

Example 3 : using a structure array

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.

Stype structure definition

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

```
struct Stype {
    int I;
    int K;
    int E;
    int M;
    double A;
};
```

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 S[SIZE] arrays  
// by assigning random number grade  
//-----  
void init_arrays (struct Stype T[])  
{  
    int i;  
  
    // srand(7) makes rand() generate  
    // the same random sequence  
    // --> easy to debug a program  
    srand(7);  
  
    for (i=0; i<SIZE; ++i) {  
        T[i].I = i+1 + 201600; // I  
        T[i].K = rand() % 101; // K  
        T[i].E = rand() % 101; // E  
        T[i].M = rand() % 101; // M  
        T[i].A = avg3(T[i].K, T[i].E, T[i].M);  
    }  
}
```

pr_table() definition

```
//-----  
// Print the original table  
//-----  
void pr_table (struct Stype T[])  
{  
    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",  
            T[i].I, T[i].K, T[i].E, T[i].M, T[i].A);  
    }  
}
```

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 (struct Stype T[])  
{  
    int i, j;  
    double B[SIZE]; // Backup Array for Sorting  
  
    for (i=0; i<SIZE; ++i) B[i] = T[i].A;  
  
    //.....  
    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] == T[j].A) break;  
  
        printf("%10d %10d %10d %10d %10.2f \n",  
            T[j].I, T[j].K, T[j].E, T[j].M, T[j].A);  
    }  
}
```


Avg() definition

```
//-----  
// Average over Integer Array  
//-----  
double Avg(struct Stype T[], int n) {  
    int i; double S=0.0;  
  
    // n is used to select  
    // Korean (n=0)  
    // English (n=1)  
    // Math (n=2)  
    // Avg (n=3)  
  
    for (i=0; i<SIZE; ++i) {  
        switch (n) {  
            case 0 : S += T[i].K; break;  
            case 1 : S += T[i].E; break;  
            case 2 : S += T[i].M; break;  
            case 3 : S += T[i].A; break;  
            defalut: S = 0; break;  
        }  
    }  
    return S/SIZE;  
}
```

pr_averages() definition

```
//-----  
// Print the Averages  
//-----  
void pr_averages(struct Stype T[]) {  
    double A1 = Avg(T, 0); // 0 for Korean  
    double A2 = Avg(T, 1); // 1 for English  
    double A3 = Avg(T, 2); // 2 for Math  
    double A4 = Avg(T, 3); // 3 for Averages  
  
    printf("%10s %10.2f %10.2f %10.2f %10.2f \n",  
           "Average", A1, A2, A3, A4);  
}
```

main() definition

```
//=====
// main
//=====
int main(void) {

    // S[i].I --> I[i] // ID of a student
    // S[i].K --> K[i] // Grade of Korean
    // S[i].E --> E[i] // Grade of English
    // S[i].M --> M[i] // Grade of Math
    // S[i].A --> A[i] // a student's Average

    struct Stype S[SIZE];

    // S is the array name
    // S is also the address like pointer variables
    // thus, the following calls are pass-by-reference
    init_arrays(S);
    pr_table(S);
    pr_sorted_table(S);
    pr_averages(S);

}
```

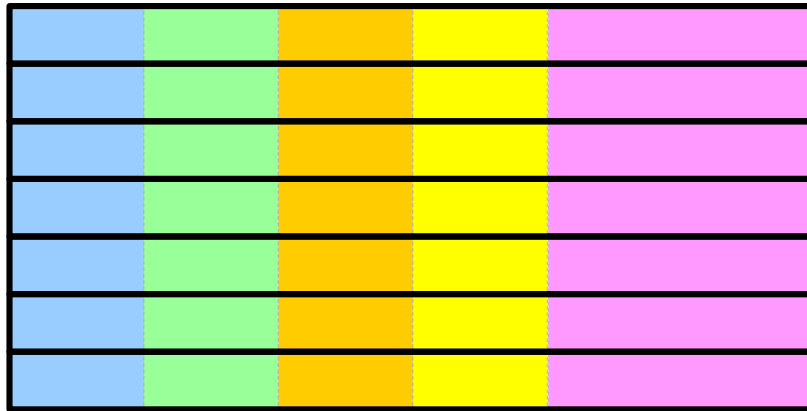
Structure Array S[SIZE]

