УДК 371.3:004.415.53+159.928

АНАЛІЗ, ДОСЛІДЖЕННЯ ТА РОЗВ'ЯЗУВАННЯ КОНКУРСНИХ ЗАДАЧ ПІД ЧАС УЧНІВСЬКОЇ ОЛІМПІАДИ З ІНФОРМАТИКИ

Жуковський С.С.
Житомирський державний університет імені Івана Франка

Пропонується методика підготовки учнів до розв'язування олімпіадних задач з інформатики. Наведена технологія аналізу задачі та її умови, побудови математичної моделі, реалізації алгоритму мовою програмування та тестування програми-розв'язку. Пропонується стратегія поведінки учня під час змагання.

Ключові слова: *олімпіада з інформатики, задача, етапи розв'язання задач, математична модель*

Концепція «Державної програми роботи з обдарованою молоддю на 2006-2010 роки» визначає залучення обдарованої молоді до науково-дослідницької, експериментальної, творчої діяльності … з обдарованою молоддю, проведення всеукраїнських олімпіад, конкурсів, конкурсів-захистів, турнірів і фестивалів та забезпечення участі обдарованої молоді у міжнародних інтелектуальних і творчих змаганнях [1, 2].

У світлі цієї концепції завдання сучасної школи є не лише глибоке засвоєння та відтворення отриманих знань, а й розвиток пошуково-пізнавальних здібностей учнів, формування всебічно розвинутої особистості з метою успішної соціалізації їх у суспільстві.

У післяшкільному житті реальною необхідністю в наші дні є безперервна освіта, що потребує повноцінної загальноосвітньої підготовки. Все більше й більше запитані спеціальності, пов'язані з використанням комп'ютерів, зокрема з програмуванням мовами професійного рівня.

Вміння розв'язувати задачі є одним з основним показником рівня математичного розвитку, глибини засвоєння навчального матеріалу. У шкільному курсі математики, фізики та інформатики навчанню розв'язування задач приділяється багато часу, але основним методом такого навчання є демонстрація способів розв'язування певних видів (класів) задач, і зовсім не даються так необхідні знання аналізу суті задачі та її розв'язку. В учнів не виробляються уміння і навики в діях, що входять у загальну діяльність по розв'язуванню задач, не стимулюється постійний аналіз учнями своєї діяльності у цьому напрямку, по виділенню в ній загальних методів та підходів, що дало б можливість, у подальшому, будувати власну стратегію дослідження та розв'язання задач такого класу.

Розробити та описати технологію розв'язування учнем задач на олімпіаді з інформатики є метою даної статті.

Задача – вимоги, або запитання, на які необхідно знайти відповідь, спираючись та враховуючи вхідні дані, які випливають з умови задачі.

Олімпіадна задача з програмування – це завдання, яке вимагає написати програму, яка повинна зчитати з консолі (файлу) певні дані, в залежності від вхідних даних розв'язати задачу і вивести в консоль (файл) певні дані (результат), відповідь на поставлену задачу. На відміну від математичних задач, в яких у більшості випадків задаються конкретні дані, задача з програмування вимагає передбачити різноманітні вхідні дані, обмежені умовою задачі, і в залежності від вхідних даних розв'язати поставлену задачу. Часто трапляється, що розв'язуючи задачу з програмування доводиться розглядати декілька випадків у залежності від вхідних даних, а інколи і декілька різних задач.

Приклад: Два кола (http://www.e-olimp.com.ua/ua/problems/4**).**

Визначити в скількох точках вони перетинаються.

Технічні умови

Вхідні дані: 6 чисел x1, y1, r1, x2, y2, r2, де x1, y1, x2, y2 – координати центрів кіл, а r1, r2 – їх радіуси. Всі числа – дійсні, не перевищують 1000000000 за модулем та задані не більш ніж з 3 знаками після коми.

Вихідні дані: одне число, яке показує кількість точок перетину. 0, 1, 2 – відповідна кількість точок перетину; -1 – безліч точок перетину.

