

情報生命科学演習 文字列

東京大学院新領域・情報生命・特任准教授

加藤 毅

kato-tsuyoshi@k.u-tokyo.ac.jp

文字列



Test09a.java

プロジェクト名: test09

```
import java.lang.Math;
public class Test09a {
    public static void main( String[] args ){
        String str;
        str = "Myogatani";
        System.out.println("str="+str);
    }
}
```

文字列定数はダブルクォーテーションで

画面

str=Myogatani

スタック

str

ヒープ

"Myogatani"

int	整数型
double	実数型
String	文字列型
int[]	整数の配列型
double[]	実数の配列型

文字型



Test09c.java

プロジェクト名: test09

```
import java.lang.Math;
public class Test09c {
    public static void main( String[] args ){
        char c;
        c = 'T';
        System.out.println("c="+ch1);
    }
}
```

文字定数はシングルクォーテーションで

画面

c=T

int	整数型
double	実数型
String	文字列型
char	文字型
int[]	整数の配列型
double[]	実数の配列型

1次元文字型配列



Test09c.java

プロジェクト名: test09

```
import java.lang.Math;
public class Test09c {
    public static void main( String[] args ) {
        char[] nucs;
        nucs = new char[]{'A','T','G','C'};
        System.out.println("nucs[0]="+nucs[0]);
        System.out.println("nucs[1]="+nucs[1]);
        System.out.println("nucs[2]="+nucs[2]);
        System.out.println("nucs[3]="+nucs[3]);
    }
}
```

1次元文字配列の初期化子

画面

```
nucs[0]=A
nucs[1]=T
nucs[2]=G
nucs[3]=C
```

int	整数型
double	実数型
String	文字列型
char	文字型
int[]	整数の配列型
double[]	実数の配列型
char[]	文字の配列型

2次元文字型配列



Test09d.java

プロジェクト名: test09

```
import java.lang.Math;
public class Test09d {
    public static void main( String[] args ){
        char[][] tab;
        tab = new char[][]{{'A','U'},{'T','A'},{'G','C'},{'C','G'}};
        System.out.println(tab[0][0]+" => "+tab[0][1]);
        System.out.println(tab[1][0]+" => "+tab[1][1]);
        System.out.println(tab[2][0]+" => "+tab[2][1]);
        System.out.println(tab[3][0]+" => "+tab[3][1]);
    }
}
```

画面

```
A => U
T => A
G => C
C => G
```

文字列の連結



Test09b.java

プロジェクト名: test09

```
import java.lang.Math;
public class Test09b {
    public static void main( String[] args ){
        String str1, str2, str3;
        str1 = "myoga";
        str2 = "tani";
        str3 = str1+str2;
        System.out.println("str3="+str3);
    }
}
```

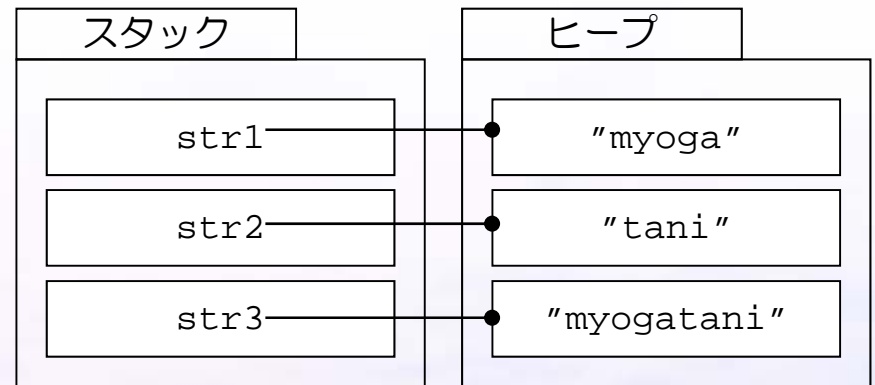
+で連結できる

str1とstr2を連結

文字列定数とstr3を連結

画面

str3=myogatani



実際には、String は文字列プールという技術を使って複雑な処理を行っているが、我々は細かいJVM実装上の工夫をあまり気にしなくてもよいだろう。

文字列の連結



Test09c.java

プロジェクト名: test09

```
import java.lang.Math;
public class Test09c {
    public static void main( String[] args ){
        String str0, str1, str2, str3;
        str0 = "myogatan";
        str1 = str0+3;
        str2 = str0+1.2;
        str3 = str0+'i'
        System.out.println("str1="+str1);
        System.out.println("str2="+str2);
        System.out.println("str3="+str3);
    }
}
```

