プログラミング論I
(15) C言語の文法
演習

電子情報工学専攻　日浦 慎作

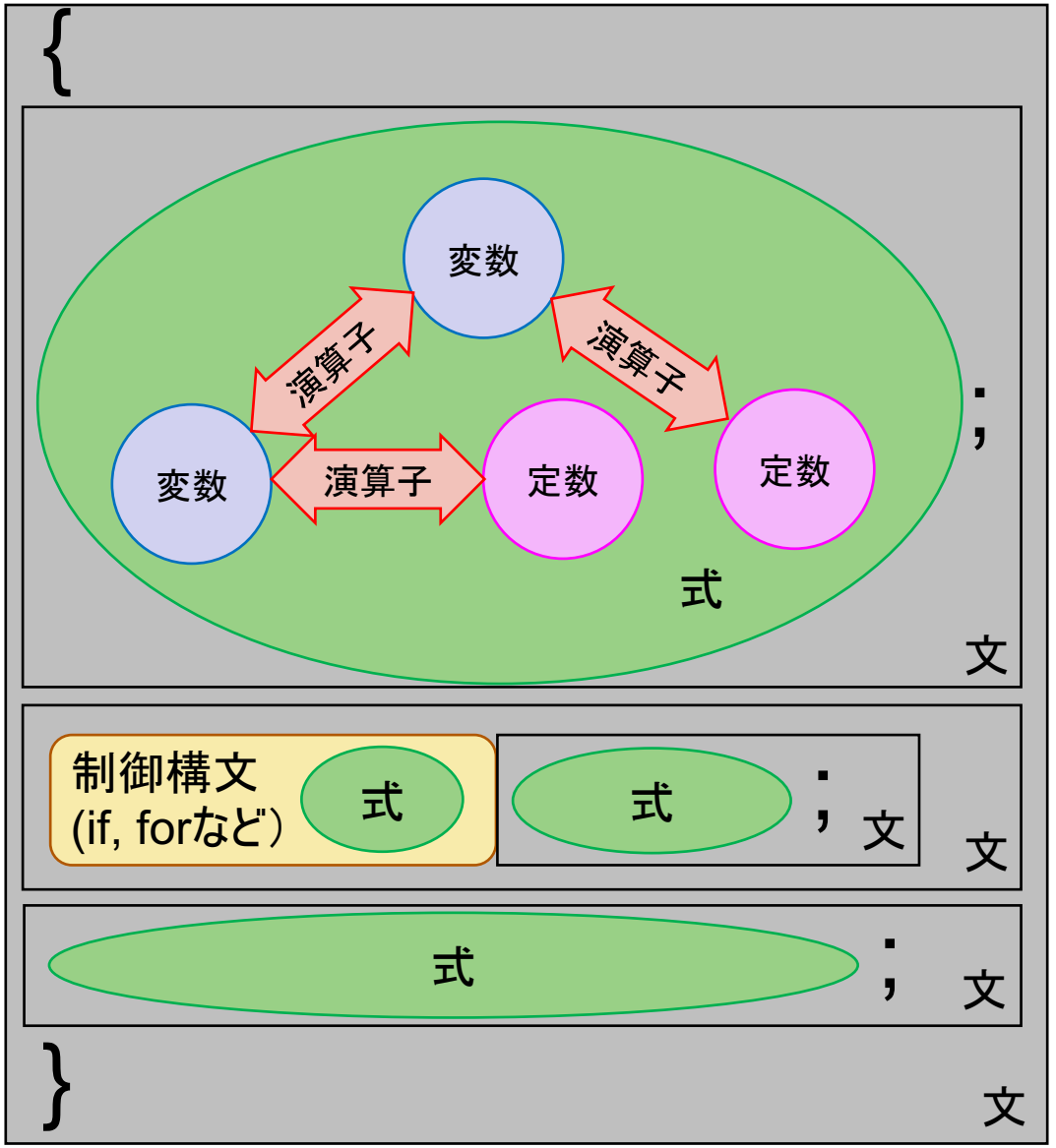
ログインしておいてください

C言語の項，式，文について

- 項
  - 変数 ： a var c01 などの名前（識別子）がつく
  - 定数 ： 2 1.3 'a' 0x5B など（整数，実数，文字等）
- 式
  - 変数や定数を演算子を用いて組み合わせたもの
    `2 * b + sin(1.3) + x[3]`
- 文
  - 式のあとに ; をつけたもの
  - 複数の文を { } （ブロック）で囲んだもの（1つの文となる）
  - 1つの文に制御構文（ifやforなど）をつけたもの

