

Example 2

Copyright (c) 2010 - 2017 Young W. Lim.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

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

This document was produced by using OpenOffice.

Using 2-d Arrays

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

avg3() definition

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

init_arrays() definition

```
//-----  
// Initialize X[4][SIZE] arrays  
// by assigning random number grade  
//-----  
void init_arrays (int X[][SIZE], 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) {  
        X[0][i] = i+1 + 201600; // I  
        X[1][i] = rand() % 101; // K  
        X[2][i] = rand() % 101; // E  
        X[3][i] = rand() % 101; // M  
        A[i] = avg3(X[1][i], X[2][i], X[3][i]);  
    }  
}
```

pr_table() definition

```
//-----  
// Print the original table  
//-----  
void pr_table (int X[][SIZE], double A[])  
{  
    int i;  
  
    printf("%10s %10s %10s %10s %10s \n", "StID",  
        "Korean", "Enlgish", "Math", "Average");  
  
    for (i=0; i<SIZE; ++i) {  
        printf("%10d %10d %10d %10d %10.2f \n",  
            X[0][i], X[1][i], X[2][i], X[3][i], A[i]);  
    }  
}
```

DbubbleSort() definition

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

pr_sorted_table() definition

```
//-----  
// Print the Sorted Table  
//-----  
void pr_sorted_table (int X[][SIZE], 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", "Enlgish", "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",  
            X[0][j], X[1][j], X[2][j], X[3][j], A[j]);  
    }  
}
```


Avg() definition

```
//-----  
// Average over Integer Array  
//-----  
double Avg(int M[], int n) {  
    int i; double S=0.0;  
  
    for (i=0; i<n; ++i) S+= M[i];  
    return S/n;  
}
```

DAvg() definition

```
//-----  
// Average over Doubl Array  
//-----  
double DAvg(double N[], int n) {  
    int i; double S=0.0;  
  
    for (i=0; i<n; ++i) S+= N[i];  
    return S/n;  
}
```

pr_average definition

```
//-----  
// Print the Averages  
//-----  
void pr_averages(int X[][SIZE], double A[]) {  
    double A1 = Avg(X[1], SIZE);  
    double A2 = Avg(X[2], SIZE);  
    double A3 = Avg(X[3], SIZE);  
    double A4 = DAvg(A, SIZE);  
  
    printf("%10s %10.2f %10.2f %10.2f %10.2f \n",  
          "Average", A1, A2, A3, A4);  
}
```

main() definition

```
//=====
// main
//=====
int main(void) {
    // X[0][SIZE] --> I[SIZE]; // ID of a student
    // X[1][SIZE] --> K[SIZE]; // Grade of Korean
    // X[2][SIZE] --> E[SIZE]; // Grade of English
    // X[3][SIZE] --> M[SIZE]; // Grade of Math

    int X[4][SIZE];
    double A[SIZE]; // Average of a student

    init_arrays(X, A);

    pr_table(X, A);

    pr_sorted_table(X, A);

    pr_averages(X, A);
}
```

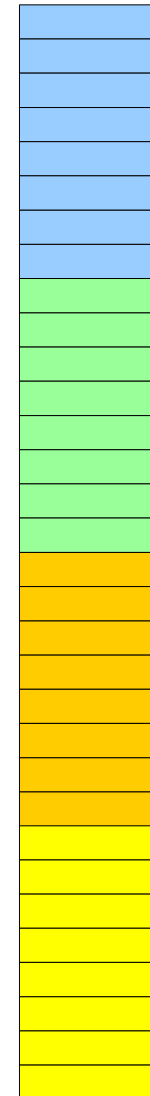
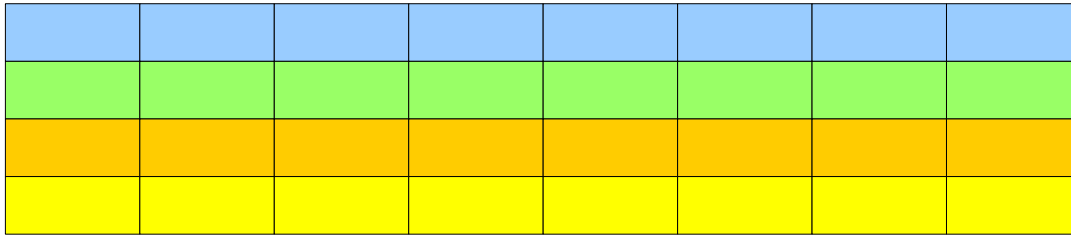
2-d Array Definition

