情報システムプログラミングⅡ（4回目）

2024年5月10日（金）

3～4限

授業内容

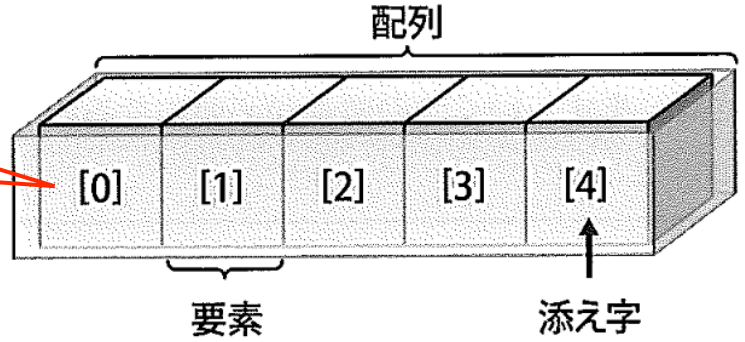
- 講義内容（教科書の216～259ページ＋α）
  - 配列の復習
  - 構造体の配列
  - 多次元配列
  - Pythonにおける配列とリスト
- 演習課題

配列の復習

■配列の概要

- 同じ型の複数のデータを順に格納するデータ構造

要素の添え字は
「0」から始まる



- 配列を定義する構文

```
要素の型 変数名 [ 要素数 ];
```

- 配列の各要素にアクセスする構文

```
配列変数名 [ 添え字 ]
```

配列の復習

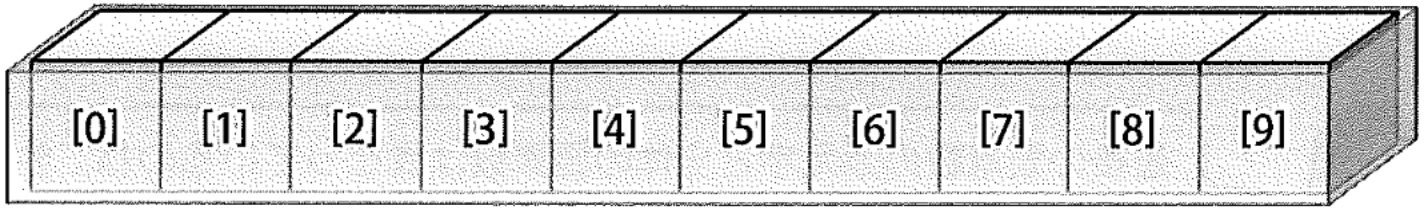
■配列の概要

- 配列の定義と各要素へのアクセスの例

```
int gems[10];
gems[1] = 3;
printf("%d¥n", gems[1]);
```

要素 gems[1] に代入

要素 gems[1] の内容を表示



gems

配列の復習

■配列の初期化

- 配列を初期化する際の構文

要素の型 配列変数名 [要素数] = { 要素 [0] の初期値 ,
要素 [1] の初期値 , …};

※ [] 内の要素数を省略した場合は初期値に指定した数が要素数となる。

- 配列の初期化の例

```
int a[5] = {0, 0, 0, 0, 0};
```

5つ分（この場合は全て）を0に

```
int a[] = {0, 0, 0, 0, 0};
```

配列 a は要素数 5 として定義される

[ ]の要素数は省略できる

```
int a[5] = {20, 30, 10};
```

要素 [3] と [4] は 0 で初期化される

```
int a[5] = {0};
```

先頭要素と以降をすべて 0 で初期化

配列の復習

■配列の利用例

- ループ（反復）処理による全要素の利用例

```
for (int i = 0; i < 配列要素数 ; i++) {
    配列変数名 [i] を使った処理
}
```

while文の例

```
int i = 0;
while (i < 配列要素数) {
    処理
    i++;
}
```

```
for (int i = 0; i < 10; i++) {
    int gemType = rand() % 4;
    gems[i] = gemType;
}
```

ループのたびに i の値が 0 ～ 9 で変化する

配列の復習

■配列の利用例

- ループ（反復）処理による集計例

```
int 集計結果の変数 = 0;
for (int i = 0; i < 配列要素数 ; i++) {
   配列変数名 [i] を調べて集計結果の変数を書き換える処理
}
```

```
int count = 0;
for (int i = 0; i < 10; i++) {
  if (gems[i] == FIRE) {
    count++;
  }
}
```

```
for (int i = 0; i < 5; i++) {
  sum += scores[i];
}
int avg = sum / 5;
```

1 科目ずつ sum に足していく

合計を科目数で割る

配列の復習

■配列の利用例

- 添え字に対応したアクセスの例

全要素の値が先頭から順番に表示される

```
for (int i = 0; i < 10; i++) {
  printf("%c ", GEM_CHARS[gems[i]]);
}
```

全要素の値が末尾から順番に表示される（要素数が10であれば末尾の添え字は9）

```
for (int i = 9; i >= 0; i--) {
    printf("%c ", GEM_CHARS[gems[i]]);
}
```

配列の復習

■配列の注意点

```
int a[] = {10, 20, 30, 40, 50};
printf("%d ", a);
```

NG！　配列変数を渡しても全要素の表示はできない

```
int a[] = {10, 20, 30, 40, 50};
int b[] = {5, 15, 25, 35, 45};
int c[] = a + b;
if (a == b) {
    :
}
```

NG！　配列変数の計算はできない

NG！　配列変数の比較はできない

配列の復習

■配列の注意点

✖

```
int a[5] = {1, 2, 3, 4, 5};
int b[5];
b = a;
```

配列 b に 1、2、3、4、5 を入れているつもり

○

```
int a[5] = {1, 2, 3, 4, 5};
int b[5];
for (int i = 0; i < 5; i++) {
  b[i] = a[i];
}
```

