Обласні олімпіади з інформатики на Чернігівщині

Горошко Ю.В., Костюченко А.О.

Вже декілька років викладачів кафедри інформатики і ОТ Чернігівського державного педуніверситету імені Т.Г.Шевченка запрошують до роботи у журі обласної олімпіади з інформатики серед школярів. В цій статті ми поділимося враженнями про олімпіади, в яких ми брали участь, та наведемо деякі задачі, які пропонувалися до розв'язання.

На даний момент сформувалося постійне журі олімпіади, кістяк якого складають викладачі кафедри інформатики і ОТ – Горошко Ю.В. (голова журі), Вінниченко Є.Ф., Костюченко А.О., Лукаш І.М. та Шкардибарда М.І. Також у роботі журі приймає участь викладач Ніжинського держуніверситету Харченко В. та кращі вчителі інформатики області. Олімпіади з інформатики багато років організовувала Данькевич С.В., нині цим займається Євтушенко Н.В. – методист з інформатики ЧОІППО (учитель вищої категорії, має звання учитель-методист).

Хотілося б відмітити вчителів, які працюють творчо і учні яких зайняли призові місця на останній обласній олімпіаді. Це Мальована Ганна Петрівна, учитель інформатики ліцею №22 м. Чернігова; Іштукін Валерій Володимирович – учитель інформатики ЗОШ І-ІІІ ст. №6 м. Прилуки, його учень Захожий Ігор мав диплом ІІ ступеня на ІІІ етапі; Бондаренко Сергій Михайлович – учитель інформатики ЗОШ І-ІІІ ст. №7, м. Прилуки, написав досить багато книжок по розв'язанню олімпіадних задач, його учень – Ляшенко Павло – диплом ІІІ ступеня. Донченко Сергій Сергійович – учитель інформатики Новгород-Сіверської гімназії №1, учень – Захожий Ігор Олександрович, диплом ІІ ступеня.

Складанням задач для обласних олімпіад займається ініціативна група, в яку входять Горошко Ю.В., Вінниченко Є.Ф., Костюченко А.О., Лукаш І.М., Шкардибарда М.І., Харченко В. Наведемо деякі задачі, що пропонувалися учням для розв'язування на обласних олімпіадах.

Задача „Терези” (запропонована Горошком Ю.В.). В літературі ця задача має назву задачі Баше-Мєндєлєєва.

Леонардо Пізанський, Лука Пачулі та ін. розглядали задачу про найкращу систему важків. Нею цікавився і Д.Мєндєлєєв, коли був директором Головної палати мір та вагів. Французький математик Клод Баше опублікував цю задачу у „Збірнику цікавих задач”, виданому у 1612 р. Ця задача існує у двох варіантах. Спочатку розглянемо більш простий, а саме, при якій системі важків, маючи їх по одному, можна зважити всі можливі вантажі, до деякого найбільшого, якщо класти важки тільки на одну шальку терезів? Маються на увазі цілочисельні вантажі.

Зрозуміло, що максимальний вантаж, який може бути зважений даною системою важків, дорівнює їх сумі. Таким чином, треба підібрати систему важків такою, щоб їх кількість була мінімальною, і за допомогою неї можна було б зважити всі цілі вантажі від 1 до максимального. Підібрати таку систему важків можна, наприклад, наступними міркуваннями. Нехай за допомогою системи важків можна зважити будь-який вантаж від 1 до M. Спробуємо вибрати наступний важок максимально великим. Це буде важок M+1. Якщо взяти важок більшої маси, наприклад, M+2, тоді вантаж масою M+1 не може бути зваженим. Так ми отримаємо систему важків 1, 2, 4, 8, ..., тобто степенів двійки. Отже, зважування вантажу за допомогою такої системи важків зводиться до запису маси цього вантажу у двійковій системі числення. Оскільки будь-яке десяткове число можна подати у вигляді двійкового числа, то доведено, що така система важків дозволить зважити будь-яке число від 1 до суми мас всіх важків. Оптимальність такої системи випливає з процесу її побудови. Процес зважування конкретного вантажу зводиться до запису його маси у двійковому вигляді, що є досить легкою задачею.

Більш складний варіант задачі можна сформулювати наступним чином: при якій системі важків, маючи їх по одному, можна зважити всі можливі

вантажі, до деякого найбільшого, якщо можна класти важки на обидві шальки терезів?

