## Астролог

Дана задача легко розв'язується з застосуванням операцій роботи з літерними величинами. Але перед цим потрібно внести уточнення до умови, так як незрозуміло, як потрібно записати дату 1 вересня 1989 року: 1091989 чи 191989. Домовимось, що номер місяця спереду не буде містити незначущих нулів, тобто для місяця використовуються дійсно їх номери від 1 до 12. Алгоритм розв'язання такий: спочатку сформуємо рядок, який буде містити перше записане число. Потім переведемо його в число і далі будемо працювати з числами. Якби ми не використовували операції для роботи з літерними величинами, то нам довелося б при формуванні початкового числа перевіряти цілий ряд умов, а саме: чи більший номер дня за 9, чи більший номер місяця за 9, у якому проміжку міститься число, що утворюється при дописуванні до числа номера місяця. Використання ж літерних величин дає змогу позбавитись від цих перевірок. Домовимось, що дані вводяться з клавіатури коректно і тільки числами. З утвореним описаним вище способом числом будемо проводити операцію додавання цифр до тих пір, доки сума цифр буде більша 9. Все інше цілком зрозуміло з тексту програми.

|  |
| --- |
| #include "fstream"using namespace std;ifstream cin("input.txt");ofstream cout("output.txt");int main(){long date,happy;cin>>date;happy=0;while (date > 9){ happy = 0; while (date > 0) { happy= happy + date%10; date= date/10; }date= happy;} cout<<happy<<endl; return 0;} |

## Робочі дні

Місяцю з 31-го дня відповідає лінійна таблиця елементів, в якій робочий день позначено 1, а неробочий – 0. Скласти алгоритм, який підраховує:

а) кількість тижнів в місяці;

б) кількість робочих днів в кожному з тижнів місяця.

Примітка: Тижнем будемо називати довільну не пусту множину одиниць між двома послідовними нулями або обмежену нулем праворуч (якщо місяць починається з робочого дня), або обмежену нулем ліворуч (якщо місяць не закінчується робочим днем).

|  |
| --- |
| #include "math.h"#include "fstream"using namespace std;ifstream cin("input.txt");ofstream cout("output.txt");int main(){int n, i, k;cin>>n;int \*a=new int[n];int \*x=new int[n];for(i=0;i<n;i++) cin>>a[i];k=1;a[n]=0;int j=0;for(i=0;i<n;i++){if(a[i]==1 && a[i+1]==1)k++;else { if(a[i]==1){x[j]=k;j++;} k=1;  }}cout<<j<<endl;for(i=0;i<j-1;i++)cout<<x[i]<<" ";cout<<x[j-1]<<endl;return 0;} |

## Обробка фотографій з космосу

Розв'язання задачі здійснюємо строго на підставі вказівок умови самої задачі. Спочатку створюємо карту, а на її основі підраховуємо кількість островів, півостровів і т.д. , тобто здійснюємо кількаразовий перегляд таблиці для визначення типу клітин. Після цього визначаємо аналіз сусідніх клітин на належність до того самого типу.

