Example 1

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice.

Young Won Lim
11/9/17

Using 1-d Arrays

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 10
```

Using 1-d Arrays

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

Using 1-d Arrays

```
//-----------------------------------------------
// 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]);
   }
}
```

Using 1-d Arrays

```
//-----------------------------------------------
// 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]);

   }
}
```

Using 1-d Arrays

```
//-----------------------------------------------
// 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;
      }
    }
  }
}
```

Using 1-d Arrays

```
//-----------------------------------------------
// 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]);
   }
}
```

Using 1-d Arrays

```
//-----------------------------------------------
// 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;
}
```

Using 1-d Arrays

```
//-----------------------------------------------
// 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;
}
```

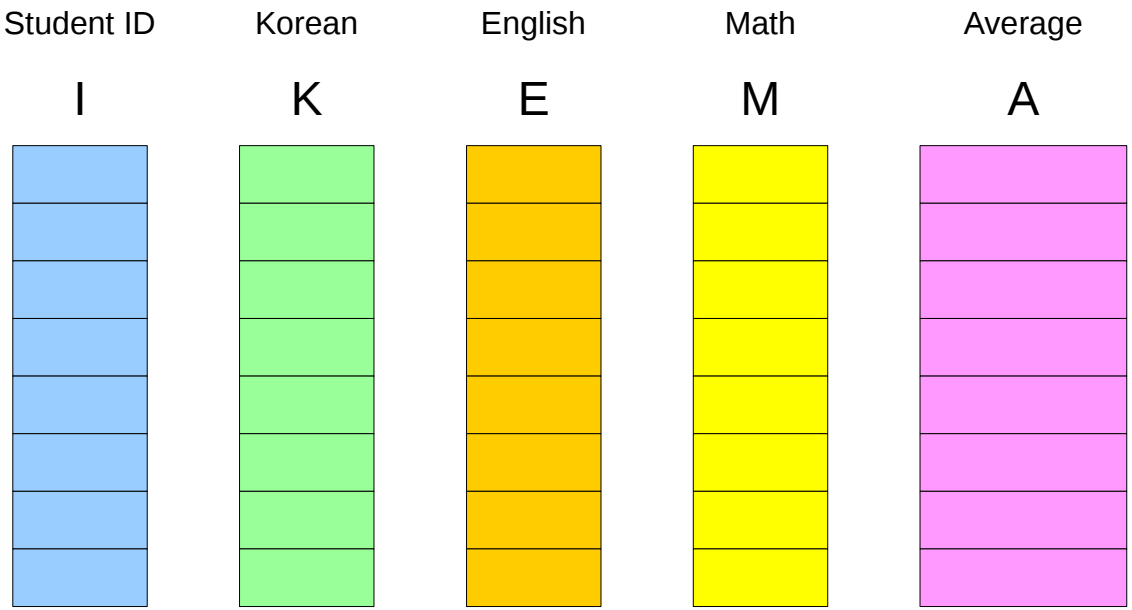
Using 1-d Arrays

