Applications of Structures (1A)

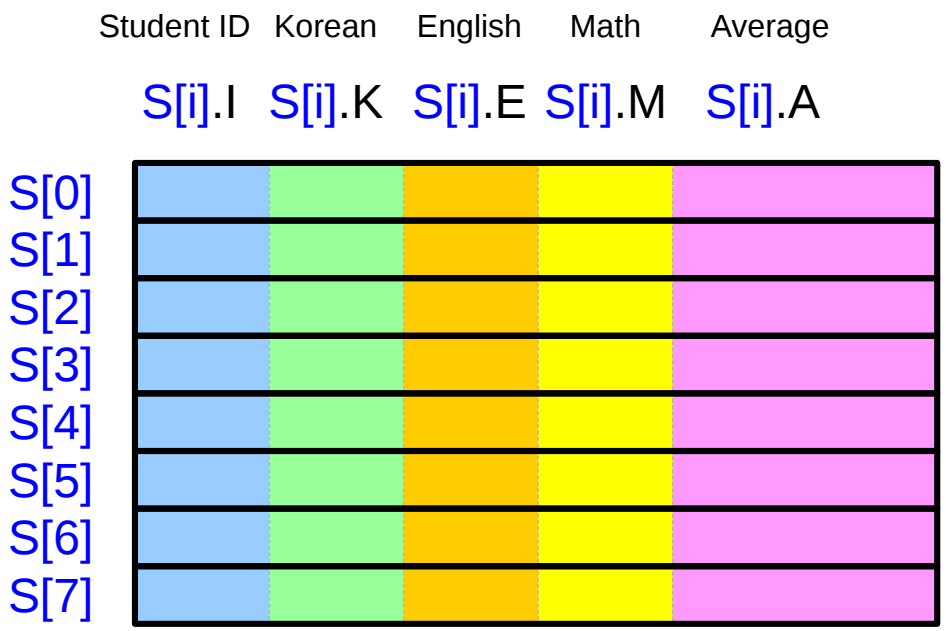
Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using LibreOffice.

Young Won Lim
11/30/17

An Array of Structures
Incomplete Definition of Structures
Self-Referential Structures

