プログラミング論I
(5) if と for の組み合わせ，応用

電子情報工学専攻　日浦 慎作

ログインしておいてください

「かつ」の，２種類の書き方

０＜ａ＜５ かどうかを調べたいとき

```
if(0 < a) {
      if(a < 5) {
            /* 成立 */
      }
}
```

２つの条件を順に確かめていく

```
if(0 < a && a < 5) {
      /* 成立 */
}
```

２つの条件の成立を一気に調べる

- どちらでも良い
  - 左の書き方では，３分岐に発展できる（次項）
    - ａが 0以下，１～4のとき，5以上　の3つへの場合分け
  - 右の書き方は簡潔で，段つけが深くならない

3つ以上への分岐(1)

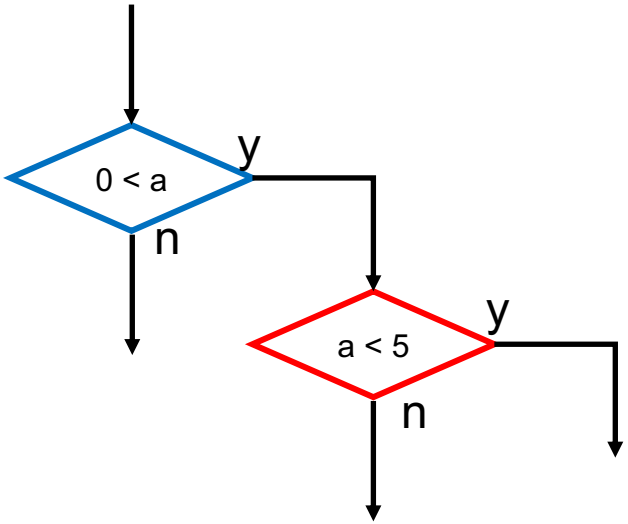
```
if(0 < a) {
    if(a < 5) {
        /* a が1以上4以下 */
    }
    else {
        /* a が5以上 */
    }
}
else {
    /* a が 0 以下 */
}
```

対応する

対応する

省略可能

省略可能



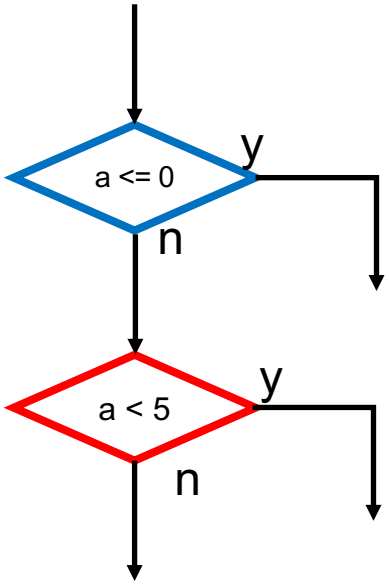
- 2分岐を二重に重ねた表現

3つ以上の分岐(2)

```
if(a <= 0) {
        /* a が 0 以下 */
}
else if(a < 5) {
        /* a が1以上4以下 */
}
else {
        /* a が5以上 */
}
```

対応する

対応する



- 1つずつ，条件が成立したものを外していく表現

(2)の実際の意味

```
if(a <= 0) {
    /* a が 0 以下 */
}
else {
    if(a < 5) {
        /* a が1以上4以下 */
    }
    else {
        /* a が5以上 */
    }
}
```

対応する

対応する

前ページとの相違部分

- 最初の else の中に2つ目の if が入る
- 書き方の問題
  - 前のページは，最初の else の後ろの {} が省略された形

ブロックについて

- { … } のことをブロックと呼ぶ
  { … } は全体として「１行」（単文）として扱われる
  { … } の中では，最初に変数の定義が出来る
  （ここで定義した変数はブロックの外では使えない）
- if (…) { … } の赤字部分もブロック

```
if (a == 0)
  printf("%d¥n", a);
```

  という風にブロックを使わない書き方も許される
  if に続く１行だけが if 文の対象になる
  （だが，個人的には，お勧めしない）
- ブロックは何重にでも出来る（入れ子構造）