項，式，文の関係

```
{
    a = 2 * b + 4;
    if(a > 5)
        a = -a;
    d = a * b;
}
```

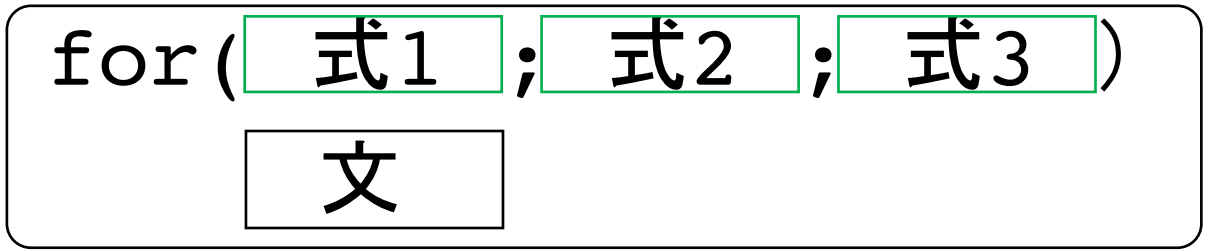


凡例：
変数
定数
演算子
制御構文

- 文がまとまって
  また文になる
- 制御構造が
  ついても文になる

C言語の文法の例

制御構文の例



のように，制御構文は以下のルールに従う

- ( ) 内に式を書くことができる（文は書けない）
  - よって，forやifにつづく( )内では{ }は使用不可
- 直後の1文を対象にする
  - 複数の文を制御の対象にする場合は { }で1つの文にする

正しいプログラミングを行うには

- 文法を意識してプログラミングすることが必要
  - 識別子と予約語など，各要素の種類を区別する
    - 英語でいうと，品詞に相当する
  - 定数や変数の型を意識する
  - 演算子の優先順位を意識する
  - それぞれのまとまりごとの，レベルを意識する
    - これは，文なのか？式なのか？
- コンパイラの仕組みのわかりやすい解説

  http://web.sfc.keio.ac.jp/~hagino/sa15/05.pdf

入れ子構造

- ある構造の中に，さらに似た構造があるもの
  - if文の中にさらにif文

```
if(a > 0)
   if(b > 0)
      return;
```

  - for文による2重ループ
  - ブロックの中にブロック
    など
- プログラミングでは
  いろいろな場面で現れる

マトリョーシカ（ロシアのおもちゃ）
人形のなかに人形，さらにその
中に人形が・・というもの

２次元配列も入れ子構造

- 二次元配列には以下のようにカッコを付けられる

```
int a[3][4] = { 0 };
int i, j;

 a[0] [0] = 1;
(a[0])[1] = 2;

...
```

