**Тема. Теорія чисел**

Перш ніж розглядати основні поняття з теорії чисел, уведемо деякі поняття.

Цифри 0, 2, 4, 6, 8 називають ***парними***, а цифри 1, 3, 5, 7, 9 - ***непарними***.

Натуральні числа називають ***парними***, якщо вони закінчуються парною цифрою, і ***непарними***, якщо вони закінчуються непарною цифрою.

При розв’язуванні олімпіадних задач розглядають системи числення з основами від 2 до 36. Це пов'язано з тим, що в алфавіті мови програмування можна використовувати 26 букв латинського алфавіту і 10 цифр.

Для того щоб представити яке небудь число в десятковій системі числення, слід обчислити суму всіх цифр даного числа, помножених на порядок системи числення, піднесеної до степеня, що дорівнює номеру розряду даної цифри. У загальному вигляді в q-тій системі запис числа $A\_{q}$, яке містить n цілих розрядів числа і m дробових розрядів числа, записується так:

$$A\_{q} =a\_{n-1}·q^{n-1}+...+a\_{0}·q^{0}+a\_{-1}·q^{-1}+...+a\_{-m}∙q^{-m}.$$

Для переведення з десяткової системи числення в систему з основою k необхідно робити ділення вихідного числа на k, запам'ятовуючи остачі від ділення і продовжуючи дану операцію з цілочисельним значенням діленням до тих пір, поки результатом ділення не буде нуль. Шуканим поданням числа буде запис, складений з отриманих остач від ділення, записаних у зворотному порядку.

Розглянемо задачу переведення цілого числа, записаного в q-ій системі числення, в число десяткової системи числення. Нехай S ‒ початковий запис числа, а X ‒ число в десятковій системі. Тоді алгоритм переведення з будь-якої системи в десяткову можна записати аналогічно до переведення з вісімкової системи числення у десяткову:

|  |  |
| --- | --- |
| Free Pascal | Python 3 |
| const q=8;var s,s1,s2:string; x,x2:real; x1:qword; i,k:longint;begin readln(s); k := pos(',',s);//обчислення цілої частини x1 := 0; for i := 1 to k-1 do x1 := x1 \* q + ord(s[i]) -ord('0');// обчислення дробової частини x2 := 0; for i := length(s) downto k+1 do x2 := (x2 + ord(s[i]) - ord('0')) / q; x := x1 + x2;{вивести всю цілу частину і 7 знаків після коми} writeln(x:0:7);end. | q = 8//зчитуємо цілу і дробову частиниs1,s2 = input().split(',')x1 = x2 =0for i in range(len(s1)) : x1 = x1 \* q + ord(s1[i]) - ord('0')for i in range(len(s2)-1,-1,-1) : x2 = (x2 + ord(s2[i]) - ord('0'))/qx = x1 + x2print("%.7f"%x) |

***Зауваження.*** *Якщо у коді поміняти константу q на потрібну, а, при* $q>10$*, у циклах застосувати оператор case у FP, чи if у Python 3, для знаходження* $x\_{1}$ *і* $x\_{2}$*, то даний алгоритм буде працювати коректно.*

Алгоритм переведення цілого числа X з десяткової системи в систему з основою К і запис його в S може бути представлений аналогічно до поданого нижче алгоритма переведення з десяткової системи в шістнадцяткову:

|  |  |
| --- | --- |
| Free Pascal | Python 3 |
| const q=16;var x,l,k:qword; s:string; c:char;begin readln(x); l:=0; repeat l:=l+1; k:=x mod q; x:=x div q; case k of 0..9: c:=char(ord('0')+k); 10: c:='A'; 11: c:='B'; 12: c:='C'; 13: c:='D'; 14: c:='E'; 15: c:='F'; end; s:=c+s; until (x=0); writeln(s);end. | q = 16x = int(input())l = 0s = ''while x > 0 : l += 1 k = x % q x //=q if 0 <= k <= 9 : c = str(k) elif k == 10 : c = 'A' elif k == 11 : c = 'B' elif k == 12 : c = 'C' elif k == 13 : c = 'D' elif k == 14 : c = 'E' elif k == 15 : c = 'F' s = c+sprint(s) |

***Зауваження.*** *Даний код працює коректно для переведення з десяткової системи числення в шіснадцяткову. Якщо у коді поміняти константу q на потрібну основу, відредагувати оператор case, то даний код буде справедливим і для переведення в інші системи числення.*

Натуральне число *c*, на яке діляться задані цілі числа *a* і *b*, називається ***спільним дільником*** цих чисел.

