COMP20005 Workshop Week 10

Preparation:

- open `grok`, `jEdit`, and `minGW` (or `Terminal` if yours is a `Mac`)
- download this slide set (`ws10.pdf`) from github.com/anhvir/c205 if you like
- open related files for assignment 2

1	`struct` revisited, Ex. 8.1
2	Ex. 8.2-8.4 combined
3	Assignment 1 review
LAB	Assignment 2: • Watch the assignment video • Q&A • Working on assignments OR • Any exercise from W5 to W10, including W5X-W10X

`struct` Revisited: building (simplified) `student_t`

on whiteboard:

Exercise 8.1:
Using RHS, suppose that we have declarations:

```
student_t bill;
student_t allstudents[10000];
```

If an `int` occupies 4 bytes of the memory, and a `char` requires 1 bytes, how many bytes each do `bill` and `allstudents` consume?

```
#define NAMELEN 40
#define MAXSUBJECTS 8

typedef namestr_t char[NAMELEN+1];

typedef struct {
   namestr_t first, others, family;
} fullname_t;

typedef struct {
   int dd, mm, yyyy;
} date_t;

typedef struct {
  int subjectcode;
  date_t enrolled;
  int status;
  int final_mark;
} subject_t;

typedef struct {
  int id;
  fullname_t name;
  date_t dob;
  int nsubjects;
  subject_t subjects[MAXSUBJECTS];
} student_t;
```

struct How to use student_t?

```
typedef struct { /* this is a simplified version */
   int id;          // 4 bytes
   name_t name;     // 3x40= 120 bytes used by name
   date_t dob;      // 3x4 = 12 bytes
} student_t;

//                id           name              dob
student_t bob={10001,{“Ali”,”Boss”,”Mo”},{01,01,2001}};
student_t new;
student_t *ps;
```

write function to read a (simplified) student

```
   typedef struct { /* this is a simplified version */
      int id;          // 4 bytes
      name_t name;     // 3x40= 120 bytes used by name
      date_t dob;      // 3x4 = 12 bytes
   } student_t;

1  ? read_stud( ? ) {
2    scanf("%d %s%s%s %d/%d/%d" ,
3           id, first, others,
4           family, dd,
5           mm, yyyy);
6
7  }
8  ...
9  read_stud(&new);
10
11
12
```

function for input – the bad

```
student_t bad_read_stud() {
  student_t s;
  scanf("%s%s%s %d %d/%d/%d", s.name.given,
        s.name.others, s.name.family,
        &s.id,
        &s.dob.dd, &s.dob.mm, &s.dob.yyyy);
  return s;
}
```

What's bad?

```
student_t stud;
stud= bad_read_stud();
```

function for input – the good

```
void read_stud(student_t *ps) {
   scanf("%s%s%s %d %d/%d/%d", ps->name.given,
     ps->name.others, ps->name.family,
     &ps->id,
     &ps->dob.dd, &ps->dob.mm, &ps->dob.yyyy);
}
```

How to use read_stud? How good is this vesrsion in comparison with the bad guy?

```
student_t stud;
read_stud(&stud)
```

and the not-ugly: using buddy variables to reduce mistakes/stress

```
void read_stud(student_t *ps) {
   fullname_t *pn= &(ps->name);  //no need ()
   fullname_t *pd= &ps->dob;
   scanf("%s%s%s %d %d/%d/%d", pn->given,
     pn->others, pn->family,
     &ps->id,
     &pd->dd, &pd->mm, &pd->yyyy);
}
```

Bravo pointers, bravo ->

Structures: important rules

DON'T:

- use a struct as a function argument
- return a struct

DO:

- use a *pointer to struct* as a function argument
- return a *pointer to struct*
- use buddy variables to reduce complexity of writing multiple struct levels
- use arrays of struct when there is a need to process a number of struct

Case Study: Polygons (Ex 8.2-8.4)

Suppose that a closed polygon is represented as a sequence of points in two dimensions. Give suitable declarations for a type `poly_t` *, assuming that a polygon has no more than 100 points.*

Write a program that includes the functions that

(i) reads a poly from `stdin`

(ii) returns the length of the perimeter of a polygon (ex 8.3, and 8.2?).

(iii) returns the area of a polygon (ex 8.4) .

(iv) return distance between the centroids of two polygon.

*Test these functions using input "*3 0 0 3 0 0 4" (which is a triangle), or something else.

Your Task: *Choose between:*

Do incremental development in your computer, keep up with Anh's speed. OR

Do incremental development is your speed, and give Anh private chat questions. OR

If you already finished this task and ***you are certain that you did well****, do some other exercises or assignment 2, and send Anh private questions on Chat [or post questions on* `grok` *if we are in lab time]*

ass1 review: a sample solution

assignment 2: new items in rubric

- avoidance of structs (eg, using skinny 2d arrays), -1.0;
- avoidance of struct pointers (eg, using whole-struct arguments), -0.5;
- inappropriate or over-complex structs, -0.5;
- other abuses of structs, -0.5;

And not new, but sometime left forgotten:

- errors in compilation that prevent testing, -4.0;
- **unnecessary warning** messages in compilation, **-1.0**;
- runtime segmentation fault on any test with no output generated, -2.0;
- runtime segmentation fault on any other test with no output generated, -2.0;

Assignment 2

(if not done,) skim the spec then watch:

then (if not yet done):

- read, understand Stage 1
- start your `ass2.c` by:
  - copying `ass2_skel.c` to `ass2.c` and sign the declaration
  - *stealing* applicable things from Alistair's movie and sample solution (there is a green light for this stealing) #define PIE 3.14159
  - implementing Stage 1 [*incremental development*]
  - read, understand Stages 2 and 3, and … implement them
- remember to use `struct` (start with a simple design just for stage 1, we can change them later to fit stages 2-4)
- *ask questions, discuss with Anh and everybody*
- submit today (and see if that compiler complains)