двох змінних N і M і повертати значення 1, якщо N більше M, або значення 0 у протилежному випадку. Визначимо функцію у такий, наприклад, спосіб:

F(N,M):=0; якщо N>M то F(N,M):=1.

Тоді обчислення суми винагороди можна реалізувати одним оператором:

V:=N*C1+F(N,M1)*(N−M1)*(C2−C1)+F(N,M2)*(N−M2)*(C3−C2)+F(N,M3)*(N−M3)*(C4−C3);

Функцію $F(N,M)$ можна записати через стандартні функції мови програмування. Наприклад, у мові програмування Паскаль є стандартна функція ord(x), яка застосовується до даних дискретного типу і повертає порядковий номер значення аргумента. У мові Паскаль логічні значення або значення типу boolean, упорядковані: type boolean=(false, true), і порядкові номери для констант типу, заданого перечисленням, рахуються з нуля, тому ord(false)=0, ord(true)=1. Отже, потрібна нам функція $F(N,M)$ — це ord ($N>M$).

Отже, розв'язок нашої задачі на мові програмування Паскаль можна записати так:

V:=N*C1+ord(N>M1)*(N−M1)*(C2−C1)+ord(N>M2)*(N−M2)*(C3−C2)+ord(N>M3)*(N−M3)*(C4−C3);

Повторюваність додатків явно провокує на використання циклу, тим більш що виникає питання, чому має бути рівно три граничних значення C, якщо ми узагальнюємо рішення? Цілком очевидно, як організувати цикл для k граничних значень, хоча і тут можливі варіанти залежно від того, як систематизувати вхідні дані. З використанням одномірних масивів для C і M одержуємо:

V:=N*C[1];
для i від 1 до k повторювати
V:=V+ord(N>M[i])*(N−M[i])*(C[i+1]−C[i]);

Здається, на цьому «ігри розуму» можна було б і припинити, але все ж таки цікаво, що працюючи над алгоритмом, ми зовсім забули про тих курчат! Рушійною силою було бажання написати простий, прозорий і звичайно ж правильний алгоритм. Тепер настав час спрямувати роздуми на осмислення виконаної роботи: для розв'язання яких задач наш алгоритм (можливо, з деякими несуттєвими модифікаціями) є застосовним? З'ясовуємо, що, наприклад, для таких, які подані далі.

1. Постачальник товару, намагаючись залучити оптових покупців до придбання якомога більшої кількості товару, застосовує систему прогресивних знижок. Визначити ціну, яку покупець має сплатити за задану кількість товару.

2. Відомо, що в гонитві за висококласними програмістами фірми приваблюють їх не тільки високим заробітком, а й його гарантованим підвищенням, розмір якого певним чином зростає зі стажем роботи працівника на фірмі. Програміст розглядає пропозиції різних фірм, кожна з яких запровадила свою систему підвищення заробітної платні. Складіть програму, яка на основі даних про цю систему визначає сумарний заробіток програміста за заданий відрізок часу роботи в фірмі.

Особливий інтерес на уроках інформатики викликає розв'язання задач, пов'язаних із повсякденною життєвою практикою учнів. Немає жодного школяра, який би не грав у хованки, а скласти алгоритм для визначення, кому водити, виявляється зовсім не простим завданням.

Задача 2. Лічилка — неодмінний атрибут гри в хованки. Щоб визначити, хто буде водити, діти стають у коло і рахуються, проговорюючи лічилку. Той, на кого прийшлося її останнє слово, виходить з кола, і рахування повторюють, починаючи з наступної дитини в колі. Врешті-решт залишається одна дитина, яка й буде водити. Розробіть алгоритм, що за заданими кількістю дітей у колі (N) і кількістю слів у лічилці (M) визначає, хто буде водити, тобто знаходить порядковий номер (P) цієї дитини в колі відносно тієї, з якої почали рахувати.

Ця задача відома як задача Йосифа Флавія або лічилка Джозефуса. З історичним підтекстом задачі, так само як і з її розв'язками можна познайомитися в Інтернеті. Цікаво, що задачу про лічилку ми запропонували на сайті кафедри інформатики (kafinfo.org.ua) в рамках проекту «Задача тижня» рік тому і проаналізували те, що з цього приводу було в Інтернеті. Нині картина змінилася: з'явилися нові описи задачі, приклади її розв'язку, і, здається, є певне пробудження нової хвилі інтересу школярів до програмування.

Спробуємо розв'язати задачу, виходячи з простих міркувань.

Розглянемо перше рахування. Оскільки воно відбувається по колу, доцільно перейти до нумерації дітей, яка починається не з одиниці, а з нуля: 0,1,... $N-1$. Тоді номер дитини, на якій завершиться лічилка з M слів, визначається як $(M-1) \bmod N$. Ця дитина вийде з кола, і наступне рахування слід буде починати з дитини з номером $M \bmod N$.

