制御文(if〜else)

山本昌志*

2006 年 7 月 12 日

概 要

条件に従い実行される文が異なる制御の構造を学習する．まずは，条件式と論理演算を学習する．そして，if〜else の構文を学習する．

1 本日の授業内容

1.1 前回の復習

先週は教科書 [1] の 3 章の p.85–99 の変数の型などについて学習した．先週の講義で理解すべき内容は，以下の通り．

- 数学と同じように C 言語でも，積や商は和や差の演算よりも優先される．C 言語でも数学と同様，括弧によって演算の順序を変えることができる．ただし，C 言語の式の演算で使えるのは小括弧 ( ) のみで，中括弧 { } や大括弧 [] は使えない．
- 数学のイコール (=) は左辺と右辺が等しい—ということを表している．それに対して，C 言語のイコールは，(1) 右辺の式を計算して，(2) 左辺の変数に代入する—というコンピューターの動作を表す．このように変数に値を代入するものを代入演算子と言う．とくに，イコール (=) は単純代入演算子と呼ばれる．
- C 言語では，イコールの他に様々な代入演算子が用意されている．表 1 に示すようなものがあり，複合代入演算子と呼ばれる．

*独立行政法人秋田工業高等専門学校電気工学科

表 1: 代表的な複合代入演算子．表中の a の値は，演算の結果である．演算の前の a の値は 7，hoge は 3 とする．いずれも整数型の変数とする．

複合代入演算子	動作	例	単純代入演算子の記述	a の値
+=	加算して代入	a+=hoge	a=a+hoge	10
-=	減算して代入	a-=hoge	a=a-hoge	4
*=	積算して代入	a*=hoge	a=a*hoge	21
/=	除算して代入	a/=hoge	a=a/hoge	2
%=	剰余算して代入	a%=hoge	a=a%hoge	1

- 代入演算子の右辺の式の計算結果を右辺値，左辺の変数の値は左辺値と言う．
- 整数型 (int) と倍精度実数型 (double) の両方が含まれる二項演算では，計算精度の高い方に自動的に型が変換される．
- また，代入演算子の左辺と右辺の型が異なる場合，右辺値は左辺の型に自動的に変換される．
- 変数に格納されている型を強制的に変換したい場合，キャストによる明示的な型変換—強制型変換—を使う．括弧を付けて型名を指定すると，その右の変数や式の値が，指定した型に変換できる．例えば，整数 (hoge=10) を整数 fuga=3 で割って，倍精度実数の値にしたい場合は，

```
a=(double)hoge/fuga;
b=hoge/(double)fuga;
c=(double)hoge/double(fuga);
```

  のようにする．倍精度実数型の変数 a,b,c には，いずれも 3.33333$\cdots$ が代入される．

1.2 本日の学習内容

本日から前期末試験まで，教科書の 4 章「制御の流れ」の学習を行う．この範囲に書かれている内容は全て理解しなくてはならない．C 言語以外の他の言語を会得するときも，同じような内容があるし，文法も似ている．したがって，この章を理解すると他の言語は簡単に習得できる．もし，この章が理解できないと，他のどんなプログラミング言語も理解できないであろう．それほど大事な内容である．

これら学習する内容は難しくなるので，予習と復習を欠かさないようにしなくてはならない．とくに 4 章はプログラミング言語を習得する上でとても重要な内容を含む．使っている教科書は，丁寧に説明している．教科書を良く読み，理解することに努めよ．

今回の授業では，4 章の p.102–117 の学習を行う．そこで，諸君が身に付けるべきことを以下に示す．

- if 文を使ったプログラムが書ける．
  - 関係演算子 (==,!=，>=，<=，>，<) が理解できる．
  - 論理演算子 (&&，||，!) が理解できる．

  - 中括弧 { } を使った複文が理解できる．

- `if`〜`else` を使ったプログラムが書ける．

2　条件式

教科書の p.102–107 の内容である．教科書では条件式と言っているが，制御式ということもある．

2.1　条件式

演算子と被演算子　先週の課題のプログラム，リスト 1 の動作を考える．このプログラムの動作は，次のとおりである．まずは,「テストの点数?」と言うメッセージが現れるので，ユーザーは点数を入力する．それに応じて，

- 50 点より低ければならば,「いわゆる欠点です．がんばらないと留年します.」と表示する．
- さもなければ—50 点以上—,「合格点です．しかし，油断は禁物です.」と表示する．

