Управління освіти і науки
Чернігівської обласної державної адміністрації

Чернігівський обласний інститут післядипломної
педагогічної освіти імені К.Д. Ушинського

# Збірник задач та розв'язків
# ІІ етапу Всеукраїнської учнівської олімпіади з інформатики
# 2014-2015 навчального року

Чернігів – 2015

**Автори-упорядники:**

**Бондаренко С.М.**, учитель математики та інформатики Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради, учитель-методист

**Зуб В.В.**, учитель математики, директор Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради, учитель-методист

**Літош Ю.М.**, завідувач відділу інформаційних технологій Чернігівського обласного інституту післядипломної педагогічної освіти імені К.Д. Ушинського

**Рецензенти:**

**Горошко Ю.В.**, завідувач кафедри інформатики та обчислювальної техніки Чернігівського національного педагогічного університету імені Т.Г. Шевченка, доктор педагогічних наук, доцент

**Покришень Д.А.**, завідувач кафедри інформатики та інформаційно-комунікаційних технологій в освіті Чернігівського обласного інституту післядипломної педагогічної освіти імені К.Д.Ушинського, кандидат педагогічних наук, доцент

Збірник містить задачі та розв'язки ІІ етапу Всеукраїнської учнівської олімпіади з інформатики, що проводилась у Чернігівській області у 2014-2015 навчальному році.

Матеріали розраховані на вчителів інформатики, слухачів курсів підвищення кваліфікації. Можуть бути використані учнями 7-11 класів для підготовки до олімпіад.

*Рекомендовано до друку вченою радою Чернігівського обласного інституту післядипломної педагогічної освіти імені К.Д. Ушинського (протокол № 5 від 28.05.2015 р.)*

**ЗМІСТ**

# Вступ

Корисними матеріалами для підготовки до інтелектуальних змагань є збірники олімпіадних задач минулих років із розв'язками. Даний збірник містить задачі та розв'язки ІІ етапу Всеукраїнської учнівської олімпіади з інформатики, що проводилась у Чернігівській області у 2014-2015 навчальному році.

У збірнику викладені алгоритми розв'язання задач, реалізовані мовою програмування Паскаль. Матеріали допоможуть учителю і учню системно організувати підготовку до олімпіад з інформатики, підвищити свій рівень знань із програмування.

У збірнику пропонуються алгоритми розв'язання задач пробного туру, першого та другого варіантів. Розв'язки задач пробного туру та першого варіанту перевірялися за допомогою автоматизованої тестувальної системи. Кожен алгоритм розв'язання вказаних задач є коректним.

ІІ етап олімпіади проводився з використанням центрального серверу автоматизованої перевірки розв´язків. Задачі для пробного туру та ІІ етапу олімпіади (перший варіант) підібрані Петровим С.О., старшим викладачем кафедри комп´ютерних наук Сумського державного університету, кандидатом технічних наук.

Алгоритми розв´язання задач, що представлені в збірнику, запропоновані Бондаренком Сергієм Михайловичем, учителем математики та інформатики Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7, учителем-методистом та Зубом Володимиром Володимировичем, директором Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради, учителем математики, учителем-методистом.

# Задачі пробного туру ІІ етапу Всеукраїнської учнівської олімпіади з інформатики

***Усі програми-розв´язки написані в середовищі FreePascal 2.6.4.***

## Задача А-Гіпотенуза

**Обмеження часу:** 1 с

**Обмеження пам'яті:** 256 М

Дано два числа a та b. Виведіть гіпотенузу трикутника із заданими катетами з точністю 6 знаків.
Вхідні дані : два числа.
Вихідні дані: шукана величина.

**Приклади**

| Вхідні дані | Результат роботи |
|---|---|
| 3<br>4 | 5.000000 |

**Розв´язок**

Для розв´язання даної задачі потрібні знання теореми Піфагора та форматованого виводу величин на мові Паскаль.

1. Теорема Піфагора. У прямокутному трикутнику квадрат гіпотенузи дорівнює сумі квадратів катетів: $c^2=a^2+b^2$.

2. Дійсні числа записуються у форматі з плаваючою комою. Тому при звичайному виведенні результату, наведеного в прикладі, на екрані отримаємо запис типу 5.0000000000000000E+0000, що відповідає стандартному вигляду $5.0 \cdot 10^0$. Для виведення чисел у звичному вигляді використовують форматований вивід: **величина:n:m**, де n – загальна кількість позицій екрану, що використовуватимуться при виведенні величини, а m – кількість знаків після коми. Форматований вивід використовується лише для дійсних величин.

```
  var a,b:int64; c:real;
begin
  read(a,b);
  c:=sqrt(a*a+b*b);
write(c:0:6);
end.
```

---

**Задача B-Кролики**

**Обмеження часу:** 1 с

**Обмеження пам'яті:** 256 М

Коли земляни, нарешті, знайшли населену планету, вони назвали її OLYMP і відправили на неї разом із космічним кораблем одного кролика. Кролику сподобався клімат нової планети і через місяць він привів на світ ще одного кролика. Далі кролики продовжили розмножуватися з такою ж швидкістю, тобто кожен місяць кожен з кроликів, присутніх на планеті, приводив на світ ще одного кролика. Однак, розмноження кроликів стримував монстр, що звідкілясь взявся на планеті. Як тільки на початку якогось місяця кроликів ставало строго більше, ніж k, він приходив і з'їдав k кроликів. Визначте, скільки кроликів буде на планеті через n місяців після прибуття туди космічного корабля з першим кроликом. Число n від 0 до 100 включно. Число k від 0 до 10000 включно. Результат роботи методу не перевершує 2000000000.

Вхідні дані: ціле число n – кількість місяців, ціле число k – кількість кроликів, що з'їдаються монстром.

Вихідні дані: ціле число, рівне кількості кроликів на планеті OLYMP через n місяців після поселення туди першого кролика.

