ソースファイル名：lbm.c，実行ファイル名：lbmc.out

```
$ cp   ../common/lbm_ori.c .
$ cp   lbm_ori.c   lbm.c
$ vi   lbm.c
$ gfortran   -o   lbmc.out     lbm.c
$ ./ lbmc.out
```

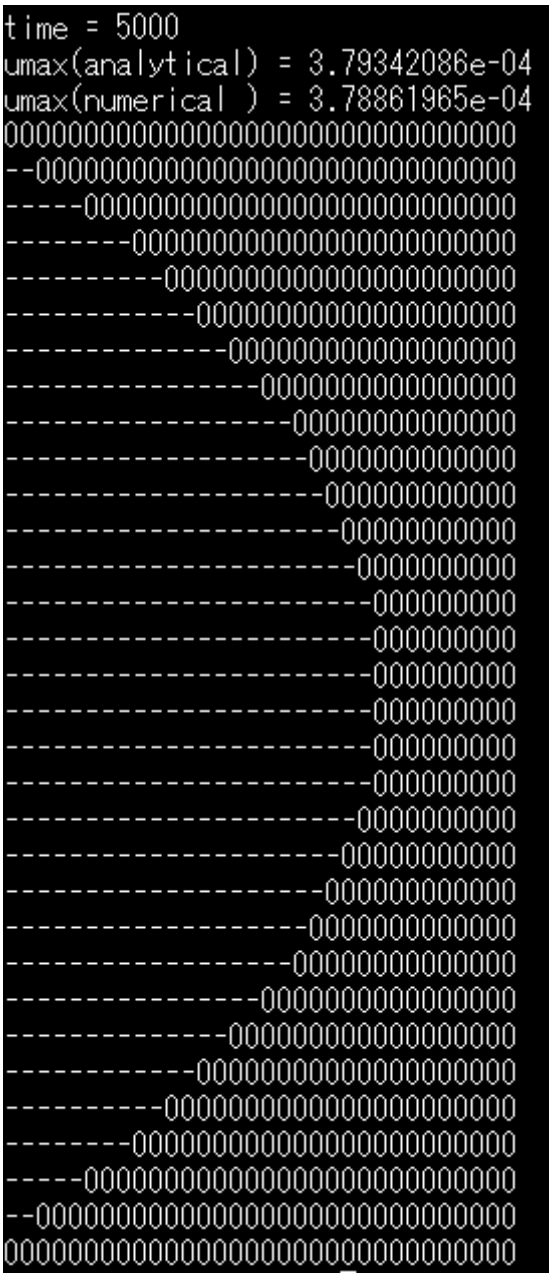
```
// Poiseuille flow
// the Lattice Boltzmann Method. The source code is written c programming language.


#include <stdio.h>
#include <stdlib.h>

# define DIM 64

main()
{
   int      nx = 32, ny = 32, time = 0., loop1, loop2;
   int      i, j, k, in, jn, nb, flag;
   float   u[DIM][DIM], v[DIM][DIM], rho[DIM][DIM], ftmp[9][DIM][DIM];
   float   f[9][DIM][DIM], f0[9][DIM][DIM], cx[9], cy[9];
   float   gx = 0.00001, gy = 0.0, tmp, u2, mu, tau = 10.;
   char     a[DIM][DIM];

   flag = 2;
   // initial condition
   mu = (tau - 0.5)/3.;

   for (i = 0; i < nx; i++) {
      for (j = 0; j < ny; j++) {
         u[i][j] = 0.; v[i][j] = 0.; rho[i][j] = 1.;
      }
   }

   cx[0] = 0.; cy[0] = 0.;
   cx[1] = 1.; cy[1] = 0.;
   cx[2] = 0.; cy[2] = 1.;
   cx[3] =-1.; cy[3] = 0.;
   cx[4] = 0.; cy[4] =-1.;
   cx[5] = 1.; cy[5] = 1.;
```

```
cx[6] =-1.; cy[6] = 1.;
cx[7] =-1.; cy[7] =-1.;
cx[8] = 1.; cy[8] =-1.;

for (i = 0; i < nx; i++) { for (j = 0; j < ny; j++) {
      u2 = u[i][j]*u[i][j] + v[i][j]*v[i][j];
      f[0][i][j] = rho[i][j]*(1. - 3./2.*u2)*4./9.;
      for (k = 1; k < 5; k++) {
         tmp = cx[k]*u[i][j] + cy[k]*v[i][j];
         f[k][i][j] = rho[i][j]*(1. + 3.*tmp + 9./2.*tmp*tmp - 3./2.*u2)/9.;
      }
      for (k = 5; k < 9; k++) {
         tmp = cx[k]*u[i][j] + cy[k]*v[i][j];
         f[k][i][j] = rho[i][j]*(1. + 3.*tmp + 9./2.*tmp*tmp - 3./2.*u2)/36.;
      }
} }

for(loop1 = 0; loop1 < 100; loop1++){
   for(loop2 = 0; loop2 < 50; loop2++){
   time++;

// collision
   for (i = 0; i < nx; i++) { for (j = 0; j < ny; j++) {
         u2 = u[i][j]*u[i][j] + v[i][j]*v[i][j];
         f0[0][i][j] = rho[i][j]*(1. - 3./2.*u2)*4./9.;
         for (k = 1; k < 5; k++) {
            tmp = cx[k]*u[i][j] + cy[k]*v[i][j];
            f0[k][i][j] = rho[i][j]*(1. + 3.*tmp + 9./2.*tmp*tmp - 3./2.*u2)/9.;
         }
         for (k = 5; k < 9; k++) {
            tmp = cx[k]*u[i][j] + cy[k]*v[i][j];
            f0[k][i][j] = rho[i][j]*(1. + 3.*tmp + 9./2.*tmp*tmp - 3./2.*u2)/36.;
         }
   } }

   for(i = 0; i < nx; i++){ for(j = 0; j < ny; j++){ for(k = 0; k < 9; k++){
            f[k][i][j] = f[k][i][j] - (f[k][i][j] - f0[k][i][j])/tau;
   } } }

   // force
```

```
for(i = 0; i < nx; i++){ for(j = 0; j < ny; j++){ for(k = 0; k < 9; k++){
        f[k][i][j] = f[k][i][j]    + rho[i][j]*(cx[k]*gx + cy[k]*gy)/6.;
} } }

// streaming
for(i = 0; i < nx; i++){ for(j = 0; j < ny; j++){ for(k = 0; k < 9; k++){
        ftmp[k][i][j] = f[k][i][j];
} } }
for(i = 0; i < nx; i++){ for(j = 0; j < ny; j++){
      in = i + 1; if(i == nx-1) in = 0;
      f[1][in][j] = ftmp[1][i][j];
} }
for(i = 0; i < nx; i++){ for(j = 0; j < ny-1; j++){
      jn = j + 1;
      f[2][i][jn] = ftmp[2][i][j];
} }
for(i = 0; i < nx; i++){ for(j = 0; j < ny; j++){