ex1.c の一部

このプログラムはエラーにならない

なぜか？

- 演算子には優先順位があり，[ ] も演算子である．
  - ( ) は演算子の優先順位を明示しているだけ．
- ２次元配列は，実際には「配列の配列」である．

配列の配列，とは？

`int a[3][4];` にカッコをつけると `int ((a[3])[4]);`

内側（aに近いところ）から読む

```
int a[3][4];
```

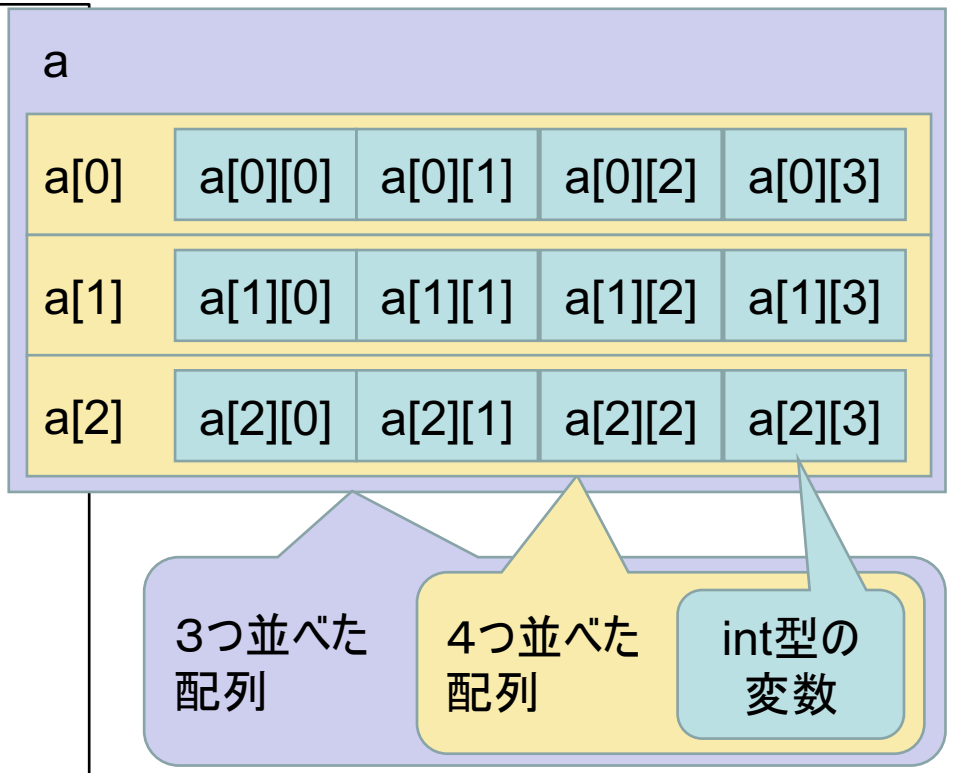
→ aは3要素の配列である．

```
int a[3][4];
```

→ その要素（a[i]）はそれぞれ4要素の配列である．

```
int a[3][4];
```

→ その要素（a[i][j]）の型はint である．



逆に読んでいくと‥

int型の変数を4つ並べた配列を，3つ並べた配列がaである．

配列の構造が入れ子になっている

```c
#include <stdio.h>

int main(void) {
  char dayname[7][10] = {
    "Sunday",
    "Monday",
    "Tuesday",
    "Wednesday",
    "Thursday",
    "Friday",
    "Saturday"
  };
  int day;

  printf("enter a number 0-6:");
  scanf("%d", &day);

  printf("%s¥n", dayname[day]);

  return 0;
}
```

ex2.c

二次元配列は途中まで書いて使うことができる

2次元配列
dayname[7][10]
を,
dayname[day]
までで使用

途中まで使った場合の「型」

```
char dayname[7][10];
```

内側（daynameに近いところ）から読む

```
char dayname[7][10];
```

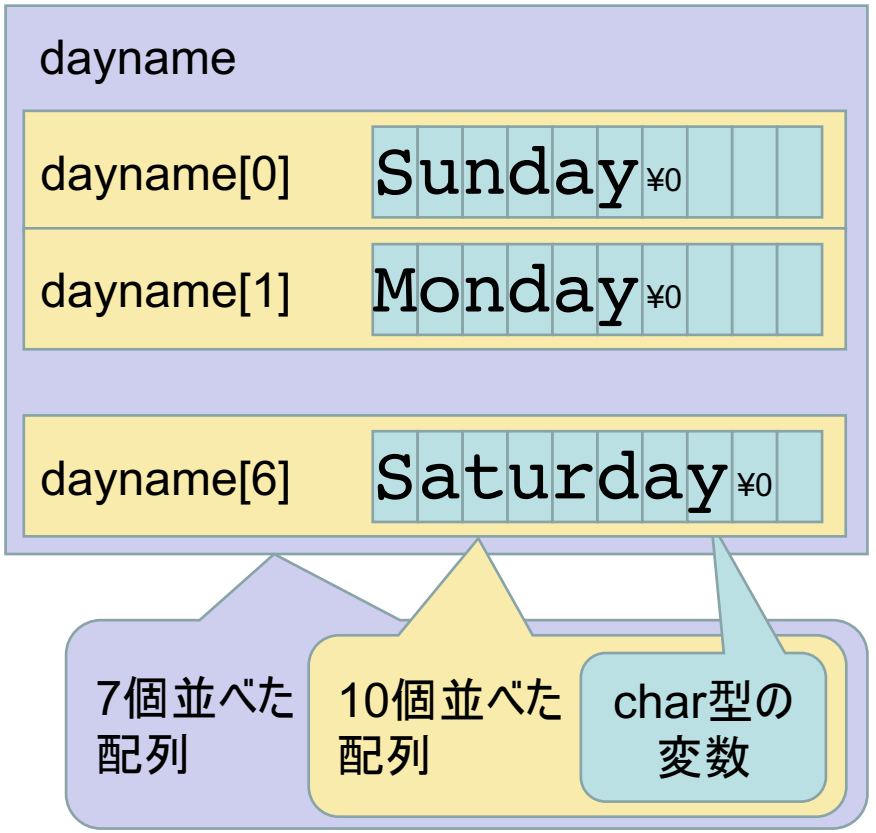
→ daynameは7要素の配列である.

```
char dayname[7][10];
```