Student ID Korean English Math Average

$S[i].I$ $S[i].K$ $S[i].E$ $S[i].M$ $S[i].A$

$S[0]$					
$S[1]$					
$S[2]$					
$S[3]$					
$S[4]$					
$S[5]$					
$S[6]$					
$S[7]$					

Memory Layouts of S

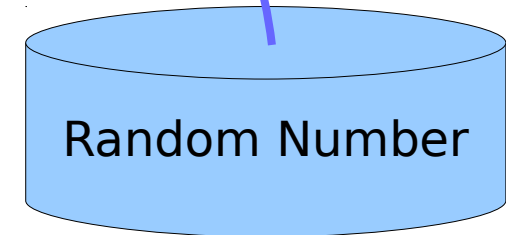


init_arrays() - filling scores

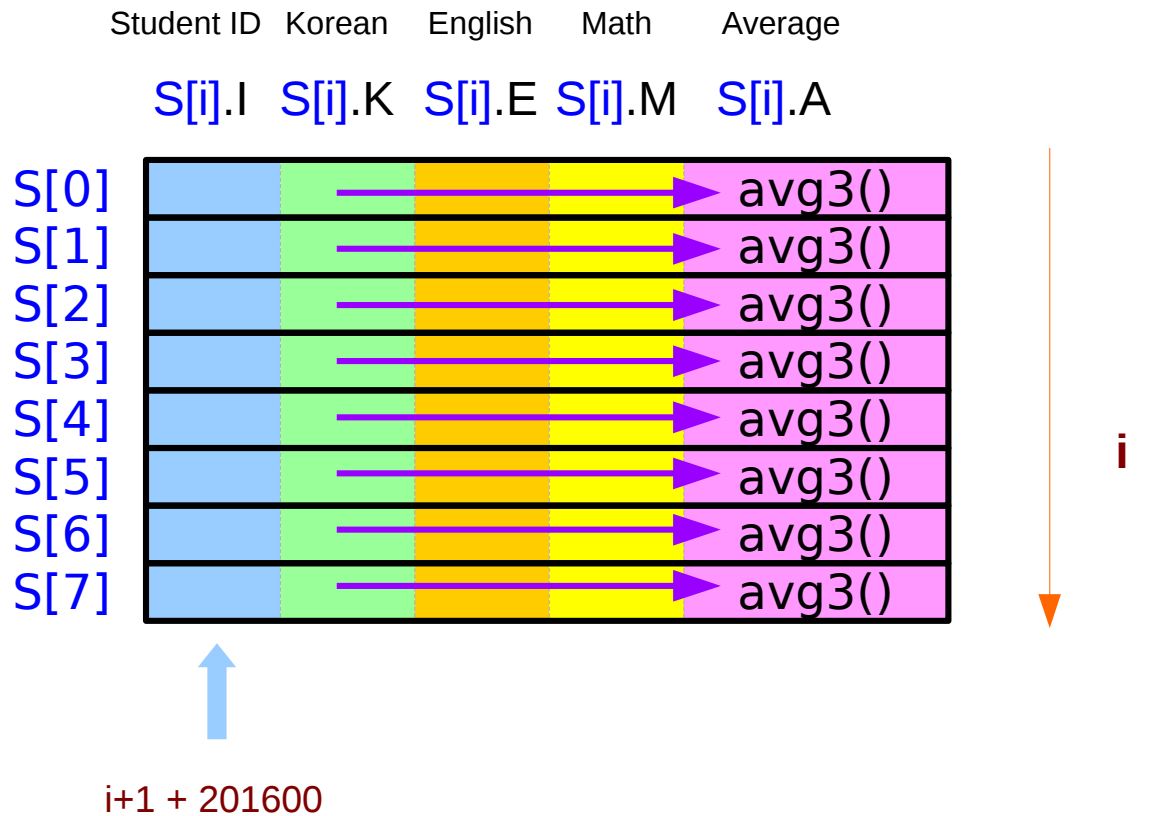
Student ID Korean English Math Average