```
//-----------------------------------------------
// 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);
}
```

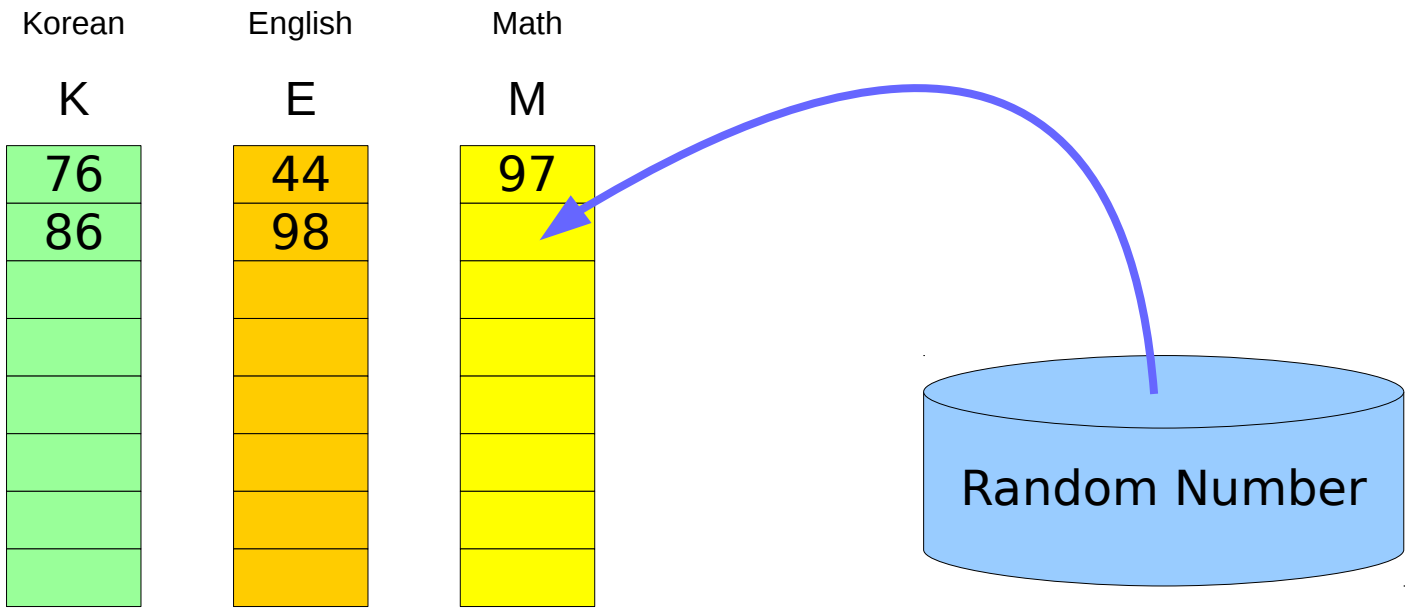
Using 1-d Arrays

