

Олександр Рудик

Код Грея

Означення 1. Кодом Грея називають непозиційну систему запису цілих натуральних чисел за допомогою двох символів 0 та 1 таким чином. Нуль кодують послідовністю нулів. При зростанні цілого числа:

- наймолодший 1-ий розряд у послідовності символів змінюють у такій послідовності: 0, 1, після чого у наступний 2-й розряд записують 1, а наймолодший розряд змінюють уже у протилежному порядку;
- два наймолодші розряди змінюють у такому порядку: 00, 01, 11, 10, після чого у 3-ій розряд записують 1, а два наймолодші розряди змінюють уже у протилежному порядку: 10, 11, 01, 00;
- три наймолодші розряди змінюють у такому порядку: 000, 001, 011, 010, 110, 111, 101, 100, після чого у 4-ий розряд записують 1, а два наймолодші розряди змінюють уже у протилежному порядку: 100, 101, 111, 110, 010, 011, 001, 000 ... ;
- k наймолодших розряди змінюють у порядку, визначеному попередніми кроками, після чого у наступний $(k + 1)$ -ий розряд записують 1, а молодші розряди змінюють уже у протилежному порядку

Код Грея називають також рефлексним (відбитим, відзеркаленим). У Таблиці 1 подано коди Грея послідовностями по 4 символи для цілих чисел від 0 до 15 включно.

Таблиця 1

й код Десятковий	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
код Двійковий	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Код Грея	0000	0001	0011	0010	0110	0111	0101	0100	1100	1101	1111	1110	1010	1011	1001	1000

Зауваження 1. Коди Грея двох послідовних натуральних чисел відрізняються лише в одному розряді.

Вказану властивість використовують для збільшення надійності роботи системи оптичних фотодіодів при встановленні кута повороту дисків-носіїв інформації.

Найпростіші задачі на опанування поняттям кодів Грея мають такий вигляд.

Визначити коди Грея послідовних натуральних чисел — орієнтовне завдання II (районного) етапу олімпіади 2008 року у місті Києві. Його можна розв'язати за допомогою такої програми, написаною мовою Turbo Pascal 7.0.

```
{ $N+ }
{ Запис кодів Грея послідовних цілих
  чисел у межах від 0 до (2^n-1) при
                                          n < 61 }

var a: array[1..61] of char;
    m, n: byte;    o: text;
    j, k: extended;
begin
assign(o, 'GRAY.DAT');    reset(o);
readln(o, n);            close(o);
assign(o, 'GRAY.RES');    rewrite(o);
for m:=1 to n do a[m] := '0';
k:=0;
repeat
  for m:=n downto 1 do write(o, a[m]);
  writeln(o);
  k:=k+1;
  m:=1;
  j:=k;
  while j-2*trunc(j/2)=0 do begin
  j:=j/2;
  inc(m)                                end;
  if m<=n then case a[m] of
    '0': a[m] := '1';
    '1': a[m] := '0' end
until m>n;
                                close(o)    END.
```

Сумнівно, щоб більшість учасників II (районного) етапу олімпіади саме таким чи аналогічним чином змогли б набрати хоча б якусь кількість балів. А цю задачу було анонсовано як таку, що посильна, можливо не у повному обсязі, більшості учасників. Пояснення доволі просте: це задача з можливим попереднім рахунком і використанням його результату у вигляді сталих у коді програми. Поки не встановлено обмежень на величину коду, такий підхід дозволяє набрати хоча б частину балів з гарантованим проходженням обмежень за часом.

1. Створити за допомогою текстового редактора файл 1.DAT такого вигляду.

```
0', '
1', '
```

2. Змінюючи назви вхідних і вихідних файлів (збільшуючи відповідні натуральні числа на 1), створити файли 2.dat, 3.dat, 4.dat, ..., 10.dat за допомогою такої програми.

```

var fi,fo:text;
si: array[1..1024] of string[10];
so: string; j,n: word; BEGIN
assign(fi,'1.DAT'); reset(fi);
assign(fo,'2.DAT'); rewrite(fo); n:=0;
repeat inc(n); readln(fi,si[n]);
writeln(fo,'0'+si[n])
until seekeof(fi);
close(fi);
for j:=n downto 1 do writeln(fo,'1'+si[j]);
close(fo) END.

```

3. За допомогою текстового редактора створюємо код програми, який містить результати попередніх розрахунків, записаних у файли 2.dat, 3.dat, 4.dat, ..., 10.dat (код подано при кількості символів n , меншій за 5).

```

const
n1=2; a1:array[1..n1] of string[1]=('0','1');
n2=4; a2:array[1..n2] of string[2]=('00','01','11','10');
n3=8; a3:array[1..n3] of string[3]=
('000','001','011','010','110','111','101','100');
n4=16; a4:array[1..n4] of string[4]=
('0000','0001','0011','0010','0110','0111','0101','0100',
'1100','1101','1111','1110','1010','1011','1001','1000');
var n: byte; o:text; k: word; BEGIN
assign(o,'gray.dat'); reset(o); readln(o,n); close(o);
assign(o,'gray.res'); rewrite(o);
case n of
1: for k:=1 to n1 do writeln(o,a1[k]);
2: for k:=1 to n2 do writeln(o,a2[k]);
3: for k:=1 to n3 do writeln(o,a3[k]);
4: for k:=1 to n4 do writeln(o,a4[k]);
end;
close(o) END.

```

Визначити код Грея даного натурального числа — завдання III (міського) етапу олімпіади 2011 року у місті Києві. Його можна розв'язати за допомогою такої програми, написаною мовою Turbo Pascal 7.0

```

{$N+}
var a: array[0..61] of extended;
j,k: shortint; s: string;
o: text; x: extended; BEGIN
{Зчитування даних}
assign(o,'gray.in'); reset(o);
readln(o,x); close(o);
s:='';

```

```

        {Визначення степенів двійки і довжини коду}
        j:=0; a[j]:=1;
repeat inc(j); a[j]:=a[j-1]*2
until x < a[j];
dec(j); {Визначення коду Грея}
if x=0 then s:='0' else
REPEAT while (x < a[j]) and (0 < j) do begin
            s:=s+'0';
            dec(j) end;
        if (0 < x) then begin
            x:=2*a[j]-x-1;
            s:=s+'1';
        if (x=0) then for k:=1 to j do s:=s+'0';
        dec(j) end;
UNTIL x=0;
            {Запис відповіді}
        assign(o,'gray.out'); rewrite(o);
writeln(o,s); close(o) END.

```

Якщо нас задовольняють розрахунки у межах базового типу цілих чисел, то можна скористатися виразом $x \text{ xor } (x \text{ div } 2)$ для коду Грея числа x .

Альтернативою до цього є попереднє обчислення кодів Грея для всіх натуральних чисел, що менші за певний степінь двійки. Далі потрібно створити програму, у якій результати розрахунків подано масивом сталих (див. вище).

Визначити натуральне число за його кодом Грея пропонуємо читачам самостійно.