$S[i].I$ $S[i].K$ $S[i].E$ $S[i].M$ $S[i].A$

$S[0]$		76	44	97	
$S[1]$		86	98		
$S[2]$					
$S[3]$					
$S[4]$					
$S[5]$					
$S[6]$					
$S[7]$					



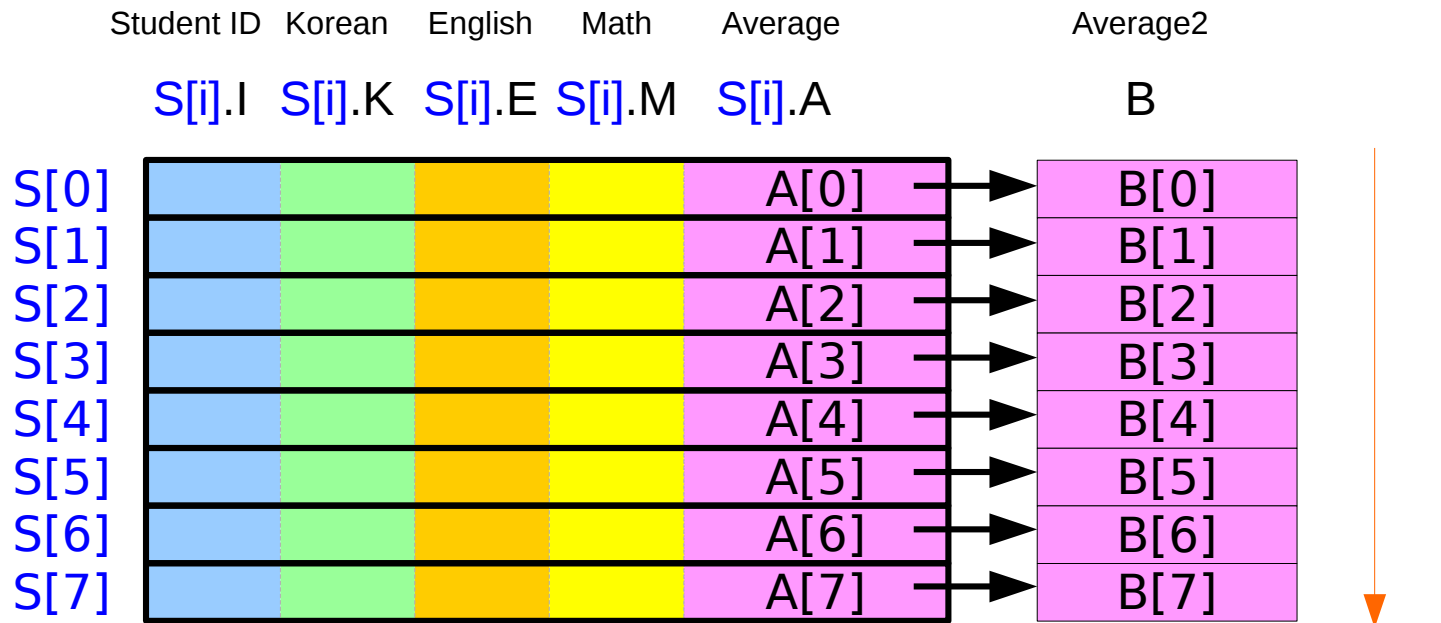
init_arrays() - computing averages



pr_table()

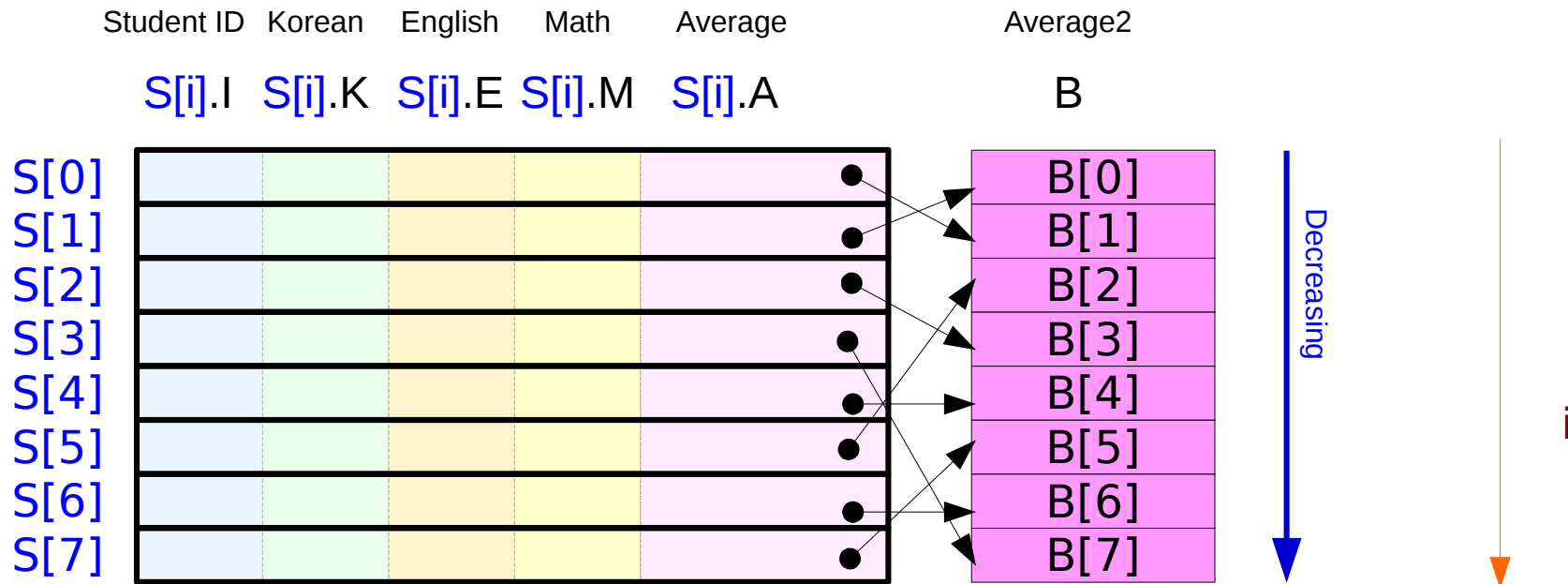
	Student ID	Korean	English	Math	Average
	$S[i].I$	$S[i].K$	$S[i].E$	$S[i].M$	$S[i].A$
i					