Student ID									
Korean									
English									
Math									
Average									

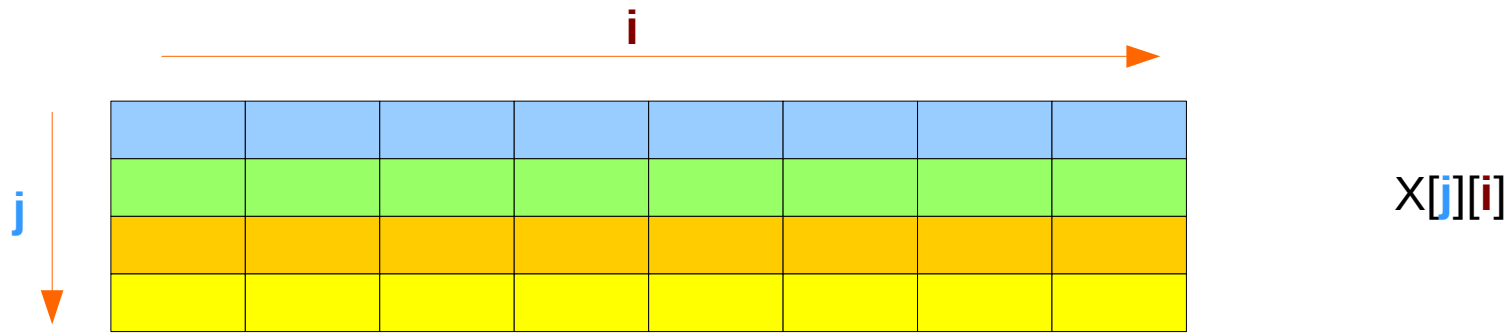
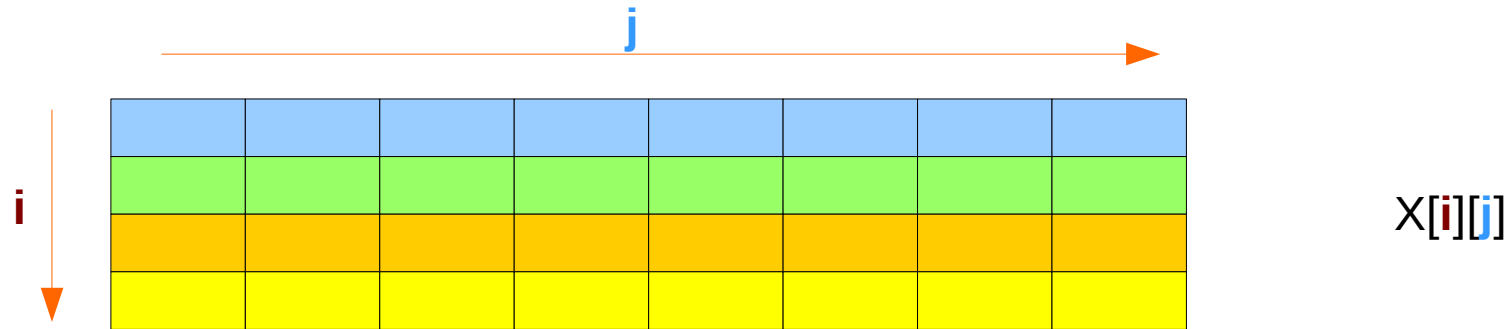
$X[i][j]$