String+String=String
String+int =String
String+double=String
String+char =String

画面

```
str1=myogatan3
str2=myogatan1.2
str3=myogatani
```

文字列のメソッド, その1



Test09e.java

プロジェクト名: test09

```
import java.lang.Math;
public class Test09e {
    public static void main( String[] args ){
        String str;
        char   ch1, ch2;
        int    len;
        str   = "ATGC";
        ch1   = str.charAt(1);
        ch2   = str.charAt(2);
        len   = str.length();
        System.out.println("ch1="+ch1);
        System.out.println("ch2="+ch2);
        System.out.println("len="+len);
    }
}
```

画面

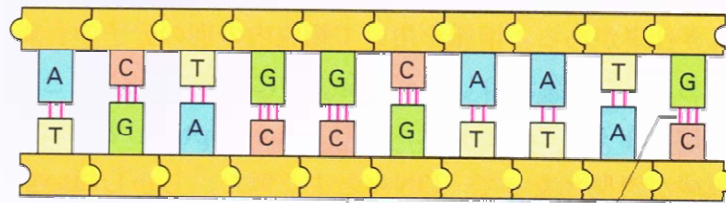
```
ch1=T
ch2=G
len=4
```

文字列.charAt(i)
(i+1)文字目を取り出す

文字列.length()
文字列の長さを返す

DNA配列の相補鎖

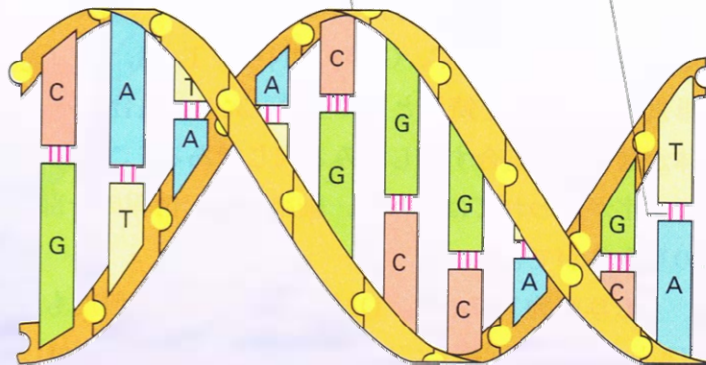
(D) 二本鎖 DNA



糖-リン酸からなる主鎖

水素結合した塩基対

(E) DNA二重らせん



塩基	相補塩基
A	T
T	A
G	C
C	G

たとえば

5' ATGGCGAACG 3'

の相補鎖は

3' TACCGCTTGC 5'

となり、5'末端を左にすると

5' CGTTCGCCAT 3'

となる

Molecular Biology of the Cell 4版, p.194

逆相補鎖の計算法



ホワイトボードで説明します

練習9-h. 逆相補鎖の計算



Test09h.java

プロジェクト名: test09

DNA配列の逆相補鎖
を計算するプログラム
を完成させよ

```
public class Test09h {
    public static void main( String[] args ){
        String  seq, compl;
        seq = "ATGGCGAACG";
        compl = comp_compl( seq );
        System.out.println( "compl="+compl );
    }
    public static String comp_compl( String seq ){
        int i, len;
        String ret;
        len = seq.length();
        ret = "";
        for ( i = 0; i < len; i++ ){
            ret = ret + seq.charAt(            );
        }
        return ret;
    }
}
```

画面

compl=CGTTCGCCAT

塩基配列間の距離



- ・ p-distance
 - 2つの同じ長さの塩基配列の間の距離
 - 異なる塩基数 / 長さ

例1

```
GGCTCAGCTC
AGCTGAGCTC
*** *****
```

p-distance=2/10=0.2

例2

```
GGCTCAGCTC
GGCTCAGCCC
***** **
```

p-distance=1/10=0.1

例3

```
GGCTCAGCTC
AGATCGGTTC
* ** * **
```

p-distance=4/10=0.4

例4

```
AGCTGAGCTC
GGCTCAGCCC
*** ** *
```

p-distance=3/10=0.3

練習9-m. p-distance の計算(1/2)