Приклад вхідних даних:	***Приклад вихідних даних:***
0 0 5 5 0 5	2

У даній задачі можна виділити до 10 різних випадків, які при вдалому групуванні можна поділити на 4 групи: кола накладаються, коли центри і радіуси співпадають; дотикаються (зовні та одне коло всередині іншого); немає точок дотику (зовні і одне коло всередині іншого); решта – перетин у двох точках.

Розв'язання задачі – робота дещо незвичайна, а саме розумова, аналітична. Для того, щоб навчитися якій-небудь справі, потрібно попередньо добре вивчити той матеріал, з яким доведеться працювати, ті інструменти, які доведеться використовувати під час розв'язування задачі.

Відома багатьом учасникам олімпіади з інформатики крилата фраза: «Я не знаю, як розв'язувати задачі. Я знаю лише, що після того, як розв'язати їх багато, то починаєш це робити краще, починаєш краще бачити можливі підходи до розв'язання задач, починаєш краще їх відчувати» [3].

Відповідь на запитання «Як розв'язати задачу?», особливо, якщо це олімпіадна задача, не завжди лежить на поверхні – тому, що пошук її є творчий процес. І єдиного підходу до таких задач знайти неможливо, але, як показує практика, є ряд методів та прийомів, використовуючи які, можна навчитися розв'язувати задачі. Багато в чому у таких випадках не обійтися без інтуїції [4].

Отже, щоб навчитися розв'язувати задачі, потрібно розібратися в тому, що таке задача, як вона побудована, із яких частин складається умова, які інструменти можна використати для її розв'язання.

Якщо придивитися до будь-якої задачі, то можна побачити запитання, на яке необхідно знайти відповідь, виходячи з тих даних, обмежень, які знаходяться в умові задачі. Тому, починаючи розв'язувати будь-яку задачу, потрібно уважно вивчити умову задачі. Це все називається проаналізувати умову задачі.

Задача з програмування, як правило, потребує спочатку побудувати її математичну модель, аналітично дослідити цю модель, розробити алгоритм розв'язування математичної задачі, а вже потім написання програми, яка буде розв'язувати задачу для довільних вхідних даних.

Власний досвід автора дає можливість виділити наступні етапи, які повинен виконати учень під час розв'язування олімпіадних задач: аналіз умови задачі, побудова математичної моделі, реалізація алгоритму розв'язку мовою програмування, тестування та відлагодження розв'язку, здача розв'язку на перевірку.

Аналіз умови задачі. Умова олімпіадної задачі з програмування складається з таких елементів: сюжету, завдання, технічних умов, прикладу вхідних та вихідних даних.

Сюжет містить опис ситуації, яка розглядається в даній задачі. Автори олімпіадних задач люблять робити громіздкий сюжет, що інколи заплутує учасника. Бувають сюжети, що прикрашають умову задачі і такі, що містять цінну інформацію. Тому потрібно уважно прочитати сюжет умови задачі, вникаючи в кожне слово.

Отримавши задачу необхідно уважно її прочитати. У будь якій задачі ми побачимо завдання, яке ми повинні виконати, щоб отримати потрібний результат.

Під час олімпіади бути уважним і спокійним дуже важко. Для прочитання умови задачі необхідно виділити досить часу. Не потрібно поспішати на даному етапі. Задачу необхідно читати не поспішаючи, не пропускаючи ліричні відступи, не рекомендується читати задачу оглядово. Автори задач досить часто люблять прикрасити умову задачі

ліричним відступом. І нерідко в таких ліричних відступах може бути приховані важливі моменти, що стосуються вхідних даних умови задачі. Неправильне розуміння умови задачі може привести до того, що буде розв'язуватися зовсім інша задача, а не та, що сформульована в умові.

Ключ до умови задачі може бути прихований також у форматі вхідних або вихідних даних. Без нього інколи задача може нести зовсім інший зміст. Тому, важливо визначити з формат введення та виведення, обмеження на вхідні, вихідні та проміжні дані.

