

# プログラミングII

<http://bit.ly/Prog3i>

## 計算モデル（1）

前期 第8週

2019/6/11

# 前期期末までの内容

以下の3つについて、プログラミングの  
応用的な内容を扱う

- ▶ 計算モデル … オートマトンについて
- ▶ 言語処理系 … 処理系の実装について
- ▶ 数値計算 … 数値計算アルゴリズムについて

# 本日は・・・

- ▶ オートマトンと状態遷移図, 正規表現
- ▶ オートマトンの例
- ▶ C言語による実装方法

# 本日は・・・

- ▶ オートマトンと状態遷移図, 正規表現
- ▶ オートマトンの例
- ▶ C言語による実装方法

# 自動販売機の例

旧式の自動販売機

1本60円のジュース

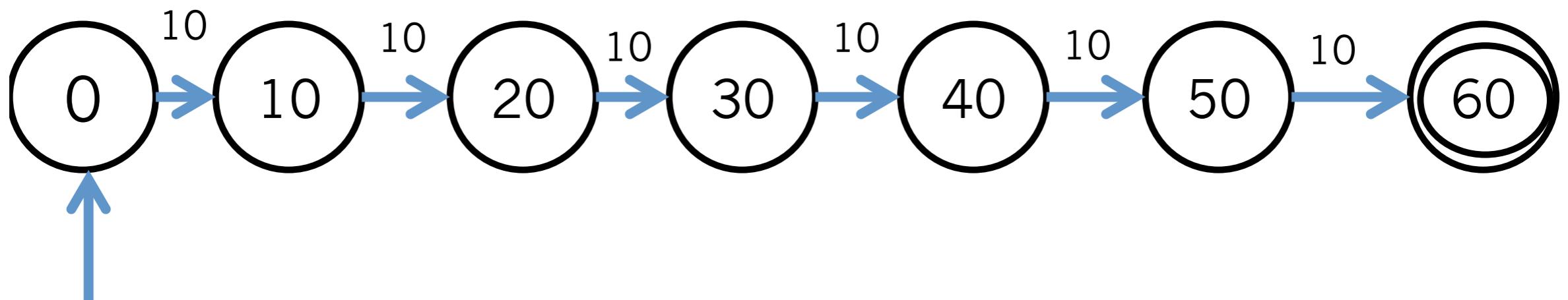
10円玉もしくは  
50円玉のみを受け付ける

60円に達したら  
