

プログラミング基礎 12/5

TAによる解説の資料

ひな形; データの読み込み

```
int main(){  
  
    int i; /* ループのためのカウンタ */  
    int n; /* 入力する数の個数 */  
    int a[100]; /* アクセスは a[0] ~ a[99] なので注意 */  
  
    /* キーボードから読み込み */  
    printf("Array size ?\n");  
    scanf("%d", &n);  
    for(i=0; i<n; i++){  
        scanf("%d", &a[i]);  
    }  
  
    /* プリント */  
    for(i=0; i<n; i++)  
        printf("%d ", a[i]);  
  
    printf("\n");  
}
```

data ファイルを作らなかった場合

```
cm10134$ gcc 1.c  
cm10134$ ./a.out  
Array size ?  
4 ← 配列の数 n ;自分で打ち込む  
1 3 5 7 ← 配列の中身 a[0]~a[3];自分で打ち込む  
1 3 5 7 ← 配列の中身 a[0]~a[3];スキャンしたものが  
プリントされる
```

for の中身の文が
一つだけの場合、
<>はつけなくても良い

data ファイルを作った場合

```
cm10134$ gcc 1.c  
cm10134$ ./a.out < data  
Array size ?  
-3 65 73 12 51 21 1 7 80 -12
```

配列の中身 ;data ファイル内にあるものが
プリントされる

平均と標準偏差の計算

メイン関数のみ

```
Int main(){
int sum,sum2; // 和 , 2乗の和
double E,S; // 平均 , 標準偏差
```

```
//平均値の計算
sum =0;//和の初期化
for(i=0; i<n; i++)
    sum += a[i];

E = (double)sum/(double)n;
/* 小数型への変換を忘れずに */
```

```
//標準偏差の計算
sum2 = 0;//二乗和の初期化
for(i=0; i<n; i++)
    sum2 += a[i]*a[i];

S = sqrt((double)sum2/(double)n - E*E);
}
```

サブ関数二つを利用する

```
Int main(){
double E,S; // 平均 , 標準偏差
    E = average(a,n);
    S = deviation(a,n);
}

double average(int data[100], int num){
    int i; /* ループのためのカウンタ */
    int sum =0;//和の初期化
        for(i=0; i<n; i++) sum += data[i];
    return (double)sum/(double)num;
}

double deviation(int data[100], int num){
    int i; /* ループのためのカウンタ */
    double E; /* 平均 */
    Int sum2 = 0;//二乗和の初期化
        for(i=0; i<n; i++) sum2 += data[i]*data[i];

    E = average(data, num);//平均値の計算

return sqrt((double)sum2/(double)num - E*E);
}
```

平均と標準偏差の計算

一つのサブ関数で平均と標準偏差両方を計算し、両方の値を返す
→ポインタによる計算が必要

```
Int main(){
double E,S; /* 平均 , 標準偏差 */

ave_devi(&E, &S, a, n); /* & マークを付けて変数のアドレスを渡す */
}

void ave_devi(double* pE, double* pS, int data[100], int num){
int i; /* ループのためのカウンタ */
int sum=0;//和の初期化
Int sum2; //二乗和の初期化

for(i=0; i<num; i++){ //和と二乗和の計算
    sum += data[i];
    sum2 += data[i]*data[i];
}
/* 平均値と標準偏差を計算 */
/* ポインタ中身にアクセスするには *マークを付ける */
*pE = (double)sum/(double)num;
*pS = sqrt((double)sum2/(double)num - *pE**pE);
}
```

累積値への更新

メイン関数のみ

```
int main(){  
/* 累積値へ変換 */  
for(i=1; i<n; i++) /*カウンタは1から*/  
    a[i] += a[i-1]; /* 一つ前の値を足しこむ */  
  
/* 出力をプリント */  
printf("accumulated: ");  
for(i=0; i<n; i++){  
    printf("%d ", a[i]);  
}  
printf("¥n");  
}
```