Часто трапляється, що після детального прочитання задачі, не знаєш, як до неї підступитися. У такому випадку рекомендується відкласти її і читати наступну. Наш мозок здатний працювати підсвідомо, навіть у той час, коли працює над іншою задачею. Під час тренування Ф. Меншиков [5] радить прочитати умови задач всього тренування відразу, спробувати розв'язати задачу, а якщо розв'язок зовсім не приходить на думку, то прочитати розв'язок, викладений у цій же книзі, але не раніше ніж через 3 дні після прочитання задачі. Учні повинні вчитися шукати розв'язок. Навіть, коли розв'язок до цієї задачі прийде не відразу, а через день, два, тиждень, учень за цей час перебере багато варіантів розв'язання задачі, розв'яже десяток інших задач, які подібні, але розв'язок можливо не пов'язаний із розв'язком даної задачі.

При навчанні розв'язування задач потрібно навчити учнів виділяти головні елементи в умові задачі, не пропускати жодних, на перший погляд неважливих, деталей. Перше – це необхідно виділити в умові складові частини і навчитися визначати необхідні дані з кожної з них.

Якщо прочитати умову будь-якої задачі, то можна виділити деяке питання, іншими словами вимогу, на яку необхідно отримати відповідь, спираючись на умову. Якщо ж уважно вивчити умову задачі, то можна побачити в ній певні твердження (що дано), вони ще називаються умовами, і певні завдання (те, що потрібно знайти).

Далі розглянемо складові частини завдання і рекомендації учням при їх розв'язуванні.

Побажання для засвоєння змісту задачі (1-й етап – аналіз умови). Не можна приступати до розв'язання задачі, не з'ясувавши чітко, у чому полягає завдання, тобто не встановивши, що дано і що необхідно знайти. Перша порада вчителя: не поспішати починати розв'язувати задачу. Ця порада не означає, що задачу треба вирішувати якомога повільніше. Це означає, що розв'язуванню задачі повинна передувати підготовка, що полягає в наступному:

а) спочатку треба ознайомитися із завданням, уважно прочитавши її зміст. При цьому вимальовується загальна ситуація, описана в завданні;

б) ознайомившись із задачею, необхідно вникнути в її зміст. При цьому потрібно слідувати такій пораді: виділити в задачі вхідні та вихідні дані, формати введення та виведення;

в) якщо задача геометрична, на теорію графів тощо, корисно зробити малюнок до неї і позначити на малюнку вхідні та вихідні дані. Намалювати малюнок, що відповідає прикладу вхідних даних умови задачі, придумати декілька прикладів вхідних даних і намалювати відповідні малюнки (це теж порада, якій повинен слідувати учень);

г) вже на першій стадії розв'язування задачі – стадії аналізу завдання – рекомендується відповісти на питання: «Чи можливо розв'язати задачу за такої умови?», «Чи завжди буде розв'язок даної задачі?», «Чи можливо декілька правильних розв'язків даної задачі?» (неоднозначний розв'язок), «Який з розв'язків необхідно вивести при неоднозначному розв'язку?», «В якому порядку виводити дані, якщо необхідно вивести всі неоднозначні розв'язки?».

Відповідаючи на ці запитання, потрібно встановити, чи вистачає даних для розв'язування задачі, чи немає зайвих даних, чи немає даних, які суперечать між собою.

Складання плану розв'язку задачі (2-й етап – пошук шляху розв'язання). Складання плану розв'язання задачі, мабуть, є головним кроком на шляху її розв'язання. Правильно складений план розв'язання задачі майже гарантує правильне її розв'язання. Але складання

плану може виявитися складним і тривалим процесом. Тому вкрай необхідно навчити учнів ставити запитання до задачі, що допомагають йому краще і швидше скласти план розв'язання задачі, фактично визначити метод її розв'язання:

а) чи відома учневі якась подібна задача? Аналогічне завдання? Якщо така задача відома, то скласти плану розв'язку задачі не буде складним. Іншими словами, чи можна застосувати метод раніше розв'язаної задачі. Але така задача відома далеко не завжди;

б) чи відома вам задача, до якої можна звести дану задачу. Якщо така задача відома, то процес складання плану вирішення даної задачі очевидний: звести задачу до задачі, яка розв'язувалась раніше. Може виявитися, що споріднене завдання невідоме вирішуваному і учень не може звести дане завдання до якого-небудь відомого. План же відразу скласти не вдається;

