数理生物学演習

第9回　人工生命

第9回：人工生命

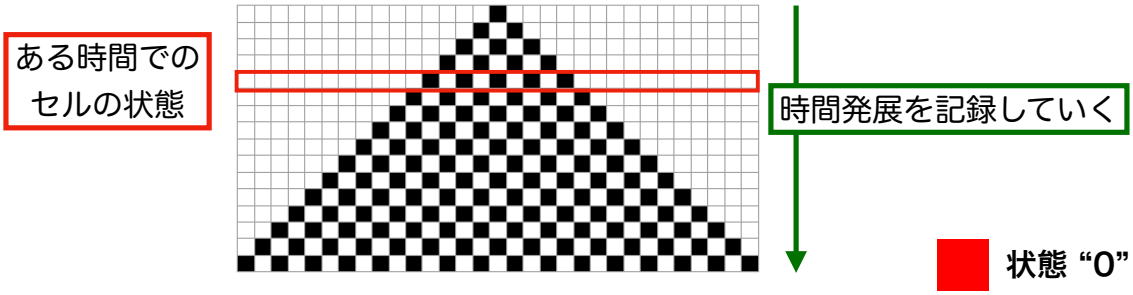
本日の目標

- セル・オートマトン
- ライフゲーム

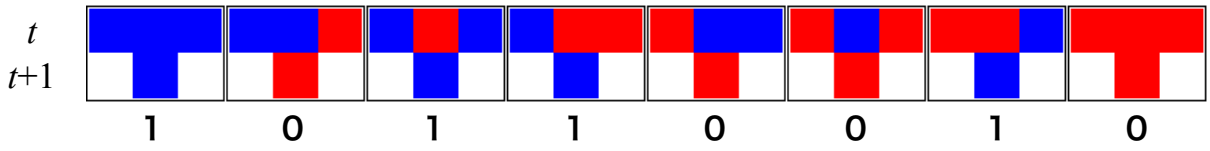
1次元セル・オートマトン

仮定

- セルが１次元的に並んでいる
- セルは状態“0”または“1”のいずれかをもつ
- セルは自身と近傍の状態により次のステップでの状態が決まる



遷移ルール（２近傍）



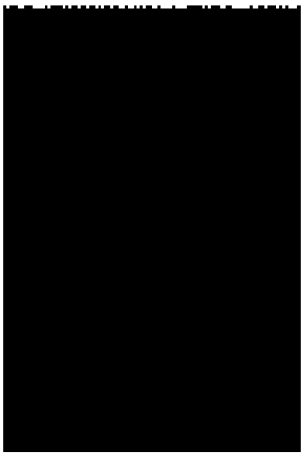
$1\times2^7+0\times2^6+1\times2^5+1\times2^4+0\times2^3+0\times2^2+1\times2^1+0\times2^0=178$

ルール178

ウルフラムのクラス

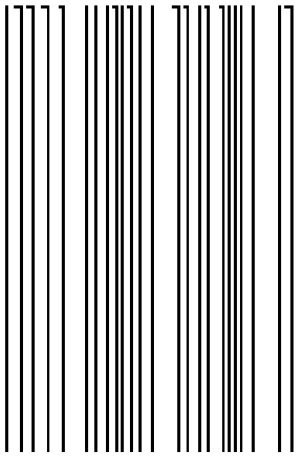
Wolfram (1983)