**Приклади**

| Вхідні дані | Результат роботи |
|---|---|
| 0 10 | 1 |
| 1 10 | 2 |
| 100 1024 | 2048 |
| 16 0 | 65536 |

**Розв´язок**

З умови задачі чітко слідує її циклічність. В умові також вказується кількість циклів, тому доречно використати цикл із параметром. У кожній ітерації потрібно подвоювати кількість кроликів і на початку кожної ітерації (місяця) потрібно перевіряти умову поїдання кроликів монстром.

```
  var i,n,k:longint;t:int64;
begin
  read(n,k);
  t:=1;
  for i:=1 to n do
    begin
      if t>k then t:=t-k;
      t:=t*2;
    end;
  write(t);
end.
```

---

## Задача C-Old hocus-pocus

**Обмеження часу:** 1 с

**Обмеження пам'яті:** 256 M

Петрик П’яточкин загадав число від 1 до $10^9$, а Вам повідомив три остачі, які утворилися при діленні загаданого числа на числа 971, 997, 1033. Зробіть фокус – швидко відгадайте число. Напишіть програму, що за даними остачами,

знаходить загадане число.

Вхідні дані: єдиний рядок вхідного потоку містить три натуральних числа.

Вихідні дані: єдиний рядок вихідного потоку має містити одне натуральне число.

**Приклади**

| Вхідні дані | Результат роботи |
|---|---|
| 5 10 15 | 835049324 |

**Розв´язок**

Перший варіант програми цілком може бути таким:

```
  var a,b,c,i:longint;
begin
  read(a,b,c);
  for i:=1 to 1000000000 do
    if (i mod 971 = a)and(i mod 997 = b)and(i mod
1033 = c)
       then begin write(i);break;end;
end.
```

Але при перевірці працездатності програми на вхідних даних побачимо, що час роботи програми далеко перевищує відведену 1с.

У курсі математики 6 класу в темі «Подільність натуральних чисел» розглядається дуже корисна для нас формула: $a=b \bullet n+r$. За її допомогою можна представити будь-яке число $a$ через неповну частку $n$ та остачу $r$ при його діленні на $b$. Застосування даної формули дасть можливість зменшити максимальну кількість ітерацій циклу з $10^9$ до менш, ніж $10^6$. До речі, якщо остачі від ділення певного числа на різні числа рівні, то ці остачі дорівнюють шуканому числу.

Спочатку знайдемо число, яке відповідає одній із умов і перевіримо його на відповідність іншим умовам. Найкраще вибирати найбільший дільник 1033. Тоді таким числом буде число $a=1033+15$ (див. вхідні дані). Воно не дає відповідних

остач при діленні на 997 та 971. Наступним буде число *a+1033*, і т.д. Тож у циклі розглядаємо числа типу *1033•n+r* і перевіряємо, чи дають вони відповідні остачі при діленні на 971 та 997.

```
  var a,b,c,i,j,k,x:longint;
begin
  read(a,b,c);
  if (a=b)and(b=c) then begin write(a);exit;end;
  x:=1033+c;
  while not((x mod 971=a)and(x mod 997=b)) do
x:=x+1033;
  write(x);
end.
```

---

## Задача D-Кількість чисел, що не діляться на 2, 3 або 5

**Обмеження часу:** 1 с

**Обмеження пам'яті:** 256 M

Задано натуральне число N. Напишіть програму, яка визначає кількість натуральних чисел, які не більші за N і не діляться ні на одне із чисел 2, 3, 5.

Вхідні дані: число N ($1 \le N \le 1000000000$).

Вихідні дані: знайдене число

**Приклади**

| Вхідні дані | Результат роботи |
|---|---|
| 10 | 2 |

**Розв´язок**

Як і в попередній задачі, можна застосувати циклічний підрахунок кількості чисел, що відповідають умові задачі. Але точно так само це потребує більше 1 с часу.

Розв’яжемо задачу за допомогою кіл Ейлера та теорієї множин із курсу математики 10 класу (академічний рівень) чи

8 класу (поглиблений рівень) (Рис. 1).

Нехай прямокутник являє собою множину натуральних чисел *N*, дану в умові задачі. Кола *B,C, D* – множини чисел, що діляться на 2, 3 та 5 відповідно. Тоді незафарбована частина прямокутника (множина A) є множиною шуканих чисел. *B∩C* – множина чисел, що діляться на 2 і на 3, тобто на 6; *B∩D* – діляться на 2 і на 5, тобто на 10; *C∩D* – діляться на 3 і на 5, тобто на 15; *B∩C∩D* – діляться на 2, на 3 і на 5, тобто на 30.

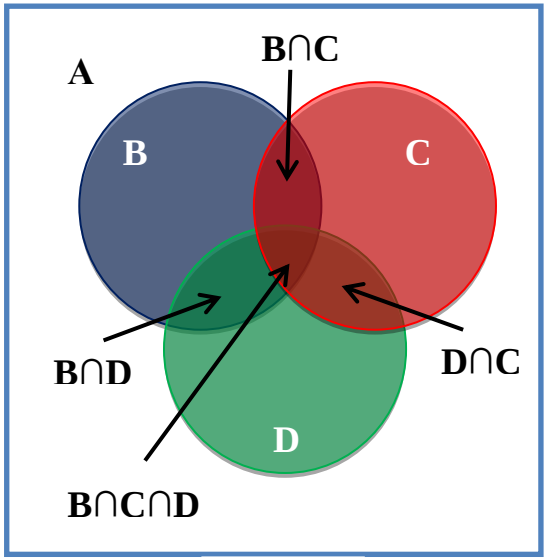


Рис. 1

Множина *B* містить *N/2* чисел, множина *C– N/3*, множина *D – N/5*. Для визначення кількості чисел, що входять в множину A досить скористатися формулою *K=N/2+N/3+N/5-N/6-N/10-N/15+N/30*.

```
  var a,n,i:int64;
begin
  read(n);
  a:=(n div 2)+(n div 3)+(n div 5)+(n div 30)-
(n div 6)-(n div 10)-(n div 15);
  write(n-a);
end.
```

### Задача E-Наступне і попереднє

**Обмеження часу:** 1 с

**Обмеження пам'яті:** 256 M

Напишіть програму, яка зчитує ціле число і виводе текст, аналогічний наведеному у прикладі. Пробіли, розділові знаки, великі і малі букви важливі!