Положити важок на ту ж шальку, де знаходиться вантаж, означає відняти її масу від маси важків, що знаходяться на іншій шальці.

Для вибору системи важків можна скористатися наступними міркуваннями. Нехай за допомогою системи важків можна зважити будь-який вантаж від 1 до M. Знову ж таки спробуємо вибрати наступний важок з максимально можливою масою. У попередній задачі це був важок масою на 1 більшою за вантаж, який можна було положити на ліву шальку. Але тепер на ліву шальку можна класти не тільки вантаж, але і важки, тому додамо до максимального вантажу суму всіх важків і виберемо наступний важок на одиницю більшим за цю суму, тобто M+M+1. Отримаємо систему важків 1, 3, 9, 27, 81, ..., тобто степені трійки. Таким чином процес зважування аналогічний запису числа у трійковій системі числення, але не звичайній (з цифрами 0, 1, 2), а системі з цифрами -1, 0, 1 (-1 – важок лежить на тій же шальці, де і вантаж, 0 – важок не бере участь у зважуванні, 1 – важок на сусідній з вантажем шальці). Виникає питання, чи можна подати будь-яке натуральне число у такій специфічній трійковій системі числення. Відповідь позитивна, можна запропонувати такий спосіб переходу від звичайної трійкової системи числення до специфічної для цієї задачі: якщо маємо на будь-якому місці запису числа у звичайній трійковій системі числення 2, ми можемо додати і відняти 1 і отримати одну одиницю вищого розряду і від'ємну одиницю попереднього розряду. Так, наприклад,

$12021_3=121(-1)1_3=2(-1)1(-1)1_3=1(-1)(-1)1(-1)1_3.$

На олімпіаді було запропоновано більш складний варіант задачі у наступному вигляді.

Завдання:

1. Визначити мінімальний набір важків для зважування за допомогою терезів предметів масою в межах від 1 до 1000000000 г (маса – натуральне число). Дані про важки розмістити у файлі

„giriset.sol”, що складається з n+1 рядка: перший рядок – кількість важків n, другий рядок – маса 1-го важка, 3-й рядок – маса 2-го важка, ... n+1 рядок – маса n-го важка.

2. Розробити програму “giri”, що розподіляє знайдені важки між терезами для зважування предмету вказаної маси. Вхідні дані – файл „giri.dat”, що містить єдине ціле число – масу предмету. Вихідний файл „giri.sol” містить n рядків, в кожному з яких записано єдине число: -1, 0, 1 за правилом: якщо в і-му рядку знаходиться -1, це означає, що і-й важок знаходиться на лівій шальці, (там де і сам предмет, що зважують), якщо 1 – і-й важок знаходиться на правій шальці, 0 – важок не бере участь у зважуванні.

Програма, що формує файл „giriset.sol” є елементарною і наводитись не буде. Для написання програми “giri” можна скористатися алгоритмом, запропонованим в аналізі задачі, а можна запропонувати і інший алгоритм, що базується на аналізі наступної таблиці, в якій записано зважування перших шістнадцяти вантажів:

	27	**9**	**3**	**1**
1	0	0	0	1
2	0	0	1	-1
3	0	0	1	0
4	0	0	1	1
5	0	1	-1	-1
6	0	1	-1	0
7	0	1	-1	1
8	0	1	0	-1
9	0	1	0	0
10	0	1	0	1
11	0	1	1	-1

12	0	1	1	0
13	0	1	1	1
14	1	-1	-1	-1
15	1	-1	-1	0
16	1	-1	-1	1
...	...	...	...	...

З таблиці видно певну циклічність у використанні важків для зважування, а саме у стовпчику для і-го важка спочатку йде кількість нулів, що дорівнює сумі попередніх важків, а потім чередуються групи, що складаються з 1, -1 та 0, причому кількість цифр у групі дорівнює значенню цього важка. Наведемо реалізацію цього алгоритму мовою Паскаль:

```
var gr,res: array[1..25] of longint;
    g, Sum, V, V1, Vtmp, Block, Nom_in_Block: longint;
    n,i: integer;
    f0,f:text;
begin
  V:=1000000000;
  assign(f0,'giri.dat');
  reset(f0);
  readln(f0,v1);
  close(f0);
{Формуємо систему важків у масиві gr, n – кількість важків у системі}
  g:=1;
  n:=0;
  Sum:=0;
  while Sum<V do
  begin
    inc(n);
    Sum:=Sum+g;
    gr[n]:=g;
    g:=g*3;
  end;
{проводимо зважування}
  Sum:=0;
  Vtmp:=V1;
  for i:=1 to n do
  begin
    Vtmp:=V1+Sum;
    Block:=Vtmp div gr[i];
    Nom_in_Block:=Block mod 3;
    if Nom_in_Block=2 then res[i]:=-1 else
      res[i]:=Nom_in_Block;
    Sum:=Sum+gr[i];
  end;
```

```
  assign(f,'giri.sol');
  rewrite(f);
  for i:=1 to n do
    writeln(f,res[i]);
  close(f);
end.
```