ジュースを出して終了



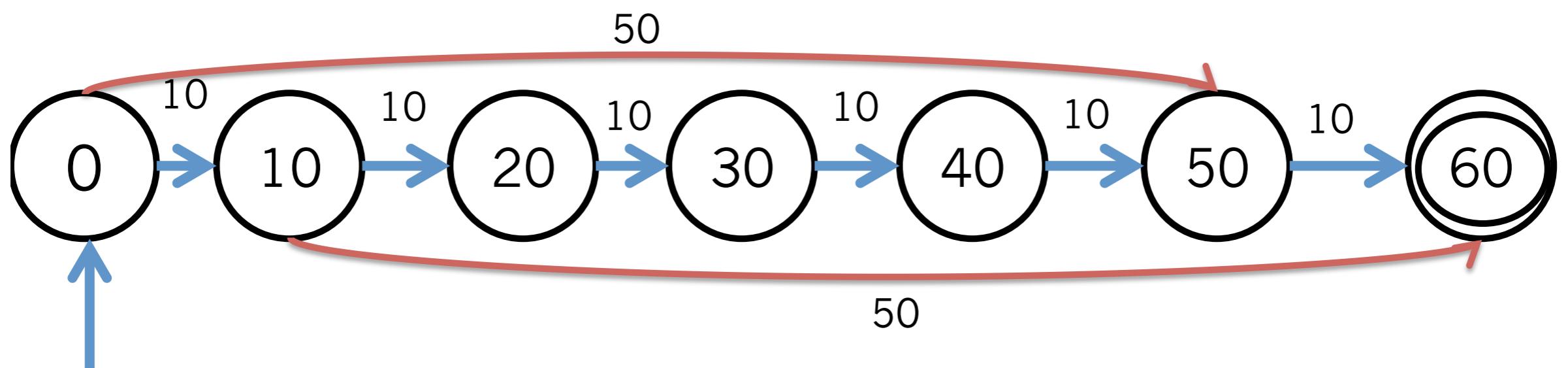
# 10円玉だけの場合

0からはじまり、10円玉を投入する度に  
次の状態に遷移していく



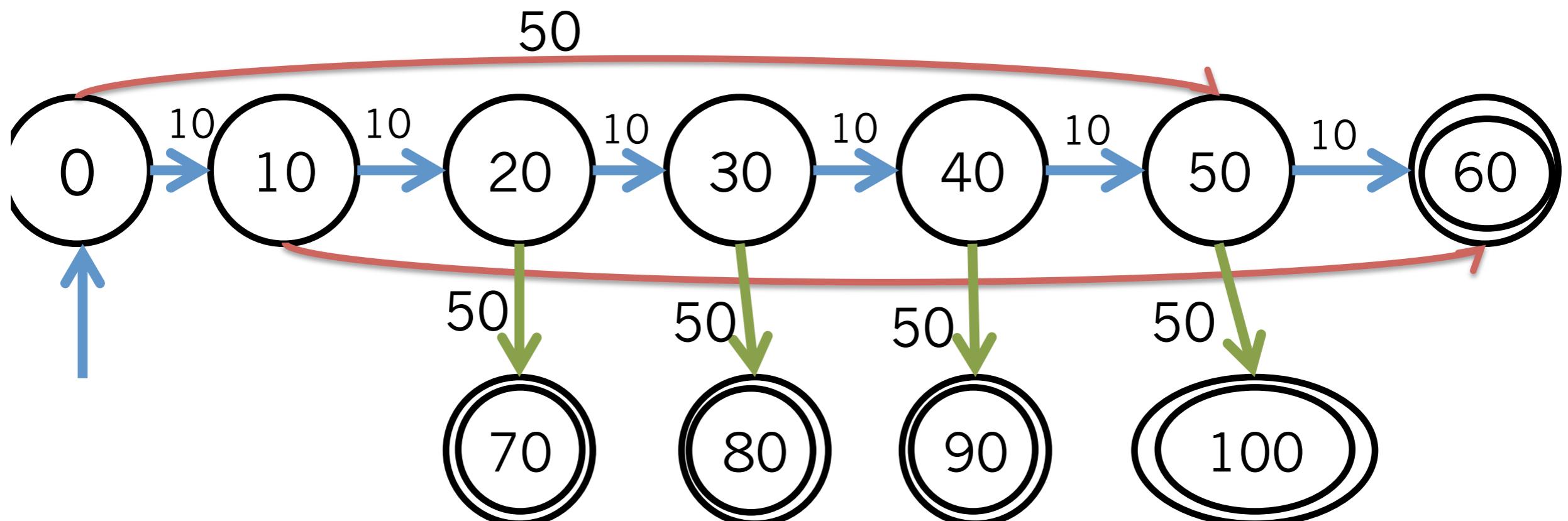
# 10円玉+50円玉だけの場合

0と10に50円の遷移を追加したが、これで十分？



# 10円玉+50円玉だけの場合

20, 30, 40, 50から50円を入れたときも  
ジュースを出して終了



# オートマトンとは？

## ▶ オートマトン

▶ 状態と状態間の遷移を表現したモデル

▶ 自販機の例では…

入っているお金の合計 → 状態

投入するコイン(10 or 50) → 遷移

## ▶ 決定性有限オートマトン

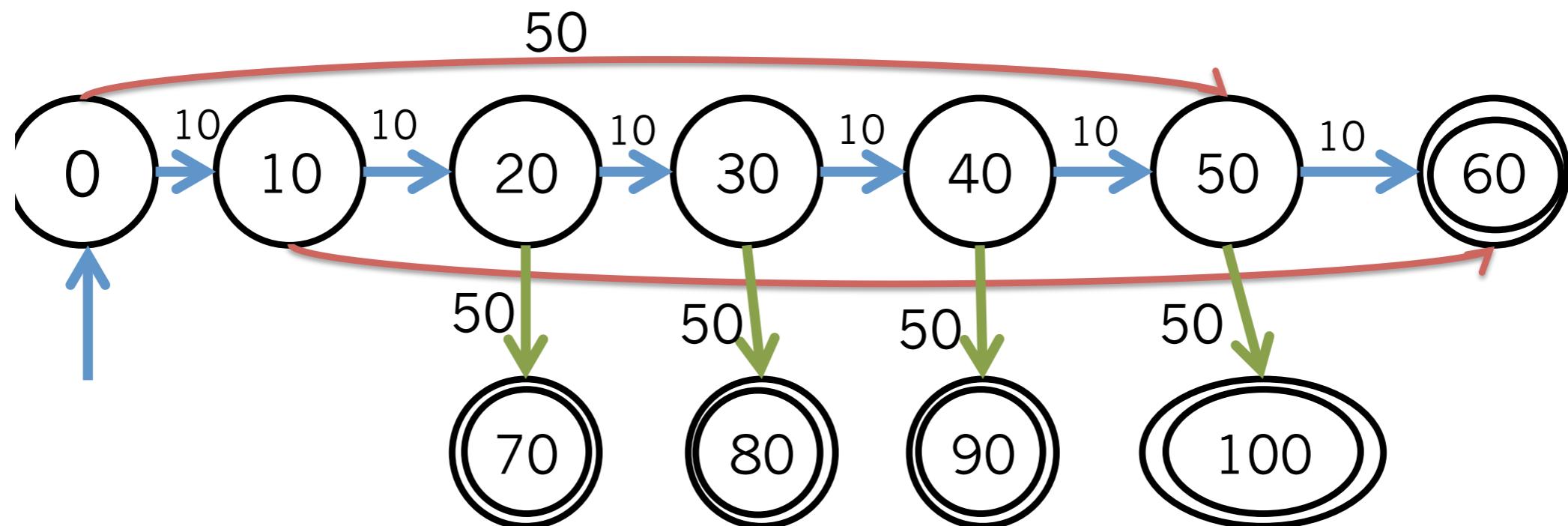
▶ 状態の数が有限個 (有限)