**Приклади**

| Вхідні дані | Результат роботи |
|---|---|
| 25 | The next number for the number 25 is 26.<br>The previous number for the number 25 is 24. |

**Розв´язок**

```
  var n:int64;
begin
  read(n);
  writeln('The next number for the number ',n,
' is ',n+1,'.');
  write('The previous number for the number ',n,
' is ',n-1,'.');
end.
```

---

### Задача F-Сума цифр числа

**Обмеження часу:** 1 с

**Обмеження пам'яті:** 256 M

Задано чотирьохзначне число N. Напишіть програму, яка визначає суму цифр даного числа.

Вхідні дані: число N.

Вихідні дані: знайдена сума.

**Приклади**

| Вхідні дані | Результат роботи |
|---|---|
| 1012 | 4 |

**Розв´язок**

Класична задача на поділ числа на цифри. Згідно умови дано чотирицифрове число. Отже, можна просто написати чотири команди виокремлення цифр за допомогою операцій div та mod, а потім їх додати.

Нехай *n=1000a+100b+10c+d*. Тоді *a=n div 1000, b=n mod 100 div 100, c=n mod 100 div 10, d= n mod 10.*

У програмі наведено розв´язок для довільного цілого числа (у межах цілих типів мови Паскаль) і використовує цикл із передумовою. Для «довгих» чисел краще використовувати роботу з рядковим типом величин.

```
  var n,s:integer;
begin
  read(n);
  while n>0 do
    begin
      s:=s+n mod 10;
      n:=n div 10;
    end;
  write(s);
end.
```

# Задачі I (першого) варіанту II етапу Всеукраїнської учнівської олімпіади з інформатики

***Усі розв'язки-програми написані в середовищі FreePascal 2.6.4.***

## Задача A – Здача

**Обмеження часу:** 100 мс

**Обмеження пам'яті:** 128 M

Смачний сніданок у шкільній їдальні коштує ***A*** гривень і ***B*** копійок. Степан заплатив ***C*** гривень і ***D*** копійок. Напишіть програму, яка визначає здачу – гривень та копійок, що отримає Степан.

Формат вхідних даних: в єдиному рядку міститься чотири натуральних числа ***A, B, C, D (0 ≤ A, B, C, D ≤ 100)***.

Формат вихідних даних: виведіть два числа – здачу Степана.

**Приклади**

| Вхідні дані | Результат роботи |
|---|---|
| 1 30 2 50 | 1 20 |

**Розв´язок**

Найпростіший спосіб – перевести суми в копійки, відняти і перевести назад (у гривні та копійки).

```
  var a,b,c,d,e:longint;
begin
  read(a,b,c,d);
  e:=(c-a)*100+(d-b);
  write(e div 100,' ',e mod 100);
end.
```

## Задача B – Ігрові дні

**Обмеження часу:** 100 мс

**Обмеження пам'яті:** 128 M

Батьки дозволяють Степану грати за комп'ютером, якщо він отримав у цей день п'ятірку (у школі, в якій навчається Степан, п'ятибальна система оцінювання), але не отримав трійок (Степан двійок та одиниць не отримує). Напишіть програму, яка визначає чи зможе сьогодні Степан пограти за комп'ютером?

Формат вхідних даних: У першому рядку міститься одне натуральне число ***N (1 ≤ N ≤ 100)*** – кількість оцінок, які отримав Степан. У другому рядку записані ***N*** чисел – оцінки, які отримав Степан, кожна з яких 3, 4 або 5.

Формат вихідних даних: Виведіть "YES" – якщо Степан зможе пограти за комп'ютером, або "NO" в іншому випадку.

**Приклади**

| Вхідні дані | Результат роботи |
|---|---|
| 3<br>4 5 4 | YES |
| 4<br>5 3 5 4 | NO |

**Розв´язок**

Розв’язок задачі може бути реалізований різними способами. Наприклад, *S* – таблиця кількості оцінок 3, 4, 5 Степана. Комірка s[5] містить кількість п’ятірок, а комірка s[3] – кількість трійок (решта оцінок не важливі). Відповідно до умови Степан гратиме на комп’ютері лише у випадку, коли s[5]>0, а s[3]=0.

Для коректної роботи рекомендується "очистити" таблицю (заповнити її нулями). Це можна робити за допомогою циклу. Але для великих таблиць, особливо дво- чи тримірних краще використовувати процедуру fillchar.

*procedure FillChar (var Buffer; Count: Integer; const Fill);*

*Опис: Buffer X - буфер, який потрібно заповнити, Count – кількість символів, Value – заповнювач (типу Byte або Char).*

```
  var n,a,i:integer;s:array[3..5] of integer;
begin
  read(n);
  fillchar(s,sizeof(s),0);{заповнення таблиці S
нулями}
  for i:=1 to n do
    begin
      read(a); {читання чергової оцінки Степана}
      inc(s[a]); {збільшення значень комірок s[3],
s[4], s[5] таблиці на 1 в залежності від a}
    end;
  if (s[5]>0) and (s[3]=0) then write ('YES') else
write ('NO');
end.
```

---

**Задача C – Податок**

**Обмеження часу:** 100 мс

**Обмеження пам'яті:** 128 M

У деякій країні інфляція досягла таких розмірів, що доходи громадян стали виражатися числами, кількість знаків десяткового запису яких доходить до 200. Це сильно ускладнило завдання збору податків. Один із податків на доходи складає 1%. Напишіть програму, яка за введеним числом ***D*** (величині доходу громадянина) обчислить цей податок. При цьому застосовуються такі правила округлення:

1. Якщо податок виражається цілим числом, то він не округлюється.
2. Якщо податок виражається дробовим числом, то він округлюється в бік більшого цілого (на користь держави).

Формат вхідних даних: У першому рядку міститься одне натуральне число $D$ $(10^5 \le D \le 10^{200})$– величина доходу громадянина.

Формат вихідних даних: Виведіть одне натуральне число – величину податку.

***Система оцінювання:*** Розв´язки, які вірно працюють при $D$ $(10^5 \le D \le 10^9)$, будуть оцінюватись у 40 балів. Розв´язки, які вірно працюють при $D$ $(10^5 \le D \le 10^{15})$, будуть оцінюватись у 60 балів.