Test09m.java

プロジェクト名: test09

```
public class Test09m {
    public static void main( String[] args ){
        String  seq1, seq2;
        double  pdist;
        seq1   = "GGCTCAGCTC";
        seq2   = "AGCTGAGCTC";
        pdist  = comp_pdist( seq1, seq2 );
        System.out.println("pdist="+pdist);
    }
    public static double comp_pdist( String seq1, String seq2 ){略}
}
```

上記は、2つの文字列 seq1 と seq2 の p-distance を計算するプログラムである。

(1) これを実装して、動作させてみよ

練習9-m. p-distance の計算(2/2)



Test09m.java

プロジェクト名: test09

```
public class Test09m {
    public static void main( String[] args ){略}
    public static double comp_pdist( String seq1, String seq2 ){
        int    i, len, cnt;
        double ret;
        assert seq1.length() == seq2.length();
        len = seq1.length();
        cnt = 0;
        for ( i = 0; i < len; i++ ){
            if ( seq1.charAt(i) != seq2.charAt(i) ){
                cnt = cnt + 1;
            }
        }
        ret = ((double)cnt) / len;
        return ret;
    }
}
```

(2) このプログラムを使って、次のペアのp-distanceを計算せよ

- GGCTCAGCTCとGGCTCAGCCC
- GGCTCAGCTCとAGATCGGTTC
- AGCTGAGCTCとGGCTCAGCCC

練習9-n. p-distance の距離行列の計算

Test09m.java

プロジェクト名: test09

```
public class Test09n {
    public static void main( String[] args ){
        String[]  seqs;
        double[][] distmat;
        seqs      = new String[]{"GGCTCAGCTC", "AGCTGAGCTC", "GGCTCAGCCC", "AGATCGGTTC"};
        distmat = comp_pdistmat( seqs );
        pri_ary2d( distmat );
    }
    public static double[][] comp_pdistmat( String[] seqs ){
        文字列集合 seqs 間の距離行列を返せ
    }
    public static double comp_pdist( String seq1, String seq2 ){略}
    public static void pri_ary2d( double[][] mat ){略}
}
```

Test09m.javaの出力

```
0.00 0.20 0.10 0.40
0.20 0.00 0.30 0.40
0.10 0.30 0.00 0.50
0.40 0.40 0.50 0.00
```