クラス1
セルの状態が
すべて同じになり，
変化が起こらない．



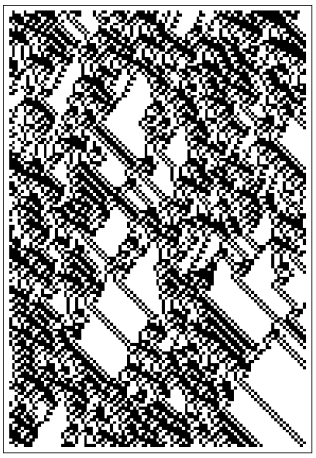
平衡点

クラス2
安定したパタンに落ち着き変化が周期的になる．



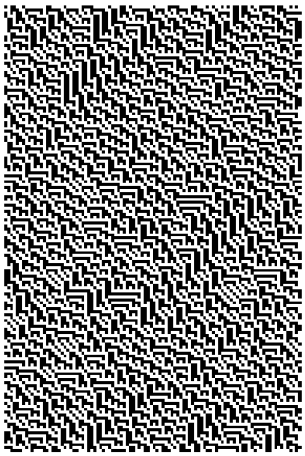
リミットサイクル

クラス4
あるときは規則的なパタンを示し，あるときはランダムに振る舞う．



複雑系

クラス3
全体がランダムに振る舞う．ただし，決定論的．



カオス

秩序
安定

無秩序
不安定

クラス4で“複雑さ”が最大になる
生命現象はここにあるのかも？

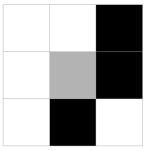
ライフゲーム Conway's Game of Life

仮定

- **各セルは状態“生”と“死”をもつ**
- **誕生，生存，死亡のプロセスを経て，“生”と“死”の状態を更新する**
- **８近傍のセルの状態により次の状態がきまる**
- **遷移ルールは誕生，維持，過疎，過密の４つ**

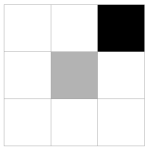
2次元のセル・オートマトンの特殊な場合．かなり，色々なパタンが観察できる．

誕生



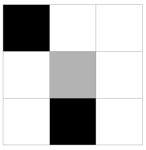
8近傍中
ちょうど3つが“生”ならば
次のステップで“生”

過疎



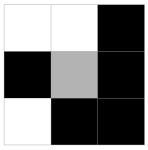
8近傍中
“生”が１つ以下ならば
次のステップで“死”

維持



8近傍中
ちょうど２つが“生”ならば
次のステップで更新なし
（“生”ならば“生”，“死”ならば“死”）

過密



8近傍中
“生”が４つ以上ならば
次のステップで“死”

ライフゲームにみられるパターンいろいろ

固定物体 still life

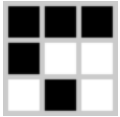
ブロック

振動子 oscillators

ブリンカー

移動物体 spaceship

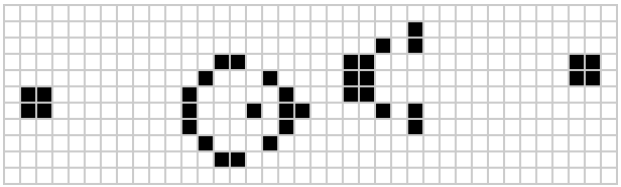
グライダー



繁殖型

銃 guns

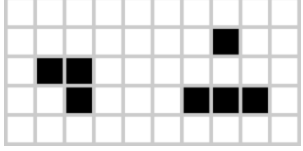
グライダー銃



シュシュポッポ列車 puffer

長寿 methuselahs

ダイハード



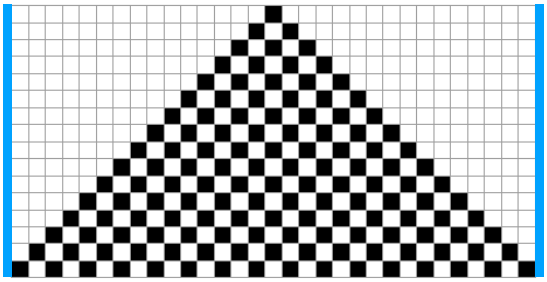
もっといろいろなパタンを知りたい人はLifeWiki http://www.conwaylife.com/wikiなどを参照

境界条件

プログラムを組むときも，
この部分の処理は注意！

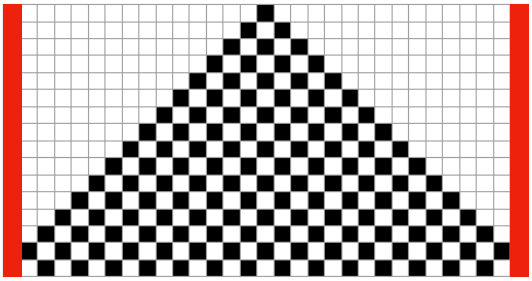
周期境界条件

“端”同士が張り合わされていると考える．



固定端

“端”の値を与えて，変動しないとする．



例えば，この端で常に状態“０”

