Codi: 24983

Gràfics per Computador I

Pràctica 1: Operacions amb polígons 2D
Curs 2004-2005

INDEX

FONAMENT TEÒRIC I DOCUMENTACIÓ ÚTIL

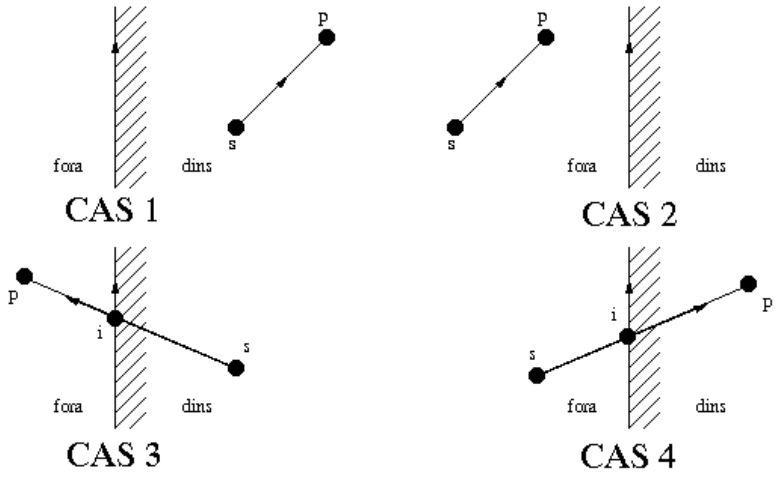
En aquesta pràctica treballarem les operacions amb polígons 2D. Els polígons 2D es defineixen com una seqüència de punts (x, y), ordenats en sentit de les agulles del rellotge. El darrer punt quedarà unit amb el primer.

Retall de polígons

Quan volem visualitzar per pantalla, moltes vegades ens caldrà "retallar" els objectes que volem mostrar per encabir-los a la finestra. Quan tenim polígons, aquest retall es complica, perquè volem que el resultat d'un polígon retallat sigui a la vegada un sol polígon, per poder-lo pintar, per exemple. Per tant, no en tenim prou amb retallar les rectes que el formen.

Algorisme de Sutherland-Hodgman

En aquest algorisme, en lloc d'atacar el problema directament, el que es fa és anar retallant el polígon sobre cada recta que defineix la finestra.
Les diferents configuracions que es poden donar amb dos vèrtexs consecutius i una recta de la finestra són quatre :



Així, l'algorisme és el següent :

```
V = Vèrtexs del polígon
Per a cada recta de la finestra
    V' = buit
    s = darrer vèrtex de V
    Per tots els vèrtexs p de V
        Mirem el cas en què ens trobem
        Si estem al CAS 1 Afegim "p" a V'
        Si estem al CAS 2 No afegim res a V'
        Si estem al CAS 3 Afegim "i" a V'
        Si estem al CAS 4 Afegim "i" i després "p" a V'
        s = p
    Fi Per
    V = V'
Fi Per
```

Càlcul de les interseccions

Estem considerant que la nostra finestra de retall és rectangular. En aquest cas, el càlcul de les interseccions és molt senzill.

Per calcular les interseccions, és a dir, el punt **`i = (x_i, y_i)`**, farem el següent :

- Intersecció amb el costat de l'esquerra :

```
y_i = y_p + m(x_costat - x_p)
x_i = x_costat
```

- Intersecció amb el costat de la dreta :

```
y_i = y_p + m(x_costat - x_p)
x_i = x_costat
```

- Intersecció amb el costat de dalt :

```
x_i = x_p + (y_costat - y_p) / m
y_i = y_costat
```

- Intersecció amb el costat de baix :

```
x_i = x_p + (y_costat - y_p) / m
y_i = y_costat
```

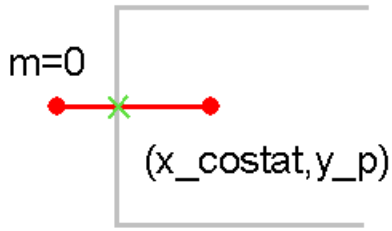
On **m** és el pendent de la recta que passa pels punts "s" i "p":

```
m = (y_s - y_p) / (x_s - x_p)
```

