# プログラミング応用

http://bit.ly/ouyou3d

## アルゴリズム

前期 第9週

2018/6/19

# 今後の授業内容

| | |
|---|---|
| 前期中間 | ☑ ソースコード管理<br>☑ OSとプログラミング |
| 前期期末 | ☐ 計算モデル<br>☐ 言語処理系など |
| 後期中間 | ☐ 再帰処理<br>☐ アルゴリズム |
| 後期期末 | ☐ ソフトウェア開発方法論<br>☐ コンピュータネットワーク |

# 今後の授業内容

| | |
|---|---|
| 前期中間 | ☑ ソースコード管理<br>☑ OSとプログラミング |
| 前期期末 | ☐ 再帰処理<br>☐ アルゴリズム |
| 後期中間 | ☐ 再帰処理<br>☐ アルゴリズム |
| 後期期末 | ☐ ソフトウェア開発方法論<br>☐ コンピュータネットワーク |

こっちに移動

※後期の順番は後程再検討

# 本日は・・・

- アルゴリズム
  - アルゴリズムとは
  - 良いアルゴリズム
- timeコマンド

# 本日は・・・

- アルゴリズム
  - アルゴリズムとは
  - 良いアルゴリズム
- timeコマンド

コンピュータは、
問題を解決するため（目的を達成するため）に、
**手順**に従って**計算をする**


手順 ＝ アルゴリズム

# アルゴリズムの例

## 3つの整数の最大値を求めるプログラム

- 3つの正の整数a, b, cが与えられる
- どの整数が最大値なのかを求めたい
- a, b, cを何回か比較すれば求まりそう・・・

# アルゴリズムの例

**3つの整数を与える**

a 20

b 30

c 10

**問題**

**アルゴリズム**

**何回か比較する**

**最大値を求める**

a 20

b 30

c 10

**解決！**

# 比較の組合せ

**3つの整数○, △, □を比較して、**

**「○が最大値となる」全ての条件を列挙してみると・・・**

**整数と不等号の組合せ**

| | | | | | |
|---|---|---|---|---|---|
| 1 | ○ | > | △ | > | □ |
| 2 | ○ | > | □ | > | △ |
| 3 | ○ | > | △ | = | □ |
| 4 | ○ | > | □ | = | △ |
| 5 | ○ | = | △ | > | □ |
| 6 | ○ | = | □ | > | △ |
| 7 | ○ | = | △ | = | □ |
| 8 | ○ | = | □ | = | △ |

**Cでの条件の表記**

| |
|---|
| ○ > △ && △ > □ |
| ○ > □ && □ > △ |
| ○ > △ && △ == □ |
| ○ > □ && □ == △ |
| ○ == △ && △ > □ |
| ○ == □ && □ > △ |
| ○ == △ && △ == □ |
| ○ == □ && □ == △ |

# 比較の組合せ

**△ = □ と □ = △ は同じなのでまとめられる**

**整数と不等号の組合せ**

| | | | | | |
|---|---|---|---|---|---|
| 1 | ○ | > | △ | > | □ |
| 2 | ○ | > | □ | > | △ |
| 3 | ○ | > | △ | = | □ |
| 4 | ○ | > | □ | = | △ |
| 5 | ○ | = | △ | > | □ |
| 6 | ○ | = | □ | > | △ |
| 7 | ○ | = | △ | = | □ |
| 8 | ○ | = | □ | = | △ |

同じ

同じ

**Cでの条件の表記**

| |
|---|
| ○ > △ && △ > □ |
| ○ > □ && □ > △ |
| ○ > △ && △ == □ |
| ○ == △ && △ > □ |
| ○ == □ && □ > △ |
| ○ == △ && △ == □ |