▶ 遷移先が一意に決まる (決定性)

# オートマトンの「受理」

入力を先頭から読み込み、末尾まで到達したときに  
終了状態に居れば「**受理**」となる

- ▶ 入力が10円玉+50円玉 → 受理
- ▶ 入力が10円玉+10円玉 → 受理しない



○は終了状態

# オートマトンの表現方法

3つの方法で表現できる

- ▶ 状態遷移図
- ▶ 形式的定義
- ▶ 正規表現

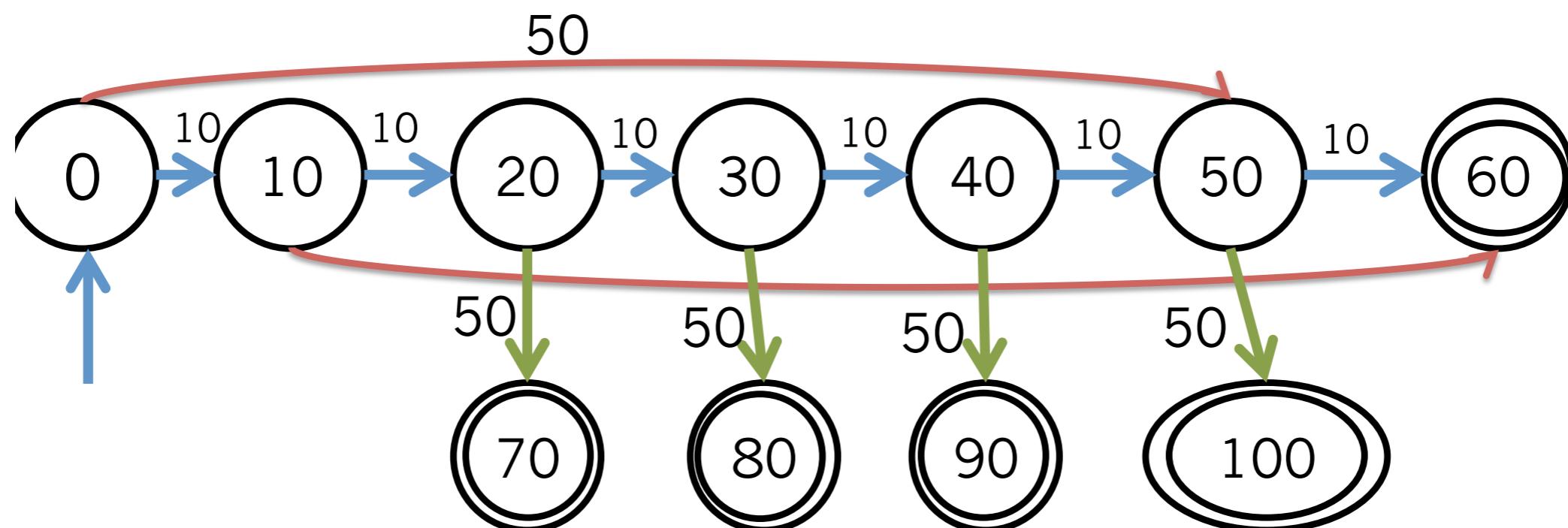
# 状態遷移図でオートマトンを表現

オートマトンの状態と遷移をグラフで表現

▶ ノードは状態を表現

初期状態には入り矢印、終了状態は二重丸 (○)

