第 A 章　教科書 1–5 章の復習

A.1　「演算と型」の復習

教 p.19

/ 演算子

int 型の式 / int 型の式

という演算では、整数としての割算（小数点以下は切捨て）としての結果が得られる。

> **参考:** オペランドのどちらかまたは両方が負の数の場合、どちら向きに切り捨てるか（例えば -7/3 が -2 になるか -3 になるか）は処理系依存である。（C 言語の仕様では定まっていない。）　教 p.20

Q A.1.1 次の命令文の出力を書け。

1. `printf("%d", 7/3);`
   答: _
2. `printf("%d", 3/4);`
   答: _

教 p.28

型と演算

double 型の式 / double 型の式

の演算では、切捨ては行われない。
一方、int 型の式と double 型の式が混じっている場合、

int 型の式 / double 型の式

や

double 型の式 / int 型の式

の場合も、int 型のオペランドが double 型に暗黙に型変換され、double 型の演算となる。

キャスト (cast)　とは、明示的に型変換することである。　教 p.30

(型) 式

という形で、「式」の値を「型」としての値に変換する。例えば、

```
int na, nb
...
(double)(na + nb) / 2
```

では、double 型としての割算が行われる。

教 p.177　（演算子の優先順位に注意する。括弧がなければ割算よりもキャストが優先する。）

Q A.1.2 次のプログラム（の断片）の出力を書け。

1.
```
double x = 7/5;
printf("%.1f", x);
```
答: ___

2.
```
double x = (double)7/5;
printf("%.1f", x);
```
答: ___

3.
```
double x = 7/5.0;
printf("%.1f", x);
```
答: ___

4.
```
double x = (double)(7/5);
printf("%.1f", x);
```
答: ___

A.2　「プログラムの流れの分岐」の復習

まず、大前提として、プログラムの文は上から順（同じ行内では左から順）に実行される。

教 p.36

if 文

if (式) 文$_1$

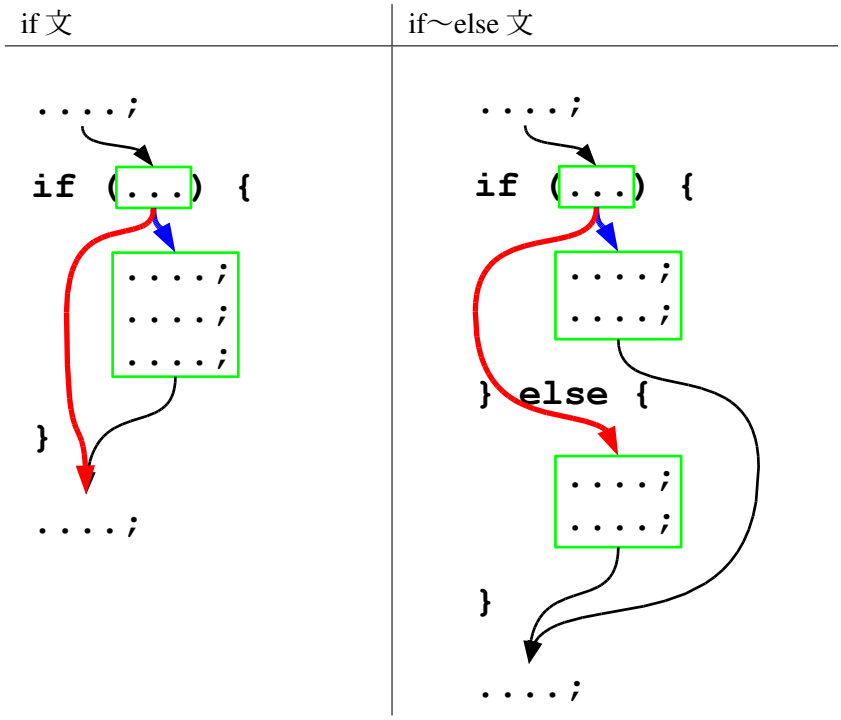
という形のこと、式を評価して、その値が真（すなわち非 0）であれば、文$_1$ を実行する。式の値が偽（すなわち 0）であるときは、何もしない

教 p.38

if〜else 文

`if ( 式 ) 文`$_1$ `else 文`$_2$

という形のこと、式を評価して、その値が真（すなわち非 0）であれば、文$_1$ を実行する。式の値が偽（すなわち 0）であるときは、文$_2$ を実行する。



教 p.45

入れ子になった if 文

```
if (no == 0) {
    puts("その数は␣0です。");
} else if (no > 0) {
    puts("その数は正です。");
} else {
    puts("その数は負です。");
}
```

これは、単に else の次の文が、また if 文になっているだけのことである。

教 p.50

複合文（ブロック）　文の並びを波括弧（ブレース — { と } —）で囲んだものを複合文またはブロックという。（文のまえにいくつかの宣言があってもよい。）複合文は構文上単一の文と見なされる。複合文中の文は上から順（同じ行内では左から順）に一つずつ実行される。