**Приклади**

| **Вхідні дані** | **Результат роботи** |
|---|---|
| 1000001 | 10001 |
| 12345600 | 123456 |

**Розв´язок**

Алгоритм дій прописаний в умові – якщо остача від ділення числа D на 100 ≠ 0, то цілу частину потрібно збільшити на 1, інакше залишити від змін. Для отримання 40 балів достатньо було використати тип *longint* і написати код приблизно так:

```
p:= d div 100;
if d mod 100 <> 0 then p:=p+1;
```

При використанні *int64* можна отримати 60 балів. Використання інших числових типів принципового виграшу не дає, оскільки жоден із них не має точності в необхідні 200 знаків. Отже, використовуємо символьні величини.

Поділити число, записане в символьному вигляді, на 100 просто – досить відрізати останні два символи. Якщо це були два нулі, то отримане число і є шуканим. В іншому разі потрібно збільшити отримане число на 1. Розглянемо два випадки:

1) остання цифра отриманого числа не 9, тоді її достатньо збільшити на 1;

2) остання цифра 9. Тоді все трохи складніше, адже 9+1=10. Тобто, при додаванні одиниці розглядають попередню цифру і здійснюють перехід через десяток. А це знову два випадки. Отже, організовується цикл, в якому ми перевіряємо цифри з кінця числа. Поки вони рівні 9, їх потрібно замінювати нулями. Першу ж цифру, яка не є дев'яткою потрібно просто збільшити на 1.

Також потрібно передбачити випадок, коли всі цифри числа – дев'ятки. Тоді ми отримаємо самі нулі – не забути добавити

першою одиницю!!!

*function Ord ( Arg : Char) : Integer;*

*Опис : функція Ord, перетворює символ Arg в його числовий код. Коди всіх символів можна побачити в кодовій таблиці ASCII. ASCII (American Standard Code for Information Interchange) – міжнародний стандарт, що прийнятий для кодування текстової інформації. Усього в ній 256 символів.*

*function Chr ( IntValue : Integer ) : AnsiChar;*

*Опис: функція Chr протилежна функції Ord. Ця функція перетворює числовий код IntValue символу в сам символ.*

```
  var a,b:string;n,i,j:integer;
begin
  read(a);
  n:=length(a)-2; {кількість цифр числа податку}
  b:=copy(a,1,n);{величина податку}
  if copy (a,n+1,2)<>'00'
    then if b[n]<>'9'
          then b[n]:=chr(ord(b[n])+1) {додавання 1
до даного розряду числа}
          else begin
               b[n]:='0';j:=1;
               while (n-j>0) and (b[n-j]='9') do
                 begin
                   b[n-j]:='0'; inc(j);
                 end;
           if j<>n then b[n-j]:=chr(ord(b[n-j])+1)
           else b:='1'+b;
                end;
  write(b);
end.
```

## Задача D – Будівництво школи

**Обмеження часу:** 100 мс

**Обмеження пам'яті:** 128 M

В Ужляндії всі будинки розташовані вздовж однієї вулиці по одну сторону від неї. По інший бік від цієї вулиці поки нічого немає, але скоро все буде – школи, магазини, кінотеатри і т.д. Для початку в Ужляндії вирішили побудувати школу. Місце для будівництва школи вирішили вибрати так, щоб сумарна відстань, яке проїжджають учні від своїх будинків до школи, була мінімальною. План Ужляндії можна представити у вигляді прямої, у деяких цілочисельних точках якої знаходяться будинки учнів. Школу також дозволяється будувати тільки в цілочисленній точці цієї прямої (у тому числі дозволяється будувати школу в точці, де розташований один з будинків – адже школа буде розташована з іншого боку вулиці). Напишіть програму, яка за відомими координатами будинків учнів допоможе визначити координати місця будівництва школи.

Формат вхідних даних: Спочатку вводиться число $N$ $(1 \le N \le 10^5)$ – кількість учнів. Далі йдуть у строго зростаючому порядку координати будинків учнів – цілі числа, що не перевищують $2*10^9$ по модулю.

Формат вихідних даних: Виведіть одне ціле число – координату точки, в якій найкраще побудувати школу. Якщо відповідей декілька, виведіть будь-яку з них.

***Система оцінювання:*** Розв´язки, які вірно працюють при $N$ $(1 \le N \le 10^3)$ для координат, які не перевищують по модулю 1000, будуть оцінюватись у 30 балів. Розв´язки, які вірно працюють при $N$ $(1 \le N \le 10^5)$ для координат, які не перевищують по модулю 100000, будуть оцінюватись у 70 балів.

**Приклади**

| Вхідні дані | Результат роботи |
|---|---|
| 4<br>1 2 3 4 | 2 |
| 3<br>-1 0 1 | 0 |

**Розв´язок**

Як початковий варіант можна запропонувати повний перебір: обчислимо всі можливі суми для кожної цілочисельної точки та виберемо серед них найменшу. Варіант такого розв'язку наведено нижче. Зрозуміло, що він далеко не