これは, seqsにある文字列集合間の距離
行列を計算するプログラムである. メ
ソッド comp_pdistmat(String[])を完成
させよ

転写因子reb1に結合する配列

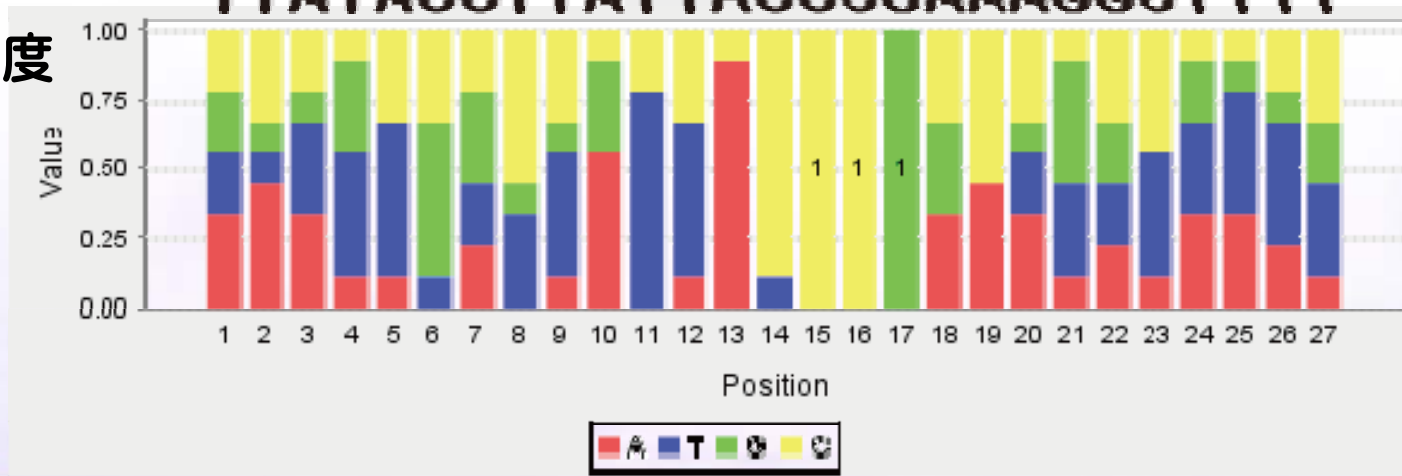


配列

```

AATCTGTTTATTACCCGACATTCTTAA
TATTTCTCTGTCACCCGGCCTCTATTT
ACTTTCATCATTACCCGGATGATTCTC
GACGCTGCTATAACCCGGCCTCCGAGG
ACATTGGGCATTACCCGCAAACTAACC
CGGGCGACAGCCCTCCGACGGAAAGACT
CCAATGGCCGCTACCCGCCTCTTCTTC
GACGCGCCGCTCACCCGCACGGCAGAG
TTATAGCTTATTACGGGAAAAGGCTTTT
    
```

頻度



頻度行列を計算しよう



TTACCCCG
TCACCCCG
TTACCCCG
TAACCCCG
TTACCCCG
CCCTCCCG
CTACCCCG
CGCTCAC
TTATTAC

A	0.000	0.111	0.778	0.000	0.000	0.222	0.000
T	0.667	0.936	0.000	0.889	0.111	0.000	0.000
G	0.000	0.111	0.000	0.000	0.000	0.000	0.778
C	0.889	0.222	0.222	0.667	0.889	0.778	0.222

配列の集合から
2次元配列で表現した
頻度行列を計算してみよう

練習9-i. 頻度行列の計算 (1/3)

Test09i.java

プロジェクト名: test09

```
public class Test09i {
    public static void main( String[] args ){
        String[]  seqs;
        double[][] freqmat;
        seqs      = new String[]{"GTATAAAAAGCGG", "CTATAAAAGGCC", "GTATAAAGGGGCG",
                                "GTATATAAGCGCG", "CTATAAAGGGGCC", "GTATAAAGGCGGG"};
        freqmat = comp_freqmat( seqs );
        pri_freqmat( freqmat );
    }
    // 頻度行列を計算する
    public static double[][] comp_freqmat( String[] seqs ){
        文字列集合 seqs 間から頻度行列を計算し, これを返せ
    }
    // 頻度行列を表示する
    public static void pri_ary2d( double[][] freqmat ){略}
}
```

練習9-i. 頻度行列の計算 (2/3)



Test09i.javaの標準出力

プロジェクト名: test09

```
0.00 0.00 1.00 0.00 1.00 0.83 1.00 0.50 0.17 0.00 0.00 0.00 0.00
0.00 1.00 0.00 1.00 0.00 0.17 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.67 0.00 0.00 0.00 0.00 0.00 0.00 0.50 0.83 0.67 0.67 0.33 0.67
0.33 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.33 0.33 0.67 0.33
```

上記のような出力が得られるように、頻度行列を計算するプログラムを完成させよ

練習9-i. 頻度行列の計算(3/3)



TTACCCCG
TCACCCCG
TTACCCCG
TAACCCCG
TTACCCCG
CCCTCCCG
CTACCCCG
CGCTCAC
TTATTAC

(2) データを左の配列集合に置き換えてみて、頻度行列を計算してみよ

A	0.000	0.111	0.778	0.000	0.000	0.222	0.000
T	0.667	0.936	0.000	0.889	0.111	0.000	0.000
G	0.000	0.111	0.000	0.000	0.000	0.000	0.778
C	0.889	0.222	0.222	0.667	0.889	0.778	0.222