в) складаючи план розв'язування задачі, завжди треба ставити собі запитання: "Чи всі дані завдання використані?". Виявлення неврахованих даних задачі полегшує складання плану її розв'язку;

г) при складанні плану розв'язання задачі інколи буває корисно змінити вхідні дані, прорахувати результат для інших вхідних даних. Змінені вхідні дані можуть підказати непередбачені моменти вашого розв'язку;

д) досить часто для розв'язання задачі доводиться розглянути деякі частинні випадки, прорахувати вручну без комп'ютера результати для спрощених даних (для 1, 2, 3…). Прорахувавши для декількох невеликих розв'язків, можна побачити певну закономірність, послідовність.

Побудова математичної моделі та схеми розв'язку. Математична модель – система математичних співвідношень, які описують досліджуваний процес або явище. Побудувати математичну модель – це описати математично процеси, факти, умови задачі.

Перш за все учень повинен формально та математично зрозуміти умову задачі. Необхідно за допомогою ручки та паперу прорахувати тест з умови задачі. Перевірити як отримати вихідні дані з вхідних даних умови задачі. Придумати декілька тестів, контртестів і також математично їх прорахувати. Таким чином для простих тестів необхідно розібратися, як розуміється умова задачі. Придумуючи тести до задачі, можна знайти правильні розв'язки, які відрізняються від тих, які приходять на думку відразу ж після прочитання умови задачі, виявляються «підводні камені» задачі.

Можливе й інше трактування побудови математичної моделі. У даному випадку побудувати математичну модель означає придумати такий розв'язок, який буде працювати на абстрактній математичній машині при необмеженій пам'яті, необмеженому часі, необмеженому діапазоні змінних і відсутності втрати точності у дійсних змінних. Можливо цей розв'язок і не ефективний, але неефективний розв'язок рівносильний розумінню умови задачі [4].

Після побудови математичної моделі необхідно скласти схему розв'язку поставленої задачі. Схема розв'язку задачі повинна складатися з відомих елементів, алгоритмів. На даному етапі не потрібно вникати у всі тонкощі реалізації алгоритму – всі моменти реалізації та підгону даного алгоритму під конкретну задачу «прийдуть» під час написання самого алгоритму. Учень повинен знати, що даний алгоритм працює з певною ефективністю, і для цієї задачі та певних обмежень він буде працювати швидко і використовувати необхідний обсяг оперативної пам'яті.

На даному етапі учасник може зіткнутися з наступними проблемами:

- погана стиковка блоків. Певні блоки погано стикуються, можливо потрібно перероблювати алгоритм, змінювати вихідні дані блоку;
- ефективність. Трапляється, що для одних даних даний алгоритм буде працювати швидко, а для інших, навіть невеликих, складність роботи алгоритму велика;
- інший тип задачі;
- наявність декількох розв'язків. При наявності декількох розв'язків потрібно вибрати самий ефективний і той який реалізовується простіше та швидше.

Реалізація алгоритму. Раціонально використаний час на олімпіаді – 90% відсотків успіху. Тому дуже важливим є уміння розподілити кожну хвилину під час самої олімпіади, мати свою тактику і стратегію.

Перед тим, як учні отримують завдання, вони мають декілька хвилин адаптації з комп'ютером, середовищем програмування. У цей час учень повинен запустити середовище програмування, написати просту програму-шаблон, яка вводить дані, робить прості обчислення, виводить дані (згідно умов олімпіади). Відкомпілювати цю програму і перевірити її виконання. Після цього зробити декілька заготовок програм для майбутніх розв'язків, які містять введення та виведення (з файлу, клавіатури, на екран, чи у файл у залежності від правил проведення олімпіади).

Під час реалізації конкретної програми використовують декілька методів написання програми: зверху вниз, знизу вверх, комбінований метод.

Перший спосіб застосовується, коли загальна картина програми відома. У даному випадку пишеться основна частина програми, а функції реалізовуються потім. Реалізація відбувається швидко, якщо розбиття на етапи вдале. У такому випадку простіше проводити відлагодження програми.