▶ 辺は遷移先を表現



# 形式的定義でオートマトンを表現

5つ要素のを使って形式的定義をする

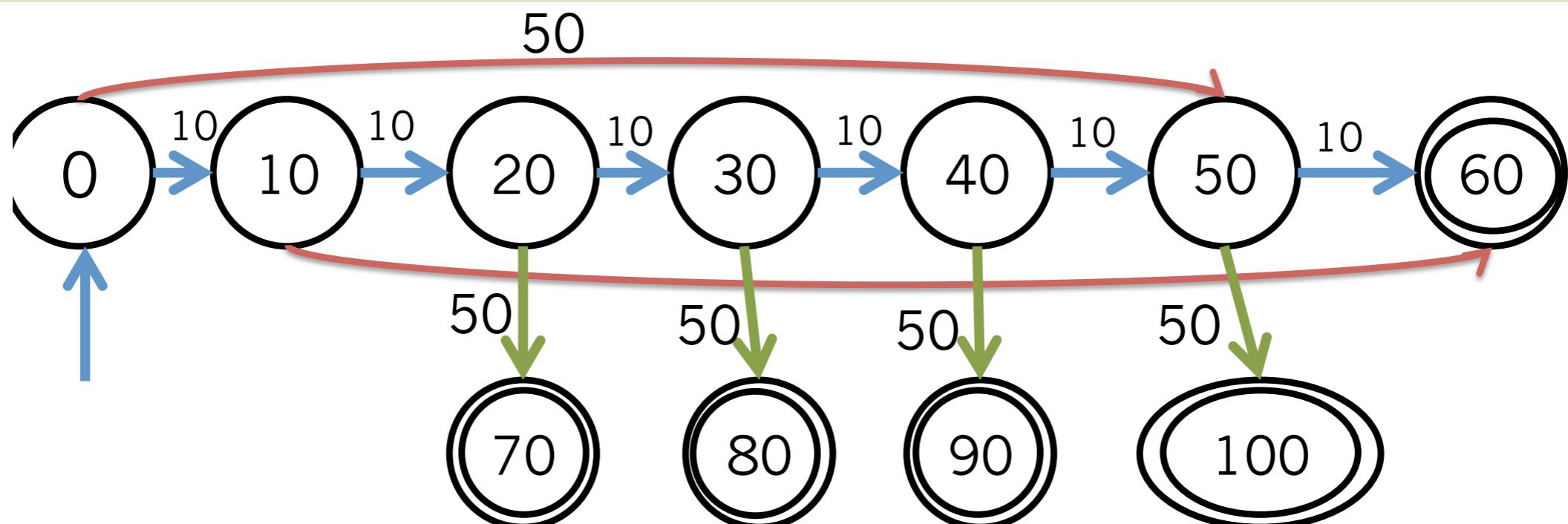
- ▶  $K$  : (状態集合)
- ▶  $\Sigma$  : (入力として受け付ける記号)
- ▶  $q_0$  : 初期状態
- ▶  $\delta$  : 状態遷移
- ▶  $F$  : 終了状態

# 形式的定義でオートマトンを表現

## 【例】自動販売機の形式的定義

例) 自動販売機の場合

1.  $K = \{0, 10, 20, 30, 40, 50, 60, \dots, 100\}$
2.  $\Sigma = \{10, 50\}$
3.  $q_0 = 0$
4.  $\delta = \{f(0, 10)=10, f(0, 50)=50, f(10, 10)=20, f(10, 50)=60, \dots, f(50, 50)=100\}$
5.  $F = \{60, 70, 80, 90, 100\}$



# 正規表現によるオートマトンの表現

正規表現を用いて記述することも出来る

- ▶ + は「または」を表す
- ▶ \* は0回以上の連続を表す

【例】  $(0+1)^*5$

「0もしくは1が0回以上連続し、  
最後が5で終わる数字列」を受理する

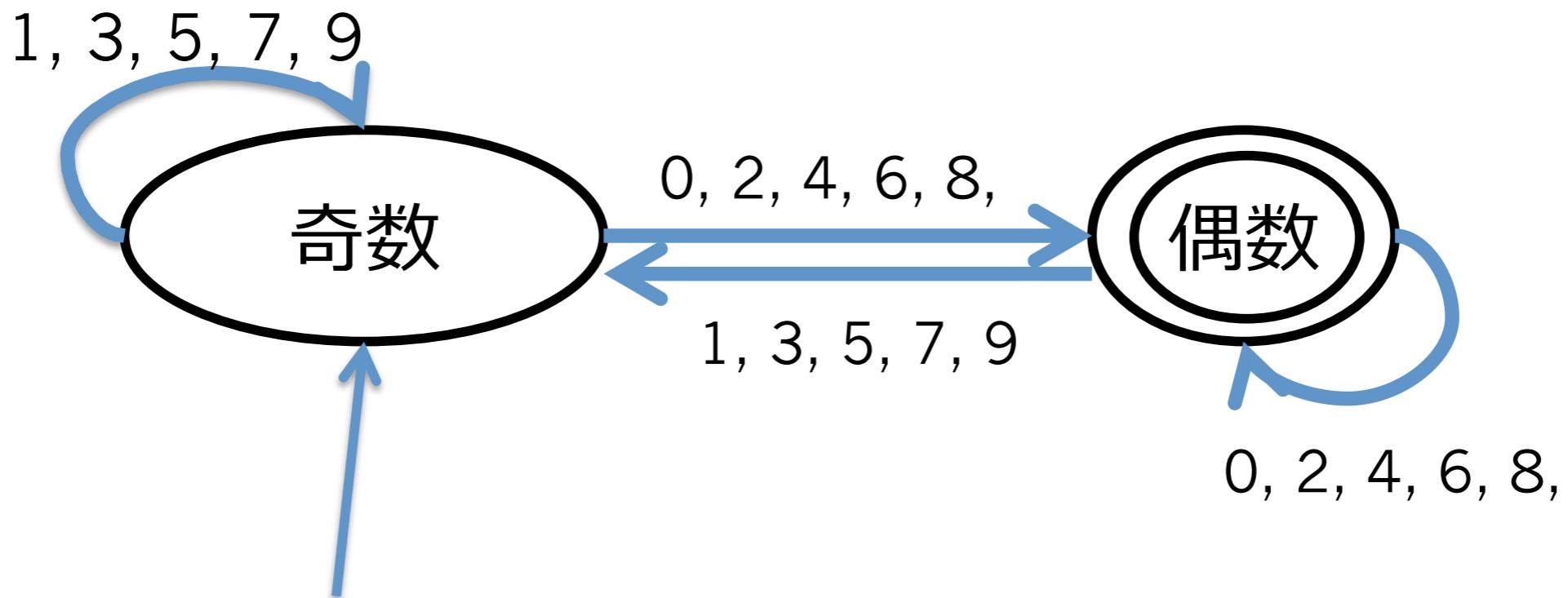
- ▶ 0005 : 受理
- ▶ 0105 : 受理
- ▶ 405 : 受理しない