pr_sorted_table - copying A to B



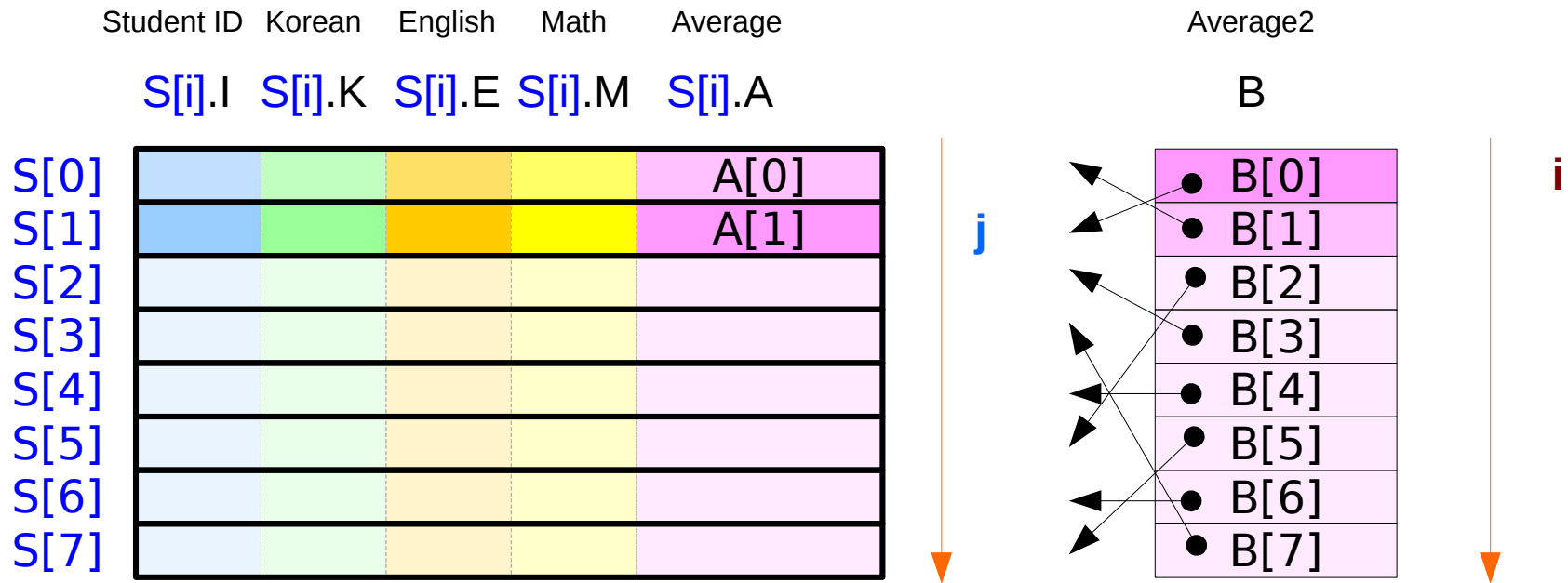
First, copy
 $A[i]$ into $B[i]$

pr_sorted_table - sorting B



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

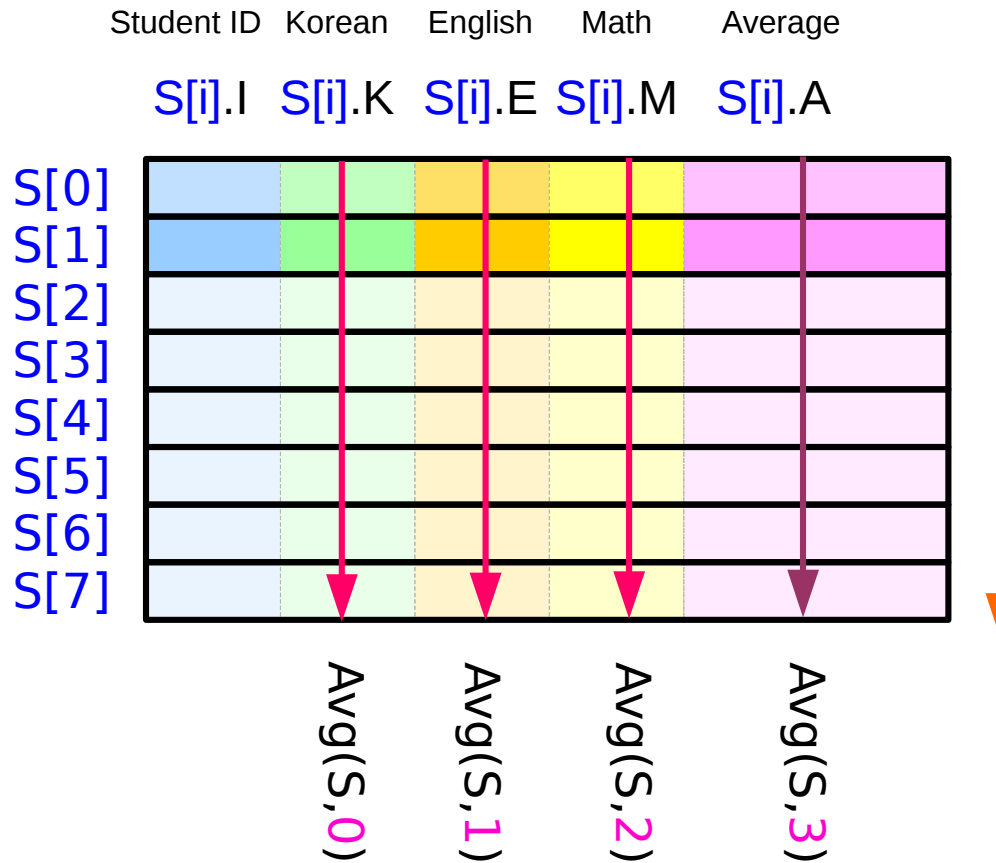
pr_sorted_table - printing by B



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

Assume that two averages have always different values

pr_averages



```
double Avg (struct Stype T[], int n);
```

```
double A1 = Avg(T, 0); // 0 for Korean  
double A2 = Avg(T, 1); // 1 for English  
double A3 = Avg(T, 2); // 2 for Math  
double A4 = Avg(T, 3); // 3 for Averages
```

```
void pr_averages (struct Stype T[]);
```

```
pr_averages(S);
```

Using Struct

```
double    avg3                (int x, int y, int z);  
void      init_arrays        (struct Stype T[]);  
void      pr_table           (struct Stype T[]);  
void      DbubbleSort        (double a[], int size);  
void      pr_sorted_table    (struct Stype T[]);  
double    Avg                (struct Stype T[], int n);  
void      pr_averages        (struct Stype T[]);
```

```
init_arrays(S);  
    T[i].A = avg3(T[i].K, T[i].E, T[i].M);
```

```
pr_table(S);  
pr_sorted_table(S);  
    DbubbleSort(B, SIZE);
```

```
pr_averages(S);  
double A1 = Avg(T, 0); // 0 for Korean  
double A2 = Avg(T, 1); // 1 for English  
double A3 = Avg(T, 2); // 2 for Math  
double A4 = Avg(T, 3); // 3 for Averages
```