Підхід знизу до верху реалізовується, коли учасник не бачить загальної схеми розв'язку, а час іде. Тоді можна написати спочатку введення, та виведення даних, певні алгоритми: функції для роботи з геометричними об'єктами, сортування, пошук, довгу арифметику, тощо, а потім робити стиковку даних алгоритмів для отримання остаточного розв'язку. У такому випадку не завадить написати невеликий коментар, які дані знаходяться в яких змінних, що повертає та чи інша функція тощо. Під час написання самих допоміжних алгоритмів може «прийти» і сам розв'язок.

На практиці найчастіше використовується комбінований із згаданих методів.

Учнівська олімпіада передбачає отримання балів за частково розв'язані задачі. Тому необхідно здавати розв'язки до кожної задачі, навіть, якщо впевнений, що алгоритм є неефективним і не пройде всі тести. Часто трапляється, що на деяких вхідних даних алгоритм працює довго. У таких випадках, якщо це можливо, необхідно запустити програму, щоб вона прорахувала результат і зберегла його в текстовому файлі. З отриманого результату створити масив констант у програмі-розв'язку. В цей час, поки програма з неефективним алгоритмом шукає розв'язки, можна зайнятися іншою задачею. Код програми, як правило, не аналізується членами журі. Потім, під час аналізу, розбору олімпіади можна буде знайти більш оптимальний розв'язок.

Тестування та відлагодження. Після того, як програма відкомпільована, необхідно перевірити на правильність її роботи. Часто буває, під час набору коду програми сплутані певні змінні, знак у формулі, неправильно розставлені дужки, тощо. Спочатку необхідно перевірити чи правильно вона працює на тестах, які дано в умові задачі. Як показує досвід роботи з учнями, при першому запуску відкомпільованої програми в половині випадків тест з умови задачі не проходить, оскільки розв'язок пишеться в умовах нервового стресу і на швидкість. Після того, як виправлені всі механічні помилки і розв'язок проходить тест з умови задачі, необхідно придумати декілька невеликих тестів, які передбачають різні моменти даної задачі. Попередні придумані тести не потрібно знищувати, адже можливо доведеться звернутися до них після виправлення помилок. Трапляється, що після виправлення помилок, або удосконалення програми вона видає правильний результат при нових тестах, а на попередніх –помилковий. Тому необхідно уважно перевіряти правильність результату, який видає програма-розв'язок, прораховувати тести вручну для перевірки.

Інколи є потреба вставляти в програму перевіряючі блоки: перевірку правильності введення вхідних даних, сортування масиву, правильність роботи навіть елементарних алгоритмів: алгоритм Евкліда, пошук в ширину, бінарний пошук тощо.

Перед здачею розв'язку необхідно протестувати його на граничних тестах. Крім «малих» тестів інколи є потреба перевірити на «великих» тестах. Для цього необхідно згенерувати тести з великою кількістю елементів, перевірити, як швидко працює даний

алгоритм. Не завжди можна перевірити правильність роботи даного алгоритму, але інколи можна за відповіддю з'ясувати, чи правильно працює алгоритм.

Здача розв'язку. Останнім часом олімпіада з програмування проводиться в он-лайн режимі. Під час олімпіади учні можуть здати на перевірку розв'язок і система перевірить його на тесті з умови задачі. Це дає можливість учням уникнути помилок, пов'язаних із неправильним форматом виведення результату.

Відправлення розв'язку на перевірку є одним із самих відповідальних моментів роботи учня на олімпіаді.

Рекомендується ще раз перечитати умову задачі, звернути увагу на формати виведення та обмеження на змінні. Можливо, при отриманому алгоритмі змінні виходять за їх межі, дані виходять за межі масиву, тощо. Це не завжди можна передбачити при розробці алгоритму.

Перед здачею програми розв'язку необхідно перевірити правильність формату вихідних даних, можливість виходу за межі масиву, видалити (або закоментувати) всі елементи відлагоджувальної інформації.

Саме на цьому етапі допущена помилка може привести до фатального результату при правильному розв'язку. Але і в такому разі трапляються моменти, коли для різних випадків пишеться окремий блок виведення і в одному з блоків залишається неправильний формат виведення або зайві дані, які не завжди перевіряються перед відправленням.

