Example 1
Using 1-d Arrays

Young W. Lim

December 13, 2017

Example 1 : using 1-d arrays   Young W. Lim

1 A spreadsheet example using 1-d arrays

[1]

- `int I[SIZE];` student ID (identification) number
- `int K[SIZE];` Korean subject score
- `int E[SIZE];` English subject score
- `int M[SIZE];` Mathematics subject score
- `double A[SIZE];` average score of three subjects

Example 1 : using 1-d arrays　　　　　　　　　　Young W. Lim

1.1　avg3() function

```
//----------------------------------------------------
// Calculating the average of three numbers
//----------------------------------------------------
double avg3(int x, int y, int z)
{
  return (x+y+z) / 3.;
}
```

- takes three integers

- returns their arithmetic average value (a double type)

Example 1 : using 1-d arrays   Young W. Lim

1.2   init_arrays() function

```
//---------------------------------------------------
// Initialize K[], E[], M[] arrays
// by assigning random number grade
//---------------------------------------------------
void init_arrays
(int I[], int K[], int E[], int M[], double A[])
{
  int i;

  // srand(7) makes rand() generate
  // the same random sequence
  // --> easy to debug a program
  srand(7);

  for (i=0; i<SIZE; ++i) {
    I[i] = i+1 + 201600;
    K[i] = rand() % 101;
    E[i] = rand() % 101;
    M[i] = rand() % 101;
    A[i] = avg3(K[i], E[i], M[i]);
  }
}
```

- takes 5 1-d array names

    - the `int` type arrays : (`I`, `K`, `E`, `M`)
    - the `double` type array : (`A`)

- returns nothing

- fill the K, E, and M arrays with random numbers

- `A[i]` is filled with the average value of `K[i]`, `E[i]`, and `M[i]`

- `I[i]` is filled with the expression `i+1+201600`

- all the 5 arrays (I, K, E, M, A) are modified

Example 1 : using 1-d arrays   Young W. Lim

1.3   pr_table() function

```
//---------------------------------------------------
// Print the original table
//---------------------------------------------------
void pr_table
(int I[], int K[], int E[], int M[], double A[])
{
  int i;

  printf("%10s %10s %10s %10s %10s \n", "StID",
          "Korean", "English", "Math", "Average");

  for (i=0; i<SIZE; ++i) {
    printf("%10d %10d %10d %10d %10.2f \n",
            I[i], K[i], E[i], M[i], A[i]);
  }
}
```

- takes 5 1-d array names

  - the `int` type arrays : (I, K, E, M)

  - the `double` type array : (A)

- returns nothing

- print the student ID, test scores of Korean, English, and Mathematics, and their average score row by row

- header

%10s	%10s	%10s	%10s	%10s
"StID"	"Korean"	"English"	"Math"	"Average"

- each row

%10d	%10d	%10d	%10d	%10.2f
I[i]	K[i]	E[i]	M[i]	A[i]

-

StID	Korean	English	Math	Average
201601	17	78	99	64.67
201602	65	87	61	71.00
201603	21	46	91	52.67
201604	100	7	31	46.00
201605	14	63	42	39.67
201606	72	78	68	72.67
201607	9	100	68	59.00
201608	84	20	2	35.33
201609	29	79	62	56.67
201610	69	53	94	72.00

- no array is modified.

Example 1 : using 1-d arrays   Young W. Lim

1.4   Dbubblesort() function

```
//----------------------------------------------------
// Bubble Sort Double Array
//----------------------------------------------------
void DbubbleSort(double a[], int size)
{
  int p, j;
  double tmp;

  for (p=1; p< size; ++p) {
    for (j=0; j< size-1; ++j) {

      if ( a[j] < a[j+1] ) {
      tmp = a[j];
      a[j] = a[j+1];
      a[j+1] = tmp;
      }

    }
  }
}
```

- takes the 1-d array name of the **double** type and the array size

- returns nothing

- there are `size-1` passes : `for (p=1; p<size; ++p) { ... }`

- in each pass, perform the following basic operation
  over `size-1` pairs of adjacent elements : `for (j=0; j<size-1; ++j) { ... }`

  - for a given j, compare the pair `a[j]` and `a[j+1]`
  - if (`a[j] < a[j+1]`) then swap each other
  - thus ensuring `a[j]` is greater than or equal to `a[j+1]`
  - the next pair to be compared will be
    * `a[j+1]` and `a[j+2]` in terms of the old value of j
    * `a[j]` and `a[j+1]` in terms of the incremented value of j

- the current element is moved to the right (increasing index)
  until the smaller element is found

- after `p-1` passes, the array elements are in the increasing order.