$A[j]$

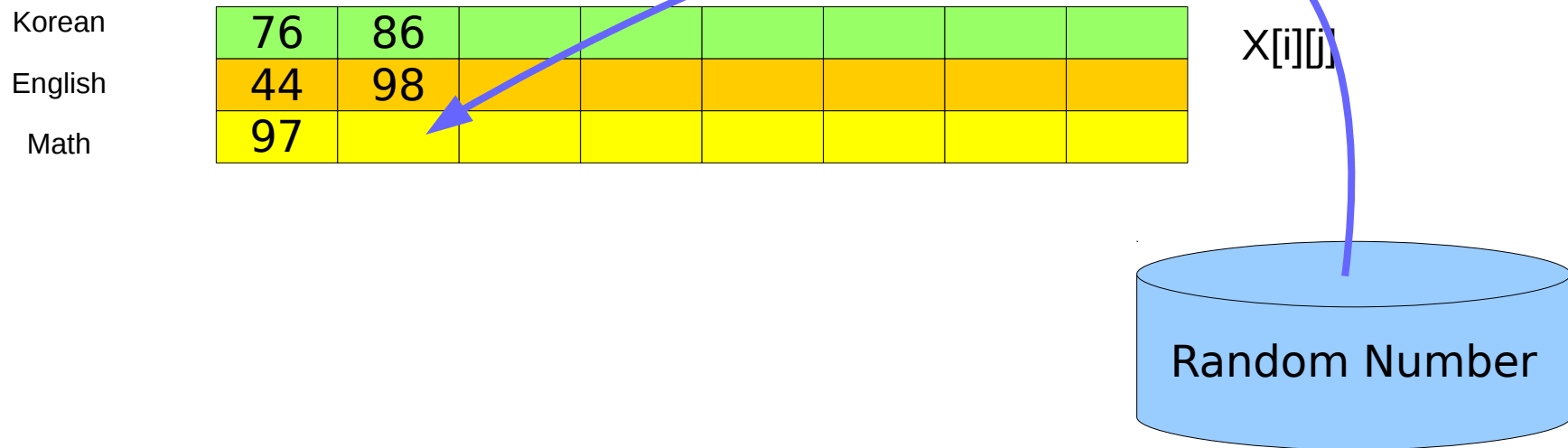
Row-by-Row : Row Major Ordering



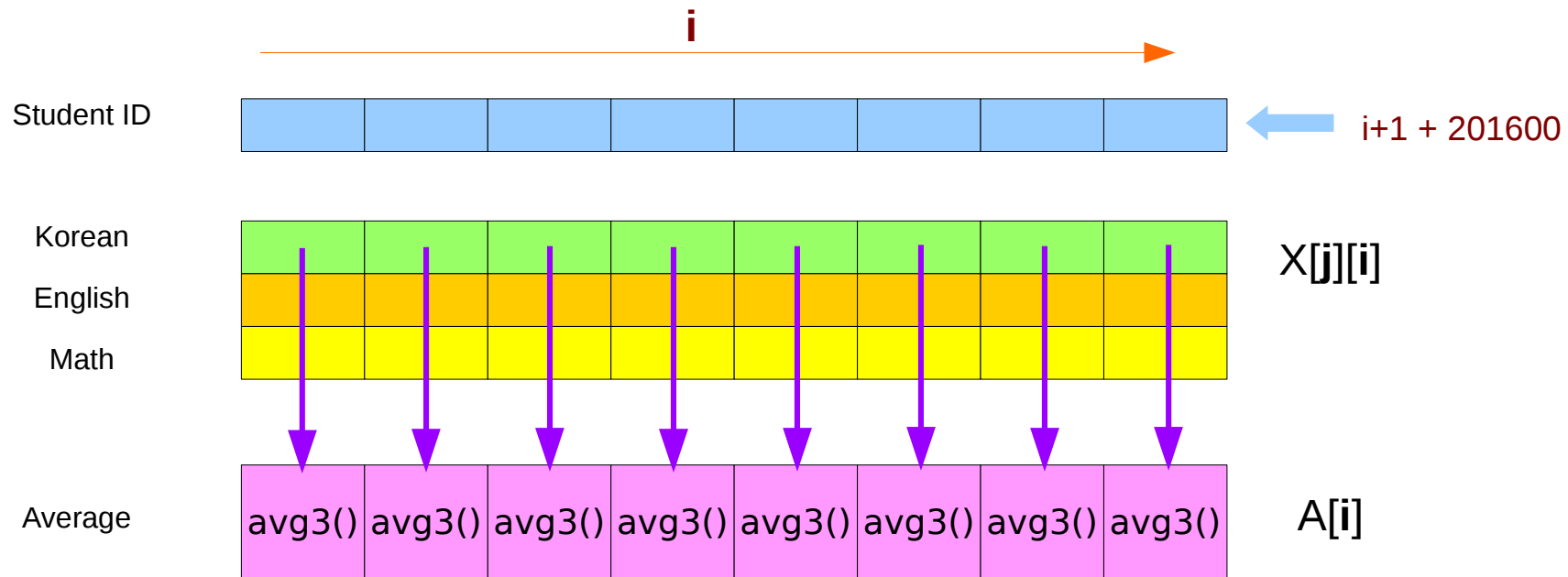
