2015年11月10日

プログラミング入門
ステートメント

横浜国立大学

倉光君郎

プログラム

プログラムは、複数のステートメントからなる。

ステートメント

ステートメント(statement)**とは、宣言や制御などを行うために言語仕様にあらかじめ組み込まれている命令を記述する文のことである。**

＜プログラム＞ ::=

＜文＞

＜文＞

＜文＞

...

ステートメント(復習)

今まで、実は次のようなステートメントを習った。

- 式
- 変数宣言
- 関数宣言
- return 文

式は、評価可能なステートメントの一種であった。

void 型

void 型とは、「値のない」ことを示す特別な型である。ステートメント（値のない式）は、void 型で型付けされている。

void 型の関数

void 型を使えば、値を返さない関数を定義することができる。

```
void main() {
  printf("hello,world\n");
}
```

これは、有名なhello,world プログラム。

制御構造

今日、新たに習うもの。

- ブロック
- if文
- while文
- do/while文
- for文

ブロック

ブロックは、中括弧{ } で囲まれたステートメントである。複数のステートメントが含まれた複文を構成するときに用いる。

```
{
    <文>;
    <文>;
    <文>;
}
```

プログラムは、原則、ブロック内を上から下に順番に実行される。

コードレイアウト

ステートメントの終端は、(ブロック以外)セミコロン(;)をつける。ブロック内は、インデント(字下げ)する。

```
{
    <文>;
    <文>;
    <文>;
}
```

コーディングルール

コーディングルールとは、コードの読みやすさを保つため、インデントや空白、改行のルールを決めておくこと。

```
{   <文>;   <文>;   <文>;   }
```

ブロック

ブロックは、ステートメントなので文として使える。型付けは、もちろん void **型となる。**

```
{
    <文>;
    {
        <文>;
        <文>;
    }
```

```
}
```

空文

空のブロックは、何もしない空文である。

```
{
}
```

コメント

コメントは、プログラミング言語から空白として無視される。

```
/* コメント */
// コメント（行末まで）

1 /* + 2 */ * 3
1 * 3 // と同じ
```

if文

if文は、もっとも基本的な制御構造である。条件分岐を記述するときに用いる。＜条件式＞がtrueのとき＜文1＞、そうでなければ＜文2＞を実行する。

```
if （<条件式>）<文1>
else <文2>
```

慣習的に＜文1＞を then 節、＜文2＞ else 節と呼ぶ。

if文の例

```
int max(int x, int y) {
  if(x > y) {
    return x;
  }
  else {
    return y;
  }
}
```

if文は、条件演算子とよく似ているが評価値はない。

elseの省略

if文は、必要なければ else 節は省略しても構わない。

```
int max(int x, int y) {
  if(x > y) {
    return x;
  }
  return y;
}
```

単文の場合は

ブロックは省略してもよい。

```
int max(int x, int y) {
  if(x > y) return x;
  return y;
}
```

ただし、多くのコーディングルールでは、ブロックを省略しない方法を推奨している。

もっと略したいときは

ブロックは省略してもよい。

```
int max(int x, int y) {
  return (x > y) ? x : y;
}
```

プログラミングは、同じ動作をするプログラムでも複数通りの書き方がある。読みやすいコードがよい。

else/if

条件が２つ以上に分岐する場合は：

```
int max(int x, int y) {
  if(x > y) {
    return 1;
  }
  else if(x == y) {
    return 0;
  }
```

```
    else {
      return -1;
    }
  }
```

if **文もステートメンドであるため、別の** if **文の** else **節に書くことができる。**

switch/case文

条件分岐の条件が値のときは、switch文で書くこともできる。

```
switch(v) {
  case 0: <文>
  case 1: <文>
  case 2: <文>
  ...
  default: <文>
```

```
    }
```

switch**文は使いこなすのは難しいため、通常は**
if/else if **で条件分岐すればよい。**

ループ

ループとは繰り返しの制御構造である。＜条件式＞が true の間、＜文＞を繰り返す。

```
while（<条件式>）<文>;
```

つまり、＜文＞において条件が変化しなければ、永遠に繰り返される。この状態を「無限ループ」と呼ぶ。

ループの例

```
int sum(int n) {
  int x = 0;
  while (n > 0) {
    x = x + n;
    n = n - 1;
  }
  return x;
}
```

break文とcontinue文

ループを中断する

```
break;
```

ループの先頭までスキップする

```
continue;
```

printf (C 言語風)

printf **は、プログラムの実行中の変数の状態を示す関数である。整数** x **の変数の値を見たいときは、次のように書く**

```
printf("x=%d\n", x);
```

println (Java風)

println() **は、プログラムの実行中の変数の状態を示す。整数** x **の変数の値を見たいときは、次のように書く**

```
println("x=" + x);
```

println (スクリプト風)

print は、プログラムの実行中の変数の状態を示す。整数 x の変数の値を見たいときは、次のように書く

```
println(‘‘‘x=${x}‘‘‘);
```

どれでもよい。

ループの例

```
int sum(int n) {
  int x = 0;
  while (n > 0) {
    x = x + n;
    n = n - 1;
    printf("x=%d, n=%d\n", x, n);
  }
  return x;
}
```

ループの構文

do/while **は、＜文＞を最低1回は実行する** while **文のバリエーション。**

```
do <文> while (<条件式>);
```

for文

for文は、while文を無限ループが起こりにくくしたもの。＜初期値＞で条件の初期化を行い、＜変更＞でループで条件式の変更を記述する。

```
for(<初期化>; <条件式>; <変更>) <文>
```

意味は、以下の文と同じ。

```
<初期化>
while(<条件式>) {
```

<文>
<変更>
}

for文 頻出イディオム

N回繰り返す

```
int i = 0;
for(i = 0; i < N; i++) {
   <繰り返す内容>
}
```

i++ は、インクリメンタル演算子。i = i + 1 の簡易の書き方

演習問題3

フィボナッチ数を求める関数をループを用いて定義せよ。

$$
\begin{aligned}
F_1 &= 1 \\
F_2 &= 1 \\
F_n &= F_{n-1} + F_{n-2}
\end{aligned}
\tag{1}
$$