真偽値とその配列

int	整数型
double	実数型
String	文字列型
char	文字型
boolean	真偽値
int[]	整数の配列型
double[]	実数の配列型
String[]	文字列の配列型
char[]	文字の配列型
boolean[]	真偽値の配列型

boolean 型のとりうる値は
true か false のみ

真偽値の演算



x と y を boolean 型変数と仮定する

x && y	x かつ y
x y	x または y
!x	x の否定

びっくりマーク

比較演算子



x と y を int 型か double 型か char 型と仮定する

true のときは

$x > y$	x は y より大きい
$x \geq y$	x は y 以上
$x < y$	x は y より小さい
$x \leq y$	x は y 以下
$x == y$	x と y は等しい
$x != y$	x と y は等しくない

これらの比較演算子はすべて boolean 型を返す

練習9-j. うるう年の判定

うるう年は、4で割り切れて、かつ、100で割り切れない、か、400で割り切れる年である。以下のプログラムは指定した年がうるう年か判定する

Test09j.java

プロジェクト名: test09

```
public class Test09j {
    public static void main( String[] args ){
        int year;
        boolean intercalary;
        year = 2008;

        intercalary = year % 4 == 0;
        intercalary = intercalary && ( year % 100 != 0 || year % 400 == 0 );
        if ( intercalary ){
            System.out.println(year+" nen wa urudoshi de aru.");
        } else {
            System.out.println(year+" nen wa urudoshi dewa nai.");
        }
    }
}
```

剰余演算を行う

2000年や1900年などを入れてみて動作を確認せよ

練習9-k.

5科目すべて30点以上なら合格とするとき、与えられた scores が合格か判定するプログラムを完成させよ

Test09k.java

プロジェクト名: test09

```
public class Test09k {
    public static void main( String[] args ){
        int[] scores;
        boolean passed;
        scores = new int[]{ 85, 72, 93, 87, 27 };
        passed = judge( scores );
        if ( passed ){
            System.out.println("Passed");
        } else {
            System.out.println("Failed");
        }
    }
    public static boolean judge( int[] scores ){
```

5科目の点数 scores がすべて 30 以上ならば true を、
さもなければ false を返せ。

```
    }
}
```

相同性検索とドットマトリックス

S F D L S T P D A V M G N P K G H G K K V L

Y F D L S H G S A Q V K G H G K K V A

この2本のアミノ酸配列のうち
似ている領域を見つけたい

相同性検索とドットマトリックス

S F D L S T P D A V M G N P K G H G K K V L

Y F D L S H G S A Q V K G H G K K V A

この2本のアミノ酸配列のうち
似ている領域を見つけたい

ドットマトリックス

	S	F	D	L	S	T	P	D	A	V	M	G	N	P	K	G	H	G	K	K	V	L
Y																						
F																						
D																						
L																						
S																						
H																						
G																						
S																						
A																						
Q																						
V																						
K																						
G																						
H																						
G																						
K																						
K																						
V																						
A																						

