COMP10002 Workshop Week 8

Outlook:

1	Structs & arrays of structs
2	Discuss: Ex 8.2
3	Discuss the use of structs as a way of managing data via a single variable, and look at Exercise 8.2
4	Lab: Working on assignment 1

The legacy of objects…

Supposing that I need to keeps MST scores of students of this class together with names and ID. Supposing that each name has maximum 30 characters.

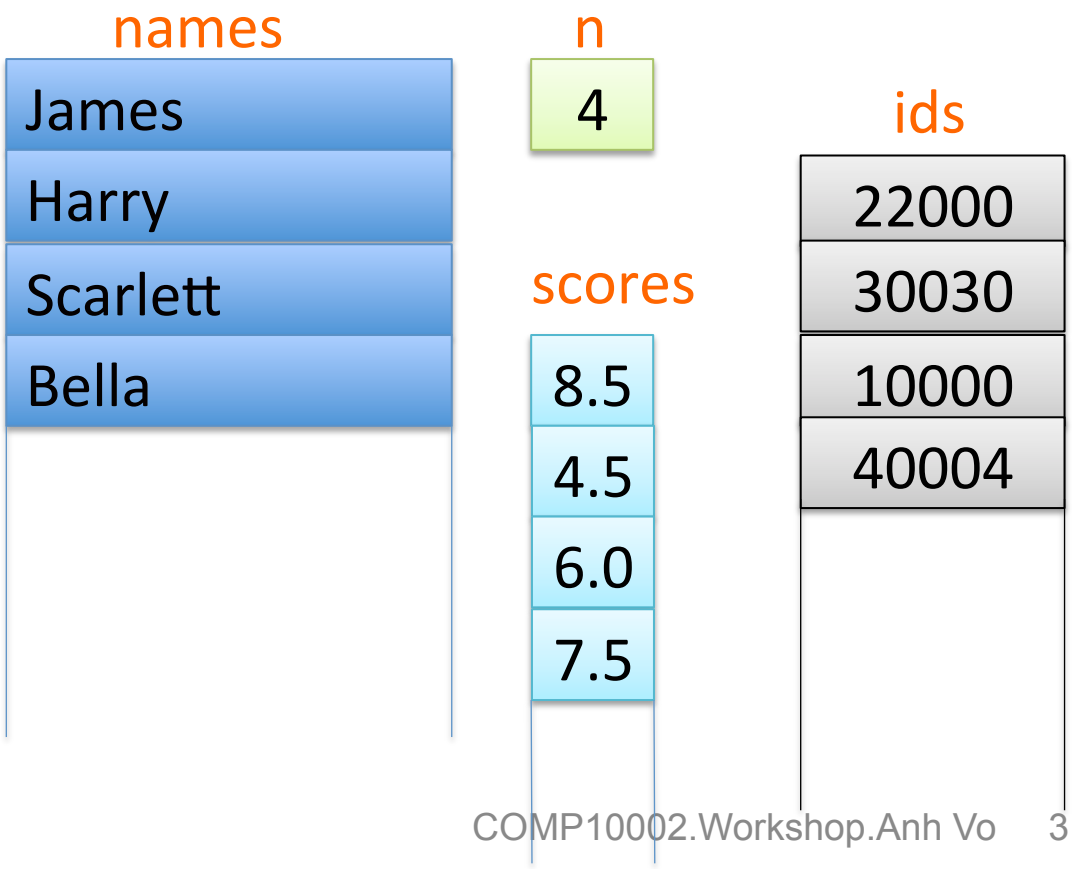
```
#define MAX_S 25
#define MAX_NAME 30
…

char names[MAX_S][MAX_NAME+1];
int ids[MAX_S];
float scores[MAX_S];
int n= 0;     // current number of students
```

Is that a *correct* design?

Is that a *good* design?

The legacy of objects…

```
char names[MAX_S][MAX_NAME+1];
int ids[MAX_S];
float scores[MAX_S];
int n= 0;
```



Hmmm…

The legacy of objects…

```
struct {
   char name[MAX_NAME+1];
   int id;
   float score;
} students[MAX_N];
int n= 0;     // current num of students
```

names	ids	scores
James	22000	8.5
Harry	30030	4.5
Scarlett	10000	6.0
Bella	40004	7.5

n
4

Structures

A structure is a compound object such as a student record. With the declaration:

```
struct {
    char name[MAX_NAME+1];
    int id;
    float score;
} stud;
```

`stud` is an object (a variable) which has 3 components:

```
stud.name
stud.id
stud.score
```

and we can use each component as a conventional variable.

Use typedef for Structures

The best way to use structures is through `typedef` as follow:

```
typedef struct {
   char name[MAX_NAME+1];
   int id;
   float score;
} student_t;
```

Use typedef for Structures

Initialising struct just like arrays:

```
student_t s1= { "Bob" , 1234 , 97.75 };
```

Processing struct by doing so with each component (like arrays):

```
scanf("%s %d %f", s1.name, &s1.id, &s1.score);
printf("name= %s, id= %d, score= %.1f\n",
        s1.name, s1.id, s1.score);
```

But, unlike arrays, we can:

- assignment: `s1= s2;`
- and hence, struct can be the output of a function:

```
student_t best_student(student_t s[], int n)
```

- but note: don't compare `struct`

Pointers to Structures

```
typedef struct {
    char name[31];    // note that "name" is an array
    int id;
    float score;
} student_t;

student_t s1= {"Bob", 1234, 97.75}, s2;
student_t *ps= &s2;
```

The following 2 lines are equivalent:

```
scanf("%s %d %f",  s2.name,  &s2.id,  &s2.score);
scanf("%s %d %f", ps->name, &ps->id, &ps->score);
```

Note: `ps->name` is just a shorthand for `(*ps).name`

Arrays of structs

Arrays of structs are popular. Examples:

- a list of student records:

```
student_t class_list[MAX_S];
int n;
```

- a polygon:

```
typedef struct {
      double x, y;
} point_t;

typedef struct {
   int n;      // number of vertices
   point_t vertices[MAX_VERTICES];
} polygon_t;
```

Exercises (including 8.2, but *changed*)

1. *Define a structure* `vector_t` *that could be used to store points in two dimensions* x *and* y *(such as on a map). Then:*

   a) *Write a function*

   ```
   double distance(vector_t *p1, vector_t *p2)
   ```

   that returns the Euclidean distance between `*p1` *and* `*p2`*.*

   b) *Define a data type for a line segment (on a map) and write a function that computes the midpoint of a segment.*

2. *Write the header of a function that returns a student with highest score from an array of* `n` *students. Use the* `student_t` *we had before.*
3. *Exercise 8.3*
4. *Exercise 8.4*

Assignment1…