Cプログラミング1(再)
第8回

講義では、Cプログラミングの基本を学び
演習では、やや実践的なプログラミングを通して学ぶ

処理の流れ

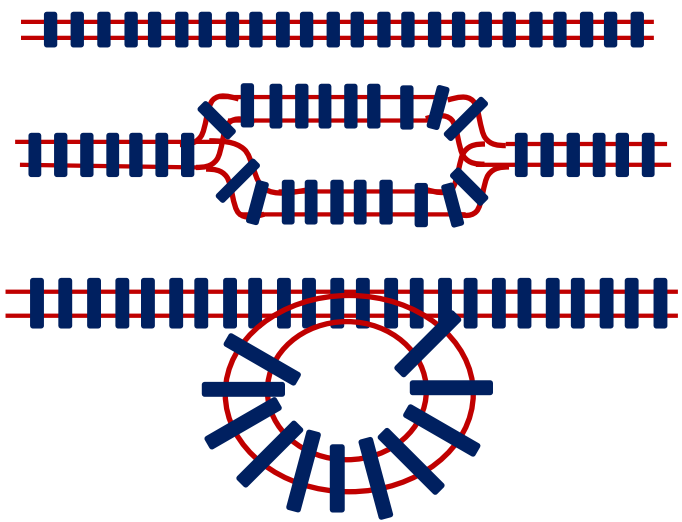
処理の流れとフローチャート

プログラムの処理の流れの典型例

順次(逐次)処理

選択処理(select)

反復処理(loop)

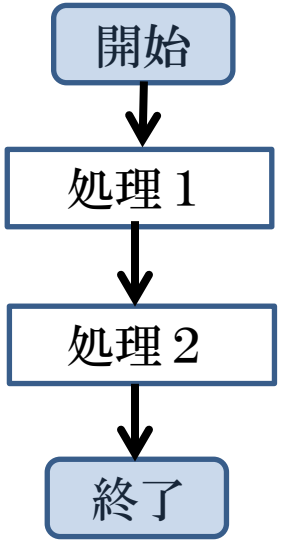


フローチャート（流れ図）

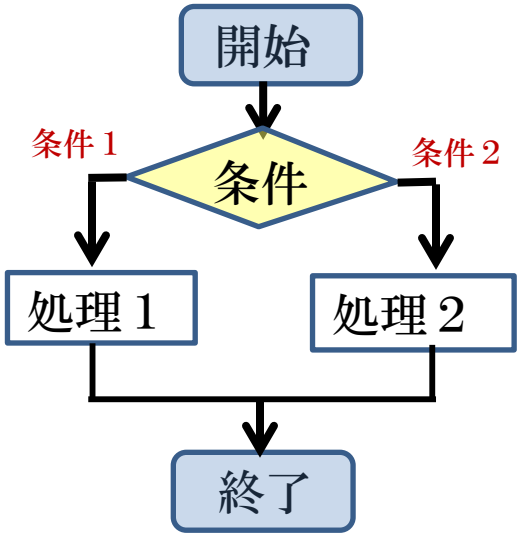
処理の流れの表現の一方法

構造化プログラミング手法　　どのプログラムも基本的にこの組み合わせ

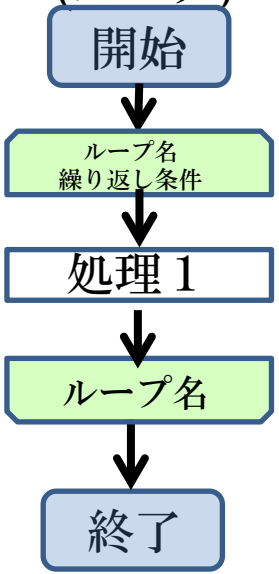
順次処理
(逐次処理)



選択処理
(条件分岐)



反復処理
(ループ)



処理の流れの基本形

順次(逐次)処理(sequential)

一連の処理を順番に行う：処理の基本

選択処理(select)

Cではif文（以外に switch/case文がある)

Cのifの条件は「０以外は真」「０は偽」

反復処理(loop)

