Topic 2: Pointer & structure

Pointer & structure

- Structure declare

```
struct reg
{
   int ID;
   int score;
}; // must have a semicolon
```

```c
#include <stdio.h>
struct person
{
    int age;
    float weight;
};
int main()
{
    struct person *person_ptr, person1; //person1 was declared as a static variable
    person_ptr = &person1;

    printf("Enter age: ");
    scanf("%d", &person1.age );   // to access static member use .
    // scanf("%d", &(person_ptr->age));

    printf("Enter weight: ");
    scanf("%f", &(*person_ptr).weight );
                    // can be &((*person_ptr).weight)
                    // *person_ptr is the content, (*person_ptr).weight → the weight member

    printf("age: %d, age: %d, weight: %.2f\n", person1.age,
                                               (*person_ptr).age,
                                               person_ptr->weight);
}
```

Pointer & structure

- I like this kind of declaration more. **(remember it!)**

```
typedef struct reg
{
    int ID;
    int score;
} tReg;


tReg *stu_ptr;
stu_ptr = (tReg *) malloc (sizeof(tReg));
stu_ptr ->ID = 10;
```

Pointer & structure (quiz)

```
#include <stdio.h>
#include <stdlib.h>
#define N 4

typedef struct reg
{
    int id;
    int score;
}tReg;

int main (void)
{

    tReg *stu_ptr;
    tReg *head;
    int i;
```

```
    stu_ptr = (tReg *) malloc (sizeof(tReg)*N);
    head = stu_ptr;

    stu_ptr->id = 1; stu_ptr->score = 99;
    stu_ptr ++;
    stu_ptr->id = 2; stu_ptr->score = 80;

    stu_ptr = head;
    stu_ptr[2].id = 40; stu_ptr[2].score = 60;



    stu_ptr = head;
    for (i = 0; i < N ; i++)
    {
        printf("id: %d with score: %d \n",
                    stu_ptr->id, stu_ptr->score);
        stu_ptr ++;
    }
    return 0;
}
```

Pointer & structure

```c
#include <stdio.h>
#include <stdlib.h>
#define N 4

typedef struct reg
{
    int id;
    int score;
}tReg;

int main (void)
{
    tReg student[N]; //not fashion
    tReg *stu_ptr;
    tReg *head;
    int i;

    student[3].id = 0;
    student[3].score = 99;
```

// Assess as a static using •

```c
    stu_ptr = (tReg *) malloc (sizeof(tReg)*N);
    head = stu_ptr; //Avoid lost your way. Store the head

    stu_ptr->id = 1;      stu_ptr->score = 99;
    stu_ptr ++;       // to the next element
    stu_ptr->id = 2;      stu_ptr->score = 80;

    stu_ptr = head; //back to the original
    stu_ptr[2].id = 40;   stu_ptr[2].score = 60;
    //Again, you can access the dynamically allocated memory
    //by the array-accessing fashion

    stu_ptr = head;
    for (i = 0; i < N ; i++)
    {
        printf("id: %d with score: %d \n",
                        stu_ptr->id, stu_ptr->score);
        stu_ptr++;
    }
    return 0;
}
```

Other coding style (I don‘t like)

```
#include <stdio.h>
#include <stdlib.h>

struct reg
{
    int id;
    int score;
}tReg;

typedef struct reg Reg;
typedef Reg *regPtr;

int main(void)
{
    regPtr start_ptr;

    regPtr new_ptr = (regPtr) malloc(xxxxx)
```

Topic 2 assignment

```
Please enter how many numbers in list1: 3
    Please input 3 numbers: 7 1 2
Please enter how many numbers in list2: 4
    Please input 4 numbers: 2 9 5 4

sorted list1: 1 2 7
sorted list2: 2 4 5 9
merged list: 1 2 2 4 5 7 9
```

- Bubble sort + merge
  - Execution result →→→→→
  - Use the following structure. In the structure, there is a p_list to store numbers

```
typedef struct num_list
{
    int counts;
    int *p_list;
}tNumList;
```

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct num_list
{
    int counts;
    int *p_list;
}tNumList;

void fill_list(tNumList *list);
void bubble_sort(tNumList *list);
void merge (tNumList *list1, tNumList *list2);
void print_list (tNumList *list);

int main (void)
{
    tNumList *list1, *list2;
    list1 = (tNumList *) malloc (sizeof(tNumList));
    list2 = (tNumList *) malloc (sizeof(tNumList));
```

You should have these four functions

You should have two number lists
→ list1 and list2

- When merge
  - Don’t apply the bubble_sort function
  - You only need to traverse list1 and list2, and then print directly
  - No new list is allowed
  - Only list1 and list2 in your program
  - Don’t change the contents of list1 and list2

- You cannot declare any array !
  - But, you can access data by the array-accessing fashion

- The p_list should be dynamically allocated