Algorithmique et Programmation

—

Examen sur machine: solution

—

G1: keriven*(at)*certis.enpc.fr G2: pons*(at)*certis.enpc.fr
G3: aganj*(at)*certis.enpc.fr G4: tousch*(at)*certis.enpc.fr
G5: segonne*(at)*certis.enpc.fr G6: chariot*(at)*certis.enpc.fr

—

07/12/07

—

```
1  // Examen sur machine 2007
2  #include <CL/Graphics/Graphics.h>
3  #include<iostream>
4  using namespace CL::Graphics;
5  using namespace std;
6
7  const int N=12;
8  const int zoom=512/N;
9  const int margin=zoom/5;
10 const double freq=0.3;
11
12 // Remplit le tableau avec des murs aux bords et une frequence 'freq' ailleurs
13 void Init(bool t[N][N]) {
14     for (int i=0;i<N;i++)
15         for (int j=0;j<N;j++)
16             t[i][j]=(i==0 ||j==0 || i==N-1 || j==N-1 || rand()<int(RAND_MAX*freq));
17 }
18
19 // Dessine une case (eventuellement avec une marge s)
20 void Draw(int i,int j,Color c,int s=0) {
21     FillRect(i*zoom+s,j*zoom+s,zoom-2*s,zoom-2*s,c);
22 }
23
24 // Dessine le terrain
25 void Draw(bool t[N][N]) {
26     for (int i=0;i<N;i++)
27         for (int j=0;j<N;j++)
28             if (t[i][j])
29                 Draw(i,j,Red);
30 }
31
32 // Désigne un point à la souris
33 void ClickPoint(int &i,int &j,bool t[N][N]) {
34     do {
35         int x,y;
36         GetMouse(x,y);
37         i=x/zoom;j=y/zoom;
38     } while (t[i][j]);
39 }
40
41 // ================= version sans memoire du meilleur chemin =============
```

```
42  // Fonction recursive
43  void Chemin1Rec(int l,int i,int j,int i1,int j1,bool t[N][N])
44  {
45      if (t[i][j])
46          return;
47      if (i==i1 && j==j1) {
48          cout << "Trouve un chemin de longueur: " << l << endl;
49          return;
50      }
51      t[i][j]=true;
52      Draw(i,j,Blue,margin);
53      Chemin1Rec(l+1,i+1,j,i1,j1,t);
54      Chemin1Rec(l+1,i-1,j,i1,j1,t);
55      Chemin1Rec(l+1,i,j+1,i1,j1,t);
56      Chemin1Rec(l+1,i,j-1,i1,j1,t);
57      t[i][j]=false;
58      Draw(i,j,White,margin);
59  }
60
61  // Appel initial
62  void Chemin1(int i0,int j0,int i1,int j1,bool t[N][N]) {
63      Chemin1Rec(0,i0,j0,i1,j1,t);
64  }
65
66  // ================== version avec memoire du meilleur chemin =============
67  // Fonction recursive
68  void Chemin2Rec(int l,int i,int j,int i1,int j1,bool t[N][N],
69                  int curi[N*N],int curj[N*N],
70                  int besti[N*N],int bestj[N*N],int& bestl) {
71      if (t[i][j])
72          return;
73      if (i==i1 && j==j1) {
74          cout << "Trouve un chemin de longueur: " << l << endl;
75          if (l<bestl) {
76              for (int c=1;c<l;c++) {
77                  besti[c]=curi[c];
78                  bestj[c]=curj[c];
79              }
80              bestl=l;
81          }
82          return;
83      }
84      t[i][j]=true;
85      Draw(i,j,Blue,margin);
86      curi[l]=i;curj[l]=j;
87      Chemin2Rec(l+1,i+1,j,i1,j1,t,curi,curj,besti,bestj,bestl);
88      Chemin2Rec(l+1,i-1,j,i1,j1,t,curi,curj,besti,bestj,bestl);
89      Chemin2Rec(l+1,i,j+1,i1,j1,t,curi,curj,besti,bestj,bestl);
90      Chemin2Rec(l+1,i,j-1,i1,j1,t,curi,curj,besti,bestj,bestl);
91      t[i][j]=false;
92      Draw(i,j,White,margin);
93  }
94
95  // Appel initial
96  void Chemin2(int i0,int j0,int i1,int j1,bool t[N][N]) {
97      int curi[N*N],curj[N*N];    // Chemin courant
98      int besti[N*N],bestj[N*N];  // Meilleur chemin
99      int bestl=N*N;
100     Chemin2Rec(0,i0,j0,i1,j1,t,curi,curj,besti,bestj,bestl);
101     if (bestl==N*N) {
102         cout << "Pas de chemin!" << endl;
```

```
103       } else {
104           cout << "Longueur minimale: " << bestl << endl;
105           for (int c=1;c<bestl;c++)
106               Draw(besti[c],bestj[c],Blue,margin);
107           Draw(i0,j0,Green,margin);
108       }
109   }
110
111   // Fonction principale
112   int main()
113   {
114       bool t[N][N];       // terrain
115       Init(t);
116       OpenWindow(N*zoom,N*zoom);
117       Draw(t);
118       int i0,j0,i1,j1;    // Points de depart et d'arrivee
119       ClickPoint(i0,j0,t);
120       Draw(i0,j0,Green,margin);
121       ClickPoint(i1,j1,t);
122       Draw(i1,j1,Green,margin);
123       //Chemin1(i0,j0,i1,j1,t);   // Sans memoire du meilleur chemin
124       Chemin2(i0,j0,i1,j1,t);             // Avec memoire
125       Terminate();
126       return 0;
127   }
```