書き方の例

```
int i, n;

printf(“自然数を入れてね :”);
scanf(“%d”, &n);

if(n <= 0)
      printf(“自然数じゃないよ¥n”);

for(i = 0; i < n; i++)
      printf(“%d “, i);
```

制御する対象が１行だけのときはブロックを省略できる

if でも for でも可

ブロックを使わないときに起こるミス

```
if(a < 0)
    printf(“a は負の値です. ¥n”);
    printf(“気をつけましょう！¥n”);
```

- 上のプログラムでは，aの値によらずに「気をつけましょう！」が必ず表示される
  - 誤ったインデント（段つけ）が，誤解を生んでいる
  - ブロックを使えば防げたミスである
- ブロックの省略はC言語らしい表現で格好いいが，僕自身はソフト開発の現場で禁じられた経験あり

もう一度見てみましょう

```
if(a <= 0) {
        /* a が 0 以下 */
}
else if(a < 5) {
        /* a が1以上4以下 */
}
else {
        /* a が5以上 */
}
```

```
if(a <= 0) {
        /* a が 0 以下 */
}
else {
        if(a < 5) {
                /* a が1以上4以下 */
        }
        else {
                /* a が5以上 */
        }
}
```

1行分とみなされて else の配下にある

同じプログラム { } は省略できる

- else の後ろのブロックが省略された形

（復習）C の文法

- 文 ・・・ ;（セミコロン）で終わる．処理実行の単位

  `seisu = 5;` ・・・ 変数 seisu に 5 を代入

  `printf(“result = %d¥n”, hensu);`

  ・・・printf により画面表示する

- 識別子 ・・・ 変数，関数などの名前

  上の例では `seisu, printf, hensu` が該当

  自分で命名できる（すでに用意されているものもある）

- 文字列 ・・・ “ と “ でくくられた文字

  コンパイラは文字列を解釈しない．定数 5 などと同等

- 関数 ・・・ 識別子(…) の形のもの

（復習）文法とプログラムの構造

```
#include <stdio.h>
int main(void) {
   int seisu;

   seisu = 5;
   printf("seisuの値は%dです¥n", seisu);
   return 0;
}
```

行頭が#はプリプロセッサ（特別扱い）

文字列
識別子
予約語
凡例

予約語一覧

auto	const	double	float	int	short	struct	unsigned
break	continue	else	for	long	signed	switch	void
case	default	enum	goto	register	sizeof	typedef	volatile
char	do	extern	if	return	static	union	while

識別子は，アルファベットか数字で作る．ただし↑の予約語は使えない
（ただし先頭はアルファベットのみ）

C言語の文法の注意点(1)

- カッコ ( ) [ ] { } の対応はきちんと取る
  - 数式計算の a * (b + c) のカッコ(優先順位)
  - 関数呼び出しの printf(“Hello!¥n”);
  - 制御構造の if(a < 0) { } の ( ) や {}
- “ “ や ‘ ‘ のように，同じ形で2個セットのものも
  - 文字列 “Hello!¥n” がきちんと閉じているか

プログラミング言語の基本です！
いい加減では動きません！
エディタが賢いので，対応関係を見てくれます！

変数名の付け方

- 変数名のルール
  - アルファベットか，アンダースコア _ から始まる
  - ２文字目からは，上記の他に数字が使える
  - アルファベットの大文字と小文字は区別される
  - 予約語や，既存の識別子と同じものは使えない
    - 変数名として for や if は使えない（予約語）
    - 変数名として printf は使えない（既存の関数名）
- 名付け方

  単語区切りのスペースの代わりに_か大文字を使う

  例：oddCount　とか　check_result とか

要するに・・

- 制御構造 for や if の支配する範囲を意識する
  - そのために段つけやブロックを用いる
- 段つけをちゃんとしていないプログラムは減点するかも！
- 美しいプログラムを書きましょう！