Множина спільних дільників чисел *a* і *b* збігається з множиною спільних дільників чисел *a* і $a\pm b$.

Найбільше натуральне число, на яке діляться задані числа, називається ***найбільшим спільним дільником*** (скорочено: НСД) цих чисел. НСД чисел *a* і *b* позначають $НСД(a,b)$ або ($a$,$b$).

Для знаходження НСД чисел *a* і *b* використовують алгоритм Евкліда. З урахуванням мов програмування, в алгоритмі використовуються такі відношення:

1. $НСД(a,0)= a$.
2. $НСД\left(a,b\right)=НСД\left(a mod b,b\right)=НСД\left(a,b mod a\right)$.

Запишемо на мовах Free Pascal і Python 3 програми обчислення НСД чисел $a$ і $b$ (GCD – Greatest Common Divisor):

|  |  |
| --- | --- |
| Free Pascal | Python 3 |
| var a,b:qword;function gcd(ap,bp:qword):qword;begin if bp =0 then gcd :=ap else gcd:=gcd(bp, ap mod bp);end;begin readln(a,b); writeln; writeln(gcd(a,b));end. | a, b = map(int, input().split())def gcd(ap,bp) : if bp == 0 : return ap else : return gcd(bp, ap%bp)print(gcd(a,b)) |

Розглянемо деякі властивості найбільшого спільного дільника:

1. Якщо $a\vdots b$ і $b>0$, то $НСД(a,b)=b$.
2. Якщо $НСД(a,b)=n$, то знайдуться такі цілі числа *c* і *d*, що $a=cn$, $b=dn$, причому $НСД(c,d)=1$.
3. Якщо $НСД(a,b)=1$, то для довільних $n, k \in N$ $НСД(a^{n},b^{k})=1$.
4. Спільний натуральний множник можна виносити з-під знака НСД: $НСД(ca,cb)=c∙НСД(a,b)$, якщо $c\in N$.
5. Якщо $a\vdots c$, $b\vdots c$, то $НСД(a,b)\vdots c$ і $НСД\left(\frac{a}{с},\frac{b}{с}\right)=\frac{НСД(a,b)}{с}$.

***Найменшим спільним кратним*** (***НСК***) двох або більше цілих чисел $a\_{1}$, $a\_{2}$, …, $a\_{n}$, які не дорівнюють нулю, називається найменше натуральне число, яке ділиться на всі ці числа. НСК чисел $a\_{1}$, $a\_{2}$, …, $a\_{n}$ позначають символом $НСК(a\_{1},a\_{2},…,a\_{n})$ або $[a\_{1},a\_{2},…,a\_{n}]$.

Щоб обчислити НСК будемо використовувати твердження

$НСК(a,b)=\frac{a∙b}{НСД(a,b)}$.

Запишемо на мовах Free Pascal і Python 3 програми обчислення НСК чисел $a$ і $b$ (LCM – Least Common Multiple):

|  |  |
| --- | --- |
| Free Pascal | Python 3 |
| var a,b:qword;function gcd(ap, bp : qword): qword;begin if bp =0 then gcd := ap else gcd:= gcd(bp, ap mod bp);end;function LCM(ap,bp : qword): qword;begin LCM := (ap div gcd(ap, bp))\*bp;end;begin readln(a,b); writeln(LCM(a,b));end. | a, b = map(int,input().split())def gcd(ap, bp) : if bp == 0 : return ap else : return gcd(bp, ap%bp)def LCM(ap, bp): return (ap // gcd(ap, bp))\*bpprint(LCM(a,b)) |

Розглянемо деякі властивості НСК:

1. $НСК\left(a,b\right)=НСК\left(b,a\right)$.
2. Якщо $a\vdots b$ і $a>0$, то $НСК(a,b)=a$.
3. Спільний множник можна виносити з-під знака НСК: $НСК(ca,cb)=c∙НСК(a,b)$, якщо $c\in N$.
4. Якщо $a\vdots k$ і $b\vdots k$, то $НСК\left(\frac{a}{k},\frac{b}{k}\right)=\frac{НСК\left(a,b\right)}{k}.$

**Простим числом** називається таке натуральне число $n>1$, яке не має додатних дільників, відмінних від *1* і *n*.

Натуральне число $n>1$ називається **складеним**, якщо воно має хоча б один додатній дільник, який відрізняється від *1* і *n*.