サブ関数を使う

```
int main(){  
    accumulate(a, n);  
/* 出力をプリント */  
printf("accumulated: ");  
for(i=0; i<n; i++){  
    printf("%d ", a[i]);  
}  
printf("¥n");  
}  
  
void accumulate(int data[100], int num){  
int i; /* ループのためのカウンタ */  
/* 累積値へ変換 */  
for(i=1; i<num; i++){  
    data[i] += data[i-1];  
}  
}
```

最小値とそのインデックスを求める (main関数のみ) ～変数、配列の宣言～

```
int main(){
```

```
int i; /* ループのためのカウンタ */
```

```
int n; /* 入力する数の個数 */
```

```
int a[100];
```

データ型 配列名 [配列の要素数];
要素数100個の配列の宣言
アクセスはa[0] ~ a[99]

```
int index;
```

最小値のインデックス

最小値とそのインデックスを求める (main関数のみ)

～入力値を配列に格納～

```
/* キーボードから読み込み */  
printf("Array size ?¥n");  
scanf("%d", &n);  
for(i=0; i<n; i++){  
    scanf("%d", &(a[i])); }
```

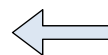
← 入力値を配列aのi番目に格納。

```
/* 入力をプリント */  
printf("Input: ");  
for(i=0; i<n; i++){  
    printf("%d ", a[i]);  
}  
printf("¥n");
```

最小値とそのインデックスを求める (main関数のみ)

～最小値を探す～

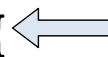
```
index = 0;
```



とりあえずa[0]が最小値であるとする。

```
for(i=1; i<n; i++){
```

```
    if(a[i] < a[index]){
```



a[0]～ a[i-1]までの最小値とa[i]を比較する。

```
        index = i;
```



a[i]の方が小さければ最小要素のインデックスをiに更新。

```
    }
```

```
}
```

```
printf("Index: %d¥n", index);
```

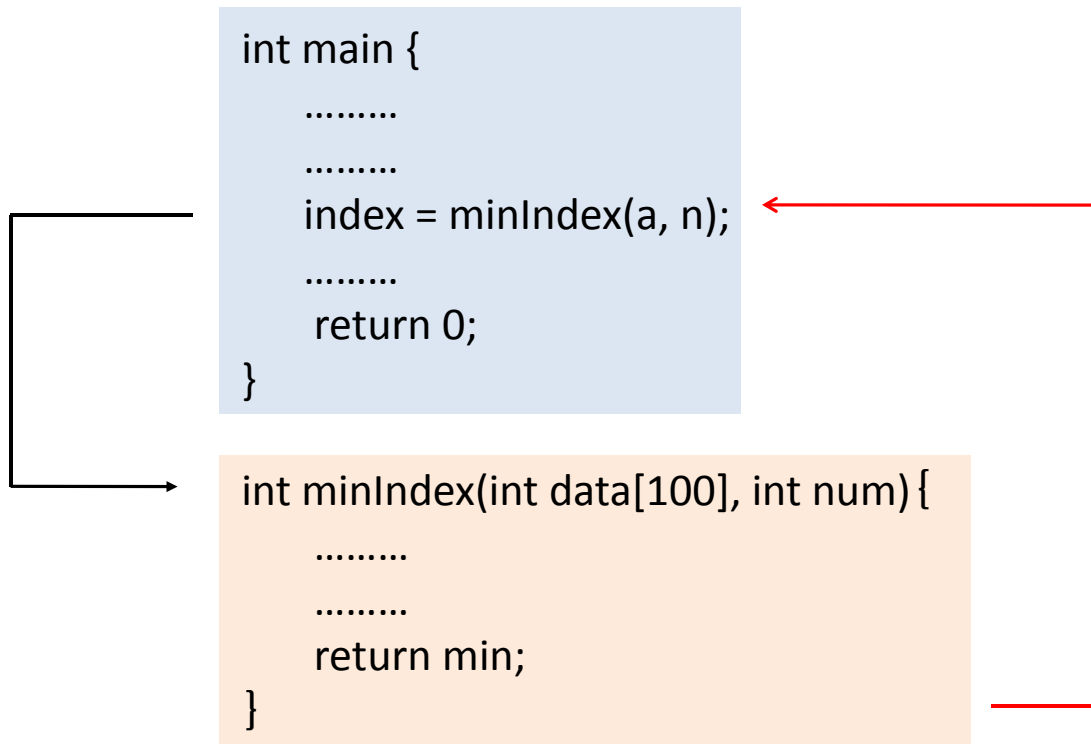
```
printf("Min. : %d¥n", a[index]);
```

```
return 0;
```

```
}
```