要素を 1 つずつコピー

配列の復習

■配列の注意点

操作	表示 printf() など	計算 +、-、*、/ など	比較 ==、!=、< など	代入（コピー） =
基本型 ※1	○	○	○	○
構造体型	×	×	×	○
配列型	☠	☠	☠	× ☠ ※2

○：可能　×：コンパイルエラー　☠：コンパイルは通るが異常動作

配列の復習

■配列に別名を与える

- 変数や構造体と同様にtypedefにより可能

```
typedef int GemList[10];
GemList gems;
```

int[10] 型に GemList という別名を与える

int gems[10]; と同じ

構造体の配列

■構造体も配列にできる（配列の要素に構造体も利用できる）

```
Monster seiryu = {"青龍", 80, 15};
Monster suzaku = {"朱雀", 100, 30};
Monster byakko = {"白虎", 100, 20};
Monster genbu = {"玄武", 120, 10};

Monster monsters[] = {suzaku, genbu, seiryu, byakko};

const String TEMPLATE = "%s：HP=%3d 攻撃力=%2d¥n";
for (int i = 0; i < 4; i++) {
  printf(TEMPLATE,
      monsters[i].name, monsters[i].hp, monsters[i].attack);
}
```

要素数４のモンスター配列型を作って初期化

ループでモンスターを順番に表示

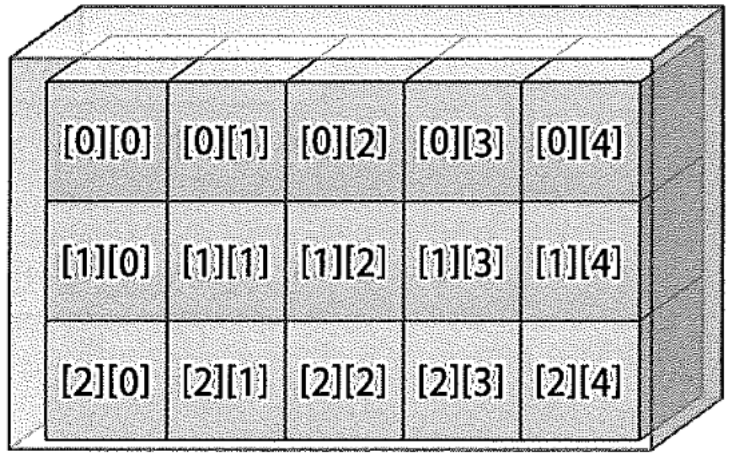
多次元配列

■多次元配列とは

- 2次元以上の配列のこと（2次元以上の配列を利用できる）
- 2次元配列の定義と利用例の構文

```
要素の型 配列変数名 [ 行数 ][ 列数 ];
```

```
配列変数名 [ 行の添え字 ][ 列の添え字 ];
```



要素数が3×5の2次元配列の
概念構造と各要素の添え字

多次元配列

■2次元配列の利用例

```
int scores[2][3];
scores[0][0] = 80;
scores[0][1] = 77;
scores[0][2] = 65;
scores[1][0] = 51;
scores[1][1] = 80;
scores[1][2] = 95;

for (int i = 0; i < 2; i++) {
  printf("%d人目の点数を表示します¥n", i + 1);
  for (int j = 0; j < 3; j++) {
    printf("%d科目め：%d¥n", j + 1, scores[i][j]);
  }
}
```

2人×3科目分の2次元配列を準備

1人目の点数を代入

2人目の点数を代入

多次元配列

■3次元配列の利用例

- 4次元以上も同様に利用できる

3次元配列を定義して各要素の値を入力するプログラム

```
int array[2][2][2];

for (int i = 0; i < 2; i++) {
    for (int j = 0; i < 2; j++) }
        for (int k = 0; k < 2; k++) {
            printf("多次元配列の[%d][%d][%d]の値", i, j, k);
            scanf("%d", array[i][j][k]);
        }
    }
}
```

Pythonにおける配列とリスト

■Pythonの配列とリストとは

- Pythonには配列（array）に加えて，リスト構造のリスト（list）もある

■Pythonの配列（C言語の配列に近いもの）

- 同じ型しか格納できない，1次元までしか利用できない
- 利用にはarrayモジュールのインポートが必要

■Pythonのリスト（基本的に配列ではなくリストを使えば良い）

- 異なる型を格納できる，多次元で利用できる
- 標準で利用できる（インポート不要）

Pythonにおける配列とリスト

■Pythonの配列とリストのプログラム例

配列の利用には
インポートが必要

型の指定
(iはint型)

```
import array

a = array.array('i', [1, 2, 3])
for i in a:
    print(i)

b = array.array('i', [1, 'two', 3])

c = [1, 2, 3]
print(c[0])
print(c[2])

d = [1, 'two', 3]
print(d)
```

○

各要素を1つずつ表示

✖

異なる型を含む
のでエラー

○

配列とリストともに
[]で添え字を
指定して各要素に
アクセスできる

○

異なる型を
含んでも大丈夫

各要素をまとめて表示

Pythonにおける配列とリスト

■Pythonの多次元リストの例

3×2の2次元リスト

```
a = [[1,2,3],[11,22,33],[111,222,333]]
for i in range(3):
    for j in range(2):
        print(a[i][j])

b = [[[1,2],[3,4],[5,6]],[[11,22],[33,44],[55,66]],[[111,222],[333,444],[555,666]]]
print(b)
```

各要素を1つずつ表示

3×3×2の3次元リスト