# 本日は・・・

- ▶ オートマトンと状態遷移図, 正規表現
- ▶ オートマトンの例
- ▶ C言語による実装方法

# 【例】偶数を受理する状態遷移図

入力を先頭から読み込み末尾の数値が偶数ならば受理



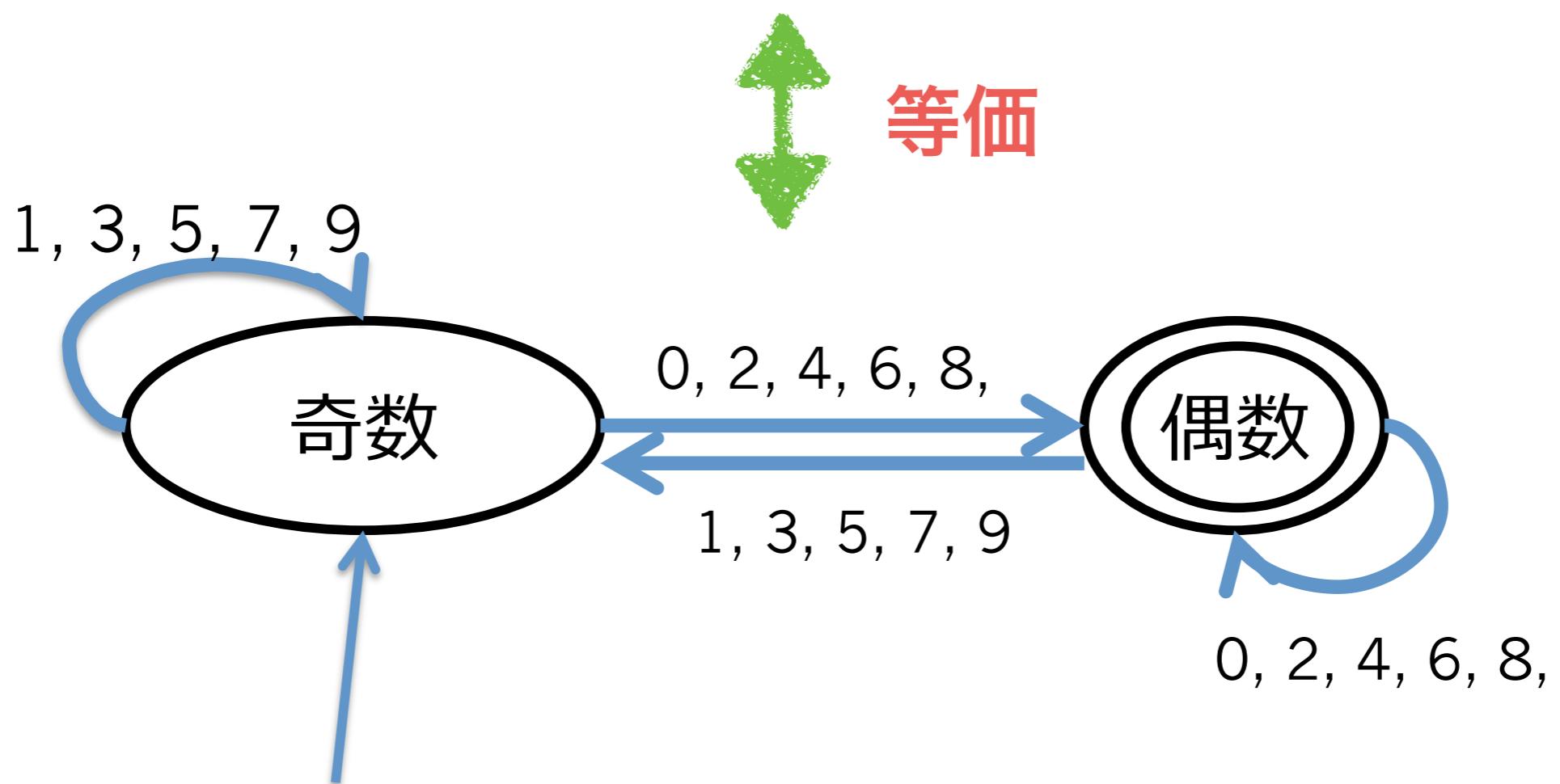
「124」を入力した場合

開始状態（奇数）

- 1（奇数のまま）
- 2（偶数に遷移）
- 4（偶数状態で終了=受理）

# 【例】偶数を受理する正規表現

$(0+1+2+3+4+5+6+7+8+9)^*(0+2+4+6+8)$



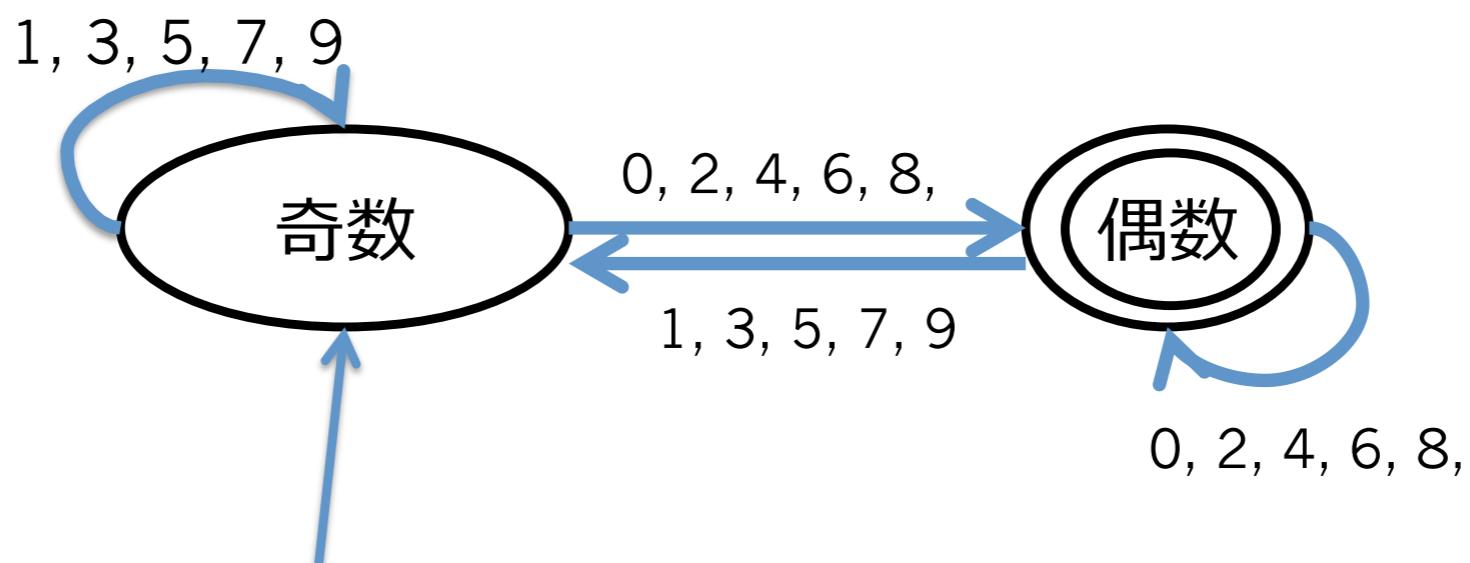
# 本日は・・・

- ▶ オートマトンと状態遷移図, 正規表現
- ▶ オートマトンの例
- ▶ C言語による実装方法

# C言語によるオートマトンの実装

## 大まかな流れ

- ①必要な変数を宣言  
(ユーザからの入力を格納する変数, 現在の状態を保持する変数)
- ②文字列を入力させる
- ③while文で先頭から1文字ずつ読み込みながら、  
switch文で状態を遷移する
- ④すべて読み込み終わったときに終了状態に居れば受理



# ①②の処理

```
/* (1) 変数を宣言 */
char input[100];           //入力を格納する配列
int i=0;                   //入力数をカウントする

//状態は、0が奇数、1が偶数の状態を表す
int current_state=0; //現在の状態（初期状態）
int fin_state=1;        //終了状態

/* (2) 文字列の入力 */
printf("数字を入力してください。\\n");
scanf("%s", input);
```

# ③の処理

```
/* (3) 先頭から順に読み込みながら状態遷移 */
while(input[i]!='\0') {  
    switch(current_state) {  
        case 0:  
            if(input[i]=='0' || input[i]=='2'  
                || input[i]=='4' || input[i]=='6'  
                || input[i]=='8') {  
                current_state = 1;  
            } else {  
                current_state = 0;  
            }  
            break;  
        case 1:  
            if(input[i]=='1' || input[i]=='3'  
                || input[i]=='5' || input[i]=='7'  
                || input[i]=='9') {  
                current_state = 0;  
            } else {  
                current_state = 1;  
            }  
            break;  
    }  
    printf("読み込んだ数値 : %c 遷移先 : %d\n", input[i], current_state);  
    i++;  
}
```

入力された文字列の終端文字まで繰り返す

現在の状態が0の場合（奇数）

次の状態に遷移

現在の状態が1の場合（偶数）

次の状態に遷移

# ④ の処理

```
/* (4) 終了状態にいるか判定 */  
if(current_state == fin_state) {  
    printf("受理する。\\n");  
} else {  
    printf("受理しない。\\n");  
}
```