Рахуємо другий раз. Тепер кількість дітей у колі $N-1$, і щоб звести цей випадок до попереднього, перенумеровуємо дітей заново так, щоб дитина, яка була з номером $M \bmod N$, отримала новий номер 0. Отже, нові номери отримуються зменшенням на $M \bmod N$. Визначаємо номер дитини, на якій завершується лічилка: $(M-1) \bmod (N-1)$. Наступне рахування починатиметься з дитини під номером $M \bmod (N-1)$.

Очевидно, що для знаходження, хто буде водити, таку процедуру рахування слід повторити $N-1$ разів, щоб зменшити кількість дітей у колі від N до 1. Відповідно будуть змінюватися і номери дітей, зменшуючись спочатку на $M \bmod N$, потім на $M \bmod (N-1)$,..., на $M \bmod 2$. В останній раз маємо «коло» з однієї дитини, її найновіший номер дорівнює 0, з неї ми мали б починати наступне рахування, але в цьому немає потреби, — вона водитиме. Наша задача фактично звелась до того, щоб «прокрутити» цикл перенумерувань назад для відновлення первісного номеру цієї останньої дитини: яким він був під час попереднього рахування, коли в колі було 2 дитини, потім коли їх було 3 і т. д. до N. Для

відновлення первісного номера дитини слід збільшувати її кінцевий номер (0) спочатку на M mod 2, потім на M mod 3 і т. д. до M mod N. Оскільки значення, одержане в результаті такого збільшення, може виявитися рівним або більшим за поточну кількість дітей у колі, необхідно кожного разу до знайденого значення застосовувати операцію ділення по модулю і, де і — поточна кількість дітей у колі.

Отже, можна записати алгоритм знаходження шуканого номера P:

P:=0;

для i від 2 до N повторювати

P:=(P+M) mod i;

P:=P+1; {після завершення циклу повертаємося до нумерації дітей з одиниці}

Зверніть увагу: в тілі циклу слід було записати оператор P:=(P+M mod i)mod i, але ми скористались тим, що (P + M mod i) mod i=(P+M) mod i.

Оскільки допитливі учні для розв'язання поставленої задачі неодмінно звернуться до допомоги колективного розуму, а в Інтернеті наводяться математичні підходи до розв'язку задачі, доцільно розглянути ще один спосіб обґрунтування алгоритму знаходження **P**, який спирається на виведення рекурентної формули для визначення P.

Шукане значення P залежить від значень M і N, тому позначимо його як $P(M,N)$. Вище ми розглядали випадки, коли кількість дітей у колі зменшувалася від N до 1, тепер здійснюватимемо розгляд у зворотному порядку.

Якщо N=1, тобто в колі всього одна дитина (з номером 0), їй і водити. Отже, $P(M,1)$=0 для будь-якого значення M.

Якщо N=2, усе залежить від парності числа M, тобто від значення остачі M mod 2. Якщо M — парне, то водитиме дитина з номером 0, а якщо M — непарне, то з номером 1. Отже, шуканий порядковий номер визначається за формулою: $P(M,2)= M$ mod 2. Цю формулу можна пов'язати з попереднім результатом. Оскільки $P(M,1)$=0, запишемо:

$$P(M,2)= (P(M,1) + M) \bmod 2.$$

Якщо N=3, то першою вийде з кола дитина з номером $(M-1)$ mod 3. У колі залишаться дві дитини, і ситуація зводиться до попередньої, але з тією різницею, що тепер рахування починатиметься з наступної дитини в колі, яка має номер M mod 3. Щоб знайти, хто ж водитиме, слід здійснити «зсув» формули, одержаної для N=2, на M mod 3:

$$P(M,3)=(P(M,2)+M \bmod 3) \bmod 3=(P(M,2)+M) \bmod 3.$$

Якщо N=4, то першою з кола вийде дитина з номером $(M-1)$ mod 4. У колі залишаться три дитини, як у попередньому випадку, але тепер починаємо рахувати з дитини, яка має номер M mod 4. Здійснюючи «зсув» попередньої формули (для N=3) на M mod 4, знаходимо номер дитини, яка водитиме:

$$P(M,4)=(P(M,3)+M \bmod 4) \bmod 4=(P(M,3)+M) \bmod 4$$

Отже, за індукцією отримуємо загальну рекурентну формулу для $N>1$:

$$P(M,N)=(P(M,N-1)+M) \bmod N.$$

Кінцеве значення $P(M,N)$, одержане у такий спосіб, слід збільшити на одиницю, щоб перейти до нумерації дітей, починаючи з одиниці.