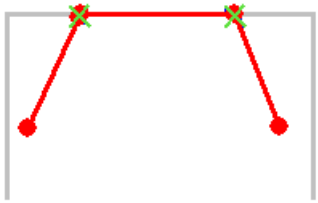
Problema: Costats horitzontals i verticals del polígon

En calcular les interseccions, ens podríem trobar amb un problema quan es tracta de costats horitzontals i de costats verticals, ja que en ambdós casos podríem tenir una **divisió per 0**. Analitzem quan podem tenir divisió per 0 :

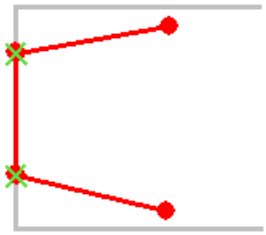
- Amb els costats horitzontals, en calcular la inversa del pendent **m**:
    - Si estem tractant un costat vertical de la finestra, la intersecció és fàcil de calcular, ja que serà la (x_costat, y_p). En aquest cas no hi ha problema, doncs es pot aplicar la fórmula directament, donat que m=0.



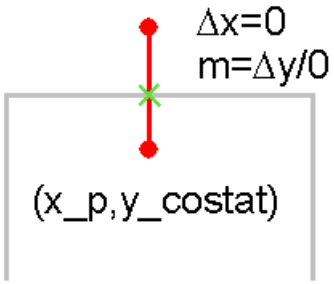
- Si estem tractant un costat horitzontal de la finestra, no es donarà el cas, perquè **<u>considerem que si un vèrtex esta sobre la finestra, està fora</u>** (considerem fronteres obertes).



- Amb els costats verticals, en calcular el pendent **m**:
  - Si estem tractant un costat vertical de la finestra, no es donarà el cas (considerem fronteres obertes).

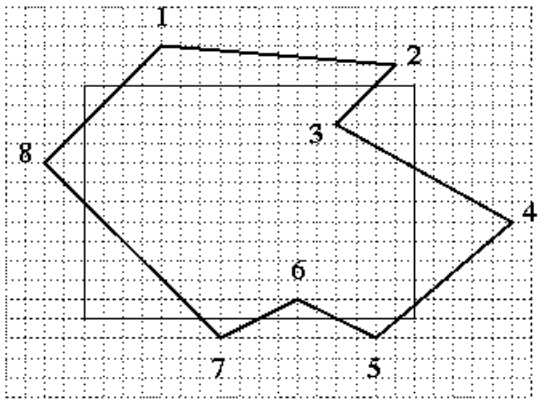


  - Si estem tractant un costat horitzontal de la finestra, la intersecció serà (x_p, y_costat). En aquest cas si que hi ha problema. Per tant, cal tenir-ho en compte en calcular el pendent **m** per aquest cas.



Exemple

A continuació teniu un petit exemple de *clipping* perquè el seguiu i pugueu testejar el vostre algorisme:

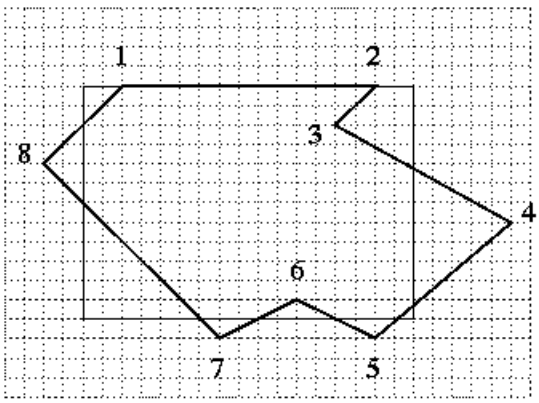


Aquest és el polígon **inicial**, definit pels punts següents:

Punt	Coordenades
1	(80, 20)
2	(200, 30)
3	(170, 60)
4	(260, 110)
5	(190, 170)
6	(150, 150)
7	(110, 170)
8	(20, 80)

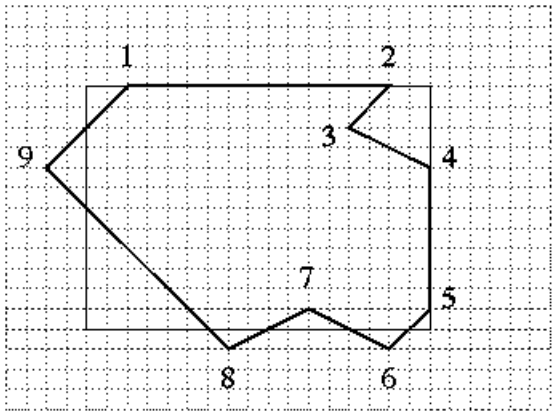
La finestra de clipping està definida pels extrems:

Punt	Coordenades
Superior esquerre	(40, 40)
Inferior dret	(210, 160)



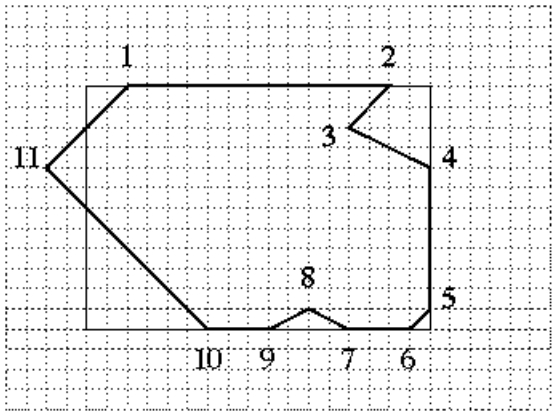
Aquest és el polígon retallat per **dalt**, els punts obtinguts són els següents:

Punt	Coordenades
1	(60, 40)
2	(190, 40)
3	(170, 60)
4	(260, 110)
5	(190, 170)
6	(150, 150)
7	(110, 170)
8	(20, 80)



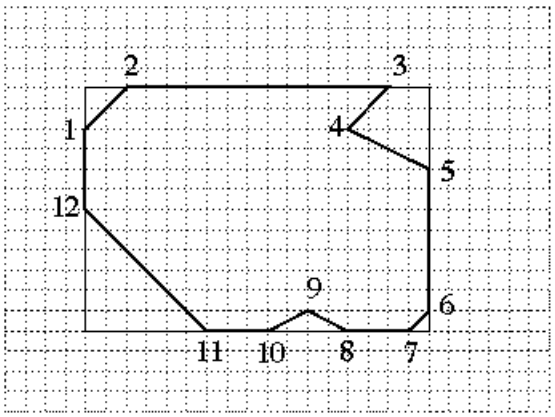
Aquest és el polígon retallat de nou, però ara per la **dreta**. Els punts són els següents:

Punt	Coordenades
1	(60, 40)
2	(190, 40)
3	(170, 60)
4	(210, 83)
5	(210, 153)
6	(190, 170)
7	(150, 150)
8	(110, 170)
9	(20, 80)



Aquest és el polígon retallat per tercer cop, ara per **baix**. Els punts són:

Punt	Coordenades
1	(60, 40)
2	(190, 40)
3	(170, 60)
4	(210, 83)
5	(210, 153)
6	(201, 160)
7	(170, 160)
8	(150, 150)
9	(130, 160)
10	(100, 160)
11	(20, 80)



Aquest és el polígon final, que ha estat retallat per l'**esquerra**. Els punts resultants són els següents:

Punt	Coordenades
1	(40,60)
2	(60, 40)
3	(190, 40)
4	(170, 60)
5	(210, 83)

6	(210, 153)
7	(201, 160)
8	(170, 160)
9	(150, 150)
10	(130, 160)
11	(100, 160)
12	(40, 100)

Emplenat de polígons

El que volem fer ara és pintar regions de la pantalla. El primer problema és com definir una regió de pantalla. En aquesta pràctica considerarem les dues possibilitats: regions definides per polígons i regions definides per píxels.

La regió que volem pintar està definida per un conjunt de punts que corresponen als vèrtexs d'un polígon. Aquí només presentarem un algorisme, que és el que s'haurà d'implementar a la pràctica, l'algorisme Y-X.

L'algorisme Y-X

La principal complicació d'aquest algoritme és la d'entendre les estructures de dades que la formen. Per una banda, tenim una taula de costats, que anomenarem TC, y per una altra tenim una llista de costats actius, que anomenarem LCA. Passem a descriure cadascun dels components :

Estructures de dades

- **Costat**
  És l'estructura bàsica amb la qual treballarem. Representarem cadascun dels costats que composen el polígon a pintar en aquesta estructura. Aquesta estructura està formada per :

Nom	Descripció
x	Inicialment, la coordenada x del vèrtex que té la **y** mínima. Durant l'execució de l'algoritme, s'anirà modificant en funció de **dx**
dx	Correspon al pendent de la recta que passa pels dos vèrtexs del costat.
ymax	Correspon a la coordenada **y** més gran dels dos vèrtexs. Ens servirà per saber quan hem de treure el costat de la llista de costats actius.