- the given array is modified

Example 1 : using 1-d arrays   Young W. Lim

1.5   pr_sorted_table() function

```
//--------------------------------------------------
// Print the Sorted Table
//--------------------------------------------------
void pr_sorted_table
(int I[], int K[], int E[], int M[], double A[])
{
  int i, j;
  double B[SIZE]; // Backup Array for Sorting

  for (i=0; i<SIZE; ++i) B[i] = A[i];

  //...................
  DbubbleSort(B, SIZE);
  //...................

  printf("\n\nSorted on a student's average\n\n");
  printf("%10s %10s %10s %10s %10s \n", "StID",
         "Korean", "English", "Math", "Average");

  for (i=0; i<SIZE; ++i) {
    for (j=0; j<SIZE; ++j) if (B[i] == A[j]) break;
    printf("%10d %10d %10d %10d %10.2f \n",
    I[j], K[j], E[j], M[j], A[j]);
  }
}
```

- takes 5 1-d array names

    – the `int` type arrays : (`I`, `K`, `E`, `M`)

    – the `double` type array : (`A`)

- returns nothing

- initially, all the 1-d arrays are sorted by the student ID

- copy `A` array into `B` array

- sort `B` array : `DbubbleSort(B, SIZE);`

- only `B` array are in the increasing order of the average score

- for each `B[i]` : `for (i=0; i<SIZE; ++i)`

    – find the index j such that `A[j]=B[i]` : `for (j=0; j<SIZE; ++j)`

    – print the (j+1)-th row of the original table

    –

%10d	%10d	%10d	%10d	%10.2f
I[j]	K[j]	E[j]	M[j]	A[j]

- no array is modified

Example 1 : using 1-d arrays   Young W. Lim

1.6 Avg() function

```
//---------------------------------------------------
// Average over Integer Array
//---------------------------------------------------
double Avg(int X[], int n) {
  int i; double S=0.0;

  for (i=0; i<n; ++i) S+= X[i];
  return S/n;
}
```

- takes a 1-d `int` array name and its size

- returns its arithmetic average value (a double type)

- $\frac{1}{n} \sum_{i=0}^{n-1} X[i]$

Example 1 : using 1-d arrays   Young W. Lim

1.7 DAvg() function

```
//---------------------------------------------------
// Average over Double Array
//---------------------------------------------------
double DAvg(double Y[], int n) {
  int i; double S=0.0;

  for (i=0; i<n; ++i) S+= Y[i];
  return S/n;
}
```

- takes a 1-d **double** array name and its size

- returns its arithmetic average value (a double type)

- $\frac{1}{n} \sum_{i=0}^{n-1} Y[i]$

Example 1 : using 1-d arrays   Young W. Lim

1.8   pr_averages() function

```
//---------------------------------------------------
// Print the Averages
//---------------------------------------------------
void pr_averages(int K[], int E[], int M[], double A[])
{
  double A1 = Avg(K, SIZE);
  double A2 = Avg(E, SIZE);
  double A3 = Avg(M, SIZE);
  double A4 = DAvg(A, SIZE);
  printf("%10s %10.2f %10.2f %10.2f %10.2f \n",
         "Average", A1, A2, A3, A4);
}
```

- takes 5 1-d array names

    - the `int` type arrays : (`I, K, E, M`)
    - the `double` type array : (`A`)

- returns nothing

- `A1` : the average score of the Korean subject

- `A2` : the average score of the English subject

- `A3` : the average score of the Mathematics subject

- `A4` : the average score of each student's average score

-
%10s	%10.2f	%10.2f	%10.2f	%10.2f
"Average"	A1	A2	A3	A4

- no array is modified

Example 1 : using 1-d arrays   Young W. Lim

1.9   main() function

```
int main(void) {
  int I[SIZE]; // ID of a student
  int K[SIZE]; // Grade of Korean Class
  int E[SIZE]; // Grade of English Class
  int M[SIZE]; // Grade of Math Class
  double A[SIZE]; // Average of a student

  init_arrays(I, K, E, M, A);

  pr_table(I, K, E, M, A);

  pr_sorted_table(I, K, E, M, A);

  pr_averages(K, E, M, A);

}
```

- declare 5 1-d arrays
    - `int I[SIZE];`
    - `int K[SIZE];`
    - `int E[SIZE];`
    - `int M[SIZE];`
    - `double A[SIZE];`
- call `init_arrays()` function
- call `pr_table()` function
- call `pr_sorted_table()` function
- call `pr_averages()` function