} 結果を出力

最小値とそのインデックスを求める(サブ関数使う)



最小値とそのインデックスを求める(サブ関数使う) ～プロトタイプ、変数、配列の宣言～

*/*プロトタイプ宣言*/*

関数は最初に使われるよりも前に宣言されていなければならない

```
int minIndex(int data[100], int num);
```

```
int main(){  
    int i;  
    int n;  
    int a[100];  
    int index;
```

変数、配列の宣言

最小値とそのインデックスを求める(サブ関数使う) ～入力値を配列に格納～

```
printf("Array size ?¥n");
scanf("%d", &n);
for(i=0; i<n; i++){
    scanf("%d", &(a[i]));
}
printf("Input: ");
for(i=0; i<n; i++){
    printf("%d ", a[i]);
}
printf("¥n");
```

キーボードから配列の要素数、要素を入力して配列に格納

配列に格納された値を出力

最小値とそのインデックスを求める(サブ関数使う) ～入力値を配列に格納～

```
index = minIndex(a, n);
```

さっきプロトタイプ宣言した関数の呼び出し。
戻り値をindexに代入。

```
printf("Index: %d¥n", index);
printf("Min. : %d¥n", a[index]);
return 0; }
```

結果の出力

ここまでmain関数

最小値とそのインデックスを求める(サブ関数使う) ～サブ関数～

```
int minIndex(int data[100], int num){
    int i;
    int min; /* 最小値のインデックス */
    min = 0;
    for(i=1; i<num; i++){
        if(data[i] < data[min]){
            min = i;
        }
    }
    return min;
}
```

この値がメイン関数内に返されindexに代入される。

昇順への並べ替え(main関数のみ) ～変数の宣言と入力値を配列に格納～

```
int main(){
    int i, j;
    int n;
    int a[100];
    int index;
    int temp;
```

変数、配列の宣言

```
    printf("Array size ?¥n");
    scanf("%d", &n);
    for(i=0; i<n; i++){
        scanf("%d", &(a[i]));
    }
```

入力値を配列に格納

```
    printf("Input: ");
    for(i=0; i<n; i++){
        printf("%d ", a[i]);
    }
    printf("¥n");
```

出力

昇順への並べ替え(main関数のみ) ～並べ替え～

```
for(j=0; j<n-1; j++){
```

```
    index = j;
```

```
    for(i=j+1; i<n; i++){
```

```
        if(a[i] < a[index]){
```

```
            index = i;
```

```
        }
```

```
    } /* j 番目と最小値を交換 */
```

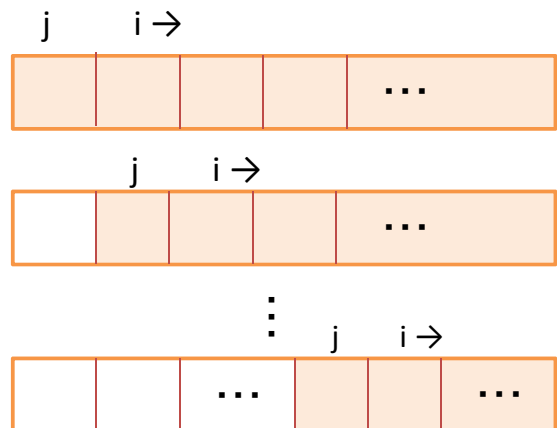
```
    temp = a[j];
```

```
    a[j] = a[index];
```