Реалізація запропонованої схеми розв'язку до конкретної задачі

Більярд (http://www.e-olimp.com.ua/ua/problems/58).

Більярд являє собою прямокутник розмірами **M x N**, де **M** і **N** – натуральні числа. З верхньої лівої лузи вилітає куля під кутом 45^0 до сусідніх сторін. Лузи розміщено тільки в кутах більярда. Визначити кількість зіткнень кулі з бортами більярда, після яких вона знову попаде в одну з луз, та номер лузи, в яку потрапить куля. Вважати, що тертя відсутнє, зіткнення абсолютно пружні, а куля – матеріальна точка.

Технічні умови

Вхідні дані:

У вхідному рядку міститься два числа **M** та **N**, **1 < = M, N <= 2000000000**. Нумерація луз за годинниковою стрілкою, починаючи з лівої верхньої лузи, з якої вилетіла куля. **M** – горизонтальна сторона більярда, **N** – вертикальна сторона більярда.

Вихідні дані:

Два числа: кількість відбивань кулі та номер лузи, в яку впаде куля записані через пропуск.

Приклад вхідних даних	*Приклад вихідних даних*
`2 1`	`1 2`

Прочитавши умову задачі можна виділити наступне:

- дано розміри більярду прямокутної форми M і N (будуємо прямокутник);
- **1 < = M, N <= 2000000000 –** змінні M і N повинні бути принаймні типу longint в Паскалі або long в C++.

Завдання

- потрібно визначити кількість зіткнень кулі з бортами більярду та номер лузи, в яку потрапить куля;
- дані результату виводяться в рядок через пропуск.

Перше, що спадає на думку, змоделювати хід кулі, поки куля не потрапить до лузи. В даному випадку необхідно для кожного удару пам'ятати кут удару, кут відбивання, в який борт вдаряється куля. По завершенні моделювання буде підраховано кількість ударів та номер лузи, в яку потрапить куля.

Є інший розв'язок цієї задачі. Побудуємо математичну модель. Зобразимо прямокутник. Розглянемо різні випадки розмірів більярду.

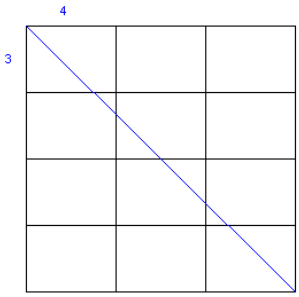
Самий вдалий випадок, коли M=N. Тоді куля потрапляє в протилежну лузу тій, з якої вона вилетіла.

Неважко визначити кількість ударів та номер лузи у випадку, коли M ділиться націло на N, або N ділиться націло на M.

Якщо M mod N =0 and (M div N) mod 2 =0, то куля потрапить у лузу №2, а якщо M mod N =0 and (M div N) mod 2=1, то у лузу №1.

Дослідимо інші випадки.

Зобразимо більярд у вигляді прямокутника (Мал.1). Намалюємо траєкторію руху кулі до першого удару. Далі



Мал. 1

продовжимо цю пряму і на стороні, де куля зіткнеться з бортом, домалюємо такий же прямокутник. Тепер ми будемо домальовувати прямокутники до тих пір, доки наша пряма не потрапить у вершину кута одного з домальованих прямокутників. Шукана нами вершина буде протилежна вершина деякого квадрата. Сторона утвореного квадрата точно ділиться на M і N. Отже, сторона квадрата дорівнює НСК(M,N). Кількість зіткнень з бортами більярду рівна кількості точок перетину прямої траєкторії кулі з бортами більярду (сторонами внутрішніх прямокутників):

НСК(M,N) div N -1+ НСК(M,N) div M -1.

Далі необхідно дізнатися, в яку лузу потрапить куля. Це з'ясовується за парністю ударів об горизонтальні та вертикальні борти більярду:

- якщо (ПГ) і (НВ) – 2 луза;
- якщо (НГ) і (ПВ) – 3 луза;
- якщо (НГ) і (НВ) – 4 луза;

П- парність, Н – непарність, Г – удари в горизонтальні борти, В –удари у вертикальні борти.

До лузи №1 куля не може потрапити тому, що для цього їй потрібно пройти по тому ж шляху, тільки в зворотному напрямку. А це можливо тільки при відбитті в лузі.