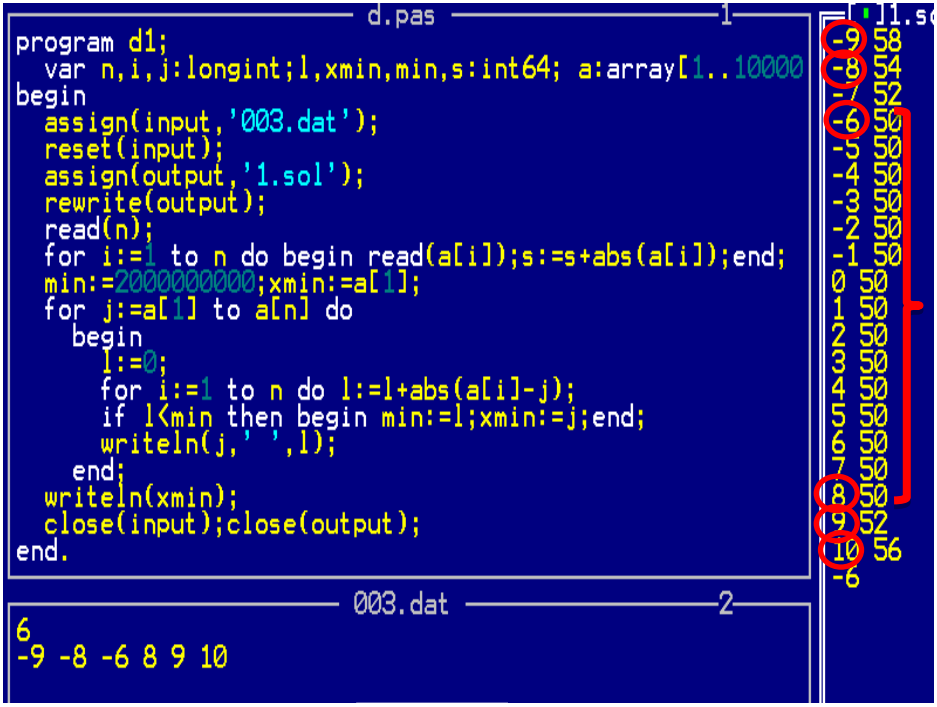


Рис. 2

оптимальний, але певну кількість балів він отримує.

```
  var n,i,j:longint;l,xmin,min:int64;
a:array[1..10000] of int64;
begin
  read(n);
  for i:=1 to n do read(a[i]);
  min:=2000000000;xmin:=a[1];
  for i:=a[1] to a[n] do
    begin
      l:=0;
      for j:=1 to n do l:=l+abs(a[j]-i);
      if l<min then begin min:=l; xmin:=i; end;
    end;
  write(xmin);
end.
```

У чому ж проблеми такого розв'язку? По-перше, потрібно виконати $(10^5)^2 = 10^{10}$ ітерацій, а це досить багато – програма не вкладеться у відведений час. По-друге, не виключено, що поточна сума може вийти за межі *int64*. І що ж робити? Використаємо вже написану програму повного перебору для аналізу розміщення школи для різних варіантів проживання учнів.

На Рис. 2 показано результат роботи програми для парної кількості учнів. Червоним виділені координати проживання учнів та місце побудови школи.

Як бачимо, при шести учнях найменшу відстань дають усі точки, розміщені між третьою та четвертою координатами. Зміна координат будь-якої точки даного прикладу не змінюють загального принципу розміщення школи – при парному N школу можна розмістити на довільній точці відрізку
[N div 2; N div 2+1].

Розглянемо приклад із непарною кількістю точок.

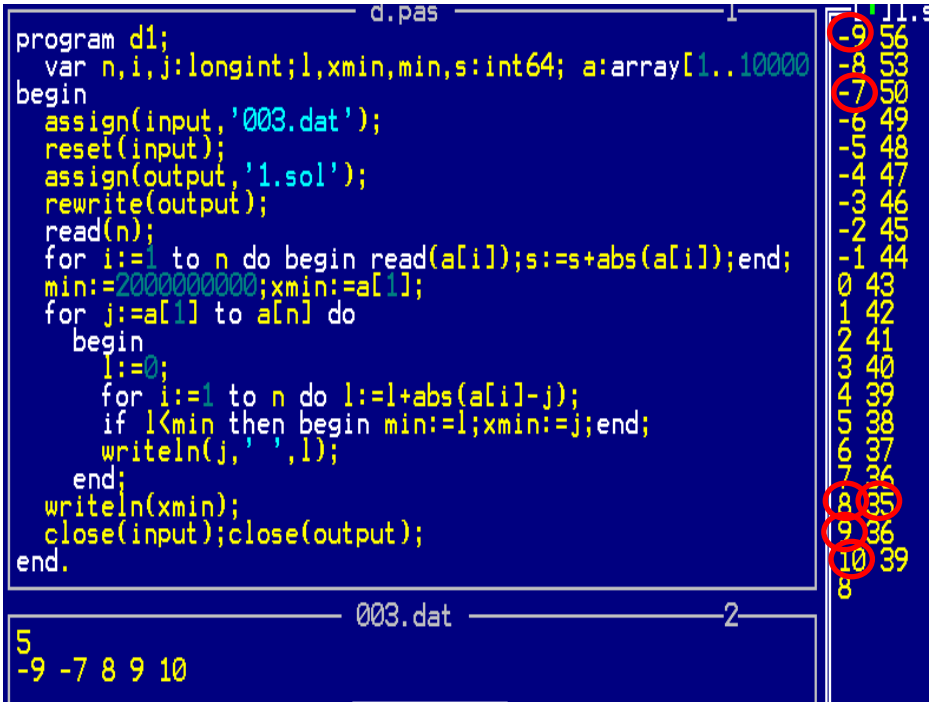


Рис. 3

Маніпуляції з координатами при непарній кількості учнів (Рис.3) підтверджують припущення – школу потрібно будувати в точці, яка відповідає середній комірці таблиці, даній в умові.

Тож маємо наступну програму.

```
  var n,i,j:longint;l,xmin,min,s:int64;
a:array[1..10000] of longint;
begin
  read(n);
  for i:=1 to n do read(a[i]);
  if n div 2 = 0 then write(a[n div 2]) else
write(a[n div 2 +1]);
end.
```

## Задача E-Свято

**Обмеження часу:** 1000 мс

**Обмеження пам'яті:** 128 M

У школах Ужляндії є цікаве свято. 20 жовтня в перший день початку інтернет-олімпіади дівчата випікають пряники, печиво і інші "вкусняшки", а потрім пригощають ними своїх однокласників. Випічка, приготовлена цього дня, зазвичай, має форму трикутників або прямокутників.

Софія приготувала на це свято N пряників різної форми і розміру і підготувала відповідну кількість однакових круглих коробок, щоб упакувати по одному прянику в коробку. Одне, що не врахувала Софія, це те, що деякі пряники не поміщаються в підготовлені коробки.

Допоможіть Софії визначити, які пряники помістяться в коробки, а які ні.

Формат вхідних даних: У першому рядку записано діаметр упаковочної коробки, а в другому – кількість приготовлених Софією пряників $N$ $(1 \le N \le 100)$. Кожен із наступних $N$ рядків містить опис одного пряника. Якщо пряник має форму трикутника, то на початку рядка записується число 1, а потім – довжини сторін цього трикутника (трикутник невироджений). Для прямокутного пряника на початку рядка записано число 2, а потім довжини суміжних сторін прямокутника. Числа розділені одним пробілом. Усі розміри – цілі додатні числа, які не перевищують $10^{17}$ (у 80% тестів ця величина не більша $10^3$). У 25% тестів пряники мають форму прямокутника.