# 「○が最大であるか？」の条件

**これらの条件のうち<u>どれか1つを満たせばよい</u>ので、**

**「<u>または</u>」で連結する**

| | | | | | |
|---|---|---|---|---|---|
| 1 | ○ | > | △ | > | □ |
| 2 | ○ | > | □ | > | △ |
| 3 | ○ | > | △ | = | □ |
| 4 | ○ | = | △ | > | □ |
| 5 | ○ | = | □ | > | △ |
| 6 | ○ | = | △ | = | □ |

```
if((○ > △ && △ > □)   ||
   (○ > □ && □ > △)   ||
   (○ > △ && △ == □)  ||
   (○ == △ && △ > □)  ||
   (○ == □ && □ > △)  ||
   (○ == △ && △ == □)) {
       result = ○;
}
return result;
```

# 比較する関数の実装

**3つのint型の変数a, b, cの最大値を求める関数を実装する**

**以下の3つの処理をif文で作る**

- **「aが最大値であるか？」を満たしたら･･･**
  **→ aの値を返す**
- **「bが最大値であるか？」を満たしたら･･･**
  **→ bの値を返す**
- **「cが最大値であるか？」を満たしたら･･･**
  **→ cの値を返す**

# 比較する関数の実装

**「aが最大値であるか？」のif文を書くには、**

**○→ a, △→ b, □→ c と置き換える**

| | | | | | |
|---|---|---|---|---|---|
| 1 | a | > | b | > | c |
| 2 | a | > | c | > | b |
| 3 | a | > | b | = | c |
| 4 | a | = | b | > | c |
| 5 | a | = | c | > | b |
| 6 | a | = | b | = | c |

```
int result = -1;
if((a > b && b > c)  ||
   (a > c && c > b)  ||
   (a > b && b == c) ||
   (a == b && b > c) ||
   (a == c && c > b) ||
   (a == b && b == c)) {
       result = a;
}
return result;
```

# 比較する関数の実装

## 同様にbとcについてもif文を作る

「bが最大値であるか？」

○→b, △→a, □→c

| | | | | | |
|---|---|---|---|---|---|
| 1 | b | > | a | > | c |
| 2 | b | > | c | > | a |
| 3 | b | > | a | = | c |
| 4 | b | = | a | > | c |
| 5 | b | = | c | > | a |
| 6 | b | = | a | = | c |

```
if((b > a && a > c)   ||
   (b > c && c > a)   ||
   (b > a && a == c)  ||
   (b == a && a > c)  ||
   (b == c && c > a)  ||
   (b == a && a == c)) {
       result = b;
}
```

「cが最大値であるか？」

○→c, △→a, □→b

| | | | | | |
|---|---|---|---|---|---|
| 1 | c | > | a | > | b |
| 2 | c | > | b | > | a |
| 3 | c | > | a | = | b |
| 4 | c | = | a | > | b |
| 5 | c | = | b | > | a |
| 6 | c | = | a | = | b |

```
if((b > a && a > c)   ||
   (b > c && c > a)   ||
   (b > a && a == c)  ||
   (b == a && a > c)  ||
   (b == c && c > a)  ||
   (b == a && a == c)) {
       result = b;
}
```

# 比較する関数の実装

**これらのif文を並べると、全ての場合を網羅した関数が完成**

```
if((a > b && b > c)  || (a > c && c > b)  ||
   (a > b && b == c) || (a == b && b > c) ||
   (a == c && c > b) || (a == b && b == c)) {
       result = a;
}
if((b > a && a > c)  || (b > c && c > a)  ||
   (b > a && a == c) || (b == a && a > c) ||
   (b == c && c > a) || (b == a && a == c)) {
       result = b;
}
if((c > a && a > b)  || (c > b && b > a)  ||
   (c > a && a == b) || (c == a && a > b) ||
   (c == b && b > a) || (c == a && a == b)) {
       result = c;
}
```

