

プログラミング基礎

<http://bit.ly/prog2d>

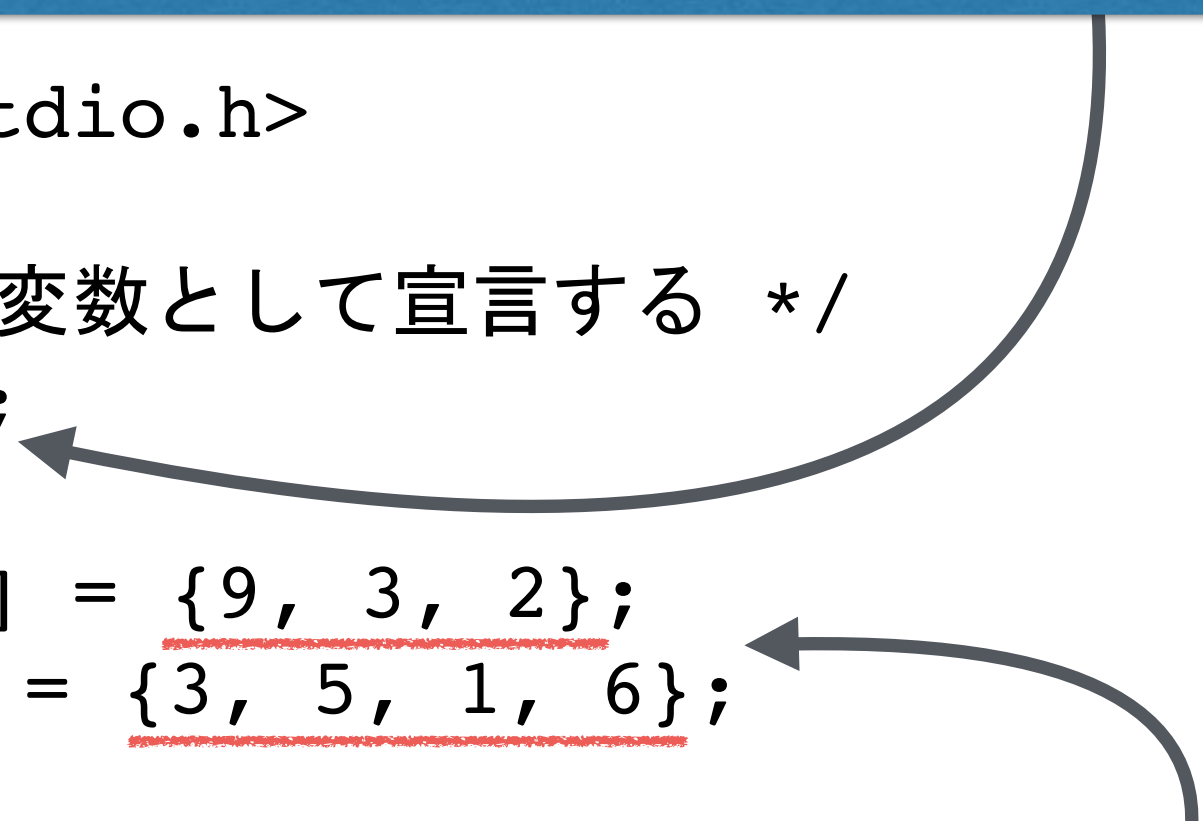
配列

前期 第3週

2017/4/24

【Point 1】 int型の5個の要素を持つ配列を宣言する
(p. 185 図7-3)

```
1: #include <stdio.h>
2:
3: /* グローバル変数として宣言する */
4: int test[5];
5:
6: int test2[3] = {9, 3, 2};
7: int test3[] = {3, 5, 1, 6};
8:
```

A diagram with two curved arrows. One arrow starts from the text '配列' (array) in Point 1 and points to the declaration 'int test[5];' on line 4. The other arrow starts from the text '初期化' (initialization) in Point 2 and points to the initialization '{3, 5, 1, 6};' in the declaration 'int test3[] = {3, 5, 1, 6};' on line 7.

【Point 2】 配列の宣言と同時に初期化する (p. 190)

【Point 3】 関数の外で宣言した変数はグローバル変数となり、プログラム中のどの関数からも参照可能

```
9:  /* 関数のプロトタイプ宣言 */
10: void show();
11:
12: /* 配列testの要素を出力する */
13: void show()
14: {
15:     int i;
16:     for(i=0; i<5; i++) {
17:         printf("test[%d]: %d\n", i, test[i]);
18:     }
19: }
20:
```

【Point 4】 配列testの添字に変数 i を使う (p. 185)