На обласній олімпіаді 2007 року Костюченком А.О. була запропонована задача „Калькулятор”.

Дано повнодужковий арифметичний вираз. Під повнодужковим записом виразу розуміють запис, в якому порядок дій задається розстановкою дужок, тобто кожна операція разом зі своїми операндами береться в єдині дужки. Арифметичність виразу означає, що в ньому можуть використовуватися лише операції додавання – „+”, віднімання – „–”, множення – „*” та ділення –„/”).

Наприклад, вирази 2+3*5; (2+3)*5; 2+(3*5); ((2+3)) – не є повнодужковими арифметичними виразами. Вирази (2+(3*5)); ((2+3)*5); (2+3) – є повнодужковими арифметичними виразами.

Завдання:

Написати програму Calculus, що за записом повнодужкового арифметичного виразу обчислює його результат.

Вхідні дані:

У вхідному файлі Calculus.dat містяться K+1 рядків. В першому рядку міститься ціле число K ($5 \le K \le 100$), що задає кількість наступних рядків. Наступні K рядків задають повнодужковий арифметичний вираз. В кожному рядку містяться дані, які можуть бути: дужкою (відкриваючою, закриваючою), алгебраїчною операцією (+, -, *, /) або числом N ($0 \le N \le 100$).

Вихідні дані:

У вихідному файлі Calculus.sol міститься одне число з двома десятковими знаками після коми, яке є результатом обчислення.

Приклад вхідних та вихідних даних:

Якщо повнодужковий арифметичний вираз має вигляд „(((5+6)*2)-2)”, то:

CALCULUS.DAT	CALCULUS.SOL
13 (((5 + 6) * 2) - 2)	20.00

Для розв’язання даної задачі можна скористатися алгоритмом Рутисхаузера.

Алгоритм полягає в тому, що оброблюючи вираз в повнодужковому запису, він присвоює кожному з елементів (дужці, алгебраїчній операції, операнду) виразу номер рівня за наступним правилом:

1. якщо це відкриваюча дужка чи операнд, то значення рівня збільшується на 1;
2. якщо знак операції чи закриваюча дужка, то значення рівня зменшується на 1.

Для виразу (((5+6)*2)-2) присвоєння значень рівня буде відбуватися наступним чином:

№ елементу	1	2	3	4	5	6	7	8	9	10	11	12	13
Елементи рядка	(	(	(	5	+	6	)	*	2	)	-	2	)
Значення рівня	1	2	3	4	3	4	3	2	3	2	1	2	1

Виконання алгоритму складається з наступних кроків:

1. Виконати розстановку рівнів;

2. Виконати пошук елементу рядка з максимальним значенням рівня;

3. Виділити трійку – два операнди з максимальним значенням рівня і операцію, яка вкладена між ними;

4. Виконати обчислення над виділеною трійкою, та результат обчислення трійки позначити допоміжною змінною;

5. З вихідного рядка вилучити трійку разом з її дужками, а на їх місце помістити допоміжну змінну, яка позначає результат, з значенням рівня на одиницю меншу, ніж отримане максимальне значення рівня у виділеної трійки;

6. Виконувати пункти 2-5 до тих пір, поки в вихідному рядку не залишиться один елемент, який визначає загальний результат виразу.