aが最大値の場合

bが最大値の場合

cが最大値の場合

# 本日は・・・

- アルゴリズム
  - アルゴリズムとは
  - 良いアルゴリズム
- timeコマンド

# 比較する関数の完成？

（なんか条件多いなぁ・・・）

**もっと条件を簡単にできないか？**

# アルゴリズム改善案 その1

## 「if～else if～else」文を使って条件を減らしてみる

**ifの条件**

| | | | | | |
|---|---|---|---|---|---|
| 1 | a | > | b | > | c |
| 2 | a | > | c | > | b |
| 3 | a | > | b | = | c |
| 4 | a | = | b | > | c |
| 5 | a | = | c | > | b |
| 6 | a | = | b | = | c |

aが最大値ではなく、
bが最大値の場合
つまり、「a!=b」である

**else ifの条件**

| | | | | | |
|---|---|---|---|---|---|
| 1 | b | > | a | > | c |
| 2 | b | > | c | > | a |
| 3 | b | > | a | = | c |
| 4 | b | = | a | > | c |
| 5 | b | = | c | > | a |
| 6 | b | = | a | = | c |

比較する必要がなくなる

aが最大値ではなく、
bも最大値ではない場合
つまり、cが最大値である

**else**

| | | | | | |
|---|---|---|---|---|---|
| 1 | c | > | a | > | b |
| 2 | c | > | b | > | a |
| 3 | c | > | a | = | b |
| 4 | c | = | a | > | b |
| 5 | c | = | b | > | a |
| 6 | c | = | a | = | b |

# アルゴリズム改善案 その2

「現時点の仮の最大値」を格納する変数を用意する

1. 変数（tmpなど）にaの値を代入する
2. tmpよりもbの方が大きいなら、tmpにbを格納する
3. tmpよりもcの方が大きいなら、tmpにcを格納する
4. tmpを最大値として返す

# どのアルゴリズムが良いか？

## 良いアルゴリズムとは

- 計算量が少ない
  - 比較, 計算, 代入など処理がより少なく済むアルゴリズムは計算量が少ない（つまり速い)
    - 最近の情報分野では扱うデータが大容量化のため、高速に処理できるアルゴリズムが求められる
- メモリ使用量が少ない
  - メインメモリを使い果たすと処理が急激に遅くなる(スワップ機能により、外部記憶装置を使うため)

# 本日は・・・

- アルゴリズム
  - アルゴリズムとは
  - 良いアルゴリズム
- timeコマンド

# timeコマンド

UNIXコマンドを実行した**時間を計ってくれる**

【使い方】

```
$ time　実行ファイル名
```

【使用例】

```
$ time ls
（「ls」にかかった時間を最後に表示する）
```

自分で作ったプログラムの実行時間も計れる

```
$ time ./a.out
```

# 【練習9-1】

講義資料のページから、ファイル「9_max1.c」をダウンロードして、「aが最大値の場合」のみ動作するプログラムを実行し、実行結果を確認してみましょう。

ただし、「bが最大値の場合」「cが最大値の場合」はまだ正しく動作しません。

# 【練習9-2】

**練習9-1のプログラムの実行時間を、timeコマンドを使って測ってみましょう。**

**timeコマンドは3つの時間「real」「user」「sys」を表示します。それぞれどのような意味なのか、調べてみましょう。**

# 【課題9-1】

**練習9-1の「9_max1.c」のプログラムを、「bが最大値の場合」「cが最大値の場合」も正しく動作するように改良してください。**

**できる人は、「改善案その1」も試してみましょう。**

# 【課題9-1】

[実行結果]

```
---max1 for a---
```
（全ての条件の組合せで30が返ってくる）

```
---max1 for b---
```
（全ての条件の組合せで30が返ってくる）

```
---max1 for c---
```
（全ての条件の組合せで30が返ってくる）

# 【課題9-2】

**講義資料のページから、ファイル「9_max2.c」をダウンロードして、「改善案その2」で最大値を求める関数max2を作って下さい。**

[ 実行結果 ]（課題9-1と同じ結果になる）

```
---max2 for a---
（全ての条件の組合せで30が返ってくる）


---max2 for b---
（全ての条件の組合せで30が返ってくる）


—max2 for c---
（全ての条件の組合せで30が返ってくる）
```