- **Taula de Costats**
  Aquesta taula emmagatzema els costats del polígon, en forma d'estructures de tipus **Costat**. La taula està formada per una llista de costats per cada coordenada **y** de pantalla. Els costats emmagatzemats en cadascuna d'aquestes llistes són els que tenen com a coordenada **y** mínima la corresponent a la posició de la taula, i estan ordenats per la **x** corresponent al vèrtex de coordenada **y** mínima. Per omplir aquesta taula, cal tenir en compte l'**algorisme de les singularitats**, que explicarem més endavant.

- **Llista de Costats Actius**
  Es tracta d'una llista de elements de tipus **Costat** ordenats per **x**. La reordenació d'aquests costats és necessària quan els costats del polígon a pintar es creuen. Aquesta llista **sempre** ha de tenir un nombre parell de costats; si això no passa és que us heu equivocat al fer l'algorisme (segurament al resoldre el problema de les singularitats).

Algorisme

L'algorisme a implementar és el següent:

```
Crear la TC
Inicialitzar la Llista de Costats Actius (LCA) com a buida
Per y des de la més petita a la més gran Fer
    Afegir els costats apuntats per TC[y] a la LCA, mantenint-la ordenada per la x
    Per a cada parella de costats consecutius de la LCA
        Pintar la línea (x_costat1, y)-(x_costat2, y)
    Fi Per
    Eliminar de la LCA els costats que per y + 1 deixen de ser actius ( ymax = y )
    Recalcular les x dels costats restants per la expressió x = x + dx
    Reordenar la LCA segons x
Fi Per
```

El problema de les singularitats

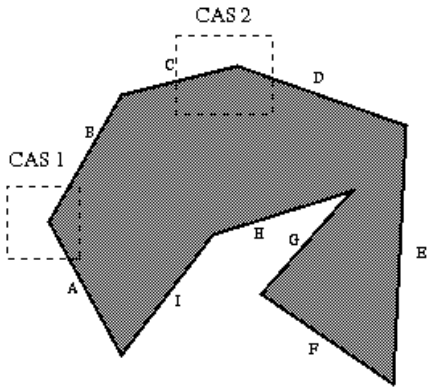
Hem vist que aquest algorisme es basa, a grans trets, en anar escanejant horitzontalment el polígon. Així, comencem a l'exterior i si creuem un costat, estem a l'interior, si tornem a creuar un costat, a l'exterior, si en creuem un tercer, a l'interior, etc. Aquest escanejat es fa gràcies a la LCA, ja que si us hi fixeu conté precisament aquesta informació, és a dir, per la coordenada **y** que està essent escanejada, la llista ordenada de les coordenades **x** dels costats que es creuen. És per això que pintem línies per parelles de costats consecutius, ja que el primer costat serà d'entrada al polígon i el segon de sortida.

Ara bé, hi ha casos en els que això no funciona.

En primer lloc, s'ha de resoldre què es fa quan tenim **costats horitzontals**, ja que amb aquests costats ens trobem amb una divisió per 0 en calcular el pendent. La solució és senzilla: no es consideren i, per tant, **no s'inclouen a la TC**.

D'altra banda, hi ha un altre problema quan creuem un vèrtex del polígon, doncs cada vèrtex pertany a dos costats i podríem estar considerant dos creuaments amb la línia d'escanejat quan en realitat només n'hi ha un.

Bàsicament, hem de distingir dos tipus de vèrtexs :



CAS 1

Els dos costats del vèrtex tenen la **mateixa direcció**, en concret, a l'exemple tots dos van cap amunt.
En els vèrtexs del tipus CAS 1, hi ha **canvi d'estat** quan el travessem (a l'exemple, passem d'estar fora a estar dins). Donat que aquest punt estarà repetit (un cop per cada costat), en realitat travessaríem dos costats, i per tant, segons l'heurística descrita fins ara, en el exemple tornaríem a estar fora, i no és així.
Per tant, el que farem serà considerar que el **costat B comença una coordenada 'y' més amunt,** per tal que només es travessi un costat en aquest vèrtex. Si en lloc de que tots dos anessin cap amunt, fos a l'inrevés, i tots dos anessin cap avall (vèrtex entre D i E, per exemple), el que faríem és que el costat (E, en aquest exemple) comencés una **y** més avall.