Генеровані трійки		Вираз												
5+6=11	Елемент													
	Рівень													
11*2=22	Елемент			1										
	Рівень													
22-2=20	Елемент	(	22	-	2	)								
	Рівень	1	2	1	2	1								
Результат		20												

Генеровані трійки		Вираз												
5+6=11	Елемент	(	(	(	5	+	6	)	*	2	)	-	2	)

	Рівень	1	2	3	4	3	4	3	2	3	2	1	2	1
11*2=22	Елемент	(	(	11					*	2	)	-	2	)
	Рівень	1	2	3					2	3	2	1	2	1
22-2=20	Елемент	(	22									-	2	)
	Рівень	1	2									1	2	1
Результат		20												

Для розв’язання задачі визначимо запис TCalc з трьома полями Element : String[60] – міститиме рядковий запис елементу; Chislo : Real – якщо елемент є операндом, буде містити його числове значення, інакше міститиме 0; NRivnj : Byte – міститиме номер рівня для елементу. Оскільки наш вираз може містити не більше ніж 100 елементів, то опишемо відповідний масив TCalcMas=Array[1..100] of TCalc.

```
Type
  TCalc=record
    Element : String[60];
    Chislo : Real;
    NRivnj : Byte;
  End;
  TCalcMas=Array[1..100] of TCalc;

Var f : Text;
       i, n : Integer;
       PosR : Integer;
       s : String;
       cm : TCalcMas;
       kol : Byte;
       TmpR : Byte;
       ch : Real;
```

При розв’язанні задачі нам необхідно буде шукати позицію елементу з максимальним рівнем. Для цього опишемо функцію FindMax, яка нам буде повертати дану позицію.

```
function FindMax : Integer;
Var j : Integer;
    maxn : Integer;
```

```
Begin
  maxn:=1;
  For j:=2 To kol Do
   If (cm[j].NRivnj>cm[maxn].NRivnj)
    Then maxn:=j;
  FindMax:=maxn;
End;

Begin
  Assign(f,'calculus.dat');
  Reset(f);
  Readln(f,kol);
  n:=0;
```
Послідовно зчитуємо елементи виразу і розставляємо рівні за вказаним правилом.
```
  For i:=1 To kol Do
   Begin
     ReadLn(f,s);
     cm[i].Element:=s;
     If (s=')') Or ((Length(s)=1) And (s[1] in ['+','*','-
','/']))
      Then n:=n-1
      Else n:=n+1;
     cm[i].NRivnj:=n;
```
Якщо поточний елемент є операндом (числом) то знаходимо його числове значення.
```
     If Not ((Length(s)=1) And (s[1] in ['+','*','-
','/','(',')']))
      Then Begin
        Val(s,ch,PosR);
        cm[i].Chislo:=ch;
      End
      Else cm[i].Chislo:=0;
   End;
  Close(f);
  While Kol>1 Do
   Begin
     PosR:=FindMax;
     TmpR:=cm[PosR].NRivnj;
```
Обчислюємо значення виділеної трійки.
```
     Case cm[PosR+1].Element[1] Of
       '+' : ch:=cm[PosR].Chislo+cm[PosR+2].Chislo;
       '-' : ch:=cm[PosR].Chislo-cm[PosR+2].Chislo;
       '*' : ch:=cm[PosR].Chislo*cm[PosR+2].Chislo;
       '/' : ch:=cm[PosR].Chislo/cm[PosR+2].Chislo;
     End;
```
Надаємо нове значення елементу.
```
     str(ch:1:4,s);
     cm[PosR-1].Element:=s;
     cm[PosR-1].Chislo:=ch;
     cm[PosR-1].NRivnj:=TmpR-1;
```
Робимо зсув елементів на місце використаної трійки і її дужок.
```
     For i:=PosR To Kol-4 Do
```

```
      Begin
        cm[i].Element:=cm[i+4].Element;
        cm[i].Chislo:=cm[i+4].Chislo;
        cm[i].NRivnj:=cm[i+4].NRivnj;
      End;
     Kol:=kol-4;
   End;
  Assign(f,'calculus.sol');
  Rewrite(f);
  WriteLn(f,cm[1].Chislo:1:2);
  Close(f);
End.
```

Інший підхід до розв’язання даної задачі може бути таким. Проходячи по елементам заданого повнодужкового виразу, починаючи з першого, знаходимо перше входження закриваючої дужки, яку можна вважати завершенням певної, цілком визначеної (операндами якої є дійсні числа, а не інші операції) арифметичної дії. Виділивши трійку (дві операнди і операцію, яка вкладена між ними), що знаходиться ліворуч від знайденої закриваючої дужки, виконуємо обчислення над нею і з вихідного рядка вилучаємо трійку разом з її дужками. Даний процес продовжуємо доки не залишиться один елемент, який визначає загальний результат виразу.

Література

1. Депман И.Я. История арифметики. Пособие для учителей. Издание второе, исправленное. - Москва, 1965.- 415 с.