Формат вихідних даних: Виведіть рядок із $N$ символів. Кожен символ рядка відповідає одному прянику (у порядку введення даних). Символ "Y" означає, що пряник можна помістити в коробку, а символ "N" – що пряник помістити не можна.

**Приклади**

| Вхідні дані | Результат роботи |
|---|---|
| 20<br>2<br>2 15 17<br>2 19 5 | NY |
| 20<br>4<br>1 20 12 16<br>1 10 10 10<br>1 20 20 20<br>2 13 15 | YYNY |

**Розв´язок**

**Випадок перший – прямокутник (Рис. 4).** Пряник можна помістити в коробку, якщо його діагональ не перевищує діаметр коробки. За відомими сторонами прямокутника *a* і *b* знайти його діагональ *c* можна за допомогою теореми Піфагора: $c = \sqrt{a^2 + b^2}$. Гарна формула і проста, але при значеннях аргументів більших за $10^9$ вона не надає необхідної точності обчислення. Краще обчислити діагональ за формулою $c = \frac{a}{\sin \alpha}$. Із обернених тригонометричних функцій мова Паскаль має лише *arctg*, тому $\alpha = arctg\frac{a}{b}$.

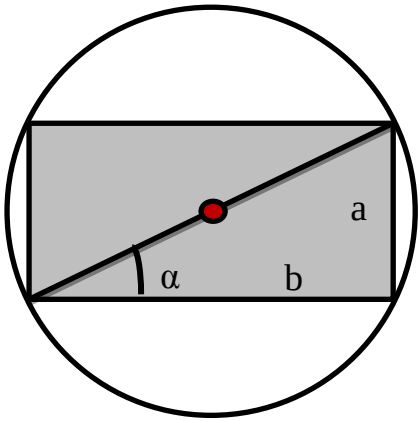

Рис. 4

**Випадок другий – трикутник (Рис.5).**

1. Якщо трикутник тупокутний чи прямокутний, то маємо

випадок, схожий із прямокутником: достатньо щоб найбільша сторона була не більшою за діаметр коробки. Отже, спочатку визначаємо найбільшу сторону і порівнюємо її з діаметром коробки.

2. Якщо трикутник гострокутний (Рис.6), то шукаємо діаметр описаного кола. Це краще зробити за теоремою синусів, а кут – за допомогою теореми косинусів:

$2R = \frac{a}{\sin\propto}$, $\sin\propto = \sqrt{1 - cos^2\propto}$, $\cos\propto = \frac{b^2+c^2-a^2}{2bc}$.

Останню формулу краще записувати у вигляді $\cos\propto = \frac{b}{2c} + \frac{c}{2b} - \frac{a}{2b}\cdot\frac{a}{c}$.

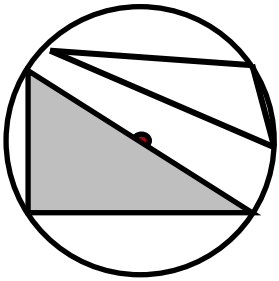

Рис. 5

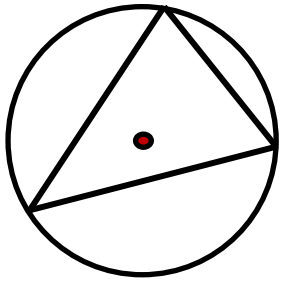

Рис. 6

```
  var n,i,t:integer; d1,d,cosc,x,y,z:extended;
a,b,c:int64;
begin
  read(d,n);
  for i:=1 to n do
    begin
      read(t,a,b);{читаємо тип фігури і перші дві
сторони}
      if t=1 then {випадок трикутника}
         begin
           read(c);{читаємо третю сторону і
знаходимо найбільшу}
```

```
           if (a>=b) and (a>=c) then begin x:=a;
y:=b; z:=c;end;
           if (b>=a) and (b>=c) then begin
x:=b;y:=a;z:=c; end;
           if (c>=b) and (c>=a) then begin
x:=c;y:=b;z:=a; end;
           if x*x>=y*y+z*z then begin {перевіряємо
тип трикутника}
              if x<=d then write('Y') else
write('N'); end
              else begin
             cosc:=a/2/b-c/2/a*c/b+b/2/a;
             d1:=c/sqrt(1-cosc*cosc);
                      if d1<=d then write('Y')
                               else write('N');
                  end;
               end
     else begin {випадок чотирикутника}
         d1:=a/sin(arctan(a/b));
         if d1<=d then write('Y') else write('N');
          end;
    end;
end.
```

# Задачі ІІ (другого) варіанту ІІ етапу Всеукраїнської учнівської олімпіади з інформатики

***Усі програми-розв´язки написані в середовищі FreePascal 2.6.4.***

## Задача А – Золотий акваріум

*(автор Корнієць О.М., старший викладач кафедри інформатики та інформаційно-комунікаційних технологій в освіті Чернігівського обласного інституту післядипломної педагогічної освіти імені К.Д.Ушинського)*

У Петрика скоро день народження і він на наступній неділі разом із батьками піде купувати собі акваріум із золотими рибками. Однак, нещодавно на природознавстві Петрик дізнався, що для нормального розведення золотих рибок потрібно, щоб на кожну рибку в акваріумі припадало не менше 3-х літрів води. Допоможіть Петрику визначити допустиму кількість золотих рибок N, залежно від об'єму, вибраного ним акваріума ***V***.

Вхідні дані: Єдине ціле число ***V***– об'єм вибраного акваріума.

Вихідні дані: Єдине ціле число ***N*** - допустима кількість золотих рибок.

Обмеження: 20 < V < 500.

**Приклад вводу**:
22

**Приклад виводу**:
7

Підказка: Очевидно, що допустима кількість рибок повинна бути цілим числом.

**Розв´язок**

Оскільки кількість рибок має бути цілим число і на кожну рибку повинно припадати не менше 3 л води, то максимальна кількість рибок шукається як ціла частина від ділення об'єму

акваріума на 3.

```
  var n,v:integer;
begin
  read(v);
  n:=v div 3;
  write(n);
end.
```

---

**Задача В – Весела абетка**

*(автор Корнієць О.М., старший викладач кафедри інформатики та інформаційно-комунікаційних технологій в освіті Чернігівського обласного інституту післядипломної педагогічної освіти імені К.Д. Ушинського)*

Петрик навчається в початковій школі. Він ще не дуже добре знає англійську абетку і тому дуже засмутився, коли Ольга Павлівна (учителька англійської мови) попросила його переставити літери слова S за абеткою (від «A» до «Z»). Допоможіть Петрику впоратися з завданням.