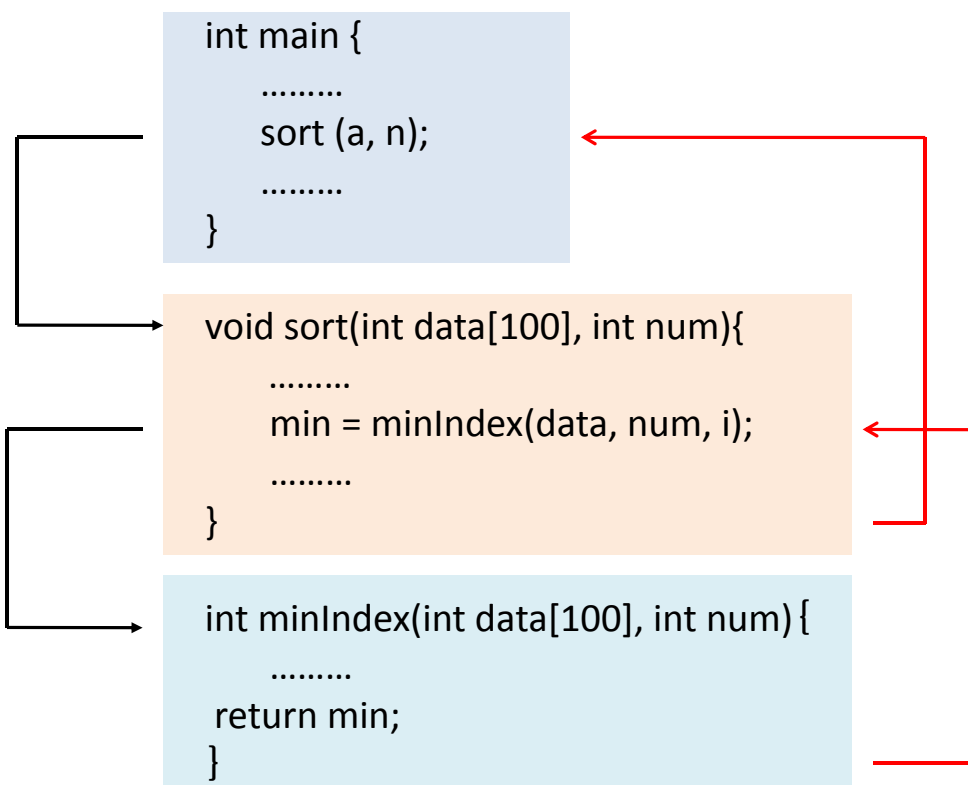
```
    a[index] = temp;
```

```
}
```

j ~ n-1 で a[i] が最小となる
i を index に代入



昇順への並べ替え(サブ関数使う)

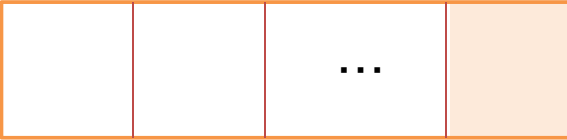


昇順への並べ替え(サブ関数使う) ～minIndex関数の定義～

```
int minIndex (int data[100], int num, int start){
    int i;
    int min;
    min = start;

    for(i=start+1; i<num; i++){

        if(data[i] < data[min]){
            min = i;
        }
    }
    return min;
}
```



The diagram shows a horizontal array of six cells. The first three cells are white and contain an ellipsis (...). The last three cells are shaded light orange and also contain an ellipsis (...). A label 'start' is positioned above the first cell of the shaded region, with a line pointing to its center.

昇順への並べ替え(サブ関数使う) ～sort関数の定義～

```
void sort(int data[100], int num){
    int i;
    int min;
    int temp;
    for(i=0; i<num-1; i++){
        min = minIndex(data, num, i);
        temp = data[i];
        data[i] = data[min];
        data[min] = temp;
    }
}
```