```
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);
}
```

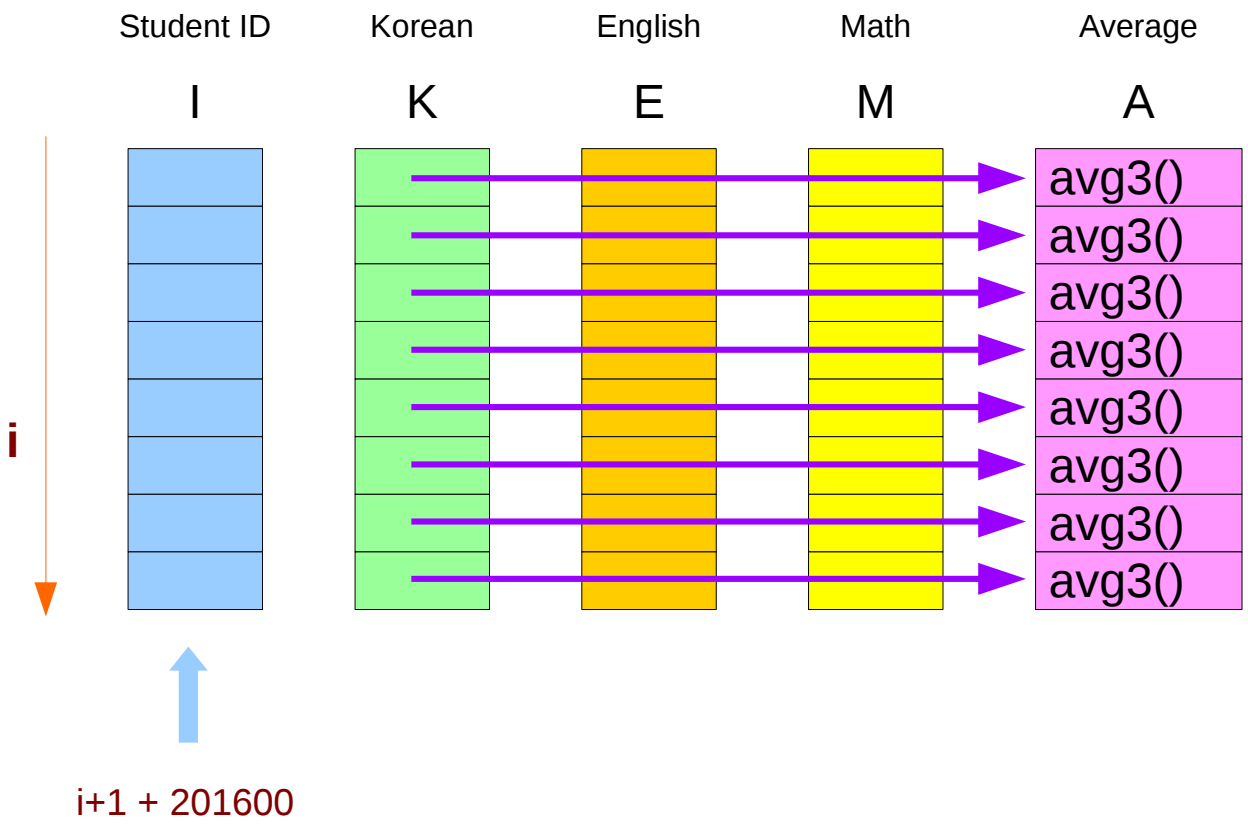
Using 1-d Arrays



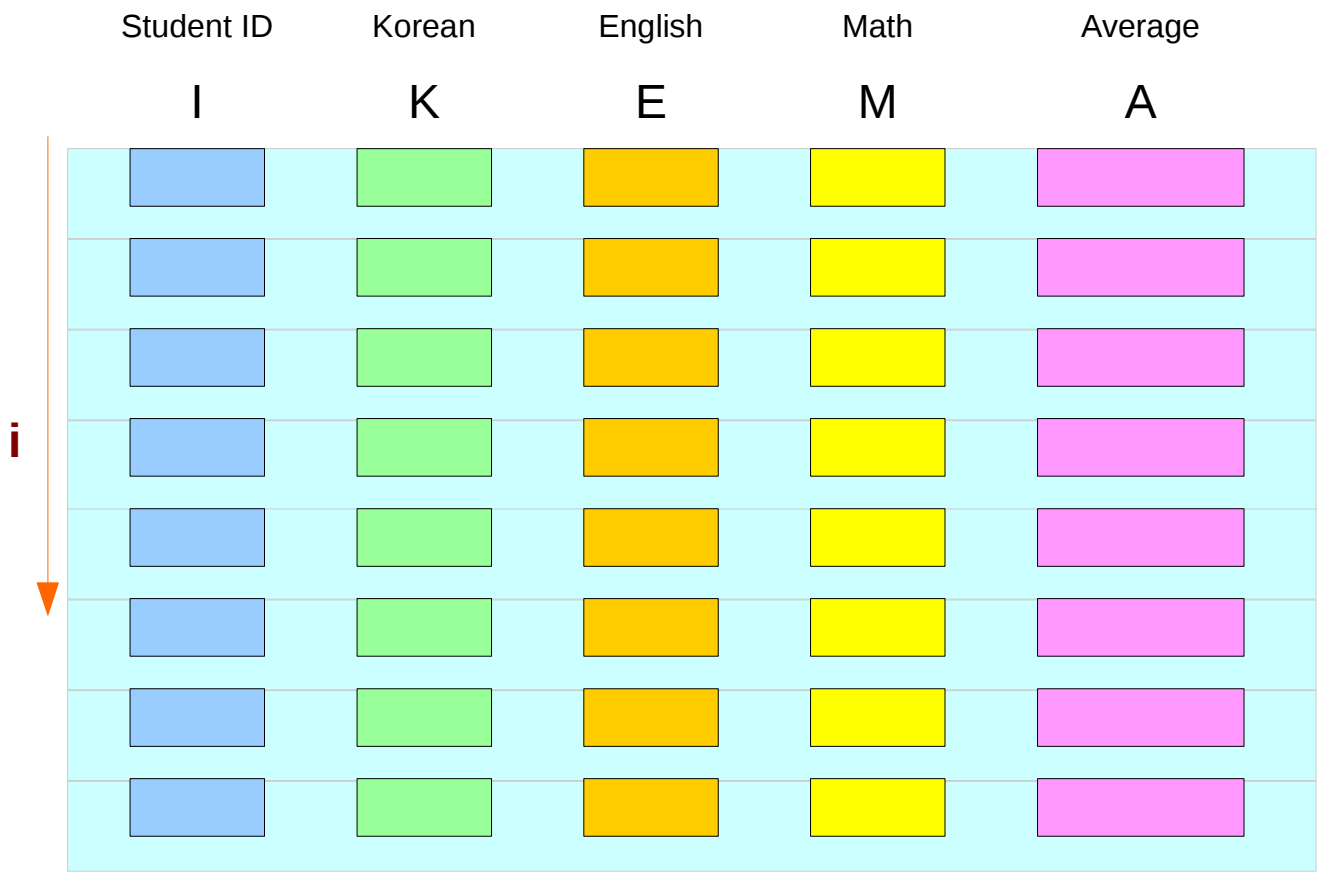
init_arrays() - filling grades



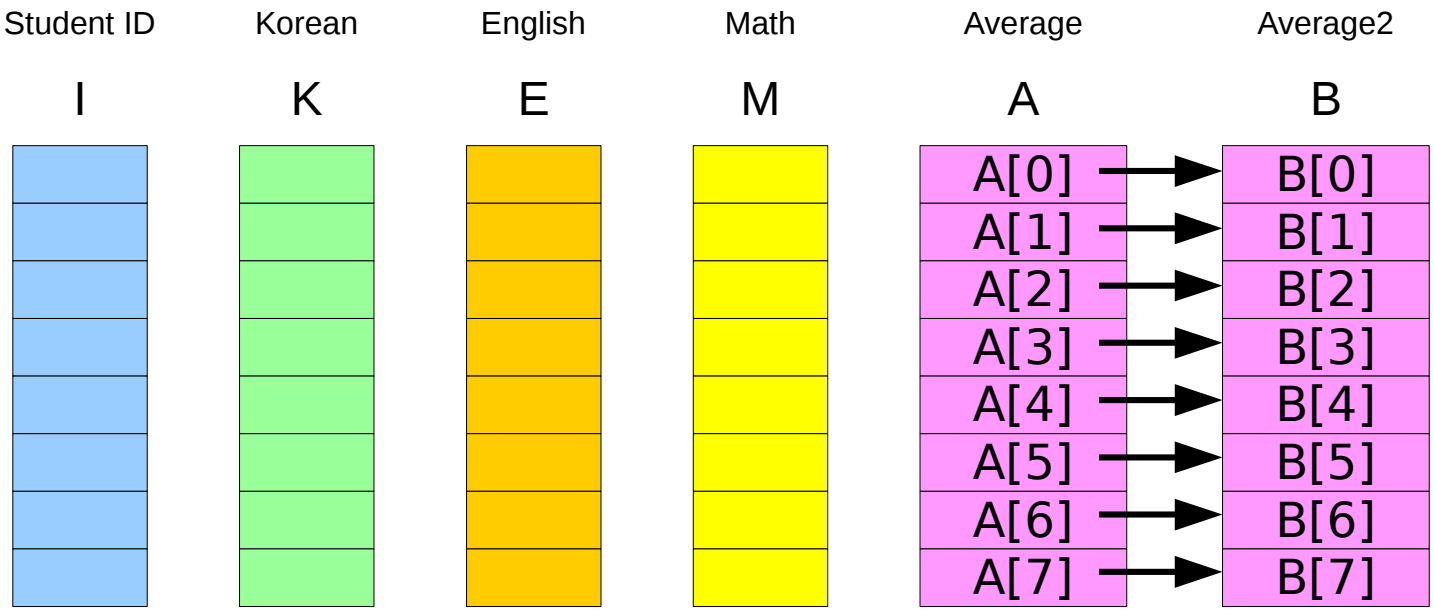