２種類の処理

(1) while による繰り返し

繰り返し回数が決まっていない場合の反復処理にむいている

「ある条件が成り立っている間」これをせよ、という書き方

基本パタン： while (条件）{ 処理; ... }

// 処理の中で「条件」の値を変更するようなものが必要

(2) forによる繰り返し

繰り返し回数が決まっている場合の反復処理に向く

基本パタン(N回の繰り返し)

for(i=0; i < N; i++) { 処理; ... }

前回の文字列の関数の解説

文字の長さを求める関数 length

```
int length(char str[]) {     // 引数は文字の配列
   int len=0;                          // 走査用の変数(初期値は0)
   while (str[len] != ‘¥0‘)     // 文字を一つずつ調べる
         len++ ;                       // 次の文字へ
   return len;          // ‘¥0’になった時のlenの値が「長さ」
}
```

文字列を逆順に: reverse

```
void reverse(char str[ ]) {        // 入力は文字の配列
  int len = length(str)-1;         // 「文字の長さ－１」が最後の文字のインデックス
                                   // lenは文字列を末尾から見ていくためのインデックス
  int p;                           // pは文字列を先頭から見ていくためのインデックス
  char c;                          // swapのためのダミー変数
 for (p=0; p < len; p++, len--) {  // スワップするたびに pに１を足し、lenから１引く
            // 「先頭と最後の文字を取り替える」ことを順に行っていく
            c=str[p];              // str[p]とstr[len]とをswapする
           str[p] = str[len];
           str[len] = c;
  }
}
```

回文チェック：pal

```
int pal(char str[ ]) {                    // 入力は文字の配列
            // pは文字列の先頭、lenは文字列最後の文字のインデックス
   int p=0, len = length(str)-1;
                        // 探索のしかたは reverse と同じ
   for(   ; p < len; p++, len--) {   // pを徐々に増やしていく
              // 先頭と最後が一致しなければ回文ではない
        if (str[p] != str[len])
                        return 0;
   }
   return 1;  // 全部のチェックをパスしたなら、回文
}
```

問題1：次の九九の表を作成するプログラム

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38
3	6	9	12	15	18	21	24	27	30	33	36	39	42	45	48	51	54	57
4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	68	72	76
5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95
6	12	18	24	30	36	42	48	54	60	66	72	78	84	90	96	102	108	114
7	14	21	28	35	42	49	56	63	70	77	84	91	98	105	112	119	126	133
8	16	24	32	40	48	56	64	72	80	88	96	104	112	120	128	136	144	152
9	18	27	36	45	54	63	72	81	90	99	108	117	126	135	144	153	162	171
10	20	30	40	50	60	70	80	90	100	110	120	130	140	150	160	170	180	190
11	22	33	44	55	66	77	88	99	110	121	132	143	154	165	176	187	198	209
12	24	36	48	60	72	84	96	108	120	132	144	156	168	180	192	204	216	228
13	26	39	52	65	78	91	104	117	130	143	156	169	182	195	208	221	234	247
14	28	42	56	70	84	98	112	126	140	154	168	182	196	210	224	238	252	266
15	30	45	60	75	90	105	120	135	150	165	180	195	210	225	240	255	270	285
16	32	48	64	80	96	112	128	144	160	176	192	208	224	240	256	272	288	304
17	34	51	68	85	102	119	136	153	170	187	204	221	238	255	272	289	306	323
18	36	54	72	90	108	126	144	162	180	198	216	234	252	270	288	306	324	342
19	38	57	76	95	114	133	152	171	190	209	228	247	266	285	304	323	342	361

ただし、

　　数を入力し、九九の表の一行を表示する関数を作り、それを利用して全体のプログラムをつくること。

　その関数を printOneLineとすると、

printOneLine(2) は以下の表示をするもの:

2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38

問題2. 簡単な暗号プログラム

次のような暗号プログラムをつくろう：

(1)キーボードから、ASCIIコードで32(スペース)以上93(大括弧閉じ)以下の文字からなる文字列(英大文字, 一部の記号)を入力する

例: IINE!

(2) それぞれの文字に適当な数だけ足して別な文字に変換する。例えば「+5」すれば IINE! ⇒ NNSJ&　これが暗号

(3)この暗号を解読するには(1)の後、それぞれの文字に同じ数だけ「マイナス」する。 NNSJ& ⇒ IINE!

(1),(2) を行うプログラムを encrypt.c

(3)を行うプログラムを decrypt.c として作成せよ。

問題3. 暗号解読

次の暗号は、先の暗号の作り方によって作られたものである。これを解読してみよ。（ただし+5したものではない）

(1)NRUHJD#¥RPHUX#QDQWH#VXJRL#QH$

次はどうだろうか？

(2) ljnjxbAufotbjeb