Вхідні дані: У рядку міститься слово S.

Вихідні дані: У рядку міститься слово, яке утворилось під час перестановки літер.

Обмеження: слово S містить від 1 до 10 символів, тільки маленькі латинські літери.

**Приклад вводу**:
champion

**Приклад виводу**:
Achimnop

Підказка: Очевидно, що слово, яке утворилося під час перестановки літер, містить стільки ж символів як початкове.

**Розв´язок**

Оскільки S – рядок, то це величина типу string, тобто таблиця символів. Для її впорядкування за абеткою можна

скористатися довільним методом сортування. Але є оптимальний спосіб, який гарантовано це зробить "за один прохід". Для цього використаємо таблицю на 26 символів англійського алфавіту і в її комірки будемо записувати кількість відповідних літер вхідного слова. Наприклад, для даного в умові слова: a['a']=1, a['b']=0, a['c']=1… Таким чином, нам не доведеться впорядковувати таблицю, що суттєво зекономить час (хоча для даних обмежень це не принципово).

Елементи таблиці індексуються довільним перерахованим типом даних. Найчастіше для цього використовують цілочисельні величини. У даній задачі ми використовуємо літери англійського алфавіту. Це впорядкований перерахований тип, тому індекси таблиці цілком можуть йому належати. Тож оголосимо таблицю *a : array['a'..'z'] of integer.*

Для виведення результату потрібно лише вивести літери, у комірках, яких значення більші за нуль. Оскільки ці значення можуть відрізнятися від 1, то це краще робити в циклі.

Для коректної роботи рекомендується "очистити" таблицю (заповнити її нулями). Це можна робити за допомогою циклу. Але для великих таблиць, особливо дво- чи тримірних, краще використовувати процедуру fillchar.

*procedure FillChar (var Buffer; Count: Integer; const Fill);*

*Опис: Buffer X - буфер, який потрібно заповнити, Count – кількість символів, Value – заповнювач (типу Byte або Char).*

```
  var s:string; a:array['a'..'z'] of integer;
i:integer; j:char;
begin
  read(s);{читаємо слово}
  fillchar(a,sizeof(a),0);{очищуємо таблицю}
  for i:=1 to length(s) do
inc(a[s[i]]);{збільшуємо на 1 значення комірки, що
відповідає і-й літері даного слова}
  for j:='a' to'z'do
    if a[j]>0 then {це необов'язковий рядок,
оскільки тіло циклу з параметром не виконується,
```

```
якщо кінцеве значення параметра менше за
початкове, тобто при a[j]<1}
      for i:=1 to a[j] do write(j);{виводимо
результат}
end.
```

---

**Задача С – Суми**

*(Задача з ІІ етапу Всеукраїнської учнівської олімпіади 2010-2011 навчального року у Львівській області)*

Петрику задано масив чисел, із нього будують квадратну таблицю таким чином, щоб елемент, який знаходиться на перетині і-ого рядка та j-ого стовпчика, дорівнював добутку і-ого та j-ого елементів заданого масиву. Ваше завдання полягає в тому, щоб порахувати суму елементів нижньої трикутної частини, тобто елементів, для яких індекс стовпчика є не більшим за індекс рядка.

Нехай є масив чисел a = {2, 1, 4. 3}. Запишемо, як відповідно до умови задачі, формуватиметься таблиця для даного масиву.

| | **2** | **1** | **4** | **3** |
|---|---|---|---|---|
| **2** | 2*2 = 4 | 2*1 = 2 | 2*4 = 8 | 2*3 = 6 |
| **1** | 1*2 = 2 | 1*1 = 1 | 1*4 = 4 | 1*3 = 3 |
| **4** | 4*2 = 8 | 4*1 = 4 | 4 * 4 = 16 | 4 * 3 = 12 |
| **3** | 3*2 = 6 | 3*1 = 3 | 3 * 4 = 12 | 3*3 = 9 |

Сума елементів нижньої трикутної частини(елементи якої виділені сірим кольором) 4 + 2 + 1 + 8 + 4+ 16 + 6 + 3 + 12 + 9 = 65 .

Вхідні дані: У першому рядку задано число N – кількість елементів масиву. У наступному рядку задано N чисел, розділених пробілом, – елементи масиву $A_j$.

Вихідні дані: Єдине ціле число.

Обмеження:

$1 \le N \le 100000$,

$0 \le A_j \le 100$.

**Приклад вводу:**
4
2 14 3

**Приклад виводу:**
65

**Розв´язок**

Розглянемо таблицю з умови завдання в компактнішому вигляді і проаналізуємо.

| | | | |
|---|---|---|---|
| 4 | 2 | 8 | 6 |
| 2 | 1 | 4 | 3 |
| 8 | 4 | 16 | 12 |
| 6 | 3 | 12 | 9 |

Комірки, для яких *i=j*, утворюють так звану головну діагональ. Оскільки від перестановки множників добуток не змінюється, то *b[i,j]=b[j,i]*. Тобто, таблиця симетрична відносно головної діагоналі. Обчислювати елементи самої таблицю не має потреби. Для обчислення шуканої суми використаємо подвійний цикл з умовою $1 \le i \le j \le n$.

```
  var n,i,j:longint; s:int64; a:array[1..100] of
longint;
begin
  read(n);
  for i:=1 to n do read(a[i]);
  s:=0;
  for i:=1 to n do
    for j:=i to n do
      s:=s+a[i]*a[j];
  write(s);
end.
```

## Задача D – Продаж слоників

*(Задача з ІІ етапу Всеукраїнської учнівської олімпіади 2010-2011 навчального року у Львівській області)*

Темніє. Сонце ще пускає останні проміння світла. На дворі вже ні душі. А Петрик П'яточкін далі рахує слоників!