CAS 2

Els dos costats del vèrtex **no** tenen la **mateixa direcció,** en concret, a l'exemple, el primer puja i el segon baixa.
En els vèrtexs d'aquest tipus, **no** hi ha **canvi d'estat** quan el travessem. Per tant, ja ens va bé que hi hagi una duplicitat, i per tant **no toquem** res.

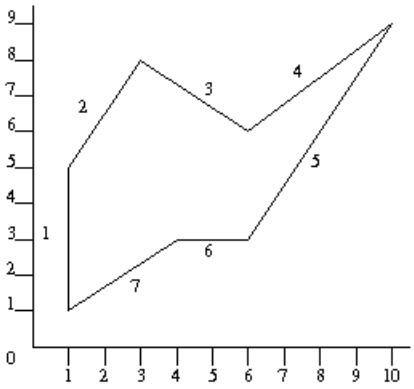
Així doncs, ara ja sabem com hem de crear la TC. L'algorisme és el següent :

```
A = Primer costat començant pel final no horitzontal del polígon
Per a tots els costats B no horitzontals del polígon Fem
    dyA = yfinalA - yinicialA
    dyB = yfinalB - yinicialB
    Si dyA y dyB tenen el mateix signe Aleshores
        (Hem de treure un punt del costat B)
        Si yinicialB < yfinalB Aleshores
            (El costat baixa)
            GuardarCostatTC amb posicioTC = yinicialB + 1,
                               dx = (xfinalB - xinicialB) / (yfinalB -
yinicialB),
                               x = xinicialB + dx,
                               ymax = yfinalB.
        Sino
            (El costat puja)
            GuardarCostatTC amb posicioTC = yfinalB,
                               dx = (xfinalB - xfinicialB) / (yfinalB -
yinicialB),
                               x = xfinalB,
                               ymax = yinicialB - 1.
        Fi Si
    Sino
        (No toquem res)
        Si yinicialB < yfinalB Aleshores
            (El costat baixa)
            GuardarCostatTC amb posicioTC = yinicialB,
                               dx = (xfinalB - xinicialB) / (yfinalB -
yinicialB),
                               x = xinicialB,
                               ymax = yfinalB.
        Sino
            (El costat puja)
            GuardarCostatTC amb posicioTC = yfinalB,
                               dx = (xfinalB - xinicialB) / (yfinalB -
yinicialB),
                               x = xfinalB,
                               ymax = yinicialB.
        Fi Si
    Fi Si
    A = B
Fi Per
```

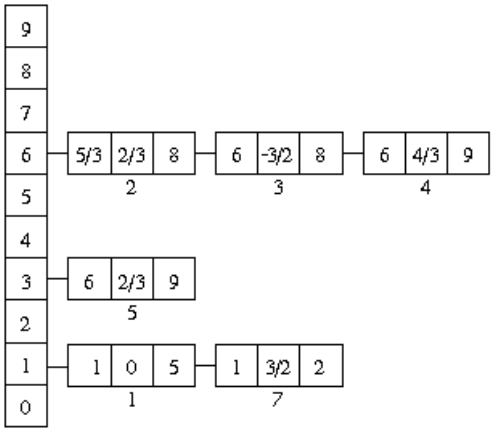
La funció `GuardarCostatTC` el que fa és crear un nou registre de tipus costats, omplir-hi les dades de `dx, x i ymax`, i ficar-ho a la llista apuntada per `TC[posicioTC]`.