Row and Column Index



init_arrays() - filling grades



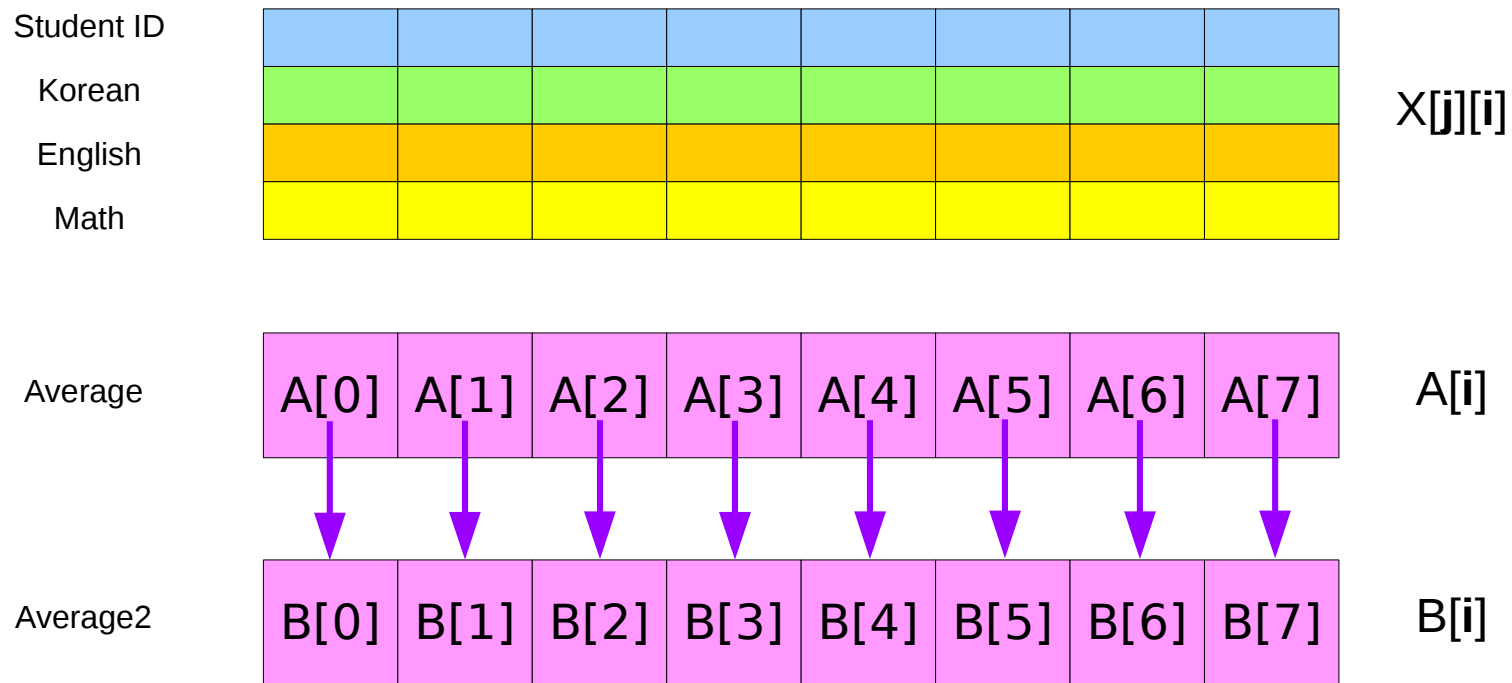
init_arrays() - computing averages



pr_table()