Вирішивши, що варто відпочити, він пішов додому дивитися перед сном ТБ. І тут на екрані пройшов рекламний рядок, який він одразу ж записав. Невже розпродаж слоників?! Так як він майже спав, він міг переплутати деякі букви, тому його цікавить таке питання: скількома способами можна вибрати K послідовних символів рядка, з букв яких можна скласти слово "slonyk" (символи в підрядку можна міняти місцями).

Вхідні дані: Перший рядок містить рекламний рядок S, який записав Петрик. Цей рядок складається з пробілів та маленьких букв англійського алфавіту. Другий рядок входу містить одне ціле число – K.

Вихідні дані: Єдине число – кількість можливих підрядків за вимогами Петрика.

Обмеження:

$1 \le K \le 10$,

$1 \le$ довжина рядка $S \le 44$.

**Приклад вводу:**

solkynssaleuptotendollars

9

**Приклад виводу:**

2

**Розв´язок**

Робота з символьними даними вимагає знань деяких функцій:

*Function Pos (Substr : String; S : String) : Byte;*

*Опис : Шукає позицію підрядка Substr у рядку S. Параметри Substr і S – рядкові вирази. Pos шукає перше входження рядка Substr у рядку S і повертає цілочисельне значення, яке є індексом*

*першого символа Substr у середині S. Якщо рядок Substr не знайдено, то Pos повертає нуль.*

*Function Copy (S : String; Index : Integer; Count : Integer) : String;*

*Опис: Повертає підрядок рядка. Параметр S – вираз рядкового типу. Index і Count – вирази цілочисельного типу. Функція Copy повертає підрядок рядка S, що містить Count символів, починаючи з символу з номером Index. Якщо значення Index більше, ніж довжина рядка S, то Copy повертає порожній рядок. Якщо значення Count більше, ніж кількість символів, що залишились у рядку з позиції Index до кінця рядка, то повертається Length(S)-Index символів.*

*Function Length(S: String : Integer;*

*Опис: Повертає довжину рядка.*

Отже, маємо послідовно перевіряти слова, утворені копіюванням послідовних *n* символів вхідного рядка, починаючи з першого і до *length(s)-n*, і в отриманому, таким чином, слові перевіряти можливість утворення слова *slonyk* переставлянням літер.

Алгоритм розв'язання задачі розіб'ємо на два кроки.

1. Отримання слова довжиною *n* з рядка *S*.
2. Перевірка наявності всіх літер слова *slonyk* у слові, одержаному на першому кроці. Для вирішення даного завдання використаємо функцію *pos*, оскільки ця функція дає нульовий результат при відсутності шуканої літери у рядку, то досить знайти добуток позицій усіх символів слова *slonyk* у скопійованому слові. Добуток буде дорівнювати нулеві при відсутності хоч однієї літери!

```
  var i,k,n:integer; s,s1:string;
begin
  readln(s);
  read(n)
  k:=0;
```

```
  for i:=1 to length(s)-n do
    begin
      s1:=copy(s,i,n);
      if
pos('s',s1)*pos('l',s1)*pos('o',s1)*pos('n',s1)*
*pos('y',s1)*pos('k',s1)>0
        then inc(k);
    end;
  write(k);
end.
```

# Література

1. Герасимчук Н.О. Розв´язання олімпіадних задач з програмування: навчальний посібник для слухачів відділення комп´ютерних наук МАН. – Луцьк: ВО МАН, 2010. – 76 с.

2. Караванова Т.П. Методика розв´язування алгоритмічних задач. Основи алгоритмізації та програмування: навчально-методичний посібник для вчителів / Т.П. Караванова. – Кам´янець-Подільський : Аксіома, 2013. – 460 с.

3. Караванова Т.П. Методика розв´язування алгоритмічних задач. Побудова алгоритмів: навчально-методичний посібник для вчителя / Т.П. Караванова. – Кам´янець-Подільський: Аксіома, 2014. – 344 с.

4. Олімпіадні задачі з інформатики: розв´язання задач ІІ етапу Всеукраїнської олімпіади з інформатики 2007, 2008рр./ В.Є. Величко, М.М. Рубан, В.П. Батуніна, С.Є. Устінов. – Слов´янськ, 2009. – 34 с.

5. Пасіхов Ю.Я. Олімпіадні задачі з інформатики / Юрій Пасіхов, Григорій Непомнящий. – К. : Шк. Світ, 2011. – 128 с. – (Бібліотека «Шкільного світу»).

6. Рекомендації до розв´язування задач Міжнародних і Всеукраїнських олімпіад з інформатики серед учнів: навч.- метод. посіб. – К.: ТОВ Редакція «Комп´ютер», 2008. – 128 с.:іл..

# Список рекомендованих ресурсів мережі Інтернет

1. ACM-Контестер: Український портал ACM-спільноти [Електронний ресурс]. – Режим доступу: http://acm.lviv.ua/ – Назва з екрану.

2. Дистанционная подготовка по информатике [Електронний ресурс]. – Режим доступу: http://informatics.mccme.ru/moodle/ – Назва з екрану.

3. Дніпропетровські олімпіади з інформатики [Електронний ресурс]. – Режим доступу: http://oi.dp.ua/ – Назва з екрану.

4. Інтернгет-портал організаційно-методичного забезпечення дистанційних олімпіад з програмування для обдарованої молоді навчальних закладів України [Електронний ресурс]. – Режим доступу:http://e-olymp.com/ – Назва з екрану.

5. Київські учнівські олімпіади з інформатики. Проведення ві результати [Електронний ресурс]. – Режим доступу: http://kievoi.ippo.kubg.edu.ua/ – Назва з екрану.

6. Матеріали Всеукраїнських учнівських олімпіад з інформатики [Електронний ресурс]. – Режим доступу: http://matholymp.org.ua/contests/types/olympiads/informatics/ – Назва з екрану.

7. Центр підтримки та проведення олімпіад школярів з використанням можливостей Internet [Електронний ресурс]. – Режим доступу: http://www.olymp.vinnica.ua/ – Назва з екрану.

8. Школа программиста [Електронний ресурс]. – Режим доступу: http://acmp.ru/ – Назва з екрану.

9. Timus Online Judge: архив задач с проверяющей системой [Електронний ресурс]. – Режим доступу:http://acm.timus.ru/ – Назва з екрану.