実際にプログラムを組んでみよう！

9-1. 1次元セル・オートマトン

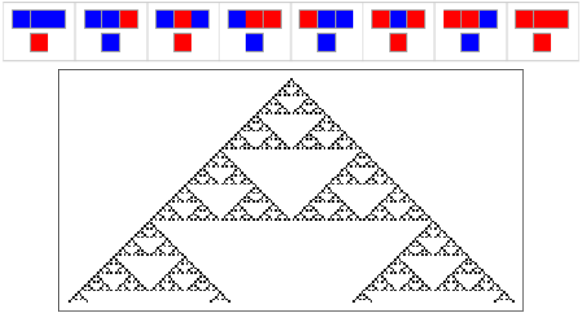
```
#include <stdio.h>

int rule(int cell1, int cell2,int cell3);

int main(void){
	int t,i;
	int cell[100];
	int tempcell[100];

	FILE *fp;
	fp=fopen("CA.csv","w");

	//配列の初期化
	for(i=0;i<100;i++){
		cell[i]=0;
		tempcell[i]=0;
	}

	//初期条件
	cell[50]=1;
	for(i=0;i<100;i++){
		fprintf(fp,"%d",cell[i]);
		if(i!=99){
			fprintf(fp, ",");
		}
	}
	fprintf(fp,"\n");

	for(t=1;t<100;t++){
		/*-----状態遷移------*/
		//境界条件処理その1

tempcell[0]=rule(cell[99],cell[0],cell[1]);
		//メイン
		for(i=1;i<99;i++){

tempcell[i]=rule(cell[i-1],cell[i],cell[i+1]);
		}
		//境界条件処理その2

tempcell[99]=rule(cell[98],cell[99],cell[0]);
		/*-----状態遷移ここまで------*/

		/*-----情報の更新と出力------*/
		for(i=0;i<100;i++){
			cell[i]=tempcell[i];
			fprintf(fp,"%d",cell[i]);
			if(i!=99){
				fprintf(fp, ",");
			}
		}
		fprintf(fp,"\n");
/*-----情報の更新と出力ここまで------*/
	}

	fclose(fp);
	return 0;
}

//ルール178
int rule(int cell1, int cell2,int cell3){
	if(cell1==1){
		if(cell2==1){
			if(cell3==1){
				return 1;
			}
			else if(cell3==0) {
				return 0;
			}
		}
		else if(cell2==0){
			if(cell3==1){
				return 1;
			}
			else if(cell3==0) {
				return 1;
			}
		}
	}

	else if(cell1==0){
		if(cell2==1){
			if(cell3==1){
				return 0;
			}
			else if(cell3==0) {
				return 0;
			}
		}
		else if(cell2==0){
			if(cell3==1){
				return 1;
			}
			else if(cell3==0){
				return 0;
			}
		}
	}
	return -1;
}
```

あまり賢い関数の定義の仕方ではない.
余裕のある人はもっと優れた実装方法を考えてみてください.

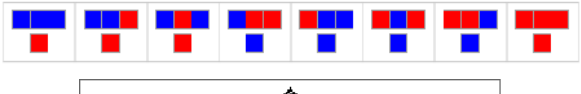
9-2. 別のルールを実装してみよう

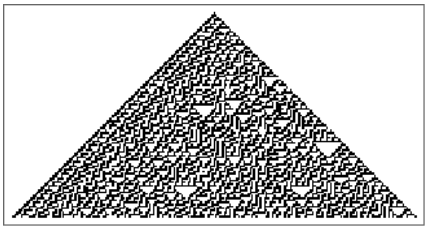
ルール0（クラス１）

ルール90（クラス２）

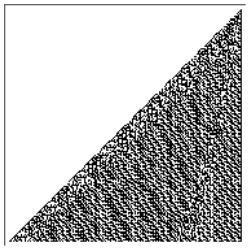


ルール30（クラス３）





ルール110（クラス４）



状態 “0”

状態 “1”