|  |
| --- |
| program Earth\_Map;const lenmap=20;var i,i1,p1,t,coltest,n,m:integer; mes, bact : array[1..lenmap, 1..lenmap] of byte; barr : array[0..10] of byte; f, f1 : text; st : string; flstop : boolean;{ скільки клітин якого типу знаходиться навколо клітини [i, i1], результат заносимо в масив barr, де barr[n] буде містити кількість клітин типу n }procedure calcbar(i, i1 : integer);var t1, t2 : integer;beginfillchar(barr, sizeof(barr), 0);for t1 := -1 to 1 dofor t2 := -1 to 1 doif abs(t1) + abs(t2) = 1 then inc(barr [ mes [i + t1, i1 + t2]])end;{ підрахунок кількості об'єктів типу a на карті }function calcColObj(a : integer) : integer;const hod : array[1..4,1..2] of integer = ((0,1),(0,-1),(1,0),(-1,0));var col, i, i1, k, x, y : integer; flStop, flF : boolean;begin{ відсіюємо всі інші типи об'єктів і залишаємо тільки тип а }for i := 1 to m dofor i1 := 1 to n doif mes[i,i1] = a then bact[i, i1] := 1 else bact[i, i1] := 0;col := 0;repeat{ шукаємо першу 1, змінюємо її на 2 і збільшуємо лічильник кількості об'єктів }flStop := true;for i := 1 to m dofor i1 := 1 to n doif (flStop) and (bact[i,i1] = 1) thenbegininc(col); bact[i,i1] := 2; flStop := falseend;{ всі 1, що стоять коло 2, змінюємо на 2 і повторюємо до тих пір, доки вони є }if flStop = false thenrepeatflF := true;for i := 1 to m dofor i1 := 1 to n doif bact[i,i1] = 2 thenfor k := 1 to 4 dobeginx := i + hod[k, 1]; y := i1 + hod[k, 2];if (x > 0) and (x <= m) and (y > 0)and (y <= n) and (bact[x,y] = 1) thenbegin bact[x,y] := 2; flF := false end;end;until flF;{ видаляємо "порахований" об'єкт }for i := 1 to m dofor i1 := 1 to n doif bact[i, i1] = 2 then bact[i, i1] := 0;until flStop;calcColObj:=col;end;{ головна програма }beginassign(f, 'input.txt'); reset(f);assign(f1, 'output.txt'); rewrite(f1);read(f, m);readln(f, n);for i := 1 to m dobeginreadln(f, st);for i1 := 1 to n do beginmes[i, i1] := ord(st[i1]) - ord('0');end;end;{ Виділення }repeatflstop := true;for i := 2 to m - 1 dofor i1 := 2 to n - 1 dobegincalcbar(i, i1);if mes[i, i1] = 0 thenbeginif barr[0] = 4 then mes[i, i1] := 2; { материк }if barr[1] = 4 then mes[i, i1] := 3; { острів }if (barr[1] = 3) or (barr[1] =2 ) and (barr[4] >= 1)or(barr[1] = 1) and (barr[4] >= 2) thenbeginflstop := false; mes[i,i1] := 4end; { півострів }end;end;until flstop;for i := 1 to m do for i1 := 1 to n dobeginif mes[i, i1] = 0 then mes[i, i1] := 5; { берег }if (i = 1) or (i1 = 1) or (i = m) or (i1 = n)then mes[i, i1] := 6; { море по краях завжди }end;repeatflstop := true;for i := 2 to m - 1 dofor i1 := 2 to n - 1 dobegincalcbar(i, i1);if (mes[i, i1] = 1) and (barr[6] > 0) thenbeginflstop := false; mes[i, i1] := 6end; { море }end;until flstop;repeatflstop := true;for i := 2 to m - 1 dofor i1 := 2 to n - 1 dobegincalcbar(i, i1);p1 := barr[2] + barr[3] + barr[4] + barr[5];if mes[i, i1] = 6 thenif (p1 = 2) or (p1 = 3) or (barr[7] = 4)or (p1 = 1) and (barr[7] >= 2) thenbeginflstop := false; mes[i,i1] := 7end; { затока }end;until flstop;for i := 1 to m dofor i1 := 1 to n doif mes[i,i1] = 1 then mes[i,i1] := 8; { озеро }{ Вивід результатів }{for i := 1 to m dobeginst := '';for i1 := 1 to ndo st := st + chr(ord('0') + mes[i, i1]);writeln(f1, st);end;}writeln(f1, calcColObj(3)); { кількість островів }writeln(f1, calcColObj(4)); { кількість півостровів }writeln(f1, calcColObj(7)); { кількість заток }writeln(f1, calcColObj(8)); { кількість озер }close(f); close(f1);end. |

## Зелена пляма

Ідея розв'язання базується на спільному використанні декількох ідей. Спочатку визначаємо, скільки вершин одного трикутника лежать всередині іншого, і запам'ятовуємо їх. Потім шукаємо точки взаємних перетинів сторін обох трикутників і також запам'ятовуємо їх. Серед усіх знайдених точок відкидаємо ті, що повторюються, а ті, що залишились, розміщуємо у порядку, який відповідає утвореному опуклому многокутнику. Для цього скористаємось наступними міркуваннями: за першу вершину многокутника вибираємо довільну точку з множини точок, що запам'ятали. За наступну вершину вибираємо таку точку, для якої пряма, проведена через попередню точку і вибрану, ділить площину на такі дві півплощини, що одна з них міститиме всі інші точки утвореної фігури, а інша – жодної.

Взагалі, задача містить в собі декілька ключових ідей, що застосовуються при розв'язанні задач з геометричним змістом, тому рекомендуємо самостійно розв'язати дану задачу і лише у випадку виникнення труднощів розібратись з розв'язком, наведеним нижче.