となる．

リスト 1: 成績判定のプログラム

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5    int tensu;
6
7    printf("テストの点数?\t");
8    scanf("%d",&tensu);
9
10   if(tensu<50){
11     printf("いわゆる欠点です．がんばらないと留年します．\n");
12   }else{
13     printf("合格点です．しかし，油断は禁物です．\n");
14   }
15
16   return 0;
17 }
```

このように，条件—ここでは点数—に応じて実行される文が変わるものを，制御構造と言う．この条件を表す式—ここでは `tensu<50`—を条件式と言う．

式は，演算子 (operator) と被演算子 (operand) から構成する．例えば，単純な加算の式 `a+b` を考える．この場合，演算子は`+`，被演算子は `a` と `b` である．制御式も式なので，演算子と被演算子とからなる．リスト 1 の場合，演算子は`<`，被演算子は `tensu` と `50` である．

条件式で良く使われる演算子をリスト 2 にまとめておく．ほとんど，数学の記号と同じなので一瞬にして憶えられるだろう．

表 2: 条件式に使われる演算子

演算子種類	C 言語	数学
関係演算子 (大小判定)	`<`	$<$
	`<=`	$\leqq$
	`>`	$>$
	`>=`	$\geqq$
等価演算子	`==`	$=$
	`!=`	$\neq$

式の値　C 言語の式は，計算の結果としての値を持つ．例えば，式 `3+5` は 8 という値を持つ．制御式も式なので値を持つ．条件式の演算の結果の値は，`1` または `0` である．制御式の内容が「真」(true:真実の) であれば `1`,「偽」(false:誤った) であれば `0` となる．まとめると，条件式の演算の結果は，

式の内容が正しい場合　⇒　計算結果は 1(真) となる．
式の内容が誤りの場合　⇒　計算結果は 0(偽) となる．

となる．これは 2 年生で学習するブール代数の同じ 規則である．

実際のプログラム例を見たほうが分かりやすいであろう．C 言語で次のように書いたとする．

```
a = 4 > 8;
b = 4 < 8;
```

演算のは次のようになる．

- 最初の行は，式 `(4 > 8)` の大小関係が誤りなので，この計算結果は `0` となる．右辺の計算結果の `0` が左辺の変数 `a` に代入される．したがって，`a` の値は `0` となる．
- 次の行は，式 `(4 < 8)` の大小関係が正しいので，この計算結果は `1` となる．右辺の計算結果の `1` が左辺の変数 `b` に代入される．したがって，`b` の値は `1` となる．

ポイント

条件式に使われる関係演算子や等価演算子を用いた式の計算結果は，正しい場合 1 で，誤りの場合 0 となる．

3　論理演算子

教科書の p.107–112 の内容である．

3.1 論理演算子が必要な理由

被演算子が 3 つ以上を組み合わせて，条件式を作る場合を説明する．例えば，次のような場合である．

- $a < b < c$ の場合，演算結果が 1(真) となる条件式を作りたい．

これを表すために，条件式として

```
a < b < c
```

と書いたらどうなるであろうか?．一見，良さそうだが C 言語では許されない．

なぜダメなのであろうか?．演算はひとつずつ実行される．まず，a<b の演算が行われ，その結果は 0 または 1 である．そして，c が前の演算結果である 0 または 1 と比較される．これは，最初意図したことと異なる．c は 0 や 1 と比較したいのではない．

これでは，困ってしまう．通常のプログラミング言語にはこれを解決する手段が用意されている．そのためには，元の命題を

- $a < b$ かつ $b < c$ の場合，演算結果が 1(真) となる条件式を作りたい．

と書き換える.「かつ」とは，両方とも同時にという意味である．そうして,「かつ」に相当する演算子を使う．C 言語では，

```
a < b && b < c
```

と書く．これで，希望通りの条件式が出来る．

C 言語では，このような条件式 (論理式) のための演算子は，表 3 に示すものが用意されている．この 3 つの演算子 (&&,||,!) を論理演算子と言う．ここでも，論理演算子を使った演算結果は，0 または 1 の値となる．

表 3: 論理演算子

演算子	説明	演算機能
&&	論理積 (かつ)	両方とも 1(真) のときのみ 1(真)
||	論理和 (または)	少なくともどちらか一方が 1(真) の時，1(真)
!	論理否定 (反転)	1(真) のとき 0(真)，0(偽) のとき 1(真)

3.2 論理演算子の動作

3.2.1 論理積

論理積演算子 (&&) を挟んだ被演算子が両方とも 1(真) の場合のみ，その結果は 1(真) となる (表 4)．このことから，この演算子は，日本語で「かつ」，英語では「AND」と呼ばれる．この演算子の演算規則を見ても明らかなように，0 と 1 の積の演算と同じである．このことから論理積と呼ばれるのである．

使い方は，先に示したように，