Наприклад, для N=5 (п'ятеро дітей), M=7 (у лічилці 7 слів), матимемо:

$$P(7,5)=(P(7,4)+7)\bmod 5=((P(7,3)+7)\bmod 4+7)\bmod 5=$$
$$(((P(7,2)+7)\bmod 3+7)\bmod 4+7)\bmod 5 =$$
$$((((P(7,1)+7)\bmod 2+7)\bmod 3+7)\bmod 4+7)\bmod 5.$$

Ураховуючи, що $P(7,1)$=0, знаходимо шукане значення: $P7(5)$=3. Отже, водитиме дитина, яка стоїть четвертою в колі.

Представимо загальну формулу для обчислення $P(M,N)$ у розгорнутому вигляді:

$$P(M,N)=(((...((0+ M) \bmod 2+M) \bmod 3 + ...$$
$$+M) \bmod (N-1)+M) \bmod N.$$

Як бачимо, вона приводить до такого самого алгоритму, який ми записали вище.

До задачі про лічилку доцільно повернутися ще не один раз, ілюструючи різні способи організації даних, наприклад, перечислення, динамічний масив.

У процесі розв'язання задач, особливо простих, часто нехтують докладним аналізом правильності алгоритму на етапі його планування, адже здається, що немає де помилитися. Третій приклад наводимо для демонстрації такої неочевидної помилки.

Задача 3. Цукерки в мішечках розставлені вздовж дороги. На кожному мішечку написана кількість цукерок, що в ньому містяться. Хлопчик може взяти в кожну руку по два мішечка, що розташовані поряд. Розробіть алгоритм знаходження найбільшої кількості цукерок, яку може взяти хлопчик.

Задача зводиться до пошуку двох пар поряд розташованих елементів одновимірного масиву, які утворюють максимальну суму. На перший погляд, задача тривіальна. Обговоримо можливі шляхи її розв'язання. Є пропозиція вирішити задачу за один прохід масиву: переглядати його, зберігаючи в пам'яті дві пари елементів, одну — з найбільшою сумою, другу — із сумою, яка за величиною посідає друге місце. На кожному кроці необхідно порівнювати значення суми поточної пари елементів з тими значеннями сум, що зберігаються, і, якщо буде потрібно, здійснювати відповідні заміщення. Мало того, що такі порівняння загромаджують алгоритм, він не є правильним. Якщо кількість цукерок у мішечках, наприклад, така: 10, 7, 12, 6, 1, 10, 2, 8, то пара з найбільшою сумою — це 7 і 12, а пара з наступним за величиною суми значенням — це 12 і 6.

Інша пропозиція — розв'язати задачу за два проходи масиву: знайти пару поряд розташованих елементів масиву, сума яких максимальна, «взяти їх» (тобто надати відповідним елементам масиву значення 0), а потім повторити перегляд масиву для пошуку другої пари з максимальною сумою. Зайві порівняння відпадають, все просто, але і таке рішення не є правильним. Можна легко переконатися, що алгоритм не спрацьовує для нашого прикладу вхідних даних (10, 7, 12, 6, 1, 10, 2, 8). Дійсно, результатом першого перегляду буде вибір другого і третього елементів (їх сума 19), наступний перегляд масиву (тепер вже це 10, 0, 0, 6, 1, 10, 2, 8) призведе до вибору шостого і сьомо-

го елементів (їх сума 12). Отже, усього хлопчик може взяти 31 цукерку. Однак легко побачити, що це не максимальна кількість, оскільки сума перших чотирьох елементів вихідного масиву дорівнює 35. Пошук правильного і компактного рішення задачі переадресуємо читачеві.

Повертаючись до вислову Е. Дейкстри про сутність роботи програміста, нагадаємо ще один із його знаменитих листів, у якому він наводить притчу про залізничні вагони [8]. Першим у світі компетентним програмістом він назвав того безвісного зчіплювача вагонів, який знайшов простий і раціональний спосіб формування составу із заданою умовою. Отже, добирати задачі необхідно так, щоб вони були цілком зрозумілими і цікавими за постановкою, доступними для вирішення простими методами, давали шлях до розв'язання більш складних задач і мотив до оволодіння більш складними методами. А розглядати кожну задачу слід не як привід для швидкого програмування, а як вправу для розмірковування, для тренування мислення, для набуття вмінь вибудовувати правильне й красиве рішення — це і є головним у навчанні школярів основ алгоритмізації та програмування.