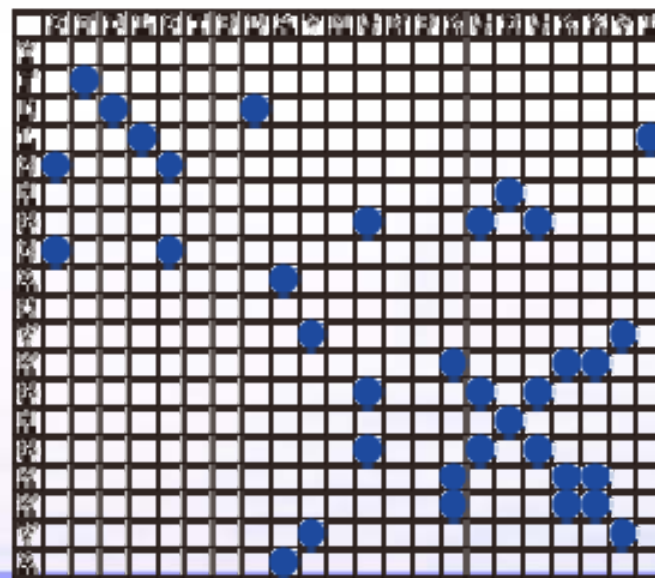
相同性検索とドットマトリックス

S F D L S T P D A V M G N P K G H G K K V L

Y F D L S H G S A Q V K G H G K K V A

この2本のアミノ酸配列のうち
似ている領域を見つけたい

ドットマトリックス



アミノ酸が一致する位置に
しるしをつける



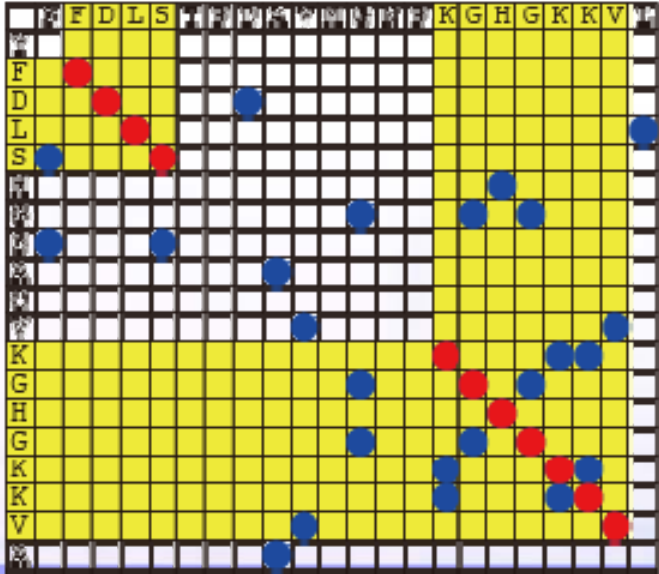
相同性検索とドットマトリックス

S F D L S T P D A V M G N P K G H G K K V L

Y F D L S H G S A Q V K G H G K K V A

この2本のアミノ酸配列のうち
似ている領域を見つけたい

ドットマトリックス



アミノ酸が一致する位置に
しるしをつける



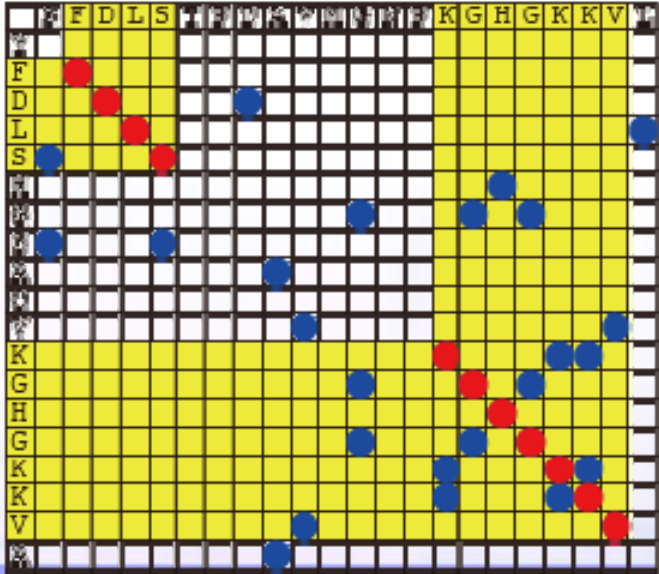
相同性検索とドットマトリックス

S F D L S T P D A V M G N P K G H G K K V L

Y F D L S H G S A Q V K G H G K K V A

この2本のアミノ酸配列のうち
似ている領域を見つけたい

ドットマトリックス



アミノ酸が一致する位置に
しるしをつける

練習9-1. ドットマトリックスの計算 (1/3)

Test091.javaの標準出力

プロジェクト名: test09

```
public class Test09k {
    public static void main( String[] args ){
        String      seq1, seq2;
        boolean[][] dotmat;
        seq1 = "YFDLSHGSAQVKGHGKKVA";
        seq2 = "SFDLSTPDAVMGNPKVKGHGKKVL";
        dotmat = get_dotmat( seq1, seq2 );
        pri_dotmat( dotmat, seq1, seq2 );
    }
    public static boolean[][] get_dotmat( String seq1, String seq2 ){
        int      i, j, len1, len2;
        boolean[][] ret;
        len1 = seq1.length();
        len2 = seq2.length();
        ret = new boolean[seq1.length()][seq2.length()];

        return ret;
    }
    public static void pri_dotmat( boolean[][] dotmat, String seq1, String seq2 ){略}
}
```