|  |
| --- |
| program zad4;var dot, newdot, dotm : array[1..12] of record xx, yy : real end;pass : set of byte;flWork : boolean;f : text;i, i1, i2, i3, i4, n, pos : integer;d, a1, a2, b1, b2 : real;{ повертає координати точки перетину двох відрізків, якщо вони перетинаються }function XLineLine(x11, y11, x12, y12, x21, y21, x22, y22 : real;var x, y : real) : boolean;{ перевірка належності точки прямокутнику }function InBox2d(xk, yk, x1, y1, x2, y2 : real) : boolean;var xl, yl, xh, yh : real; beginif x1 < x2 thenbeginxl := x1;xh := x2end elsebeginxl := x2; xh := x1end;if y1 < y2 thenbeginyl := y1; yh := y2end elsebeginyl := y2; yh := y1end;InBox2d := ((xk >= xl) and (xk <= xh) and (yk >= yl) and (yk<=yh));end;beginXLineLine := false;a1 := x12 - x11; a2 := y12 - y11;b1 := x22 - x21; b2 := y22 - y21;d := a1\*b2 - a2\*b1;if d = 0 then exit;if abs(a2) > abs(a1) thenbeginy := (a1\*b2\*y11-a2\*b2\*(x11-x21)-a2\*b1\*y21)/d;x := a1/a2\*(y-y11)+x11end elsebeginx := -(a2\*b1\*x11-a1\*b1\*(y11-y21)-a1\*b2\*x21)/d;y := a2/a1\*(x-x11)+y11end;XLineLine := (InBox2d(x,y,x11,y11,x12,y12)) and(InBox2d(x,y,x21,y21,x22,y22));end;{ обчислення площі многокутника }function SNcut(col : integer) : real;var i, i1 : integer; s : real;begins:=0;for i := 1 to col dobeginif i < col then i1 := i + 1 else i1 := 1;s := s + dotm[i].xx\*dotm[i1].yy - dotm[i1].xx\*dotm[i].yy;end;Sncut := abs(s/2);end;{ перевірка належності точки трикутнику }function In3cut(x, y, x1, y1, x2, y2, x3, y3 : real) : boolean;var s, s1, s2, s3 : real;beginwith dotm[1] do begin xx := x; yy := y end;with dotm[2] do begin xx := x1; yy := y1 end;with dotm[3] do begin xx := x2; yy := y2 end;s1 := sncut(3);with dotm[2] do begin xx := x2; yy := y2 end;with dotm[3] do begin xx := x3; yy := y3 end;s2 := sncut(3);with dotm[2] do begin xx := x1; yy := y1 end;with dotm[3] do begin xx := x3; yy := y3 end;s3 := sncut(3);with dotm[1] do begin xx := x2; yy := y2 end;s := sncut(3);in3cut := (s = s1 + s2 + s3);end;{ головна програма }beginassign(f, 'input.txt'); reset(f);for i := 1 to 6 do read(f, dot[i].xx, dot[i].yy);close(f); { точки першого трикутника всередині другого }pos := 0;for i := 1 to 3 do with dot[i] doif in3cut(xx,yy, dot[4].xx,dot[4].yy, dot[5].xx,dot[5].yy,dot[6].xx, dot[6].yy) thenbegininc(pos);newdot[pos].xx := xx; newdot[pos].yy := yyend; { точки другого трикутника всередині першого }for i := 4 to 6 do with dot[i] doif in3cut(xx, yy, dot[1].xx, dot[1].yy, dot[2].xx, dot[2].yy,dot[3].xx, dot[3].yy) thenbegininc(pos);newdot[pos].xx := xx; newdot[pos].yy := yyend; { точки перетину сторін трикутників } for i := 1 to 3 dobeginif i<3 then i1 := i + 1 else i1 := 1;for i2 := 1 to 3 dobeginif i2 < 3 then i3 := i2 + 1 else i3 := 1;with newdot[pos+1] doif xlineline(dot[i].xx, dot[i].yy, dot[i1].xx, dot[i1].yy,dot[3+i2].xx, dot[3+i2].yy, dot[3+i3].xx, dot[3+i3].yy, xx, yy)then inc(pos);end;end;{ видаляємо точки, що повторюються }i := 0;repeatinc(i);repeatn := 0;for i1 := i + 1 to pos doif (newdot[i].xx = newdot[i1].xx)and (newdot[i].yy = newdot[i1].yy) then n:=i1;if n<>0 thenbeginfor i1 := n to pos - 1 do newdot[i1] := newdot[i1+1];dec(pos);end;until n = 0;until i >= pos;if pos >= 3 thenbegin{ визначаємо порядок обходу вершин многокутника }pass := [1]; dotm[1] := newdot[1]; n := 1;repeatflWork := true;with dotm[n] dofor i := 1 to pos do if (not (i in pass))and(flWork) thenbegina1 := newdot[i].xx - xx;a2 := newdot[i].yy - yy;i3 := 0; i4 := 0;for i1 := 1 to pos dobegind := (newdot[i1].xx - xx)\*a2 - (newdot[i1].yy - yy)\*a1;if d < -0.000000001 then inc(i3);if d> 0.000000001 then inc(i4);end;if not((i3 > 0) and (i4 > 0)) thenbegininc(n); pass := pass+[i];dotm[n] := newdot[i]; flWork := falseend;end;until n = pos;d := SNcut(pos);endelse d := 0;assign(f, 'output.txt'); rewrite(f);writeln(f, pos);writeln(f, d:1:2);close(f);end. |