За створеною математичною моделлю пишеться програма-розв'язок, яка перевіряється за допомогою тестів.

В нашій програмі використовується алгоритм НСК, де при взаємно простих числах НСК(M,N)=M*N. В умові задачі числа M і N мають тип long, а добуток виходить за межі цього типу. Тому для розв'язання цієї задачі необхідно використати самий великий цілочисельний тип даних int64 в Паскалі та __int64 або long long в C++.

Наведемо текст програми-розв'язку.

```
#include<iostream.h>
__int64 NSK(__int64  a,__int64 b)
{
  __int64 m=a, n=b;
    while(a!=0 && b!=0)      if (a>b) a%=b; else b%=a;
   return m/(a+b)*n;
}
int main()
{
  __int64 a, b, c, r, g, v;
   cin>>a>>b;
    c= NSK(a,b);
    g=c/b-1;     v=c/a-1;     r=g+v;
    if (v%2==1) cout<<r<<" "<<4<<"\n"; else {
    if (g%2==0) cout<<r<<" "<<3<<"\n"; else cout<<r<<" "<<2<<"\n"; }
return 0;
}
```

Отже, під час підготовки до олімпіади з інформатики необхідно не тільки вчити учнів методів розв'язування задач та основним алгоритмам. Учнів потрібно вчити вмінню

керування своєю діяльністю, розподілом часу на розв'язання кожної задачі, вмінню розбивати процес розв'язання задачі на етапи, виділяти певні етапи та їх виконувати. З часом дані етапи будуть виконуватися, не задумуючись над кожним з них.

До олімпіади учнів необхідно готувати теоретично, практично та психологічно. Для того, щоб в учнів не було стресу під час олімпіади, регулярно потрібно проводити шкільні олімпіади з усіма вимогами міської, обласної олімпіад та підведенням рейтингу.

Після розв'язання задачі необхідно робити детальний аналіз розв'язку та процесу його отримання. Розв'язок потрібно аналізувати на оптимальність (швидкість виконання, затрати пам'яті, простоту реалізації).

Для успіху у змаганнях необхідно використовувати кожну можливість набрати максимальну кількість балів. Якщо повний розв'язок задачі побудувати не вдається, то потрібно передбачити хоча б часткові розв'язки для мінімальних та критичних даних. Саме на таких задачах учень вчиться виходити із складної ситуації, що нерідко трапляється в повсякденному житті.

Кожного року в світі проходить велика кількість олімпіад з програмування різного рівня складності. В Інтернеті з'являються нові сайти з інтерактивними архівами задач, та он-лайн змаганнями (www.olymp.vinnica.ua, http://www.acm.lviv.ua, http://e-olimp.com.ua, http://www.ttb.by, http://acm.timus.ru/). На них можна не тільки прочитати умову задачі, але й перевірити розв'язки і відразу ж отримати результат перевірки. На таких сайтах ведуться рейтинги учасників, проводяться он-лайн змагання.

Надіємося, що викладений досвід не залишиться поза увагою вчителів новаторів та талановитої молоді і принесе значну користь у підвищені своєї майстерності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Державна цільова програма роботи з обдарованою молоддю на 2007-2010 роки від 8 серпня 2007 р. № 1016. [Електронний ресурс] // Режим доступу: http://www.nau.kiev.ua.
2. Концепція Державної програми роботи з обдарованою молоддю на 2006-2010 роки розпорядженням Кабінету Міністрів України від 12 квітня 2006 р. № 202-р [Електронний ресурс] // Режим доступу: http://www.nau.kiev.ua.
3. Волков Л., Шамгунов Н. Как стать чемпионом Урала по програмированию. [Електронний ресурс] // Режим доступу: http://contest.ur.ru/library/shv.htm
4. Оршанский С.А. О решении олимпиадных задач по программированию формата ACM ICPC // Мир ПК.- №9 (Додаток до журналу). – 2005. – 30 с.
5. Меншиков Ф. Олимпиадные задачи по программированию // С.-Пб.: Питер, 2007. – 314 с.

Рецензент: Осипова Н.В.