Exemple

A continuació us mostrem un exemple d'execució d'aquest algorisme, per a que el seguiu:



Aquest és el polígon que s'ha de pintar. Si apliquem l'algoritme de creació de la TC, ens queda:



Ara que ja tenim la Taula de Costats, podem començar a iterar:

y = 1 :
Entren dos nous costats.

x	dx	ymax
1	0	5
1	3/2	2

Es pinta la línia: (1, 1)-(1, 1)

y = 2 :

x	dx	ymax
1	0	5
5/2	3/2	2

Es pinta la línia: (1, 2)-(2, 2)
Eliminem el segon costat perquè ja no estarà actiu.

y = 3 :

Entra un nou costat, que col·loquem en segon lloc.

x	dx	ymax
1	0	5
6	2/3	9

Es pinta la línia: (1, 3)-(6, 3)

y = 4 :

x	dx	ymax
1	0	5
20/3	2/3	9

Es pinta la línia: (1, 4)-(7, 4)

y = 5 :

x	dx	ymax
1	0	5
22/3	2/3	9

Es pinta la línia: (1, 5)-(7, 5)
Eliminem el primer costat perquè ja no estarà actiu.

y = 6 :
Entren tres nous costats, que col·loquem davant.

x	dx	ymax
5/3	2/3	8
6	-3/2	8
6	4/3	9
8	2/3	9

Es pinten les línies: (2, 6)-(6, 6) i (6, 6)-(8, 6)

y = 7 :

x	dx	ymax
7/3	2/3	8
9/2	-3/2	8
22/3	4/3	9

26/3	2/3	9

Es pinten les línies: (2, 7)-(4, 7) i (7, 7)-(9, 7)

y = 8 :

x	dx	ymax
3	2/3	8
3	-3/2	8
26/3	4/3	9
28/3	2/3	9

Es pinten les línies: (3, 8)-(3, 8) i (9, 8)-(9, 8)
Eliminem els dos primers costats que després ja no estaran actius.

y = 9 :

x	dx	ymax
10	4/3	9
10	2/3	9

Es pinta la línia: (10, 9)-(10, 9)

NOTA: Els punts de les línies poden variar d'un, depenent si esteu arrodonint o truncant. En aquest exemple hem arrodonit.

QUÈ S'HA DE FER?

A la primera sessió s'explicaran els algorismes a implementar, així com les pautes a seguir en la programació de la pràctica. De tota manera, us pot ser molt útil que porteu la pràctica començada. Com a mínim, llegiu-vos bé l'enunciat, feu una ullada als fitxers de la pràctica i implementeu una funció per dibuixar finestres (`void finestra(int x0,int y0,int x1,int y1)`) i una altra per dibuixar polígons (`void dibuixa_poligon(poligon *pol`)). A la llibreria "graphics.h" del TURBO C trobareu funcions per dibuixar línies, rectangles, etc. que heu d'utilitzar per implementar aquestes funcions.

Al llarg de les dues sessions, heu d'implementar els algorismes i provar que funcionin correctament amb els exemples que se us proporcionen. Abans del final de la segona sessió, s'haurà d'entregar la pràctica (execució a l'aula amb els exemples i entrega del fitxer 'prac.c' en disquet). És molt important que abans d'entregar la pràctica, hàgiu provat la pràctica exhaustivament.

Per superar l'entrega de la pràctica, aquesta ha de funcionar correctament amb TOTS els exemples. A més a més, l'execució del programa ha de finalitzar correctament, sense que el PC es quedi bloquejat ni surtin missatges d'error ('NULL POINTER ASSIGNMENT', etc...). El codi font entregat en disquet, serà avaluat posteriorment. Recordeu que una pràctica copiada suposa el suspens de l'assignatura.

Si en finalitzar la segona sessió, encara no teniu acabada la pràctica, teniu de temps per entregar-la (en hores de consulta) fins el dijous 18 de novembre, però la pràctica us puntuarà sobre 6.5. Aquesta segona data d'entrega és improrrogable. Després del 18 de novembre no s'acceptarà cap pràctica.

El mínim per aprovar la pràctica és la part obligatòria funcionant correctament. La part opcional no s'avaluarà si la part obligatòria no supera les proves.

Part obligatòria

Heu d'implementar :

- Una funció que faci el retall d'un polígon segons l'algorisme de Sutherland-Hodgman. El prototipus ha de ser :

```
void retall_SH(int x0,int y0,int x1,int y1,poligon *pol);
```

- Una funció que pinti un polígon segons l'algorisme Y-X. El prototipus ha de ser :

```
void emplenat_Y_X(poligon *pol);
```

Part opcional

Heu d'implementar :

- Una funció que ompli una regió definida per píxels segons l'algorisme de la llavor millorat (veure bibliografia). El prototipus ha de ser :

```
void llavor_millorat(int x0,int y0,poligon *pol)
```

IMPORTANT: Les operacions sobre el polígon original s'han d'anar acumulant, és a dir, que el polígon resultat d'una operació de retall o emplenat és l'entrada per a la següent operació.