# 【課題9-3】

**講義資料のページから、ファイル「9_prime1.c」をダウンロードして、「引数で与えられた整数が素数かどうかを判定する」関数isprimeを作って下さい。**

[ この関数のプロトタイプ宣言 ]

```
int isprime(int num);

  /* 変数iが2からnumまで変化する繰り返し処理を作り、
     その中で「numをiで割った余りが0」であれば繰り返しを中断する */
  /* 繰り返し後に、「i==num」であれば、
     （つまり、割り切れる整数がnumのみであれば）numを返す*/
  /* そうでなければ、
     （つまり、途中で割り切れる整数があれば）、0を返す */
```

# 【課題9-3】

[実行結果]（100までの素数が出力される）

```
2, 3, 0, 5, 0, 7, 0, 0, 0, 11, 0, 13, 0, 0, 0, 17, 0, 19,
0, 0, 0, 23, 0, 0, 0, 0, 0, 29, 0, 31, 0, 0, 0, 0, 0, 37,
0, 0, 0, 41, 0, 43, 0, 0, 0, 47, 0, 0, 0, 0, 0, 53, 0, 0,
0, 0, 0, 59, 0, 61, 0, 0, 0, 0, 0, 67, 0, 0, 0, 71, 0, 73,
0, 0, 0, 0, 0, 79, 0, 0, 0, 83, 0, 0, 0, 0, 0, 89, 0, 0, 0,
0, 0, 0, 0, 97, 0, 0, 0,
```

# 【課題9-4】

講義資料のページから、ファイル「9_prime2.c」をダウンロードして、「エラトステネスのふるいで素数を求める」関数prime2を完成させてください。
「エラトステネスのふるい」のアルゴリズムがわからない場合は、Wikipediaなどで調べてみましょう。

[この関数のプロトタイプ宣言]

```
void prime2(int *a, int num);

   /* 以下の部分の処理だけを自分で書き足して、完成させてみましょう。 */
   /* 変数jを2から開始し「i*jがnum以下」の間j++する繰り返すforを作る */
   /* 繰り返し処理の中で、配列aのi*j番目に0を代入する
      （つまり、「iの倍数」番目の整数は素数にはならないので0にする） */
```

# 【課題9-4】

[実行結果]（課題9-3と同様に、100までの素数が出力される）

```
2, 3, 0, 5, 0, 7, 0, 0, 0, 11, 0, 13, 0, 0, 0, 17, 0, 19,
0, 0, 0, 23, 0, 0, 0, 0, 0, 29, 0, 31, 0, 0, 0, 0, 0, 37,
0, 0, 0, 41, 0, 43, 0, 0, 0, 47, 0, 0, 0, 0, 0, 53, 0, 0,
0, 0, 0, 59, 0, 61, 0, 0, 0, 0, 0, 67, 0, 0, 0, 71, 0, 73,
0, 0, 0, 0, 0, 79, 0, 0, 0, 83, 0, 0, 0, 0, 0, 89, 0, 0, 0,
0, 0, 0, 0, 97, 0, 0, 0,
```

# 【課題9-5】

- 課題9-3と課題9-4のプログラムの実行時間を、timeコマンドで計ってください
  （どちらの実行時間もほとんど差がでません）
- 2つのプログラムを「300000までの素数を表示する」ように変更して（Nを変更すればよい）、timeコマンドで計ってください（どちらが速いのか確認してください）

課題9-3は、計算量は多いがメモリ使用量は少ない
課題9-4は、計算量は少ないがメモリ使用量は多い

# まだ余裕のある人は… 【課題9-6】

- **自分のPCでの実行時間はどのくらいなのか調べてみしまょう**
- **「表示する素数を増やしてみて」演習室のPCではどの程度のなのか調べてみましょう**