```
a < b && b < c
```

のように，演算子の左右に式を書く．もちろん式でなくて，値を書いても良い．

0 と 1 以外の値を書いた場合どうなるか?．C 言語の規則では，0 以外の場合は，全て真として取り扱われる．即ち

```
a = 1 && 5
b = -5 && 20.5
```

の場合，a=1，b=1 となる．5 や-5，20.5 も真として取り扱われる．これは，以降の論理和でも論理否定でも同じである．

表 4: 論理積

a	b	a && b
0	0	0
0	1	0
1	0	0
1	1	1

3.2.2 論理和

論理和演算子 (||) を挟んだ被演算子の少なくとも一方が 1 (真) の場合，演算の結果は 1(真) となる (表 5)．このことから，この演算子は，日本語で「または」，英語では「OR」と呼ばれる．これは，に示す，この演算子の演算規則を見ても明らかなように，0 と 1 の和の演算と同じである[1]．このことから論理和と言われる．

使い方は，論理積と同様に，

```
a < b || b < c
```

と，演算子の左右に式を書く．もちろん式でなくて，値を書いても良い．

これも，0 と 1 以外の値を書くことができる．論理積同様，0 以外の場合は，全て真として取り扱われる．

表 5: 論理和

a	b	a || b
0	0	0
0	1	1
1	0	1
1	1	1

[1] 1 と 1 の論理和が 1 となるところが通常の数学とは異なる．ブール代数では 1+1=1 となるのである．2 年生で学習する．

3.2.3 論理否定

論理否定の演算は，演算子 (!) の後ろに書く被演算子の値が 0(偽) の場合，1(真) となる．一方，被演算子の値が 1(真) の場合，0(偽) となる (表 6).

使い方は，

```
!(a < b)
```

のように，演算子の後に被演算子を書く．

これも，0 と 1 以外の値を書くことができる．論理積同様，0 以外の場合は，全て真として取り扱われる．

表 6: 論理否定

a	!a
0	1
1	0

3.3 演算子の優先順位

いろいろな演算子が出てきたので，その計算の優先順位が問題となる．数学では，和や差よりも積や商が優先されるように，C 言語でも優先順位がある．今まで，出てきた演算子の優先順位を表 7 にまとめておく．常識通りの順序になっているので，あまり迷わないと思う．もし，迷ったら，括弧 () がもっとも優先順位が高いことを思い出し，それを使えばよい．これは，数学と全く同じである．

また，優先順位が同じ場合，左側から評価するのも数学と同じである．

表 7: 演算子の優先順位 (上のほうが優先順位が高い)

種類	演算子
括弧	()
論理否定	!
乗除	* /
加減	+ -
比較	< <= > >=
等価	== !=
論理積	&&
論理和	||

4 if else 文

プログラム中で,「もし○○ならば，△△する」というような処理をしたい場合，if という命令を使う．また,「もし，○○ならば△△する，さもなければ□□する」という場合は，if と else を使う．ここでは，

この if や else の使い方を示す.

4.1 処理が 1 つの場合

最初は一番単純な,「もし○○ならば，△△する」という構文を示す (教科書 p.112–113). とくに，△△の部分が 1 つの文で表せる場合である．このような場合は，次のように，書く.

書式

```
if(条件式) 文;
```

条件を表す○○の部分が条件式で，△△の部分を文で表すのである．これは,「条件式が正しい (真) ならば，文を実行する」となる．もし，条件式が誤り (偽) であれば，この文は実行されず次の行に移る．図 1 にこの構文のフローチャートを示す.

以下のようなプログラムが，この構文の使用例である.