通常、if 文の制御する文（後述の while 文、for 文などでも同様）は、たとえ一つの文でも（間違いを避けるため）波括弧で囲んでブロックにする。

△ 望ましくないコード

```
if (n1 > n2)
    max = n1;
else
    max = n2;
```

◎ 望ましいコード

```
if (n1 > n2)  {
    max = n1;
} else {
    max = n2;
}
```

Q A.2.1 次のプログラム（の断片）の出力を書け。

1.
```
/* プログラム断片 ⓐ */
if (n > 3)  {
    printf("A");
}
if (n > 2) {
    printf("B");
}
if (n > 1) {
    printf("C");
}
```

答: n==1 のとき _____,　n==2 のとき _____,
n==3 のとき _____,　n==4 のとき _____

2.
```
/* プログラム断片 ⓑ */
if (n > 3)  {
    printf("A");
} else if (n > 2) {
    printf("B");
}
if (n > 1) {
    printf("C");
}
```

答: n==1 のとき _____,　n==2 のとき _____,
n==3 のとき _____,　n==4 のとき _____

3.

```
/* プログラム断片 © */
if (n > 3)  {
    printf("A");
}
if (n > 2) {
    printf("B");
} else if (n > 1) {
    printf("C");
}
```

答: n==1 のとき _____,　n==2 のとき _____,
n==3 のとき _____,　n==4 のとき _____

4.

```
/* プログラム断片 ⓓ */
if (n > 3)  {
    printf("A");
} else if (n > 2) {
    printf("B");
} else if (n > 1) {
    printf("C");
}
```

答: n==1 のとき _____,　n==2 のとき _____,
n==3 のとき _____,　n==4 のとき _____

A.3 「プログラムの流れの繰返し」の復習

教 p.68

while 文

while (式) 文

式が真（非 0）であるあいだ、文（ループ本体）を繰り返し実行する。
注: ループ本体が一度も実行されないことがある。

教 p.74

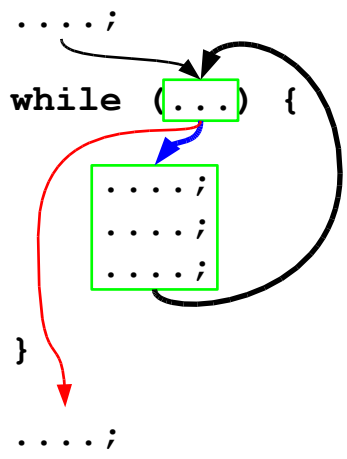
for 文

for (式$_1$; 式$_2$; 式$_3$) 文

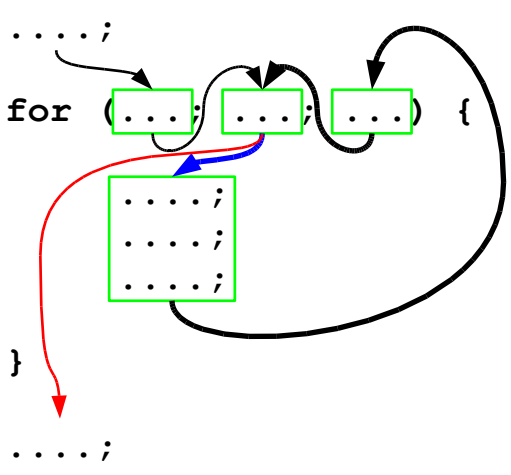
ループに入る前にまず式$_1$を実行する。
式$_2$が真（非 0）であるあいだ、文, 式$_3$を繰返し実行する。

> **詳細:** 式$_1$〜式$_3$は省略可能である。式$_2$を省略したときは、1（つまり真）と書くのと同じ意味になる。

while 文



for 文



Q A.3.1 次のプログラム（の断片）の出力を書け。

1.
```
int i;
for (i=0; i<2; i++) {
    printf("%d", i);
}
```

答: ____

2.
```
int i;
for (i=0; i<=2; i++) {
    printf("%d", i);
}
```

答: ____

3.
```
int i;
for (i=2; i>0; i--) {
    printf("%d", i);
}
```

答: ____

A.4 「配列」の復習

教 p.88

配列 同一の型のデータを集めて、番号（添字）でアクセスできるようにしたもの。C 言語の配列の添字は 0 から始まる。つまり、要素数が N の配列は最後の要素の添字は $N-1$ である。

```
int va[5];                    /* 初期化しないとき */
int vb[5] = { 15, 20, 30 }; /* 配列の初期化は、式をコンマで
  区切って { } で囲む。（残りの要素は 0で初期化される。） */
int vb[] = { 15, 20, 30 };  /* 初期化するときは要素数を省略できる。 */
```