Розглянемо кілька теорем, які часто використовуються при знаходженні простих чисел.

Якщо натуральне число *n* складене, то воно має хоча б один простий дільник не більший від $\sqrt{n}$.

Будь-яке просте число, яке більше ніж 3, має вигляд $6n-1$ або $6n+1$. Зауважимо, що обернене твердження є хибним, тобто не кожне число виду $6n-1$ або $6n+1$ є простим.

Натуральне число $n>1$ просте тоді і тільки тоді, коли число $\left(n-1\right)!+1$ ділиться на *n.*

Напишемо на мові Free Pascal функцію, яка визначає чи просте число *n*. При цьому будемо шукати такі числа $x (1<x\leq \sqrt{n})$, які є дільниками числа *n*.

|  |  |
| --- | --- |
| Free Pascal | Python 3 |
| function isprime(n: LongInt): boolean ;var i, j: LongInt;begin if (n=2)  then isprime:= true else begin j:=round(sqrt(n)); for i:= 2 to j do if (n mod i = 0)and(i<>n) then begin isprime:= false; exit; end; isprime:=true; end;end; | def isprime(x): if x == 2 : return True else : j = round(x\*\*0.5) for i in range(2,j+1): if x % i ==0 and i != 0 : return False return True |

Коли слід знайти всі прості числа від 2 до N, доречно використовувати решето Ератосфена. Візьмемо 2, як перше просте число, і будемо викреслювати з допомогою перевірки кратності всі числа, які кратні двійці. Потім візьмемо перше ще не закреслене число і повторимо з ним такі ж операції, що і з 2. Для того, щоб "викреслити" число з масиву, в комірку, індексом якої є дане число, присвоїмо значення False. Даний алгоритм буде виконувати $O(nlog\left(logn\right))$дій, що при великих значеннях *n* суттєво. Для оптимізації алгоритму врахуємо, що серед парних чисел тільки 2 є простим, а інші прості числа – непарні. Тому в алгоритмі використовуємо цикл:

на Free Pascal

"for i:=3 to n do p[i]:=odd(i);"

на Python 3

“for i in range(3,n+1):

 p[i] = ((i%2) == 0)”.

Щоб знайти всі прості числа до n достатньо зробити просіювання тільки простими числами, які не більші за $\sqrt{n}$:

на Free Pascal

"for i:=3 to round(sqrt(n)) do"

на Python 3

“for i in range(3,round(n\*\*0.5)+1):”

Просіючи числа, йдемо від $i^{2}$, оскільки всі менші числа, кратні *i*, обов’язково матимуть простий дільник, що менший за *i*. Таким чином, вони будуть відсіяні раніше. Оскільки при піднесенні до квадрату великого числа можна здійснити переповнення типу, то виконуємо перевірку:

"if i\*i<=n then" (FP)

“if i\*i <= n :” (Python 3).

Наведені удосконалення суттєво зменшують потрібний для алгоритму об’єм пам’яті і кількість дій.

|  |  |
| --- | --- |
| Free Pascal | Python 3 |
| var p:array [2..1000000] of boolean;……………procedure eratosfen(n:longint); var i:longint; j:qword;begin p[2]:=true; for i:=3 to n do p[i]:=odd(i); for i:=3 to round(sqrt(n)) do if (p[i]) then if i\*i<=n then begin j:=i\*i; while j<=n do begin p[j]:=false; j:=j+i; end; end;end; | p = [ (i%2) == 1 for i in range(n+1)]p[1] = Falsep[2] = Truedef eratosphen(n): for i in range(3, round(n\*\*0.5)+1) : if p[i] : if i\*i <= n : j = i\*i while j <= n: p[j] = False j = j + i return p |

Кожне натуральне число, крім одиниці, можна розкласти на добуток простих множників єдиним способом. Тоді число n записують $n==p\_{1}^{α\_{1}}p\_{2}^{α\_{2}}…p\_{k}^{α\_{k}}$ і цей запис називають канонічним розкладом даного числа.

