



知能情報工学演習I 第13回(後半第7回)

岩村雅一

masa@cs.osakafu-u.ac.jp

前回の課題

■ 課題1

□ 小数点以下第1位を四捨五入する関数を作りなさい。

- 例1: 60.2 → 60
- 例2: 5.8 → 6

```
#include<stdio.h>

int func(float x){
    return (int)(x+0.5);
}

int main(void){
    float i;
    printf("小数を入力: ");
    scanf("%f",&i);
    printf("四捨五入:");
    printf("%d¥n",func(i));

    return(0);
}
```

前回の課題

■ 課題2

□ 階乗(1からnまでの自然数の積)を計算する関数を作り、順列と組み合わせを表示しなさい

■ 順列

$${}_n P_r = n(n-1)(n-2)\cdots(n-r+1) = \frac{n!}{(n-r)!}$$

■ 組み合わせ

$${}_n C_r = \frac{{}_n P_r}{r!} = \frac{n!}{r!(n-r)!}$$

前回の課題の回答例1 (for文を使った場合)

```
#include <stdio.h>

/* 階乗を計算する関数 */
int fact(int x) {
    int i, fact = 1;
    for(i = 2; i <= x; i++) {
        fact *= i;
    } fact = 1 * 2 * 3 * 4 * ...
    return(fact);
} 0!や1!もok
```

```
int main (void){
    int n, r, p, c;

    printf("n: "); scanf("%d", &n);
    printf("r: "); scanf("%d", &r);


    p = fact(n) / fact(n-r);
    c = fact(n) / fact(n-r) / fact(r);

    printf("nPr = %d¥n", p);
    printf("nCr = %d¥n", c);

    return(0);
}
```

前回の課題の回答例2 (関数の再帰的呼び出し)

main関数は説明
のために消去



```
#include <stdio.h>
```

```
/* 階乗を計算する関数 */
```

```
int fact(int x) {  
    if (x==1 || x==0) {  
        return(1);  
    } else {  
        return(x*fact(x-1));  
    }  
}
```

```
int main (void){  
    int n, r, p, c;  
  
    printf("n: "); scanf("%d", &n);  
    printf("r: "); scanf("%d", &r);  
  
    p = fact(n) / fact(n-r);  
    c = fact(n) / fact(n-r) / fact(r);  
  
    printf("nPr = %d\n", p);  
    printf("nCr = %d\n", c);  
  
    return(0);  
}
```

前回の課題の回答例2 (関数の再帰的呼び出し)

例: fact(2)の場合

```
#include <stdio.h>
```

```
/* 階乗を計算する関数 */
```

```
int fact(int x) {  
    if (x==1 || x==0) {  
        return(1);  
    } else {  
        return(x*fact(x-1));  
    }  
}
```

||
fact(1)
||
1

fact(1)の計算

```
/* 階乗を計算する関数 */
```

```
int fact(int 1) {  
    if (x==1 || x==0) {  
        return(1);  
    } else {  
        return(1*fact(1-1));  
    }  
}
```

前回の課題の回答例2 (関数の再帰的呼び出し)

```
#include <stdio.h>
```

```
/* 階乗を計算する関数 */
```

```
int fact(int x) {  
    if (x==1 || x==0) {  
        return(1);  
    } else {  
        return(x*fact(x-1));  
    }  
}
```

$$\begin{aligned} & \text{fact}(4) \\ = & 4 * \text{fact}(3) \\ = & 4 * 3 * \text{fact}(2) \\ = & 4 * 3 * 2 * \text{fact}(1) \\ = & 4 * 3 * 2 * 1 \end{aligned}$$

後半の予定

7. 6月1日 プログラミング環境(テキスト1,2章)
8. 6月8日 変数とデータ型(3章)、演算子(4章)
9. 6月15日 コンソール入出力(6章)、配列(3章)
10. 6月22日 制御文1(テキスト5章)
11. 6月29日 制御文2(テキスト5章)
12. 7月13日 関数(テキスト7章)
13. 7月20日 配列(3章)、応用プログラム

本日のメニュー

- 多次元配列
- 配列の初期化
- 応用問題：文字認識

配列 (テキストP.57)

- 同種のデータ型を連続してメモリに確保したもの

- 配列の宣言

```
int a[10];
```

int型の変数10個で構成される配列a

```
double b[5];
```

double型の変数5個で構成される配列b

配列 (復習)