全ての入力に対して遷移をした後に、  
終了状態ならば「受理する」

# 【課題の準備】

演習室で作業する前に、以下のコマンドを入れるだけで準備が完了する

```
$ mygitclone 「自分のGitHubユーザ名」  
$ cd prog3i-(ユーザ名)  
$ ./myconf
```

※本体をシャットダウンするまでは、  
上記「mygitclone」と「myconf」の設定は有効です

# 【課題の準備】

以下の流れで、課題のプログラムを作るためのフォルダを準備しましょう。

1. 端末を起動して、以下のコマンドを実行して**前期第8週のフォルダ**を作る

```
$ cd prog3i-(ユーザ名)          ←既に移動しているなら不要)
$ mkdir week108
$ cd week108
```

※課題で作るファイル名は各自で決めて構いません。

# 【練習8-1】

「sample108.c」のサンプルプログラムをコンパイルして実行結果を確認しましょう。

# 【課題8-1】

「入力を先頭から読み込み末尾の数値が**奇数**ならば受  
理」するプログラムを作成して下さい。  
(「sample108.c」の終了状態に注目)

# 【課題の提出】

以下の流れで、作ったCプログラムをGitHubにプッシュして、Webサイトで確認してみましょう。

1. 端末内で、以下のコマンドで課題を提出

```
$ git add -A  
$ git commit -m "課題8-1提出"  
$ git push origin master
```

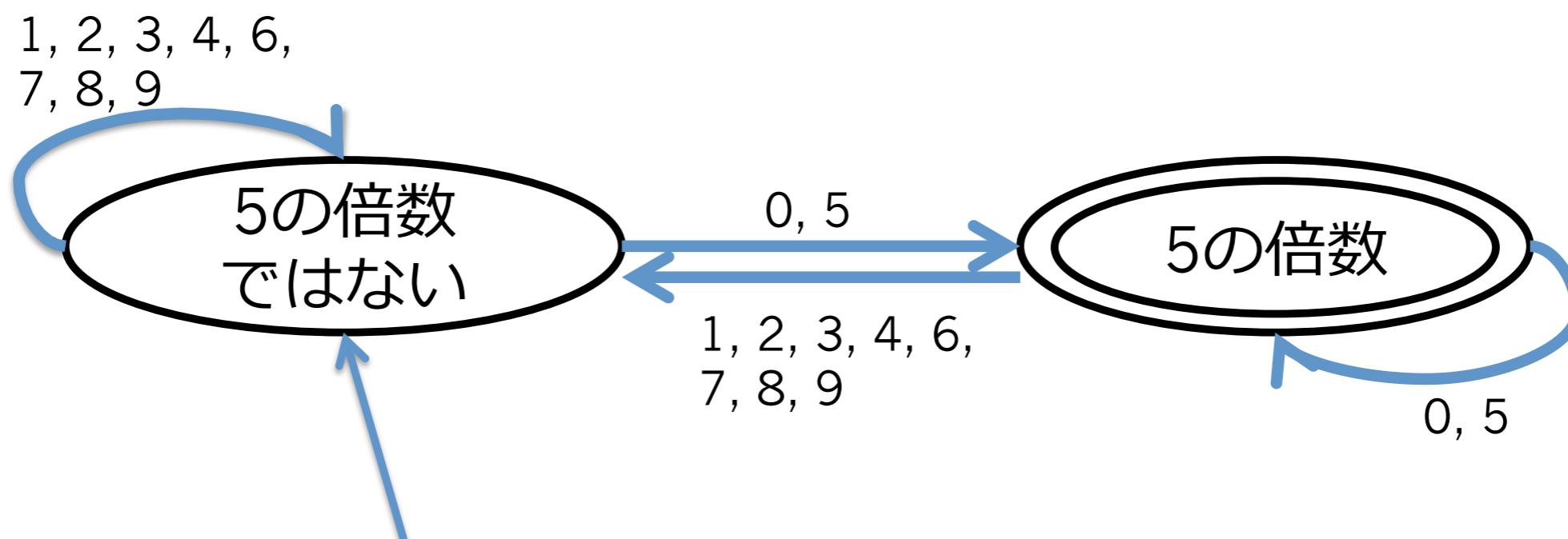
2. 自分のリポジトリを開いて、提出したファイルがプッシュされているか確認する

[https://github.com/nit-ibaraki-prog3i/prog3i-\(ユーザ名\)](https://github.com/nit-ibaraki-prog3i/prog3i-(ユーザ名))

# 【課題8-2】

「入力した数値が5の倍数ならば受理」するプログラムを作成して下さい。

- ▶ 5の倍数は、末尾が0もしくは5
- ▶ つまり、入力を先頭から読み込んでいき、末尾の数値が0, 5ならば受理
- ▶ 以下の状態遷移図を参考に作れる



# 【課題の提出】

以下の流れで、作ったCプログラムをGitHubにプッシュして、Webサイトで確認してみましょう。

1. 端末内で、以下のコマンドで課題を提出

```
$ git add -A  
$ git commit -m "課題8-2提出"  
$ git push origin master
```