→ その要素はそれぞれ10要素の配列である.

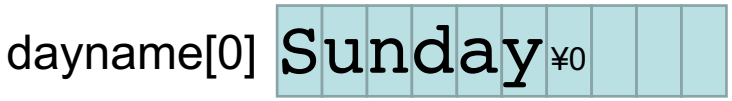
```
char dayname[7][10];
```

→ その要素の型は char である.



ここだけを見ると・・

dayname[i] は，char型の変数10個の配列である



つまり dayname[day]と書いた場合，その式の型は，文字列である.

同じ変数でも，書き方（演算子の付け方）で，その式の型が変わる

先ほどの例

- `dayname[j][i]` ・・・ char型
  なぜなら `char dayname[7][10];`
  は，赤字部分が青字の型である という意味．
- `dayname[j]` ・・・ char型の配列
  なぜなら `char dayname[7][10];`
  は，赤字部分が青字の型である という意味．
  `char str[10];` の `str` と同じ．

練習課題(要提出)

1．2020年1月d日のdを入力すると，
その曜日を出力するプログラムを作成せよ．

prac1.c を配布

- ex2.c を少し改造すればできる．
- なお2020年1月1日は水曜日である．

2．2020年m月d日のmとdを入力すると，
その曜日を出力するプログラムを作成せよ．

prac2.c を配布

- 各月の日数を配列で定義しておき，m−1月までの合計を計算することで，1月1日からの日数を計算すれば良い．

```
int dayNum[12] = {31, 29, 31, 30, ..};
```

- C言語の配列は0から始まるが，
月や日は1から始まることに注意．

応用課題(提出任意)

3．2020年m月d日のmとdを入力すると，以下のような形式で日付と曜日を出力するプログラムを作成せよ.

- 2月4日(m=2, d=4 の場合)
  Tuesday, February 4

さらにできれば，以下のようにしてみよう
(switch-caseを活用すればよいし，配列を使っても良い)

- 2月4日(m=2, d=4 の場合)
  Monday, February 4th
- 5月21日(m=5, d=21 の場合)
  Thursday, May 21st

```
1st
2nd     20th
3rd     21st
4th     22nd
...     23rd
10th    24th
11th    ...
12th    30th
...     31st
```

応用課題(提出任意)

4．2020年の月(1–12)を入力すると，カレンダーを表示するプログラムを作成せよ.

prac4.c を配布

```
Enter a month of 2020.
month:1
Sun Mon Tue Wed Thu Fri Sat
                1   2   3   4
  5   6   7   8   9  10  11
 12  13  14  15  16  17  18
 19  20  21  22  23  24  25
 26  27  28  29  30  31
```

```
Enter a month of 2020.
month:4
Sun Mon Tue Wed Thu Fri Sat
                1   2   3   4
  5   6   7   8   9  10  11
 12  13  14  15  16  17  18
 19  20  21  22  23  24  25
 26  27  28  29  30
```

ヒント： m–1月までの日数の合計を計算するところは，練習問題2と同じ.

回収について

- ソースプログラム（〇 〇 〇 .c）を直接，
  回収ボックスに入れてください
- ファイル名は任意ですが，
  課題1, 2, 3, 4がわかるようにしてください
- 3, 4 は好きな方からやってもらって良いです
- 授業終了５分前に回収します