**Числом Армстронга** називається натуральне число, яке дорівнює сумі своїх цифр, піднесених до степеня, який дорівнює кількості його цифр. Наприклад, $153=1^{3}+5^{3}+3^{3}$.

Для знаходження числа Армстронга, як і в деяких інших задачах, потрібно піднести число до степеня. Традиційний алгоритм, у якому потрібне число множать на саме себе N разів, має складність O(N), де N – показник степеня. При піднесенні до степеня число збільшується дуже швидко, а значить, доведеться виконувати операцію множенння з довгими числами.

Розглядаючи степінь деяких чисел, можна здогадатися про метод, який дає швидший результат. Наприклад, для піднесення 5 в 4-ий степінь потрібно виконати 3 операції множення. Однак, якщо змінити порядок дій на такий: $(5^{2})^{2}$, то буде потрібно всього два множення. Слід пам’ятати, що при піднесенні степеня числа в який-небудь інший степінь показники цих степенів перемножуються. Саме на скороченні парних степенів ґрунтується ідея швидкого піднесення до степеня.

Наведемо текст функції, яка підносить число *a* до степеня *n*:

|  |  |
| --- | --- |
|  |  |
| function pow(a,n:integer): qword;var b,c,k:qword;begin k:=n; b:=1; c:=a; while (k<>0) do if (k mod 2 = 0) then begin k:=k div 2; c:=c\*c; end else begin dec(k); b:=b\*c; end; pow:=b;end; | def qpow(aa,nn): b = 1 c = aa while nn != 0 : if nn % 2 == 0: nn //= 2 c \*= c else : nn -= 1 b \*= c return b |

Досить часто у олімпіадних задачах використовують різні "цікаві" числа. Перейдемо до їх розгляду.

**Числами-близнятами** називаються два простих числа, які відрізняються лише на дві одиниці, тобто є послідовними непарними числами. Наприклад, 5 і 7, 11 і 13 тощо.

**Числами Ферма** називаються числа виду $F\_{k}=2^{2^{k}}+1$, де $k=0, 1, 2…$ .

**Числами Мерсена** називаються числа типу $M\_{n}=2^{n}-1$, де n – довільне натуральне число. Для цих чисел справедливе твердження: *якщо n складене число, то й відповідне число Мерсена* $M\_{n}$ *також буде складеним.* При знаходженні простих чисел Мерсена використовують таку теорему: *якщо n – просте число, то всі натуральні дільники числа* $M\_{n}$ *повинні мати вигляд* $2∙n∙k+1$*, де* $k=0, 1, 2…$ *.*

**Трикутним числом** називається число кругів, з яких можна скласти правильний трикутник, як подано на рисунку 1.

Уважається, що $T\_{0}=0$, а сама послідовність трикутних чисел записується: 0, 1, 3, 6, 10, 15, 21, … . У загальному вигляді n-е трикутне число виражається формулою: $T\_{n}=\frac{n(n+1)}{2}$. Із властивостей таких чисел важливою є наступна: $T\_{n-1}+T\_{n}=n^{2}.$



$T\_{1}=1$ $T\_{2}=3$ $T\_{3}=6$ $T\_{4}=10$

*Рис. 1. Форма зображення трикутних чисел*

**Квадратним числом** називається ціле додатне число, яке може бути записане у вигляді квадрата деякого іншого числа. Геометричною інтерпретацією такого числа може бути площа квадрата зі стороною, яка є цілим числом. У загальному вигляді n-е квадратне число виражається формулою: $K\_{n}=n^{2}$. При знаходженні суми квадратних чисел використо-вують формулу: $1^{2}+2^{2}+…+n^{2}=\frac{n\left(n+1\right)\left(2n+1\right)}{6}.$

Установлено, що між трикутними і квадратними числами існує така залежність $8T\_{n} + 1 = K\_{2n+1}$.

**Фігурними числами** називаються числа, які можна представити у вигляді дискретних геометричних об’єктів (наприклад, множин кругів чи куль), які щільно заповнюють правильні геометричні фігури.

Нехай $F\_{n}^{m}$– n-не m-кутне число, тоді його можна знайти за формулою $F\_{n}^{m}=\frac{n(\left(m-2\right)\left(n-1\right)+2)}{2}$.