      in = i - 1; if(i == 0) in = nx-1;
      f[3][in][j] = ftmp[3][i][j];
} }
for(i = 0; i < nx; i++){for (j = 1; j < ny; j++){
      jn = j - 1;
      f[4][i][jn] = ftmp[4][i][j];
} }
for(i = 0; i < nx; i++){ for(j = 0; j < ny-1; j++){
      in = i + 1; if(i == nx-1) in = 0;
      jn = j + 1;
      f[5][in][jn] = ftmp[5][i][j];
} }
for(i = 0; i < nx; i++){ for(j = 0; j < ny-1; j++){
      in = i - 1; if(i == 0) in = nx-1;
      jn = j + 1;
      f[6][in][jn] = ftmp[6][i][j];
} }
for(i = 0; i < nx; i++){ for(j = 1; j < ny; j++){
      in = i - 1; if(i == 0) in = nx-1;
      jn = j - 1;
      f[7][in][jn] = ftmp[7][i][j];
} }
for(i = 0; i < nx; i++){ for(j = 1; j < ny; j++){
```

```
      in = i + 1; if(i == nx-1) in = 0;
      jn = j - 1;
      f[8][in][jn] = ftmp[8][i][j];
} }

// bounce back boundary condition
for(i = 0; i < nx; i++){
   if(flag == 1) {
// Bounce Back
   f[2][i][     0] = f[ ][i][     0];
   f[5][i][     0] = f[ ][i][     0];
   f[6][i][     0] = f[ ][i][     0];
   f[4][i][ny-1] = f[ ][i][ny-1];
   f[7][i][ny-1] = f[ ][i][ny-1];
   f[8][i][ny-1] = f[ ][i][ny-1];
   } else{

// Q. Zou and X. He, Phys. Fluids 9, 1591 (1997).

   rho[i][ny-1] =f[0][i][ny-1] + f[1][i][ny-1] + f[3][i][ny-1]
                + 2*(f[2][i][ny-1] + f[5][i][ny-1] + f[6][i][ny-1]);
   f[4][i][ny-1] = f[ ][i][ny-1];
   f[7][i][ny-1] = f[ ][i][ny-1] + 0.5*(f[ ][i][ny-1] - f[ ][i][ny-1]);
   f[8][i][ny-1] = f[ ][i][ny-1] - 0.5*(f[ ][i][ny-1] - f[ ][i][ny-1]);

   rho[i][0] = f[0][i][0] + f[1][i][0] + f[3][i][0]
             + 2*(f[4][i][0] + f[7][i][0] + f[8][i][0]);
   f[2][i][0] = f[ ][i][0];
   f[5][i][0] = f[ ][i][0] - 0.5*(f[ ][i][0] - f[ ][i][0]);
   f[6][i][0] = f[ ][i][0] + 0.5*(f[ ][i][0] - f[ ][i][0]);
   }
}

// physics
for(i = 0; i < nx; i++) { for(j = 0; j < ny; j++) {
      rho[i][j] = f[0][i][j]; u[i][j] = 0; v[i][j] = 0;
      for( k = 1; k < 9; k++){
         rho[i][j] = rho[i][j] + f[k][i][j];
            u[i][j] =     u[i][j] + f[k][i][j] * cx[k];
            v[i][j] =     v[i][j] + f[k][i][j] * cy[k];
```

```
            }
            if(rho[i][j] != 0){
               u[i][j] = u[i][j]/rho[i][j];
               v[i][j] = v[i][j]/rho[i][j];
            }else{
               u[i][j] = 0; v[i][j] = 0;
            }
     } }

     } //loop2

     printf("time = %d¥n",time);
     printf("umax(analytical) = %10.8e¥n",gx/8/nu*(ny-1)*(ny-1));
     printf("umax(numerical ) = %10.8e¥n",u[nx/2][ny/2]);

     for(j = 0; j < ny; j++) {for(i = 0; i < nx; i++) {
          a[i][j]='0';
     } }

     for(j = 0; j < ny; j++) {
          nb = u[nx/2][j] * 20000*nu;

          for(i = 0; i < nb; i++) {
               a[i][j]= '-';
          }
     }

     for(j = 0; j < ny; j++) {
          for(i = 0; i < nx; i++) {
               printf("%c",a[i][j]);
          }
          printf("¥n");
     }


   } //loop1

   return 0;
}
```


time = 5000
umax(analytical) = 3.79342086e-04
umax(numerical) = 3.78861965e-04