参考 https://qiita.com/syougun360/items/5e8e55e22a9ec5fc34a9

while ループ

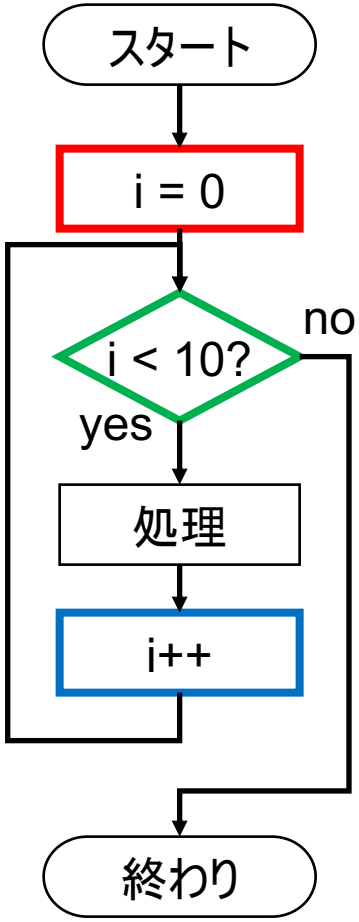
for のうち，条件判断部分しかないもの

```
for (i = 0; i < 10; i++) {
    /* 実行部分 */
}
```

同じ

うまく使い分けるべし.

```
i = 0;
while (i < 10) {
    /* 実行部分 */
    i++;
}
```



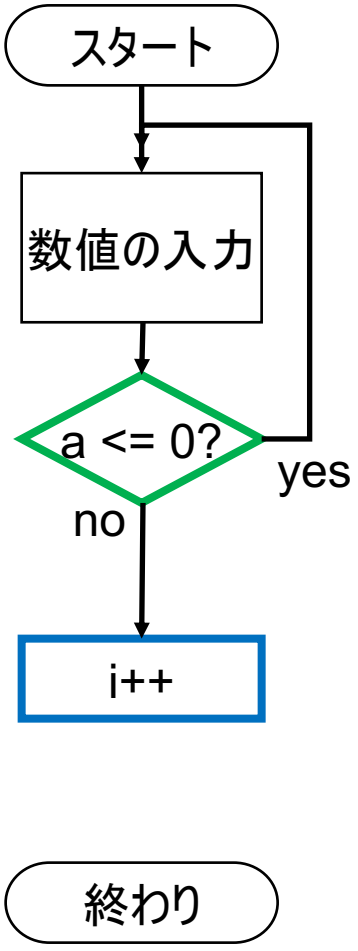
do – while ループ

- 条件判定を最後に行いたいときに使う
  - 処理の結果，やり直す場合など

```
do {
    printf(“正の数を入力せよ:”);
    scanf(“%d”, &a);
} while(a <= 0);
```

使用例：条件を満たさない入力があれば，
再度入力させる

うまく使える場面は非常に少ないと言われているが，
たまに「きれいに書ける」場合がある



繰り返しと分岐

- 繰り返しや分岐は自由に「入れ子」に出来る
  - 繰り返しの中に if が入ることは多い

120 の約数を全て求める

```
int i, num = 120;

for (i = 2; i < num; i++) {
     if (num % i == 0) {
          printf("%d は %d で割り切れる. ¥n", num, i);
     }
}
```

結果

```
120 は 2 で割り切れる.
120 は 3 で割り切れる.
120 は 4 で割り切れる.
120 は 5 で割り切れる.
120 は 6 で割り切れる.
120 は 8 で割り切れる.
120 は 10 で割り切れる.
120 は 12 で割り切れる.
120 は 15 で割り切れる.
120 は 20 で割り切れる.
120 は 24 で割り切れる.
120 は 30 で割り切れる.
120 は 40 で割り切れる.
120 は 60 で割り切れる.
```

繰り返しを中断するには？

- 繰り返し処理の途中で不都合が起こった
- 繰り返しの残りを処理する必要がなくなった
  など → break を使う

91 を割り切る最小の数を調べる

```
int i, num = 91;

for (i = 2; i <= num; i++) {
        if (num % i == 0) {
                printf("%d は %d で割り切れる. ¥n", num, i);
                break;
        }
}
```

break; を実行すると，ループから強制的に脱出できる

break の使い方

- ループ（繰り返し）を即座に抜ける
  - 使えるのは for のほか，while, do-while の中もOK
- ループを入れ子にしていたときは，
  最も内側のループだけを抜ける

```
for (i = 0; i < 10; i++) {
      for (j = 0; j < 10; j++) {
            if (何かの条件) {
                  break;
            }
      }
      /* break ではここへ抜けてくる */
}
```