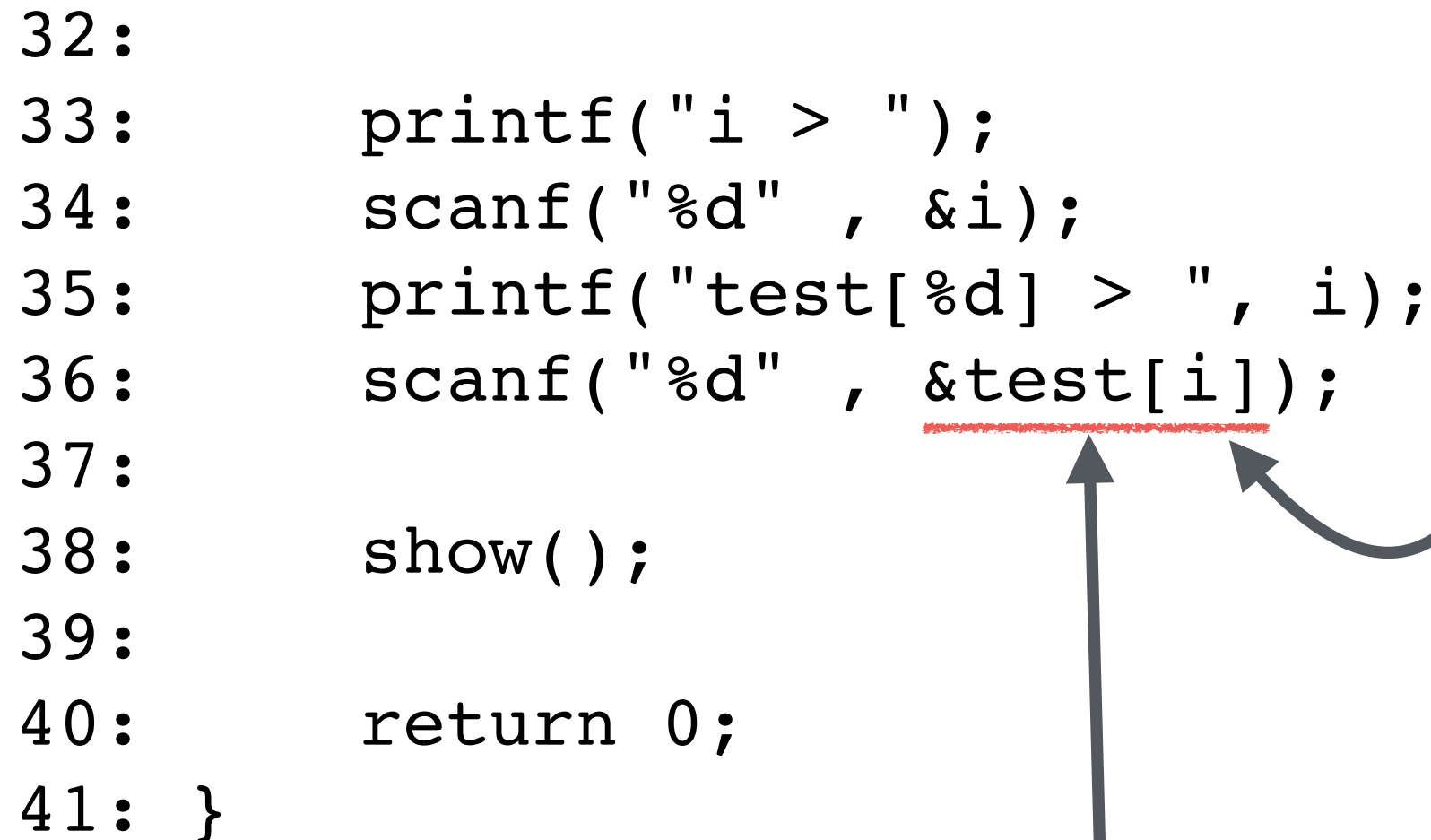
```
21: int main(void)
22: {
23:     int i;
24:
25:     test[0] = 80;
26:     test[1] = 60;
27:     test[2] = 22;
28:     test[3] = 50;
29:     test[4] = 75;
30:
31:     show( );
32:
```

【Point 5】 配列testの0番目～4番目の要素に値を代入する (p. 185 図7-3)



【Point 6】 scanf()による入力の際は、何番目の要素なのかを添字で指定し、&を付ける (p. 187)

```
32:
33:     printf("i > ");
34:     scanf("%d" , &i);
35:     printf("test[%d] > ", i);
36:     scanf("%d" , &test[i]);
37:
38:     show();
39:
40:     return 0;
41: }
```



【Point 7】 配列の添字を指定する場合、配列の大きさを超えないようにする必要がある（ここで、i を0より小さい または 5以上の値を指定すると、存在しない要素にアクセスすることになってしまう） (p. 188)