Example 1 : using 1-d arrays   Young W. Lim

1.10   When students have the same average

test cases   the previous codes do not work in such case.

```c
/-----------------------------------------------
// Initialize K[], E[], M[] arrays
// by assigning random number grade
//-----------------------------------------------
void init_arrays
(int I[], int K[], int E[], int M[], double A[])
{
  int i;

  // srand(7) makes rand() generate
  // the same random sequence
  // --> easy to debug a program
  srand(7);

  for (i=0; i<SIZE; ++i) {
    I[i] = i+1 + 201600;
    K[i] = rand() % 101;
    E[i] = rand() % 101;
    M[i] = rand() % 101;
    A[i] = avg3(K[i], E[i], M[i]);
  }

    K[3] = K[2];
    E[3] = E[2];
    M[3] = M[2];
    A[3] = A[2];

    K[5] = K[2];
    E[5] = E[2];
    M[5] = M[2];
    A[5] = A[2];

    K[7] = K[2];
    E[7] = E[2];
    M[7] = M[2];
    A[7] = A[2];

}
```

Example 1 : using 1-d arrays   Young W. Lim

- rows having index values of 2, 3, 5, 7 have the same score.

- only one student ID (201603) is repeated.

```
   StID     Korean    English      Math    Average
 201610         69         53        94      72.00
 201602         65         87        61      71.00
 201601         17         78        99      64.67
 201607          9        100        68      59.00
 201609         29         79        62      56.67
 201603         21         46        91      52.67
 201603         21         46        91      52.67
 201603         21         46        91      52.67
 201603         21         46        91      52.67
 201605         14         63        42      39.67
Average      28.70      64.40     79.00      57.37
```

Example 1 : using 1-d arrays   Young W. Lim

1.11   Handling the same average cases - Method 1

using extra memory

- using index array : `int C[SIZE];`

- swap both average score and its index : `DbubbleSort()`

```
//----------------------------------------------
// Bubble Sort Double Array
//----------------------------------------------
void DbubbleSort(double b[], int c[], int size)
{
  int p, j, t;
  double tmp;

  for (p=1; p< size; ++p) {
    for (j=0; j< size-1; ++j) {

      if ( b[j] < b[j+1] ) {
        tmp = b[j];
        b[j] = b[j+1];
        b[j+1] = tmp;

        t = c[j];
        c[j] = c[j+1];
        c[j+1] = t;
      }

    }
  }
}
```

Example 1 : using 1-d arrays   Young W. Lim

```
//-------------------------------------------------
// Print the Sorted Table
//-------------------------------------------------
void pr_sorted_table
(int I[], int K[], int E[], int M[], double A[])
{
  int i, j;
  double B[SIZE]; // Backup Array for Sorting
  int    C[SIZE]; // index array

  for (i=0; i<SIZE; ++i) {
    B[i] = A[i];
    C[i] = i;
  }

  //...................
  DbubbleSort(B, C, SIZE);
  //...................

  printf("\n\nSorted on a student's average\n\n");
  printf("%10s %10s %10s %10s %10s \n", "StID",
         "Korean", "English", "Math", "Average");

  for (i=0; i<SIZE; ++i) {
    j = C[i];

    printf("%10d %10d %10d %10d %10.2f \n",
    I[j], K[j], E[j], M[j], A[j]);
  }
}
```

Example 1 : using 1-d arrays   Young W. Lim

1.12   Handling the same average cases - Method 2

using extra computation

- counting the same average cases: `int cnt;`

- search all indices for the same average : `j`

```
//--------------------------------------------------
// Print the Sorted Table
//--------------------------------------------------
void pr_sorted_table
(int I[], int K[], int E[], int M[], double A[])
{
  int i, j;
  double B[SIZE]; // Backup Array for Sorting
  int    cnt;

  for (i=0; i<SIZE; ++i) B[i] = A[i];

  //...................
  DbubbleSort(B, SIZE);
  //...................

  printf("\n\nSorted on a student's average\n\n");
  printf("%10s %10s %10s %10s %10s \n", "StID",
         "Korean", "English", "Math", "Average");

  for (i=0; i<SIZE; ++i) {
    cnt=1;
    while (B[i-cnt]==B[i]) cnt++;

    j=0;
    while (cnt>0) {
      cnt--;
      while (A[j]!=B[i]) j++;
      if (cnt > 0) j++;
    }

    printf("%10d %10d %10d %10d %10.2f \n",
    I[j], K[j], E[j], M[j], A[j]);
  }
}
```

Example 1 : using 1-d arrays   Young W. Lim

References

[1] C Programming Exercieses, `https://cprogramex.wordpress.com/`