	Student ID	Korean	English	Math	Average
	I	K	E	M	A
i					

pr_sorted_table - copying A to B

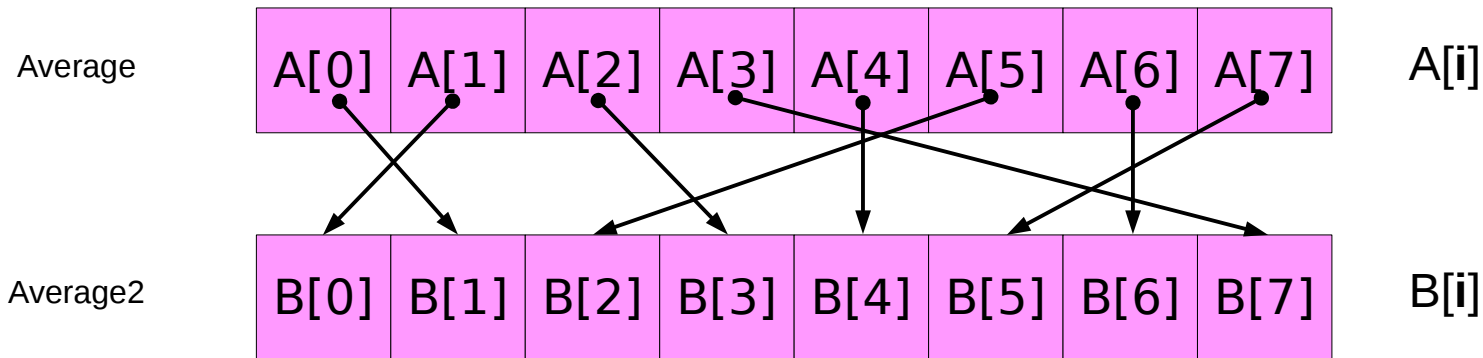


First, copy
 $A[i]$ into $B[i]$

pr_sorted_table - sorting B

Student ID							
Korean							
English							
Math							

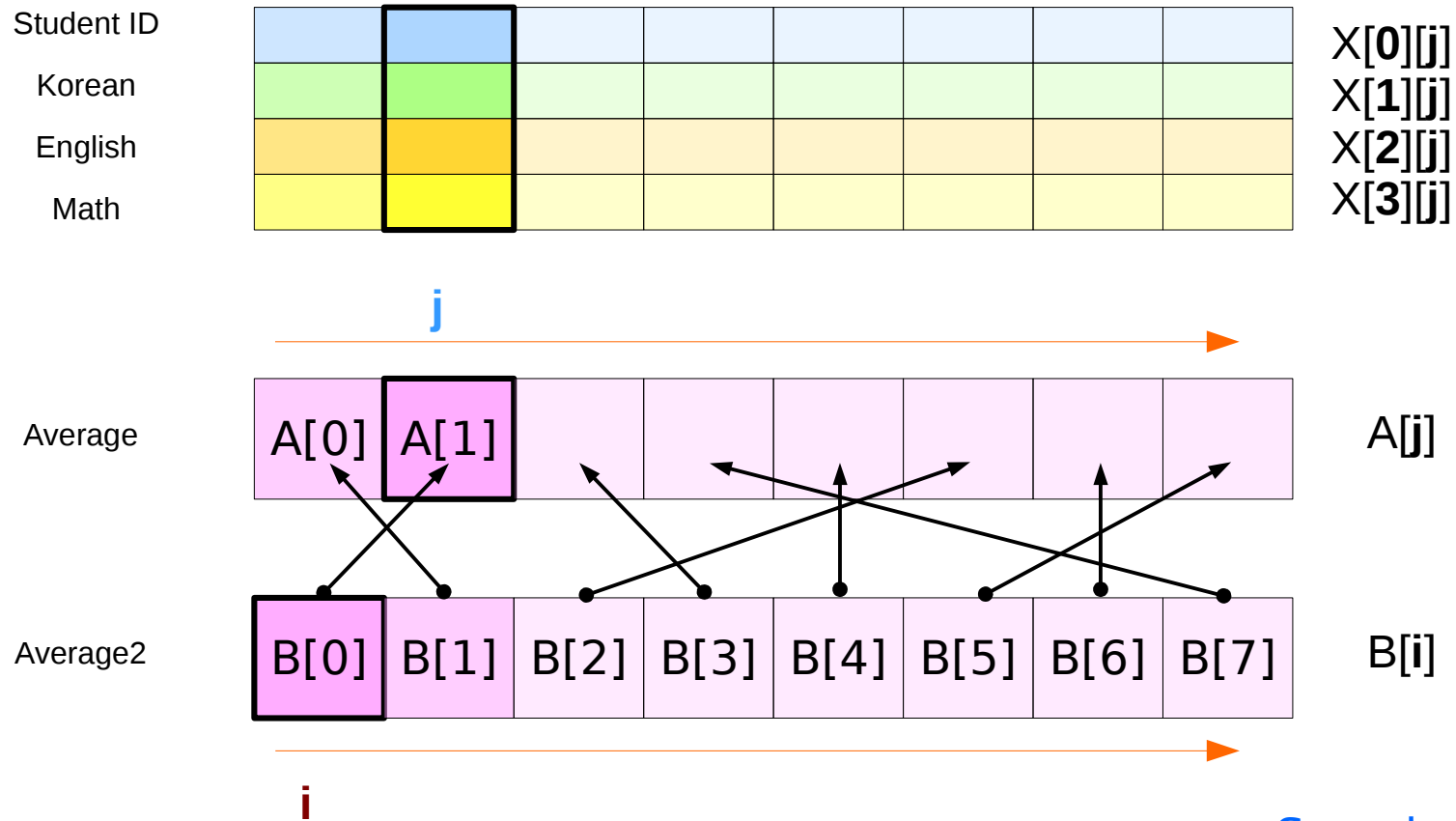
$X[j][i]$



Decreasing

after DbubbleSort()
 $B[j] > B[j+1]$
A, B: different order

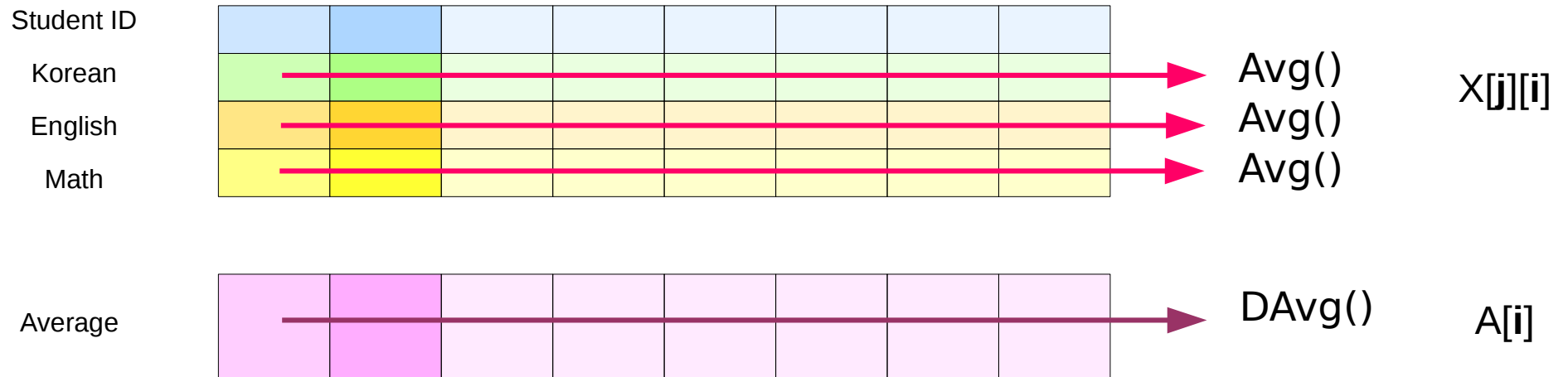
pr_sorted_table - printing by B



Search $A[j] = B[i]$

Assume that two averages have always different values

pr_averages



Function Prototypes and Function Calls

```
double    avg3          (int x, int y, int z) ;
void      init_arrays  (int X[][SIZE], double A[]);
void      pr_table     (int X[][SIZE], double A[]);
void      DbubbleSort  (double a[], int size);
void      pr_sorted_table (int X[][SIZE], double A[]);
double    Avg          (int X[], int n);
double    Davg         (double Y[], int n);
void      pr_averages  (int X[][SIZE], double A[]);
```

```
init_arrays(X, A); ..... in main()
    A[i] = avg3(K[i], E[i], M[i]); ..... in init_arrays()
```

```
pr_table(X, A); ..... in main()
```

```
pr_sorted_table(X, A); ..... in main()
    DbubbleSort(B, SIZE); ..... in pr_sorted_table()
```

```
pr_averages(X, A); ..... in main()
    double A1 = Avg(X[1], SIZE); ..... in pr_averages()
    double A2 = Avg(X[2], SIZE); ..... in pr_averages()
    double A3 = Avg(X[3], SIZE); ..... in pr_averages()
    double A4 = DAvg(A, SIZE); ..... in pr_averages()
```

2-d Array Definitions

```
int main(void) {  
    // X[0][SIZE] --> I[SIZE]; // ID of a student  
    // X[1][SIZE] --> K[SIZE]; // Grade of Korean  
    // X[2][SIZE] --> E[SIZE]; // Grade of English  
    // X[3][SIZE] --> M[SIZE]; // Grade of Math  
  
    int X[4][SIZE];  
    double A[SIZE]; // Average of a student  
  
    init_arrays(X, A);  
  
    pr_table(X, A);  
  
    pr_sorted_table(X, A);  
  
    pr_averages(X, A);  
}
```