Q A.4.1 次のプログラムの断片が 35 と出力するとき ？ の部分にはいる整数は何か？

```
int vb[] = { 15, 25, 35 };
printf("%d", vb[？]);
```

教 p.90

配列と for 文　配列は for 文と相性が良い。*N* 個の要素を持つ配列の各要素に対して同じ操作を行なうときには次のような for 文を使う。

```
#define N  ...

  int a[N] = {  ...  };

  for (____(空欄 A.4.1); ____(空欄 A.4.2); ____(空欄 A.4.3)) {
     a[i] =  ... ;
  }
```

Q A.4.2 配列の要素を最後から順に操作したい。つまり、以下のプログラムで 7532 と出力したい。空欄を埋めよ。

```
#define N 4

int main(void) {
    int a[N] = { 2, 3, 5, 7 };
    int i;
    for (__________________)  {
       printf("%d", a[i]);
    }
    return 0;
}
```

A.5　インデンテーションの復習

インデンテーション（字下げ・段付け）とは、プログラムを人間にとって理解しやすく整形することで、一定の規約に従って行なう。

プログラミング II では以下のような規約を採用する。

1. 原則として、一行には一つしか文は書かない。ただし、次の例のように密接に関連している文の場合はこの原則にこだわる必要はない。

- プロンプト（入力をうながすメッセージ）を出力する printf 文と scanf 文
- 関連する変数への代入文

Q A.5.1 良いインデンテーションか悪いかを判定せよ。悪い場合は、どこを修正すれば良いか指摘せよ。

(a)
```
int main(void) {
    printf("Hello␣World!"); return 0;
}
```

(b)
```
int main(void) {
    printf("Hello␣World!");
    return 0;
}
```

2. ブレースのなかは、外よりも 4（または 8）字分を字下げする。（ただし、首尾一貫していれば、4 や 8 という数字にこだわる必要はない。）

- ただし、case〜:や default:などのラベルは別

Q A.5.2 良いインデンテーションか悪いかを判定せよ。悪い場合は、どこを修正すれば良いか指摘せよ。

(a)
```
int main(void) {
    printf("Hello␣");
    printf("World!");

    return 0;
}
```

(b)
```
int main(void) {
    printf("Hello␣");
  printf("World!");

        return 0;
}
```

3. タブ文字を使わずに空白文字で字下げする。

インデンテーションにタブを使うのが良いか空白を使うのが良いかは議論があるが、いずれにしても首尾一貫していなければいけない。(タブでインデントするなら、空白文字を使わずにタブ文字だけを使う。)

4. 開きブレースは `if` や `switch`, `do`, `while`, `for` などのキーワードと**同じ行に改行せずに**書く。

Q A.5.3 良いインデンテーションか悪いかを判定せよ。悪い場合は、どこを修正すれば良いか指摘せよ。

(a)
```
for (i=0; i<n; i++)
    {
    printf("Hello␣");
}
```

(b)
```
for (i=0; i<n; i++) {
    printf("Hello␣");
}
```

5. 閉じブレースは `if` や `switch`, `do`, `while`, `for` などのキーワードのはじめの文字と列をそろえて、独立した行に書く。

   - ただし、`else` や `do〜while` の `while` は直前の閉じブレースと同じ行に続けるほうがよい。

Q A.5.4 良いインデンテーションか悪いかを判定せよ。悪い場合は、どこを修正すれば良いか指摘せよ。

(a)
```
for (i=0; i<n; i++) {
    printf("Hello␣");
}
printf("World!");
```

(b)
```
for (i=0; i<n; i++) {
    printf("Hello␣"); }

printf("World!");
```

(c)

```
 for (i=0; i<n; i++) {
     printf("Hello␣");
     }
 printf("World!");
```

(d) ___

```
 for (i=0; i<n; i++) {
     printf("Hello␣");
} printf("World!");
```

6. if や for などでは、選択されたり、繰り返したりされる文が**一つだけの場合**も、ブレース（{〜}）に囲む。教科書のプログラム例は必ずしもそうなっていないので、特に注意する。

   - ただし、else のあとにすぐ if が続く else if というかたちは除く。

   Q A.5.5 良いインデンテーションか悪いかを判定せよ。悪い場合は、どこを修正すれば良いか指摘せよ。

   (a) ___

```
 for (i=0; i<n; i++)
     printf("Hello␣");
     printf("World!");
```

   (b) ___

```
 for (i=0; i<n; i++) {
     printf("Hello␣");
 }
 printf("World!");
```

7. 関数本体の開始を表す開きブレース（{）は、関数頭部のあとに続けて改行せずに書くか、改行して 1 列目に書く。

   Q A.5.6 良いインデンテーションか悪いかを判定せよ。悪い場合は、どこを修正すれば良いか指摘せよ。

   (a) ___

```
 void foo(int x)
     {
         printf("Hello␣%d\n", x);
     }
```