init_arrays() - computing averages



pr_table()

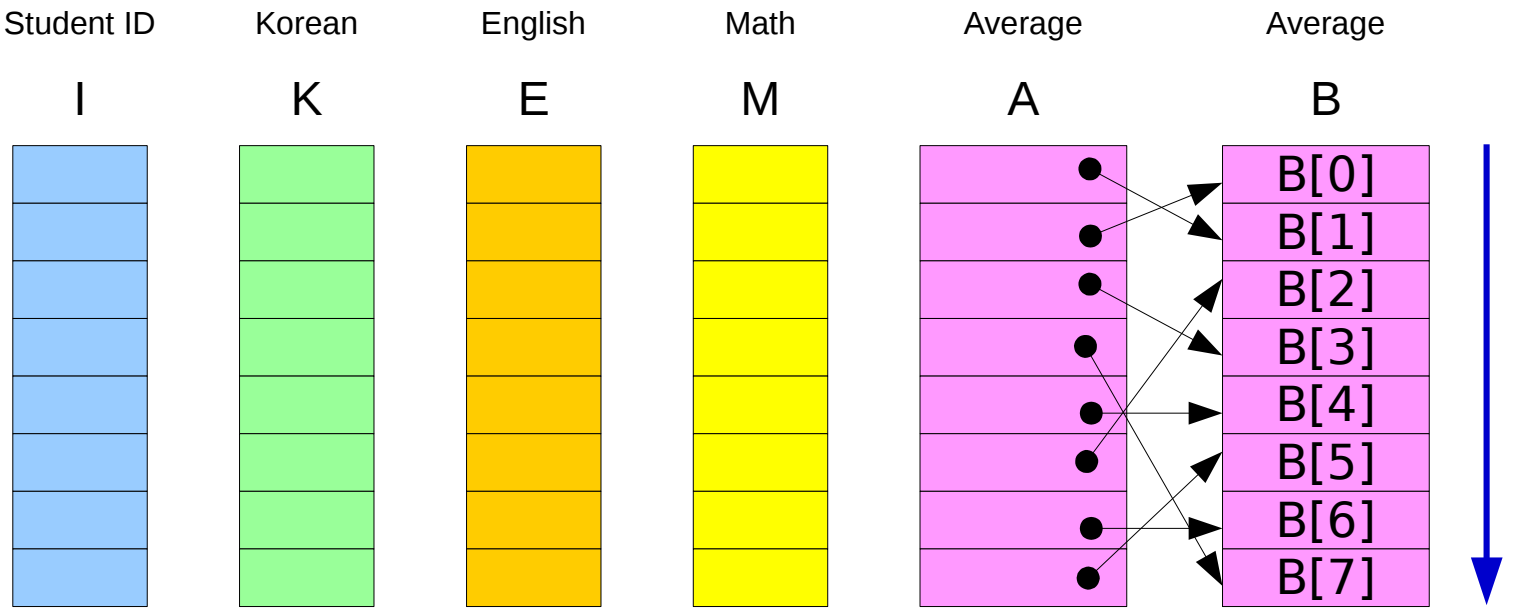


pr_sorted_table – copying A to B



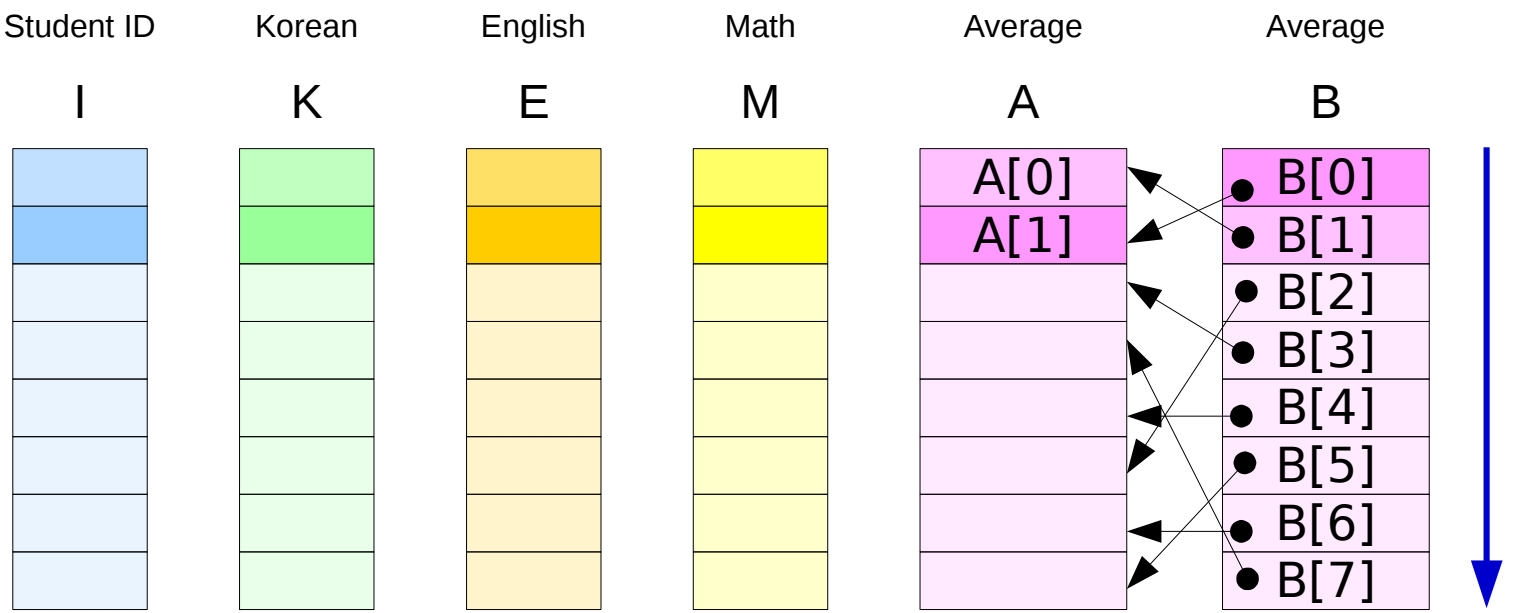
First, copy
A[i] into B[i]

pr_sorted_table – sorting B