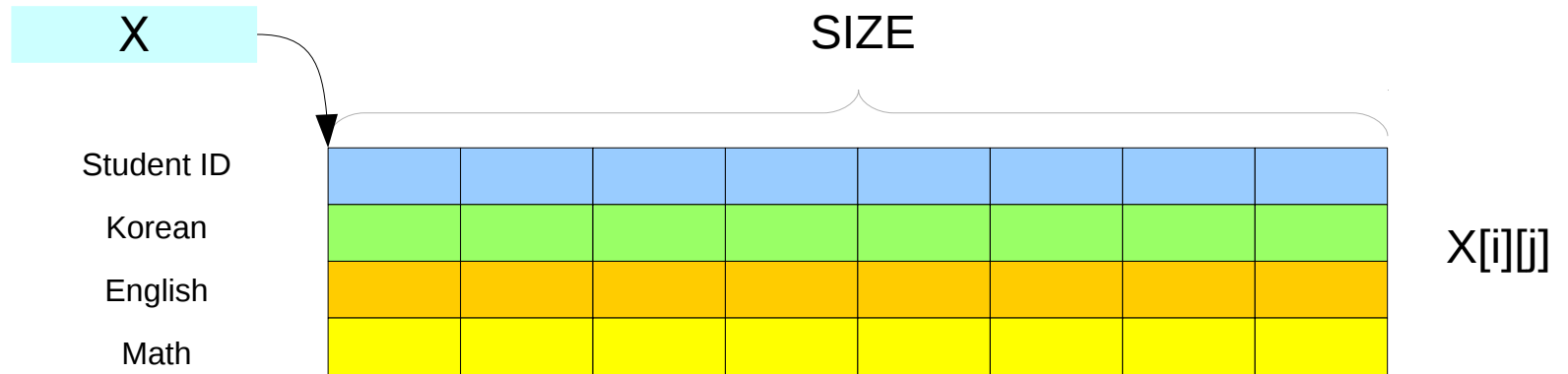

2-d Array Definition

Student ID									
Korean									
English									
Math									
Average									

$X[i][j]$

$A[j]$

int X[][SIZE] : Formal Parameter



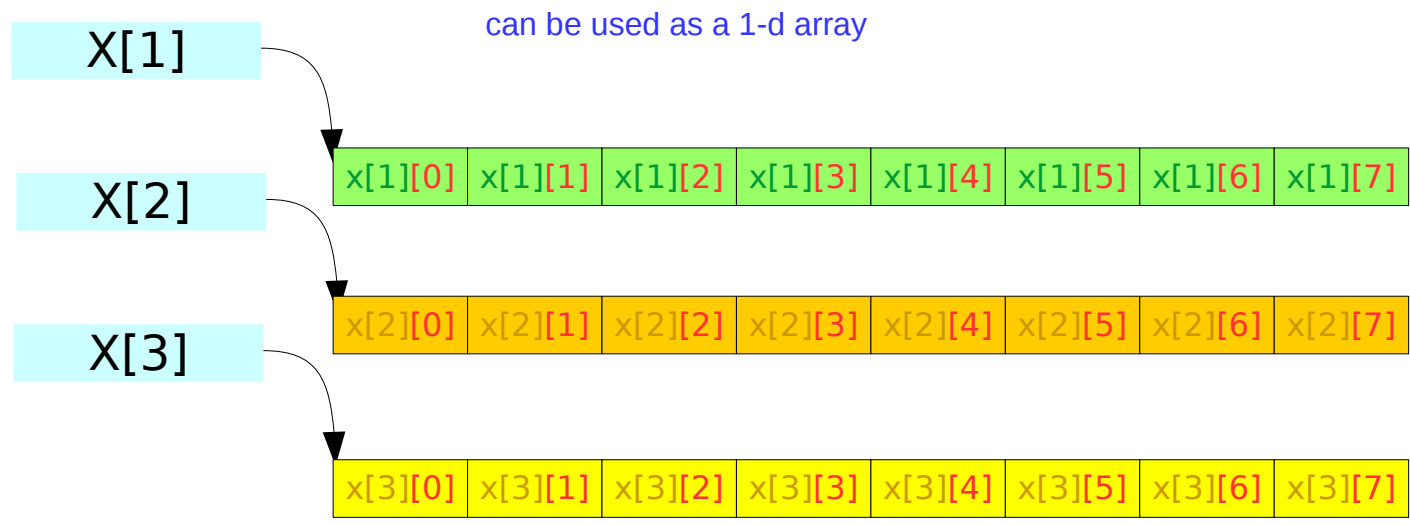
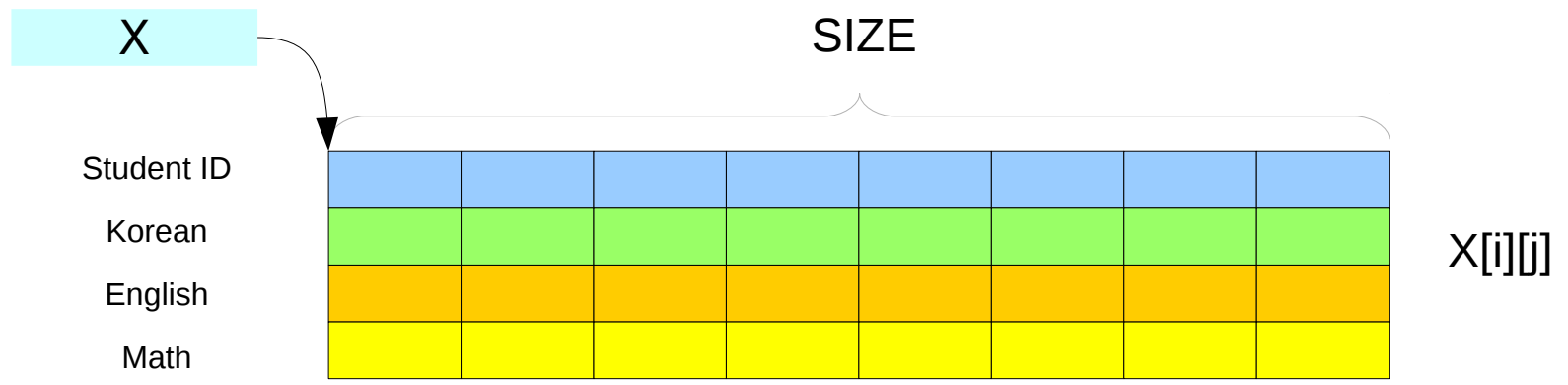
```
int X[4][SIZE];
```