ドットマトリックス dotmat を計算せよ。ただし、len1 行 len2 列とする。
dotmat[j][i]には、seq1の(j+1)文字目とseq1の(i+1)文字目が等しければ
true, さもなければ false を入れる。

練習9-1. ドットマトリックスの計算 (2/3)

Test091.javaの標準出力

プロジェクト名: test09

```
public class Test09k {
    public static void main( String[] args ) {
        String      seq1, seq2;
        boolean[][] dotmat;
        seq1 = "YFDLSHGSAQVKGHGKKVA";
        seq2 = "SFDLSTPDAVMGNPKVKGHGKKVL";
        dotmat = get_dotmat( seq1, seq2 );
        pri_dotmat( dotmat, seq1, seq2 );
    }
    public static boolean[][] get_dotmat( String seq1, String seq2 ){略}
    public static void pri_dotmat( boolean[][] dotmat, String seq1, String seq2 ){
        int          i, j, len1, len2;
        len1 = seq1.length();
        len2 = seq2.length();
        System.out.println( " "+seq2 );
        for ( j = 0; j < len1; j++ ){
            System.out.print(seq1.charAt(j));
            for ( i = 0; i < len2; i++ ){
                if ( dotmat[j][i] ){
                    System.out.printf("*");
                } else {
                    System.out.printf(" ");
                }
            }
            System.out.println();
        }
    }
}
```

2010年夏, 情報生命科学演習, 加藤.

練習9-1. ドットマトリックスの計算 (3/3)

Test091.javaの標準出力

プロジェクト名: test09

```
SFDLSTPDAVMGNPKVKGHGKKVL
Y
F *
D * *
L * *
S* *
H *
G * *
S* *
A * *
Q
V * * *
K * * **
G * * *
H *
G * *
K * * **
K * * **
V * * *
A * *
```

- (1) 上記のような出力が得られるように、ドットマトリックスを計算して表示するプログラムを完成させよ
- (2) 配列 seq2 を「 SFDLSTPDAVMGNPKVKGHGKKVLSFDLSTPDAVMGNPKVKGHGKKVL 」と2度繰り返したものに変更したときのドットマトリックスを計算してみよ

まとめ(1/3) : String, char, boolean



int	整数型
double	実数型
String	文字列型
char	文字型
boolean	真偽値
int[]	整数の配列型
double[]	実数の配列型
String[]	文字列の配列型
char[]	文字の配列型
boolean[]	真偽値の配列型

まとめ(2/3) : String のメソッド



文字列 + 文字列
文字列を連結する

文字列.charAt(i)
(i+1)文字目を取り出す

文字列.length()
文字列の長さを返す

文字列.substring(i, j)
(i+1)文字目からj文字目
までの部分文字列を返す

文字列1.equals(文字列2)
文字列1と文字列2が等しいか
等しくないか返す

まとめ(3/3): 論理演算子と比較演算子

論理演算子

x と y を boolean 型変数と仮定する

x && y	x かつ y
x y	x または y
!x	x の否定

比較演算子

x と y を int 型か double 型か char 型と仮定する

x > y	x は y より大きい
x >= y	x は y 以上
x < y	x は y より小さい
x <= y	x は y 以下
x == y	x と y は等しい
x != y	x と y は等しくない

これらの比較演算子はすべて boolean 型を返す

補足: 文字列のメソッド, その2



Test09e.java

プロジェクト名: test09

```
import java.lang.Math;
public class Test09e {
    public static void main( String[] args ){
        String str1, str2;
        str1 = "abcdefg";
        str2 = str1.substring(3,6);
        str3 = "def";
        System.out.println("str2="+str2);
        if ( str2.equals(str3) ){
            System.out.println("str2 is equal to str3");
        }
    }
}
```


文字列.substring(i, j)
(i+1)文字目からj文字目
までの部分文字列を返す

文字列1.equals(文字列2)
文字列1と文字列2が等しいか
等しくないか返す

画面

```
str2=def
str2 is equal to str3
```

補足：標準ライブラリ



標準ライブラリには String 以外にも様々なクラスが用意されている

標準ライブラリのドキュメントは以下からダウンロードできる
<http://java.sun.com/javase/ja/6/download.html>