Теорема Коші: *Довільне натуральне число може бути подане у вигляді суми не більше ніж m m-кутних чисел.*

**Досконалим числом** називається натуральне число, яке дорівнює сумі всіх своїх дільників крім самого числа. Наприклад, число 28 – досконале число, бо 28=1+2+4+7+14.

Розглянемо деякі властивості досконалих чисел.

1. Всі парні досконалі числа (крім 6) є сумою кубів послідовних непарних натуральних чисел.
2. Всі парні досконалі числа є трикутними числами.
3. Сума всіх чисел, обернених до дільників досконалого числа (включаючи саме число), дорівнює 2.
4. Всі парні досконалі числа, крім 6 і 496, закінчуються в десятковому запису на 16, 28, 36, 56 або 76.
5. Всі парні досконалі числа в двійковому запису містять спочатку p одиниць, за якими йдуть $p-1$ нулів.
6. Сума всіх цифр парного досконалого числа (крім 6) дорівнює 1. Наприклад, (2+8=10, 1+0=1; 4+9+6=28...).

**Дружніми числами** називаються два натуральних числа, для яких сума всіх дільників першого числа (крім нього самого) дорівнює другому числу і сума всіх дільників другого числа (крім нього самого) дорівнює першому числу. Наприклад, числа 220 і 284 є дружніми. Іноді окремим випадком дружніх чисел вважаються досконалі числа: кожне досконале число дружнє саме собі.

**m-самозакоханим числом** називається натуральне число, яке дорівнює сумі своїх цифр, піднесених до степеня m, де m – деяке натуральне число. Числа Армстронга – окремий випадок таких чисел.

**Числом Сміта** називається таке складене число, сума цифр якого (в даній системі числення) дорівнює сумі цифр всіх його простих співмножників. Наприклад, числом Сміта є 202, оскільки $2+0+2=4$, і $2+1+0+1=4$ ($202=2∙101$).

**Цілою частиною** дійсного числа *x* називається найбільше ціле число, яке не більше ніж *x*. Ціла частина числа *x* зазвичай позначається як $\left[x\right]$.

В інформатиці поряд з функцією **цілої частини** використовують функції **"підлога"** (англ. *floor*) та **"стеля"** (англ. *ceiling*).

Функція **"підлога"** позначається як $y=\left⌊x\right⌋$ та дорівнює найбільшому цілому числу *n* для якого виконується, що $n\leq x$. Функція **"стеля"** позначається як $y=\left⌈x\right⌉$ та дорівнює найменшому цілому числу *n*, для якого виконується, що $n\geq x$. Наприклад, $\left[\sqrt{3}\right]=1$, $\left[-\sqrt{3}\right]=-2$, $\left⌊-\sqrt{3}\right⌋=-2$, $\left⌈-\sqrt{3}\right⌉=-1$.

Дробовою частиною дійсного числа *x* називається невід’ємне дійсне число $x-\left[x\right]$, де $\left[x\right]$ – ціла частина дійсного числа *x*. Зазвичай дробову частину дійсного числа *x* позначають $\left\{x\right\}$.

Розглянемо деякі співвідношення, які справедливі для цілої та дробової частин числа.

1. Якщо *n* – ціле число, то $\left[m+n\right]=\left[m\right]+n$, $\left\{m+n\right\}=\left\{m\right\}$.

2. $\left[m+n\right]=\left\{\begin{matrix}\left[m\right]+\left[n\right], якщо \left\{m\right\}+\left\{n\right\}<1, \\\left[m\right]+\left[n\right]+1, якщо \left\{m\right\}+\left\{n\right\}\geq 1.\end{matrix}\right.$

3. $\left\{m+n\right\}=\left\{\begin{matrix}\left\{m\right\}+\{n\}, якщо \left\{m\right\}+\left\{n\right\}<1, \\\left\{m\right\}+\left\{n\right\}-1, якщо \left\{m\right\}+\left\{n\right\}\geq 1.\end{matrix}\right.$

4. $\left[\frac{n}{m}\right]+\left[\frac{n+1}{m}\right]+\left[\frac{n+2}{m}\right]+…+\left[\frac{n+m-1}{m}\right]=n$.

При написанні матеріалу за основу взято матеріал:

Харченко В.М.Готуємо до олімпіади з інформатики. – Ніжин: НДУ ім. М. Гоголя, 2015. – Ч. 2.