興味のある人は4近傍へもチャレンジしてみよう

配列

同じ型を持つ変数の集まり

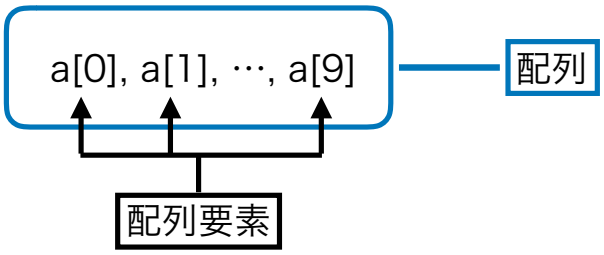
型　配列名[配列サイズ];
型　配列1[サイズ], 配列2[サイズ],…, 配列n[サイズ];

たくさんの変数を個別に宣言するのは面倒！

```
4-1. 配列
#include <stdio.h>

int main(void){
  int i;
  int a[10];

  for(i=0;i<10;i++){
    a[i]=i;
  }

  for(i=0;i<10;i++){
    printf("%d\n",a[i]);
  }

  return 0;
}
```

i番目の要素にiを代入

- 配列のなかのそれぞれの変数を配列要素という

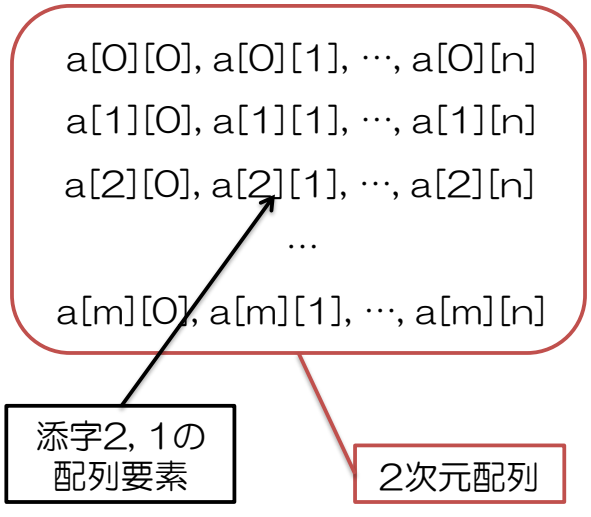


- 各要素へは添字によってアクセスする

特に注意！！
添字は0から始まり，(サイズ-1)で終わる
int a[10];　で定義したならば，
a[0]～a[9]までの要素が存在する

2次元配列

型　配列名[配列サイズ][配列サイズ];
型　配列1[サイズ][サイズ]，配列2[サイズ][サイズ]，
　…，配列n[サイズ][サイズ];



基本は1次元の場合と同じ．
より多次元の配列も定義できます！

```
10-1. 2次元配列
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(void){
  int i,j;
  int a[10][10];

  srand(time(NULL));

  for(i=0;i<10;i++){
    for(j=0;j<10;j++){
      a[i][j]=rand()%10;
    }
  }

  for(i=0;i<10;i++){
    for(j=0;j<10;j++){
      printf("%d",a[i][j]);
      if(j!=9){
        printf(", ");
      }
    }
    printf("¥n");
  }

  return 0;
}
```

ライフゲーム

9-3.
ライフゲームのプログラムを組んでみてください.

- 境界条件は周期または固定のいずれか好きな方を採用して良い
  （ただし固定端の場合は境界はすべて“死”）
- 格子のサイズは100×100で作る（もっと大きくしても良い）

やるべきこと
- 初期条件の設定
- 状態遷移ルールの実装
- 境界条件の処理
- 結果の出力
- Pythonでのデータの可視化

本日の課題

ノーマル：2つ選ぶ

ハード：全部

1. １次元セル・オートマトンのクラス1, 2, 3, 4をそれぞれ見つけ出し，示せ．また，どういった時に各クラスが出現するか考察せよ.
2. ライフゲームで固定物体，振動子，移動物体，繁殖のパタンをそれぞれ探しだし，その遷移課程を示せ.
3. 1次元セル・オートマトン，ライフゲームそれぞれについて，どういった生物学的解釈が出来るか？自分なりに考察せよ.
4. 質問，意見，要望等をどうぞ.

課題をPDFファイルにまとめて，Google フォームにて提出すること

次回予告
第10回：中間発表
6月18日

復習推奨

- 演習のこれまでの内容