■ 普通の変数

```
int a;  
a = 10;
```

a

10

■ 配列

```
int a[5];  
a[0] = 1;  
a[1] = 2;  
a[2] = 5;  
a[3] = 10;  
a[4] = -3;
```

a[0] a[1] a[2] a[3] a[4]

a

1 2 5 10 -3

多次元配列 (テキストP.59)

- 同種のデータ型を連続してメモリに確保したもの

- 多次元配列の宣言

```
int a[10][10];
```

int型の変数10x10個で構成される配列a

```
double b[5][5];
```

double型の変数5x5個で構成される配列b

多次元配列

```
int a[3][3];  
a[0][0] = 1;  
a[0][1] = 2;  
a[0][2] = 5;  
a[1][0] = 10;  
a[1][1] = -3;  
a[1][2] = 4;  
a[2][0] = 1;  
a[2][1] = 2;  
a[2][2] = 5;
```

	a[][1]		
	↓	a[][2]	
a[][0]	1	2	5
a[1][]	10	-3	4
a[2][]	8	-1	-5

多次元配列のサンプルプログラム

```
#include <stdio.h>
```

```
int main(void) {  
    int i, j, a[2][3];  
    for (i=0; i<2; i++) {  
        for (j=0; j<3; j++) {  
            printf("Input a[%d][%d]: ", i, j);  
            scanf("%d", &a[i][j]);  
        }  
    }  
}
```

```
for (i=0; i<2; i++) {  
    for (j=0; j<3; j++) {  
        printf("a[%d][%d]  
        = %d¥n", i, j, a[i][j]);  
    }  
}  
  
return 0;  
}
```

配列の初期化 (テキストP.67)

- 変数は宣言するときに値を設定できる

```
int a=10;
```

- 配列は宣言するときに値を設定できる

```
int a[3]={10,20,30};
```

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



```
int a[3];  
a[0] = 10;  
a[1] = 20;  
a[2] = 30;
```

省略可能

応用問題：文字認識

- 文字認識

- パターン認識の一つ

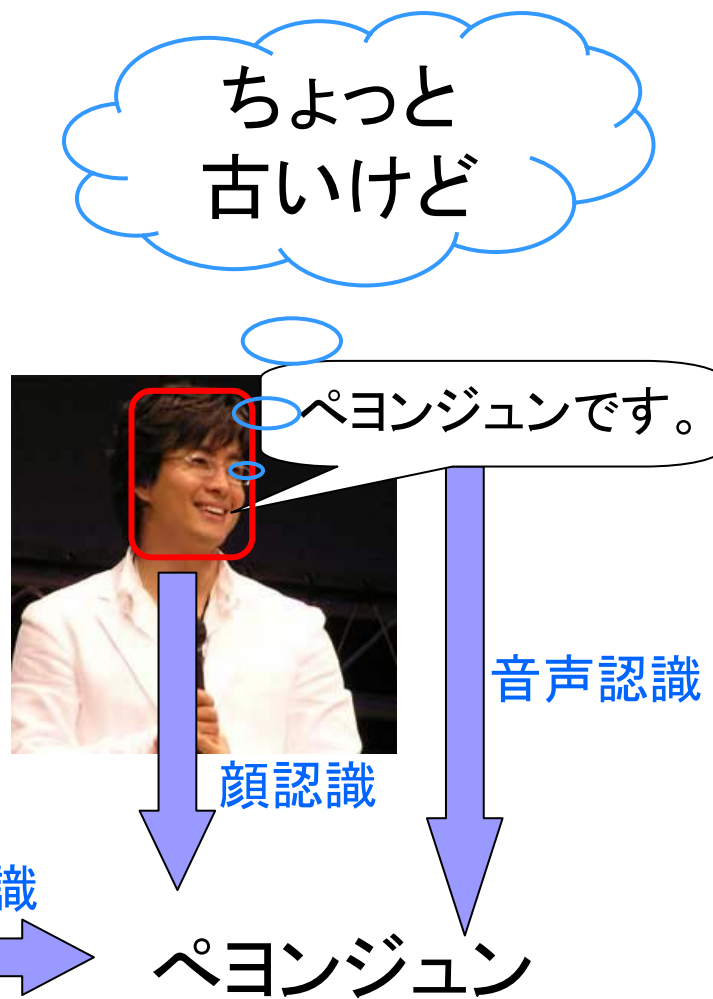
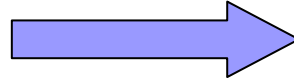
パターン認識