無限ループの作り方

- ```
  while(1) {
      /* 処理 */
  }
  ```

どちらでもよい

- ```
  for ( ; ; ) {
      /* 処理 */
  }
  ```

- ループを抜ける手段（break）がないと使えない
  - プログラムは永遠に走り続ける

次の処理へ進める

- ループ内の続きの処理を省略し，次の処理へ進める場合 continue を使う

120 を素因数分解する

```
int i, num = 120;

for (i = 2; i <= num; i++) {
        if (num % i != 0) { /* 割り切れない場合は次の値へ */
                continue;
        }
        printf("%d / %d = %d¥n", num, i, num / i);
        num /= i;
        i = 1;
}
```

残りの処理が飛ばされる

結果

```
120 / 2 = 60
60 / 2 = 30
30 / 2 = 15
15 / 3 = 5
5 / 5 = 1
```

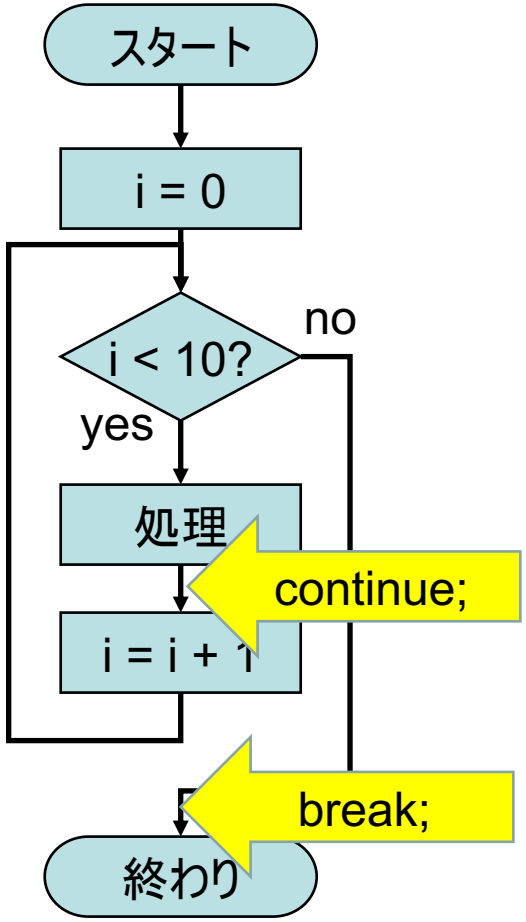
break と continue まとめ

```
int i;

for (i = 0; i < 10; i++) {

        /*なにかの処理 */
        /*なにかの処理 */
        /*なにかの処理 */

        /* continue ではここへ飛んでくる */
}
/* break ではここへ飛んでくる */
```



二重ループの抜け方

いろいろあるけど‥gotoを使うのが簡単

```
for (i = 0; i < 10; i++) {
      for (j = 0; j < 10; j++) {
            if (何かの条件) {
                  goto OUT;
      }
}
OUT:
/* 処理の続き */
```

- goto は「使っちゃダメ！」みたいに言われるが‥
  - プログラムがごちゃごちゃになりやすい（スパゲッティプログラム）
  - が，上のような場合なら，まあ，使っても良い

gotoを使うと

- if と gotoで，繰り返し処理もできますが・・

```
#include <stdio.h>

int main() {
  int i;

  i = 0;
 LOOPTOP:
  printf("Hello world!¥n");
  i++;
  if(i < 10)
    goto LOOPTOP;
}
```

練習：素因数分解のプログラム

```
#include <stdio.h>

int main(void) {
  int i, num = 120;

  for (i = 2; i <= num; i++) {
    if (num % i != 0) {
      continue;
    }
    printf("%d / %d = %d¥n", num, i, num / i);
    num /= i;
    i = 1;
  }
  return 0;
}
```