Structure Array Definition

```
struct Stype {  
    int I;  
    int K;  
    int E;  
    int M;  
    double A;  
};
```

```
//=====
```

```
// main
```

```
//=====
```

```
int main(void) {
```

```
    struct Stype S[SIZE];
```

```
    // S is the array name
```

```
    // S is also the address like pointer variables
```

```
    // thus, the following calls are pass-by-reference
```

```
    init_arrays(S);
```

```
    pr_table(S);
```

```
    pr_sorted_table(S);
```

```
    pr_averages(S);
```

```
}
```

```
// S[i].I --> I[i] // ID of a student
```

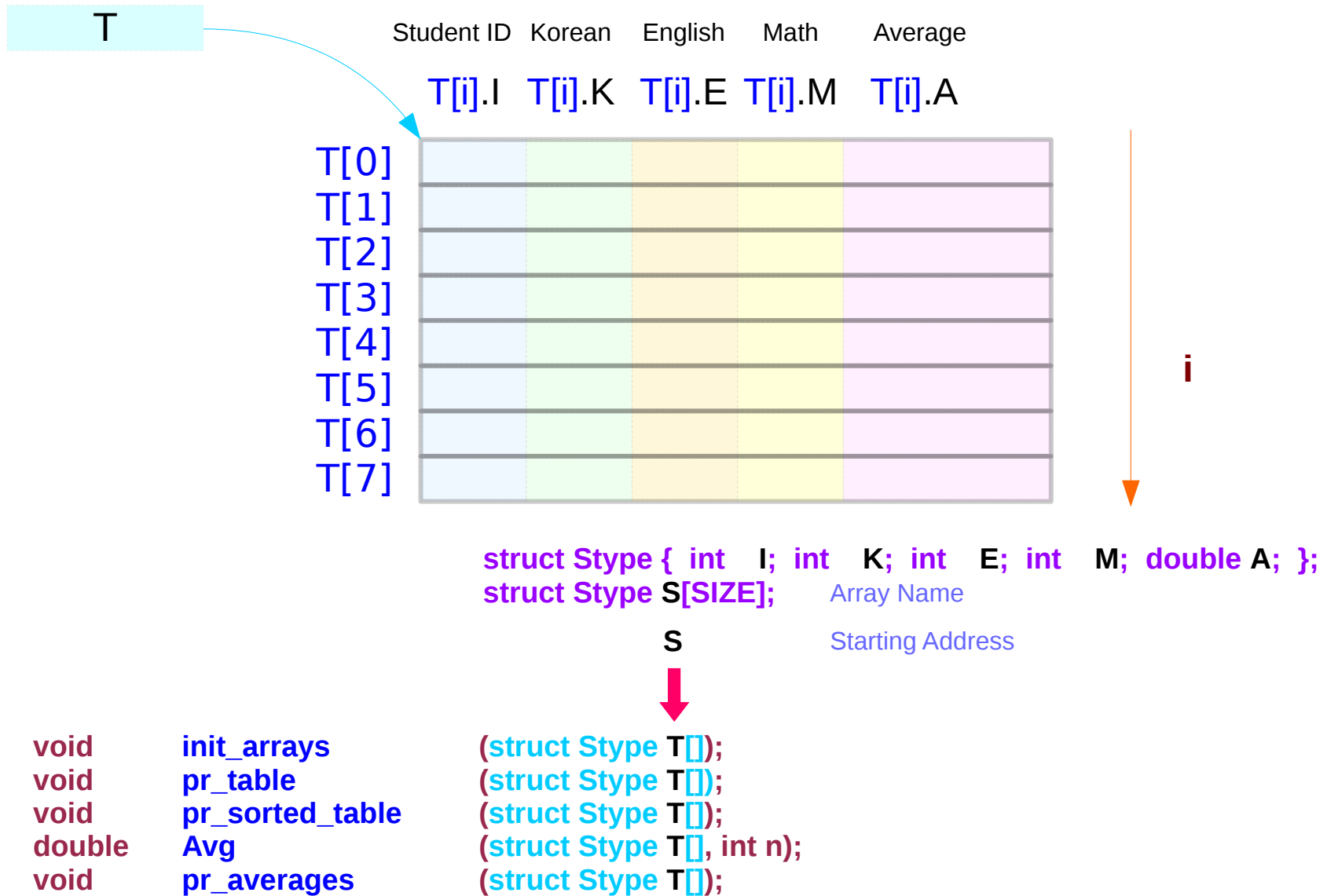
```
// S[i].K --> K[i] // Grade of Korean
```

```
// S[i].E --> E[i] // Grade of English
```

```
// S[i].M --> M[i] // Grade of Math
```

```
// S[i].A --> A[i] // a student's Average
```

struct Stype T[] : formal parameter



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