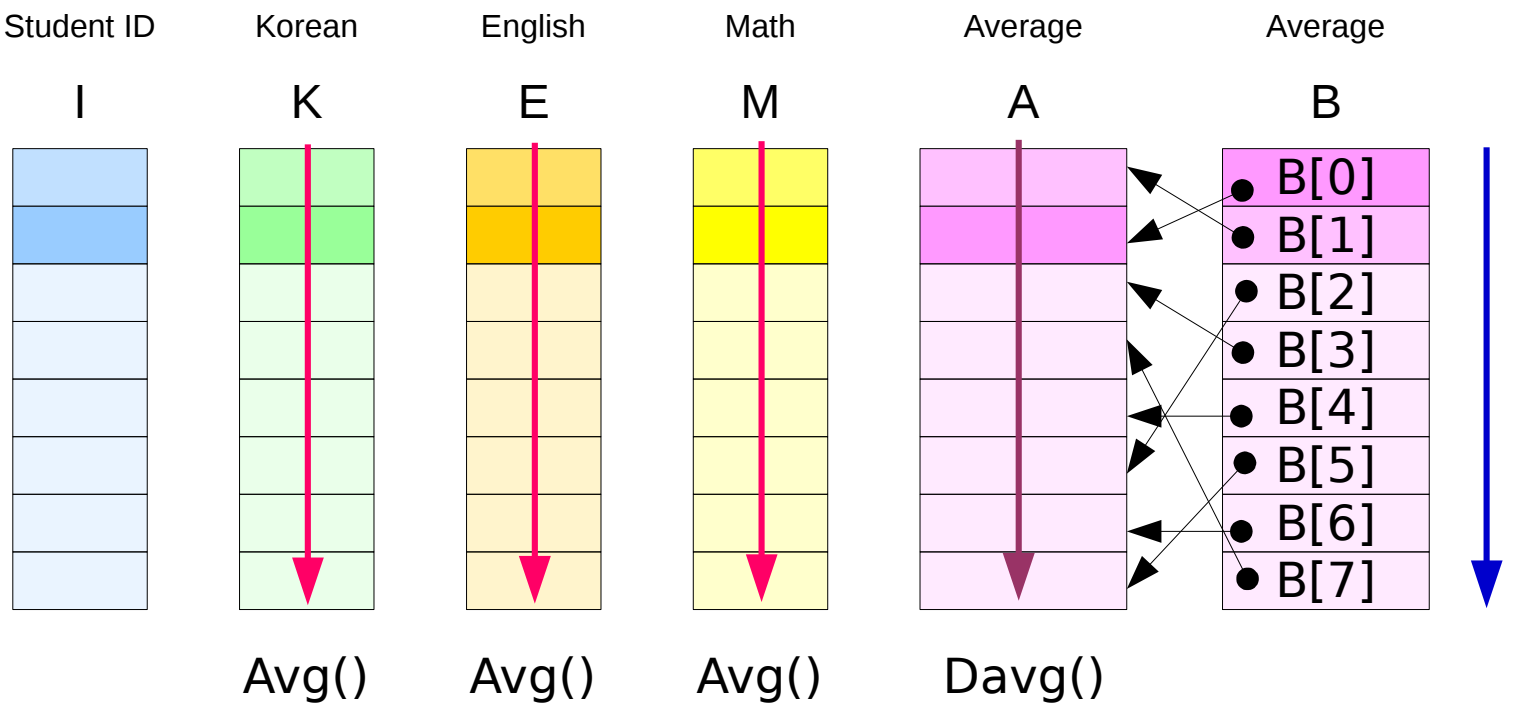
after DbubbleSort()
B[i] > B[i]
A, B: different order

pr_sorted_table – printing by B



Search A[j] = B[i]

pr_averages


Student ID
Korean
English
Math
Average
Average
I
K
E
M
A
B
B[0]
B[1]
B[2]
B[3]
B[4]
B[5]
B[6]
B[7]
Avg()
Avg()
Avg()
Davg()

References

[1] Essential C, Nick Parlante
[2] Efficient C Programming, Mark A. Weiss
[3] C A Reference Manual, Samuel P. Harbison & Guy L. Steele Jr.
[4] C Language Express, I. K. Chun
[5] cprogramex.wordpress.com