Fitxers de la pràctica

Els algorismes que implementeu, s'han d'integrar al fitxer PRAC.C que és el codi font d'un intèrpret de comandes. En aquest fitxer, ja teniu les capçaleres de les funcions, així que només heu d'omplir el cos de les funcions amb el vostre codi. Un cop compilat, la crida al programa és la següent:

prac *fitxer_de_comandes fitxer_de_dades*

Els fitxers de comandes (extensió '.gr1') contenen una seqüència d'ordres de l'estil:

```
REM Fitxer de comandes
CLS
OMPLE
PAUSA
RETALLA 40 40 210 160
PAUSA
RETALLA 100 100 300 300
LLAVOR 150 150
```

Els fitxers de dades (extensió '.pol') contenen la informació referent als vèrtexs d'un polígon. Tenen la forma:

```
8
80 20
200 30
170 60
260 110
190 170
150 150
110 170
20 80
```

El primer número (en aquest cas el 8) correspon al número de vèrtexs que té el polígon. A continuació venen les parelles de coordenades (x,y) corresponents als vèrtexs del polígon.

El programa '**prac.exe**' llegeix el polígon que li passem i el carrega en l'estructura **`poligon.`** La definició del tipus **`poligon`** és la següent :

```
typedef struct {
    int X;
    int Y;
} Punt;

typedef struct {
    int n;
    Punt *P;
} Poligon;
```

'n' és el nombre de vèrtexs del polígon i '*P' és un apuntador a la llista de punts corresponents als vèrtexs.

Tots els altres tipus que us facin falta els heu de definir vosaltres mateixos. El polígon es crea, es llegeix i s'allibera en el programa, per tant, no cal que ho feu vosaltres. Ara bé, si en creeu de nous, ho heu de controlar.

Un cop carregat el polígon, l'intèrpret llegeix les instruccions del fitxer de comandes i fa les crides a les funcions corresponents. En el cas de les instruccions 'RETALLA', 'OMPLE' i 'LLAVOR', les crides són a les funcions que heu d'implementar.

Fitxers de proves

Fitxers de dades:

EXEMPLE.POL
EXEMPLE2.POL
CUC.POL
ESTRELLA.POL
RARO.POL

Fitxers de comandes:

C1.GR1 per provar amb els polígons 'exemple', 'exemple2' i 'cuc'
C2.GR1 per provar amb el polígon 'estrella'
C3.GR1 per provar amb el polígon 'raro'

Per a qualsevol dubte, envieu un mail a **robert@cvc.uab.es**

BIBLIOGRAFIA

Algorismes de retall de polígons

- *Apunts de Gràfics*, Juny 2000 , pàgs 53-57
- D. Hearn, M.P. Baker, *Computer Graphics*, 2nd edition. Prentice-Hall, 1994, pàgs 224-248
- F. Foley, A. van Dam, S.K. Feiner, J.F. Hughes, *Computer Graphics, Principles And Practique*, 2nd edition, Addison-Wesley 1990, pàgs 110-127, pàgs 924-945

Algorismes d'emplenat de polígons

- *Apunts de Gràfics*, Juny 2000, pàgs 37-41

- D. Hearn, M.P. Baker, *Computer Graphics*, 2nd edition. Prentice-Hall, 1994, pàgs 117-126
- F. Foley, A. van Dam, S.K. Feiner, J.F. Hughes, *Computer Graphics, Principles And Practique*, 2nd edition, Addison-Wesley 1990, pàgs 92-99

Algorismes de la llavor

- *Apunts de Gràfics*, Juny 2000, pàgs 41-43
- D. Hearn, M.P. Baker, *Computer Graphics*, 2nd edition. Prentice-Hall, 1994, pàgs 127-130
- F. Foley, A. van Dam, S.K. Feiner, J.F. Hughes, *Computer Graphics, Principles And Practique*, 2nd edition, Addison-Wesley 1990, pàgs 979-986