```
if(a<=10) printf("a は，10 以下です\n");
```

- a が 10 以下ならば，
  - 「a は，10 以下です」と表示する.

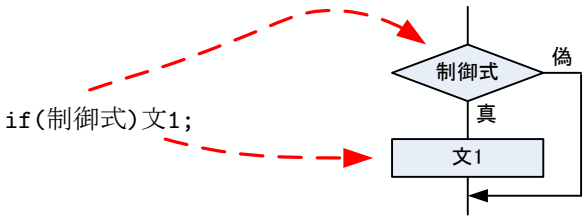

図 1: 条件式が真の場合，1 つの処理を実施する if 文

4.2 ブロックで処理する場合

先ほどの構文では，実行できる文は 1 個に限られる.「もし○○ならば，△△し，□□し，···」のように複数の文を実行したい場合がある．このようなときは，次に示すように，括弧 { } でくくり，ブロック化して，複数の文を書く．これを複文 (教科書 p.113–115) う．このブロック内には，さらに if〜else などの制御文を書くこともできる.

書式

```
if(条件式){
    文 1;
    文 2;
    文 3;
}
```

これは,「条件式が正しい (真) ならば，文 1 と文 2，文 3 を実行する」となる．もし，条件式が誤り (偽) であれば，これら文は実行されず，ブロックの外側に出る．図 2 にこの構文のフローチャートを示す．

以下のプログラムが，この構文の使用例である．

```
if(0<=a && a<=10){
  printf("a は，0 以上\n");
  printf("かつ\n");
  printf("a は，10 以下です\n");
}
```

- もし，a が 0 以上，かつ，10 以下ならば，
  - 「a は，0 以上」と表示する．
  - 「かつ」と表示する．
  - 「a は，10 以下です」と表示する．

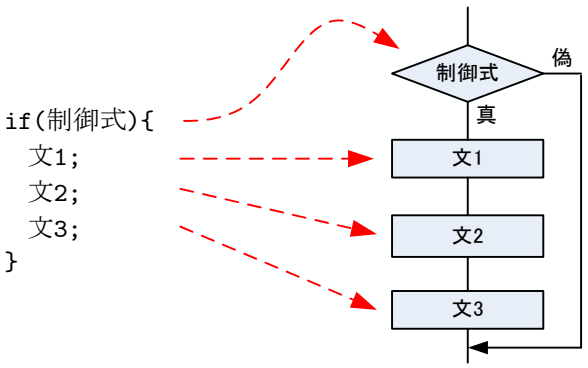

図 2: 条件式が真の場合，ブロックで処理を実施する if 文

4.3 2 分岐の場合

「もし○○ならば△△し，さもなければ□□する」というように，条件により二者択一の選択処理が必要な場合がある (教科書 p.116–117)．これは，次のように書く．

書式

```
if(条件式){
   文 1;
   文 2;
   文 3;
}else{
   文 4;
   文 5;
   文 6;
}
```

これは,「条件式が正しい (1:真) ならば，文 1 と文 2，文 3 を実行する．さもなければ，文 4 と文 5，文 6 を実行する.」となる．実行される文が複数であるので，ブロックになっていることに注意．文が 1 つの場合,, { と } でくくり，ブロック化しなくても良い．図 3 にこの構文のフローチャートを示す.

以下のようなプログラムが，この構文の使用例である．

```
if(0<=a && a<=10){
  printf("a は，0 以上\n");
  printf("かつ\n");
  printf("a は，10 以下です\n");
}else{
  printf("a は，0 未満\n");
  printf("または\n");
  printf("a は，10 より大きい\n");
}
```

- もし，a が 0 以上，かつ，10 以下ならば，
  - 「a は，0 以上」と表示する．
  - 「かつ」と表示する．
  - 「a は，10 以下です」と表示する．
- さもなければ
  - 「a は，0 未満」と表示する．
  - 「または」と表示する．
  - 「a は，10 より大きい」と表示する．

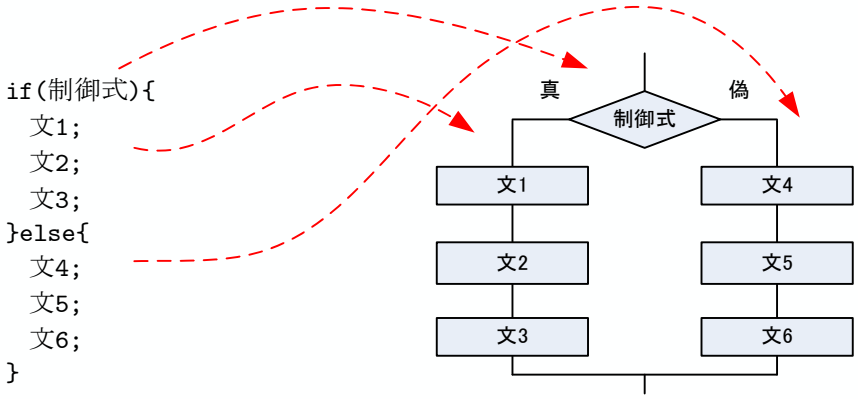

図 3: else を使って，二者択一の処理をする構文

5　プログラム作成の練習

本日の学習内容の理解を深めるために，以下の練習問題のプログラムを作成して，実行してみよ．

[**練習 1**] キーボードから二次方程式

$$ax^2 + bx + c = 0$$

の係数 a, b, c を読み込む．その方程式の解によって，

- 判別式が正の場合:「異なる 2 つの実数解があります」
- 判別式がゼロの場合:「重根 (実数) です」
- 判別式が負の場合:「異なる 2 つの虚数解があります」

と表示するプログラムを作成せよ．

[**練習 2**] 整数型の変数 a,b,c に以下の値を格納する．

