```
0  # Jeu de la vie (ISN)
   # Initialisation
1  OuvirConsole ; // il faudra donner les règles ici
2  Nb_ligne ? → y ;   Nb_colonne ? → X ;
3  Côté = 30 px ;
4  Hauteur = y * (Côté + 1) ;   Largeur = X * (Côté + 1) ;
5  Def = Hauteur * Largeur ;

6  Créer_Obj : Point (x, y) {
     • présence = 0 ;
     • Si (x+côté+1 < Largeur) : Voisin_0 = Point (x+côté+1 , y) ;
       Si (y-côté-1 > 0) : Voisin_1 = Point (x , y-côté-1) ;
       Si (x-côté-1 > 0) : Voisin_2 = Point (x-côté-1 , y) ;
       Si (y+côté+1 < Hauteur) : Voisin_3 = Point (x , y+côté+1) ;

     • Pour (i=0 , Nb_Voisins = 0 ; i < 4 ; i++) {
       Si (Voisin_i)  :  Nb_Voisins += Voisin_i . présence ;
           Sinon  Nb_Voisins += 0 ;
       }

     • colorier (const Couleur = 0) {
           Pour (j = y ; j < y+Côté ; j++) {
               Pour (k = x ; k < x+Côté ; k++) {
                   Pixel [k][j] . Color (Couleur , Couleur , Couleur) ;
               }
           }
       }
   }

7  Nb_Génération ? → G ;
```

```
8   Tab_Point [X][Y];
9   Pour (b=0; b<Y; b++){
        Pour (a=0; a<X; a++){
            Tab_Point [a][b] = Point(a*(côté+1)+1, b*(côté+1)+1);
        }
    }

10  # Traitement
    # Partie A : sélectionner la situation de Départ.
11  Ouvrir Fenêtre Graph (Largeur, Hauteur);
12  Pour (j=0, k=côté; j<Hauteur; j++){
        Tant que (k<Largeur){
            Pixel [k][j]. Color (0, 0, 0);
            k += côté + 1;
        }
    } // affiche les colonnes de la grille
13  Pour (k=0, j=côté; k<Largeur; k++){
        Tant que (j<Hauteur){
            Pixel [k][j]. Color (0, 0, 0);
            j += côté + 1;
        }
    } // affiche les lignes de la grille.
```

```
14  Quand 'click' dans la Fenêtre Graph {
        Obtenir Coordonnées_Souris (w, z);
        Case_Loc x = w - (w % (côté + 1));
        Case_Loc y = z - (z % (côté + 1));
        Tab_Loc a = (Case_Position x - 1) ÷ (côté + 1);
        Tab_Loc b = (Case_Position y - 1) ÷ (côté + 1);
        Tab_Point [Tab_Loc a] [Tab_Loc b]. colorier ();
        Tab_Point [Tab_Loc a] [Tab_Loc b]. présence = 1;
    }

15  Si (ENTER : 'down') : alors continuer;

16  # Partie B : exécution

17  Pour (g = 0; g < G; g++) {
        Pour (b = 0; b < Y; b++) {
            Pour (a = 0; a < X; a++) {
                Si (Tab_Point [a] [b]. Nb_Voisins = 2):
                    Tab_Point [a] [b]. colorier ();
                Sinon: Tab_Point [a] [b]. colorier (255);
            }
        }
        Pour (a = 0, b = 0, k = 0, j = 0; (a < X) ou (b < Y); ) {
            Tant que (j < Hauteur) {
                k = 0;
                Tant que (k < Largeur) {
                    Si (Pixel [k] [j]. Color = (0, 0, 0)):
                        Tab_Point [a] [b]. présence = 1;
```

```
                Sinon : Tab_Point[a][b].presence = 0;
                k += côté + 1 ;
                a += 1 ;
            }
        j += côté + 1 ;
        b += 1 ;
    }
Si ((a < x) OU (b < y)) :
    Fermer FenêtreGraph ;
    Alert ( Erreur ) ;
    Return 0 ; // fin de programme
}
} // fin boucle G

# Fin d'execution
Si n'importe quel bouton est pressé :
    Fermer FenêtreGraph ;
Alert ( Fin du programme, press 'q' to quit) ;
Si 'q' est pressé : Return 0 ;
```