【練習3-1】

サンプルプログラムをコンパイルして、
実行結果を確認しましょう。

【課題3-1】

関数show()を参考にして、引数で指定した範囲の要素に対して値を出力する関数show_range()を作成してください。

[この関数のプロトタイプ宣言]

```
void show_range(int s, int e);
```

```
/* show() の繰り返し処理がs～e番目になるように作ってみる */
```

```
/* s <= e となることを前提にしてよい */
```

```
/* (つまり s > e の場合を考慮しなくてもよい) */
```

【課題3-1】

[main()での処理]

```
show_range(1, 3);  
show_range(2, 4);
```

[実行例]

```
test[1]: 60      (←1番目から3番目の要素出力されている)  
test[2]: 22  
test[3]: 50  
test[2]: 22      (←2番目から4番目の要素出力されている)  
test[3]: 50  
test[4]: 75
```


【課題3-2】

引数で指定した2つの要素が持つ値を入れ替える関数
`swap_array()`を作成してください。

[この関数のプロトタイプ宣言]

```
void swap_array(int i, int j);  
/* 配列testのi番目の要素とj番目の要素の値を入れ替える */
```

【課題3-2】

[main()での処理]

```
/* 配列testには、サンプルプログラム25～29行目のよう  
   に値が代入されているとする */
```

```
swap_array(2, 3);  
swap_array(1, 4);  
show();
```

[実行例]

```
test[0]: 80  
test[1]: 75  
test[2]: 50  
test[3]: 22  
test[4]: 60
```

【課題3-3】

配列testの要素が持つ値に対して、**何番目の要素が最小値なのか**を調べる関数min_array()を作成してください。

[この関数のプロトタイプ宣言]

```
int min_array();
```

```
/* まず、最初の最小値の場所を0番目とする */
```

```
/* 0～4番目（1～4番目でも可）の要素に対して、最小値との比較を繰り返す */
```

```
/* 比較の際、最小値よりも小さいの値を持つ要素があった場合は、  
   最小値の場所を更新する */
```

```
/* 最小値は何番目の要素なのかを戻り値とする */
```

【課題3-3】

[main()での処理]

```
/* 課題3-2の処理後の配列testに対して実行した場合 */  
printf("min_array: %d\n", min_array());
```

[実行例]

min_array: 3 (「3番目の要素が最小値である」という意味)

【課題3-4】

配列testの指定した範囲内の要素が持つ値に対して、
何番目の要素が最小値なのかを調べる関数
`min_range()`を作成してください。

[この関数のプロトタイプ宣言]

```
int min_range(int s, int e);
```

```
/* 先程の課題で作ったmin_array()を参考に、  
   s番目～e番目の要素に対する処理になるように作ってみる */  
/* s <= e となることを前提にしてよい */  
/* (つまり s > e の場合を考慮しなくてもよい) */
```

【課題3-4】

[main()での処理]

```
/* 課題3-2の処理後の配列testに対して実行した場合 */  
printf("min_range: %d\n", min_range(0, 2));  
printf("min_range: %d\n", min_range(2, 4));
```

[実行例]

```
min_range: 2  
min_range: 3
```

まだ余裕のある人は…【課題3-5】

配列testに対して、要素の値を昇順（小さい順）に並び替える関数sort_array()を作成してください。

関数min_range()とswap_array()を利用します。

【課題3-5】

[この関数のプロトタイプ宣言]

```
void sort_array();  
/* 配列testの0番目～4番目までの要素に対して、  
   次のような処理を繰り返す（繰り返し処理中の要素はi番目とする） */  
/*     min_range()を使って、i番目から4番目の中から、  
        どこに最小値があるのかを探す */  
/*     swap_array()を使って、i番目の値と、最小値を入れ替える */  
/* このような処理を繰り返すと...、  
    • 0番目の要素には、0～4番目の最小値が格納され（つまり一番小さい値）  
    • 1番目の要素には、1～4番目の最小値が格納され（つまり次に小さい値）  
    • 2番目の要素には、2～4番目の最小値が格納され（つまり次の次に小さい値）  
    • 3番目の要素には、...  
    となり、結果として、配列の値が昇順に整列される */
```


【課題3-5】

[main()での処理]

```
/* 配列testには、サンプルプログラム25～29行目のよう  
   に値が代入されているとする */
```

```
sort_array();  
show();
```

[実行例]

```
test[0]: 22  
test[1]: 50  
test[2]: 60  
test[3]: 75  
test[4]: 80
```

小テストについて

小テストの注意点

- 他人の力は借りずに、自分だけでプログラムを作成する。（つまり定期試験と同様）
- 小テスト中は、演習室外へのネットワークアクセスは遮断される。

小テストについて

小テスト中に参照できるもの

- 教科書, 配付資料
- 自分のホームディレクトリ（ホームフォルダ）以下に保存されているファイル
- 小テストでは紙媒体のものは参照可能
- 上記以外の情報を参照することはカンニング行為とする
例：USBで接続された機器に保存されているファイルの参照
ネットワークを介した情報の参照、など