X

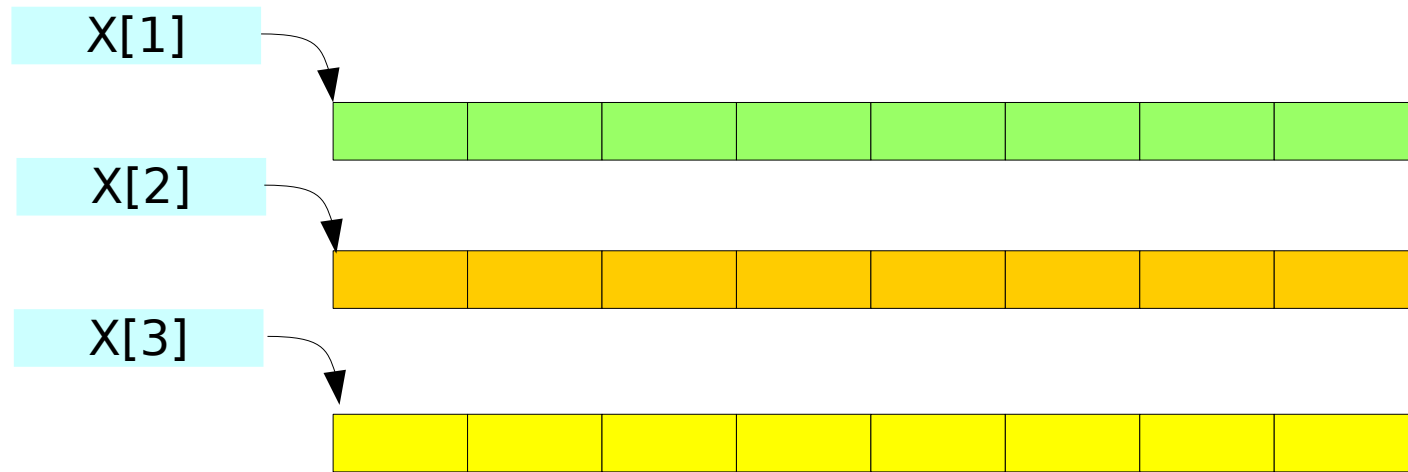


```
Init_arrays( int X[ ][SIZE], ... )  
pr_table( int X[ ][SIZE], ... )  
pr_sorted_table( int X[ ][SIZE], ... )  
pr_averages( int X[ ][SIZE], ... )
```

X[1], X[2], X[3] : 2nd, 3rd, 4th rows



int X[] : Formal Parameter



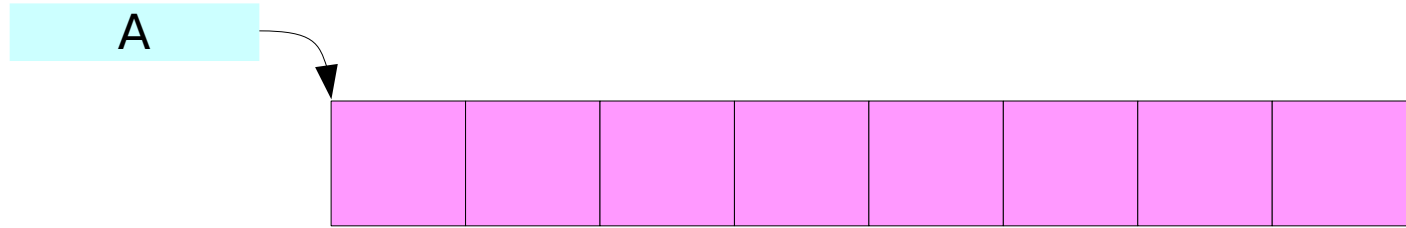
int X[4][SIZE];

X[1]
X[2]
X[3]



Avg(int X[], ...)

double Y[] : Formal Parameter



`int A[SIZE];`

A



`DAvg(double Y[], ...)`

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