■ パターン認識とは？

- パターンを概念に対応付けること
 - 文字認識
 - 音声認識
 - 顔認識
- 人には簡単
- 機械には難しい



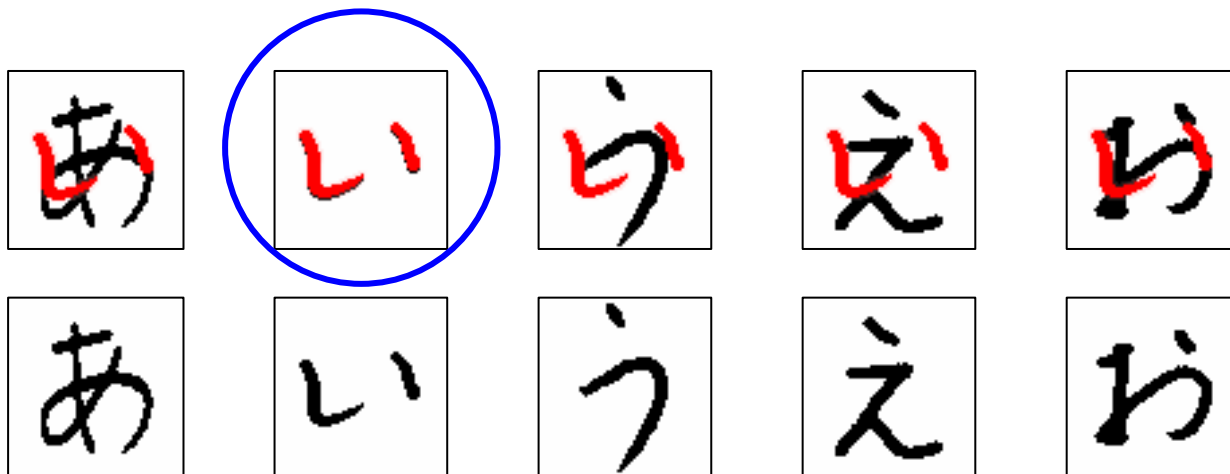
文字認識



簡単な文字認識の仕組み

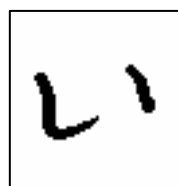
重なり部分が一番多い

学習パターン



全ての文字との重ね合わせ

認識するパターン



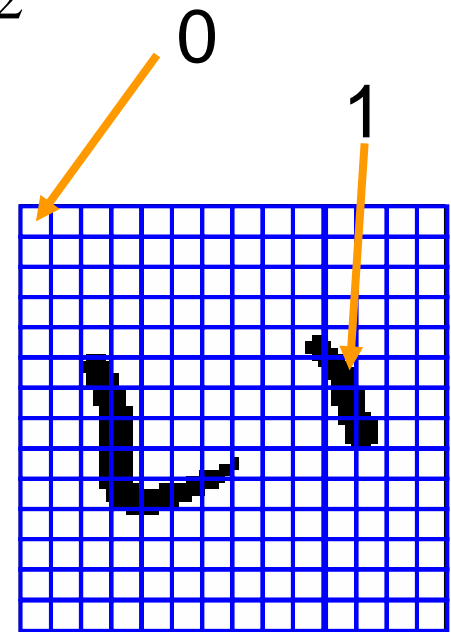
2枚の画像の重なり具合

尺度1
$$d_1 = \sum_i \sum_j |a_{ij} - b_{ij}|$$

尺度2
$$d_2 = \sum_i \sum_j (a_{ij} - b_{ij})^2$$

a_{ij} : 画像Aの(i,j)画素

b_{ij} : 画像Bの(i,j)画素



画素の値は、白色なら0、黒色なら1

文字認識のまとめ

- 全ての文字と重ね合わせて、重なるの度合いが最も大きい文字が正解
- 文字画像は(画素の)配列で表すとよい
- 重なるの度合いは画素単位で調べる
 - 画素の値: 白色は0、黒色は1