Якщо ставити питання, яку мову слід вивчати в шкільному курсі програмування, то відповідь може бути такою: безсумнівно, вибір мови програмування теж має неабияке значення, і це ще один пласт цікавої роботи над задачею. Однак слід зважити на те, що зазначити певну мову в програмі шкільного курсу не так важко, а реалізувати продуктивне навчання школярів програмування цією мовою — зовсім інша справа, і вчитель має бути до цього готовим. Перші вчителі інформатики (як і школярі, і студенти) із захопленням вивчали мови програмування — Бейсік, Паскаль, добре володіли мовами, проте із зниженням значимості теми «Алгоритмізація і програмування» в шкільному курсі зменшувалася і потреба в таких знаннях, і ми, на жаль, втратили такий цінний інтелектуальний учительський ресурс. Нині мало вчителів-програмістів серед тих, хто працює в школі, і тому, на мій погляд, доцільно повернутися до мови Паскаль, ураховуючи реальний рівень готовності до цього працюючих учителів і наявний потужний ресурс навчально-методичного забезпечення, яке представлено в друкованому вигляді і надається в Інтернеті для вільного використання.

Мов програмування багато, вони змінюватимуться, заманюючи розробника новими, усе більш зручними і потужними можливостями, але перш за все потрібно пробудити інтерес учня, бажання йти по цьому шляху, навчити його знаходити в цьому особистий сенс і власне задоволення. А стосовно тих, хто не планує стати програмістом, то розвинене логічне й раціональне мислення ще нікому не заважало.

Проте вирішуючи питання щодо програми шкільного курсу інформатики, хотілося б не «латати дірки на дорозі», а системно підійти до розгляду проблеми побудови неперервної шкільної інформатичної освіти, сформулювавши її ключові засади і коло провідних змістових ліній, які відрізняють саме інформатику серед інших навчальних предметів, становлять її незалежне ядро. А щодо застосувань масових інформаційних технологій, то всі, мабуть, читали про ефіопських дітей з віддаленого селища, де мешкає геть неграмотне населення, і про те, що сталося, коли їм дали планшети...

Висновки. Оволодіти методами і вміннями діяльності можна тільки в процесі самої діяльності. За своєю сутністю, шкільний курс інформатики є задачно орієнтованим. Отже, на перший план виходить робота над задачею, і вчителю слід правильно добирати задачі і правильно організовувати роботу учнів над задачею, не шкодувати часу на вироблення різних підходів до її рішення, прискіпливо аналізувати кожну пропозицію, крок за кроком просуватися по шляху вилучення помилок і вдосконалення алгоритму, занурювати школярів у цей процес, навчати їх знаходити в ньому і сенс, і смак, і можливість виявити власну індивідуальність.

★ ★ ★

Белоусова Л.И. Прелесть простых задач или к вопросу об использовании языков программирования в процессе изучения школьниками информатики

Аннотация. В статье раскрывается значимость обучения школьников основам алгоритмизации и программирования; на конкретных примерах рассмотрены те возможности реализации ведущих целей обучения информатике, которые предоставляет надлежащая организация работы над задачей на разработку алгоритма.

Ключевые слова: информатика, основы алгоритмизации и программирования, обучение школьников, задачи на разработку алгоритма, работа над задачей.

★ ★ ★

Abstract. The paper reveals the importance of secondary school students' training the fundamentals of algorithms design and programming; the possibilities of realization of leading goals of computer science teaching and learning, which provided by proper organized process of algorithmic problem solving, are considered through the specific examples.

Keywords: computer science, basic algorithms and programming, training school, the tasks for the development of the algorithm, the work on the task.

Література

1. *Кнут Д.* Информатика и ее связь с математикой / Д.Кнут // Современные проблемы математики. — М.: Знание, 1977. — С. 4–32.
2. *Балл Г.А.* Теория учебных задач: Психолого-педагогический аспект. Педагогический энциклопедический словарь / Г.А. Балл. — М.: Педагогика, 2002. — 321 с.
3. *Пойя Д.* Как решать задачу / Д. Пойя. — М.: Либроком, 2010. — 208 с.
4. *Пономарьова Н.О.* Педагогічні умови використання пізнавальних задач у навчанні інформатиці: дис. … канд. пед. наук : 13.00.01 / Пономарьова Наталія Олександрівна. — Харків, 1998. — 175 с.
5. *Пожар Н.В.* Групові форми організації пізнавальної діяльності старшокласників в умовах інформатизації навчання: дис… канд. пед. наук: 13.00.01 / Н.В. Пожар. — Х., 1999. — 167 с.
6. *Белоусова Л.И.* Сборник задач по курсу інформатики /Л.И.Белоусова, С.А.Веприк, А.С. Муравка — М.: Экзамен, 2008. — 256 с.
7. *Присяжнюк А.В.* Turbo Pascal. Основи / А.В. Присяжнюк, С.А. Присяжнюк [Електронный ресурс]. — Режим доступу: http://www.libraries.com.ua/?p=276.
8. *Дейкстра Едсгер Вайб.* Программистские басни. Притча [Электронный ресурс]. — Режим доступа: http://bookre.org/reader?file=101078.