2. 自分のリポジトリを開いて、提出したファイルがプッシュされているか確認する

[https://github.com/nit-ibaraki-prog3i/prog3i-\(ユーザ名\)](https://github.com/nit-ibaraki-prog3i/prog3i-(ユーザ名))

# 【課題8-3】

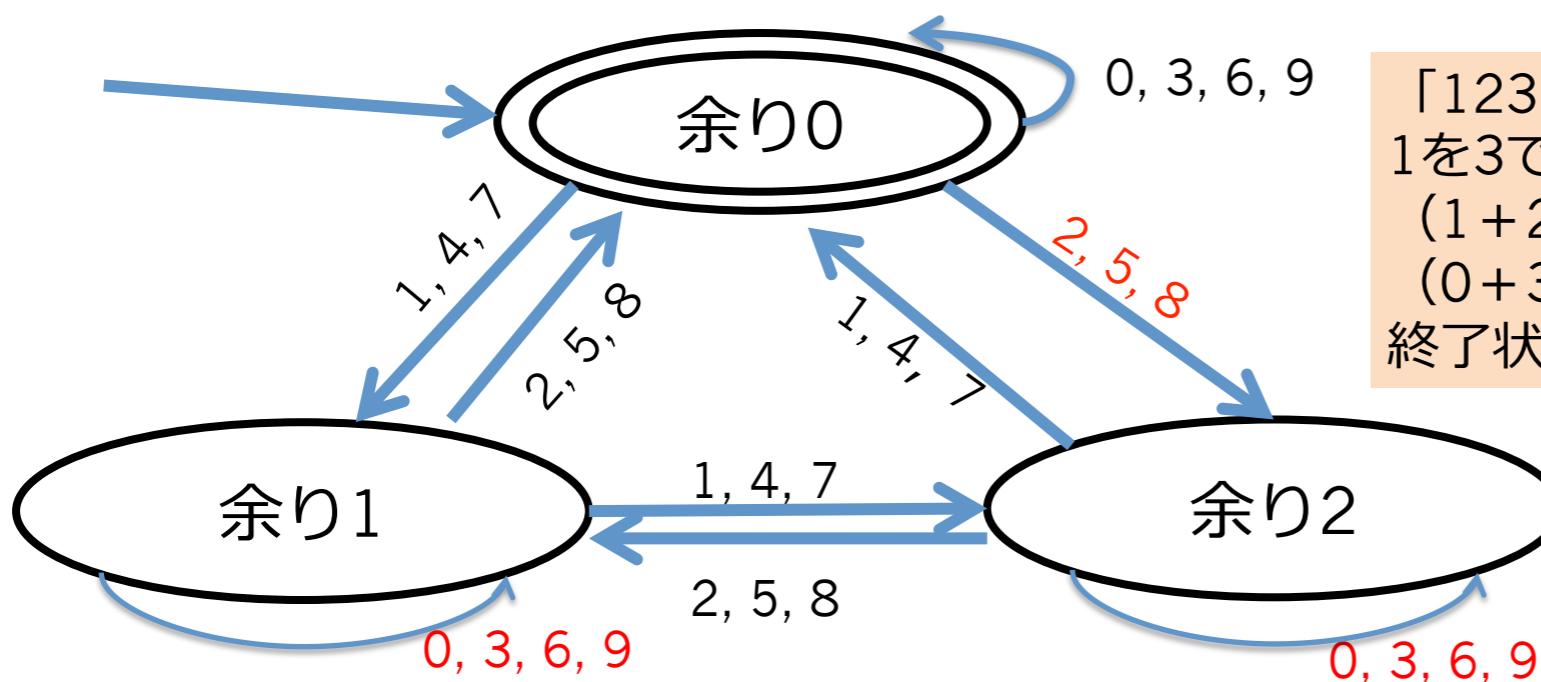
「入力した数値が**3の倍数ならば受理**」するプログラムを作成して下さい。

- ▶ 3の倍数は、各位の数字の和が3の倍数

【例】 123:  $1 + 2 + 3 = 6$  (3の倍数)

124:  $1 + 2 + 4 = 7$  (3の倍数ではない)

- ▶ 以下の状態遷移図を参考に作れる (状態が**3つ**に増えることに注意)



「123」を入力した場合：  
1を3で割ると余り1なので余り1に遷移  
(1+2)を3で割ると余り0なので余り0に遷移  
(0+3)を3で割ると余り0なので遷移せず。  
終了状態なので123は受理

# 【課題の提出】

以下の流れで、作ったCプログラムをGitHubにプッシュして、Webサイトで確認してみましょう。

1. 端末内で、以下のコマンドで課題を提出

```
$ git add -A  
$ git commit -m "課題8-3提出"  
$ git push origin master
```

2. 自分のリポジトリを開いて、提出したファイルがプッシュされているか確認する

[https://github.com/nit-ibaraki-prog3i/prog3i-\(ユーザ名\)](https://github.com/nit-ibaraki-prog3i/prog3i-(ユーザ名))