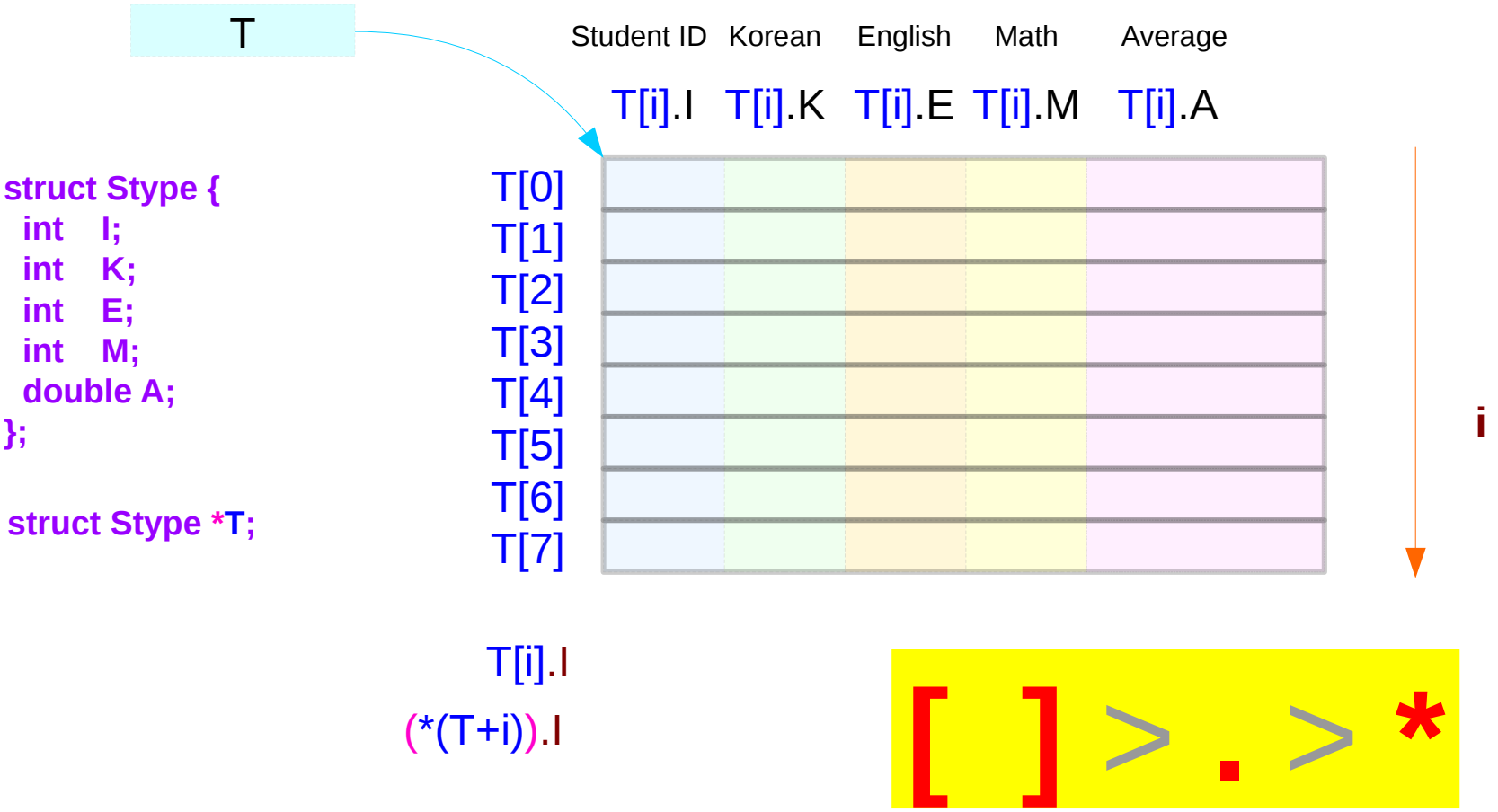
Defining an array of a structure




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

struct Stype S[SIZE];
```

Pointer to a Structure Array



struct Stype T[] : formal parameter

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

struct Stype S[SIZE];
```

Array Name

Starting Address

```
void      init_arrays      (struct Stype T[]);
void      pr_table         (struct Stype T[]);
void      pr_sorted_table  (struct Stype T[]);
double    Avg              (struct Stype T[], int n);
void      pr_averages      (struct Stype T[]);
```

Struct Variable Declaration Summary

```
struct aaa {
        int     i;
        short   s;
        char    c;
} ;
```

```
struct aaa  var;
```

```
struct aaa {
        int     i;
        short   s;
        char    c;
} ;
```

```
typedef struct aaa  ATYPE ;
```

```
ATYPE  var;
```

```
struct aaa {
        int     i;
        short   s;
        char    c;
} var ;
```

```
typedef struct aaa {
        int     i;
        short   s;
        char    c;
} ATYPE ;
```

```
ATYPE  var;
```

Incomplete Definition Range

```
struct aaa { int  i;  short  s;  char  c;  };
```

At all these points, the structure type definition is **incomplete**

From here, the definition complete.

incomplete definition
can be used in cases where the exact **size** information is not required

Dereferencing requires size information
Member assignment require size information
Local variable declaration require size information
Local pointer variable takes always 8bytes on a 64-bit machines

Global variable

Incomplete Definition Usages

```
struct bbb { int i; struct bbb *s; };
```

incomplete definition
can be used with pointer variable declaration

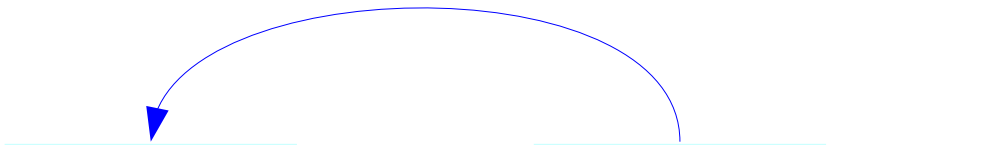
```
typedef struct ccc Ctype;
```

incomplete definition
can be used with **typedef**

```
void func( struct ddd *Ctype );
```

incomplete definition
can be used with pointer parameter declaration

Self-Referential Structures



incomplete definition
can be used with pointer variable declaration

Incomplete Structure Definitions

structure type

```
struct aaa {
int          data;
struct aaa *next;
} ;
```

```
struct aaa   var;
```

structure type

```
struct aaa {
int          data;
struct aaa *next;
} ;
```

```
typedef struct aaa   ATYPE ;
```

```
ATYPE   var;
```

structure type

```
struct aaa {
int          data;
struct aaa *next;
} var ;
```

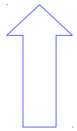
structure type

```
typedef struct aaa {
int          data;
struct aaa *next;
} ATYPE ;
```

```
ATYPE   var;
```

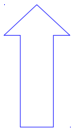
Changing Orders : Global Struct Variable Declaration

```
struct aaa  var;
```



```
typedef struct aaa  ATYPE ;
```

```
ATYPE  var;
```



```
struct aaa  {
int          data;
struct aaa *next;
}  ;
```

```
struct aaa  {
int          data;
struct aaa *next;
} ;
```

Global Structure Variable Declaration is possible before the complete structure definition.

Local Structure Variable Declaration is not possible

```
struct aaa  {
int          data;
struct aaa *next;
}  var ;
```

```
typedef  struct aaa  {
int          data;
struct aaa *next;
}  ATYPE ;
```

```
ATYPE  var;
```

Changing Tag Names

```
struct Atype  var;
```

```
typedef  struct Atype  Atype ;
```

```
Atype  var;
```

```
struct Atype  {
int              data;
struct Atype *next;
}  ;
```

```
struct Atype  {
int              data;
struct Atype *next;
} ;
```

aaa ➡ Atype : tag name

Atype : new type

global variable : var

```
struct Atype  {
int              data;
struct Atype *next;
}  var ;
```

```
typedef  struct Atype  {
int              data;
struct Atype *next;
}  Atype ;
```

```
Atype  var;
```

Types in the List Data Structures

```
struct node {
int     data;
node *next;
} ;
```

```
typedef struct node   node ;
```

```
node          var;
```

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] https://pdos.csail.mit.edu/6.828/2008/readings/pointers.pdf