```
a=1        b=3        c=5
```

以下の演算の結果を予想せよ．そして，プログラムを作成して演算結果を確かめよ．ひとつのプログラムで，全てを計算すること．

```
a==1       a==2       a!=b       a!=b+c-7
a<=1       2<b        c<a+b      6+3<b+c
```

[**練習 3**] 整数型の変数 a,b,c に以下の値を格納する．

```
a=1        b=3        c=5        d=7
```

以下の演算の結果を予想せよ．そして，プログラムを作成して演算結果を確かめよ．ひとつのプログラムで，全てを計算すること．

```
a==1 && b<=3          a>5 || 3<d            !(5==c)
0<a && a<=10          a<0 || 10<a           !(b<=a)
```

[**練習 4**] キーボードから二次方程式

$$ax^2 + bx + c = 0$$

の係数 a, b, c を読み込む．その方程式の解によって，以下のように振る舞うプログラムを作成せよ．

- 判別式がゼロ以上の場合:「解は，`3.533686 +- 5.639685` です」と「判別式の値は，30.806046」，解と判別式の値を表示する．
- 判別式が負の場合:「`1.236987 +- 8.245963i`」と「判別式の値は，-67.995905」，解と判別式の値を表示する．

6 課題

次回の講義の日 (7 月 19 日) の AM8:45 までに，以下の課題をレポートとして提出すること．表紙等は，いつもの通り．表紙のタイトルは「制御文 (if〜else)」とすること．

課題に書かれている (復) と (予) は，その問題が復習と予習のどちらに属するか—を表している．

[問 1] (復) 教科書の p.102–117 を繰り返し 3 回，読め．そして，理解できない内容があれば，クラスの分かってそうな者に聞け．さもなければ，オフィスアワーを利用して私 (山本) に質問せよ．

[問 2] (復) 整数型の変数の値が，

```
a=-4      b=-2      c=0       d=0       e=0
```

の場合，次の演算の結果はどうなるか?．プログラムを作成しないで，手計算すること．

```
a > b                                          a == -4 && c < d
(a < -5 || c <= 0) && d >= 0                   !(a+b < -5) || !(c+d) > 5
a*a >= b*c                                     d < 3 && 0 < 3 && -1 > 5
(a < 0)+(b < 0)+(c < 0)+(d < 0)+(e < 0)        a < 5 && !(c+d) || a+e < b+c
```

[問 3] (復) 次の条件の if 文を書け．

1. -30<a の場合,「end」と画面に書き出す．
   [解等例]
   ```
   if(-30 < a){
      printf("end\n");
   }
   ```
2. a の値が，10 以上 100 未満の場合,「end」と画面に書き出す．
3. a の値が，-100 以下または 100 以上の場合,「end」と画面に書き出す．
4. a と b の値の両方が負の場合,「end」と画面に書き出す．
5. a も b もゼロでない場合,「end」と画面に書き出す．
6. a と b の合計が c と d の合計よりも小さいとき,「end」と画面に書き出す．
7. a,b,c の全てが負，あるいはその合計が-10 以下のとき,「end」と画面に書き出す．
8. a と b の和が 0 以上かつ c 以下のとき,「end」と画面に書き出す．
9. a,b,c のうち少なくとも 1 つが 0 以上で，a<b<c のとき,「end」と画面に書き出す．
10. a,b,c のどれかひとつが負の場合,「end」と画面に書き出す．

[問 4] (予) 教科書の p.117–131 を 5 回，繰り返し読め．そして，理解できない内容を簡単に箇条書きせよ．ただし，cofee break(p.130) の内容は分からなくても良い．

参考文献

[1] 内田智史監修, (株) システム計画研究所編. C 言語によるプログラミング　基礎編　第 2 版. (株) オーム社, 2006.